НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

НАУКОВІ ЗАПИСКИ НаУКМА
КОМП'ЮТЕРНІ НАУКИ

Том 8 ♦ 2025

Науковий журнал ♦ Щорічник ♦ Заснований у 1996 р.

Київ
2025

**Національний університет «Києво-Могилянська академія»
заснував видання «Наукові записки НаУКМА» 1996 року**

«Наукові записки НаУКМА. Комп'ютерні науки» (англ. NaUKMA Research Papers. Computer Science) — науковий рецензований журнал відкритого доступу, що висвітлює результати наукових досліджень у галузі комп'ютерних наук та інформаційних технологій.

До 2018 р. виходив друком як частина багатосерійного видання «Наукові записки НаУКМА» (серія «Комп'ютерні науки»). З 2018 р. — «Наукові записки НаУКМА. Комп'ютерні науки». Виходить раз на рік.

Пропонований том висвітлює творчі здобутки вчених, аспірантів, студентів факультету інформатики Національного університету «Києво-Могилянська академія», а також фахівців з інших наукових центрів України, що співпрацюють з НаУКМА в галузі інформатики, кібернетики, програмування, за останній рік. Дослідження авторів статей спрямовані на розв'язання теоретичних проблем і практичних завдань у галузі сучасної інформатики.

Журнал адресовано науковцям, фахівцям, викладачам, докторантам, аспірантам, студентам, а також усім зацікавленим у наукових дослідженнях у галузі комп'ютерних наук та інформаційних технологій.

Мови видання: українська, англійська

Здійснюється подвійне анонімне рецензування матеріалів

Засновник і видавець:
Національний університет
«Києво-Могилянська академія»

Ідентифікатор у Реєстрі суб'єктів у сфері медіа: R40-04349

Внесено до Переліку наукових фахових видань України,
в яких можуть публікуватися результати дисертаційних робіт
на здобуття наукових ступенів доктора наук,
кандидата наук і ступеня доктора філософії,
категорія «Б» (наказ МОН України від 02.07.2020 № 886)

ПЕРЕДМОВА

Пропонуємо вашій увазі випуск наукового журналу «Наукові записки НаУКМА. Комп'ютерні науки». Цей збірник відображає творчі досягнення вчених і магістрів факультету інформатики Національного університету «Києво-Могилянська академія», а також провідних науковців із різних наукових центрів України, що взаємодіють із нашим університетом у галузі інформатики, кібернетики та програмування.

У статтях цього тому розглянуто теоретичні аспекти кібернетики і програмування, а також практичні застосування програмних технологій у галузі сучасної інформатики та прикладної математики. Цей випуск є результатом досліджень наших вчених і втілює наше прагнення збагатити та розширити горизонти знань у цих стратегічних галузях.

Висловлюємо щиру вдячність керівництву НаУКМА за їхню підтримку. Це видання стало можливим завдяки їхній допомозі та вірі у значущість наших досліджень.

До публікації у збірнику редакційною колегією рекомендовано 31 статтю за такими тематичними напрямами: нейронні мережі та машинне навчання, комп'ютерні науки, програмна інженерія.

Щиро дякуємо всім авторам за співпрацю, а науковцям за рецензування.

Голова редколегії збірника
М. М. Глибовець

УДК 004.032.26
DOI: 10.18523/2617-3808.2025.8.4-14

S. Medvid

MORPHONAS-BENCH: A BENCHMARK SUITE FOR MORPHOGENETIC NEURAL NETWORK GENERATION

We present MorphoNAS-Bench, a benchmark and toolkit for neural architecture search (NAS) using a generative, developmentally inspired design space. Unlike current NAS benchmark datasets (NAS-Bench-101, NATS-Bench) that use static graph encodings of networks, in MorphoNAS-Bench networks are simple, compact genomes that drive morphogenetic development, allowing for a variety of richly defined, spatially embedded recurrent architectures that emerge through different forms of deterministic growth. The following local developmental rules are used in MorphoNAS to grow genomes: morphogen diffusion, cell division, differentiation, and axon guidance as key mechanisms. The seed benchmark dataset presented in this work consists of 1,000 genome-architecture pairs, taken from a pool of over 50,000 generation attempts using the following quality thresholds: a minimum 5 neurons, 3 edges, and 70% out-degree coverage. The dataset was constructed using Latin Hypercube Sampling (LHS) with orthogonal array design to ensure comprehensive parameter space coverage. The attempts were conducted using both fully stratified parameter sampling and a biologically inspired Genome.random() sampling method, ensuring a reasonable level of coverage of the search space while being plausible. Each sample includes detailed annotations of graph entropy, hierarchy scores, core-periphery structure, transitivity, reciprocity, and structural balance metrics. We share an analysis of the emergent properties like size, modularity, grouping, and efficiency, demonstrating that both generation strategies can produce structured networks that are rich in their nontriviality. The provided Python toolkit provides the means of investigation to test how genomes develop into neural networks, with associated structural analysis, framing MorphoNAS-Bench as a reproducible and biologically inspired testbed for any research studies exploring architecture diversity, evolution, and emergent structure in NAS.

Keywords: neural networks, developmental encoding, morphogenetic development, neural architecture search, benchmark toolkit, emergent modularity, indirect encoding.

1. Introduction

Neural Architecture Search (NAS) attempts to automate the design of neural networks. Typically, NAS has searched a fixed set of architectures that are normally explicitly encoded as graphs or modules. Benchmarks like NAS-Bench-101 and NATS-Bench have been critical for reproducible research, but still use a top-down framework that constrains variability and is completely different from the biological processes generating real neural systems.

MorphoNAS-Bench is a benchmark that addresses this gap by being grounded in a generative search space. MorphoNAS-Bench does not directly encode networks, but instead encodes a compact genome for networks that develop into a neural architecture through simulated morphogenetic development, inspired by biological embryogenesis. Each morphogenetic genome specifies local morphogen diffusion, cell division, differentiation, and axon guidance, and as a result a diverse population of spatially embedded, recurrent neural networks arise through deterministic simulation.

This paper will present MorphoNAS-Bench as a dataset and toolkit, including a stratified sample of 1,000 seed architecture genomes generated for several significant developmental parameters, including fully stratified sampling and a biologically plausible Genome.random() method. Each network genome generates a spatially embedded neural network with recorded metrics based on node number, degree distributions, clustering, and spatial organization. Also, we include Python scripts and utility programs for the creation, development, and evaluation of genome neural networks.

Our project fosters reproducible experimentation with developmental NAS methods by introducing additional populations of neural architecture designs that are biologically grounded for additional algorithmic exploration. MorphoNAS-Bench can provide the context to evaluate emergent properties of the resulting networks, as well as design NAS algorithms for a space where architectures can develop, as opposed to the creation of neural networks using a space defined by a human designer.

We built the benchmark on the theoretical framework we described in more detail previously in [6]. It outlined the theoretical intent and generative approach based on morphogenetic growth, inspired by the Free Energy Principle. While that paper primarily deals with the foundational aspects of MorphoNAS-Bench, this work provides a benchmark to investigate structural diversity and architecture design potential of genomically grown networks.

2. Related Work

2.1. Neural Architecture Search and Benchmarks

Neural Architecture Search (NAS) has become an important framework for automating the design of neural networks by exploring large search spaces using optimization techniques such as reinforcement learning, evolutionary strategies, and differentiable approaches. One notable advance has been the manual development of the NAS benchmarks that facilitate reproducible and fair comparisons among algorithms.

NAS-Bench-101 [15] was the first detailed tabular benchmark which consists of over 423,000 cell-based architectures and is evaluated on CIFAR-10. Each architecture was trained using a standardized protocol, with a record of performance metric values in a lookup table that allowed comparative performance checks with little latency. Following that study, **NAS-Bench-201** [4] developed a smaller and more controlled search space and observed the performance across CIFAR-10, CIFAR-100, and ImageNet-16. This expanded and allowed for generalization analysis and also provided an increase in analytical efficiency for testing NAS methods.

NATS-Bench [3] added on to the paradigm by also considering macro and micro architectures, and allowing training-free evaluations and weight-sharing methods. These benchmarks have allowed for a shift in the community towards more rigorous, transparent, and standardized evaluation of NAS architectures.

Other approaches have pioneered the use of differentiable NAS, such as **DARTS** [8], where architecture weights are optimized with model parameters, providing a fast one-shot training method. While these approaches represent advances, they typically only consider very constrained, manually encoded architecture spaces.

One common attribute in current NAS benchmark studies is that they all rely on **explicitly encoded architectures** as directed acyclic graphs (DAGs) or operation lists. As such, the graphs are generic, static, and result in non-generative architecture spaces, which limit researchers' exploration of open-ended or biologically inspired architecture spaces.

2.2. Developmental and Generative Approaches to Architecture Design

There is increasing interest in the composition of **developmental** and **indirectly encoded** neural systems outside of the classical NAS paradigm, where the architecture arises from a compact generative set of rules rather than a directly specified list of components.

There are experimental examples of this emerging approach beginning with NEAT [13], in which both topologies and weights are evolved from direct mutations and crossover of graph-type structures, and **HyperNEAT** [12], which used **Compositional Pattern Producing Networks (CPPNs)** to indirectly encode relationships between elements in a connectivity pattern based on geometric distances. This approach demonstrated the potential of indirect encodings in terms of potentially producing binary representations for traditional regular scalable networks.

Generative systems have started emerging only recently, in areas such as neural tissue simulation [10], modular robot morphogenesis [14], and procedural graph generation [2].

Most of such frameworks still remain largely domain specific, and there does not yet exist a more generalized evaluation platform that permits systematic algorithmic comparisons across tasks and domains. There are some benchmarks (e.g. PCG Benchmark [7], Evolution Gym [1]) that give a little bit of structure within a single domain, but there is still no broadly applicable platform that allows for head-to-head evaluation of generative algorithms across datasets that include neural simulation, robot morphology, and graph-based generation tasks [5].

2.3. Positioning of MorphoNAS-Bench

MorphoNAS-Bench is a step forward with a benchmark for neural architectures designed by developmentally generating them. It combines a genome encoding inspired by biology that defines morphogen dynamics and axonal growth, a deterministic simulator that generates spatially embedded recurrent graphs, and a curated database of valid genomes and their structural metadata. Unlike previous NAS benchmarks [3, 4, 15], MorphoNAS-Bench provides a model for a generative architecture space where topology and function emerge through simple local interaction rules and can be employed to explore evolvability/compactness/biological realism, which none of the existing tabular datasets can do. MorphoNAS-Bench can contribute to the neuroevolution, NAS, and developmental computation communities by providing a vehicle for looking at architectures that grow organically, as opposed to just by refinement. It is an additional, open-ended avenue for architecture search.

3. MorphoNAS-Bench Overview

This section provides descriptions of the components of the MorphoNAS-Bench search space, the operational genome encoding format, and the resulting networks' properties.

3.1. Genome Encoding

The genome representation in MorphoNAS-Bench defines morphogen diffusion, cell fate thresholds, and axon guidance rules that drive network development. Broadly, these mechanisms follow the generative model in a separate theoretical paper [6]. The specification of the genome structure, parameter definitions, and configuration options are available on the MorphoNAS-Bench GitHub repository https://github.com/sergemedvid/MorphoNAS-Bench.

3.2. Morphogenetic Development Process

The network growth is implemented using a deterministic simulation of developmental dynamics based on the genome encoding provided above. A detailed structure and model of these dynamics, including biological motivation, is provided in the supplementary theoretical paper [6].

3.3. Comparison with Traditional NAS Spaces

MorphoNAS-Bench introduces a **complementary design space** for NAS, which enables research of the evolutionary and generative methods that are often incompatible with standard benchmarks.

Table 1. **Comparison of MorphoNAS-Bench with traditional NAS spaces**

Property	MorphoNAS-Bench	Traditional NAS Benchmarks
Encoding Type	Indirect (genome → process)	Direct (graph / op list)
Topology Generation	Emergent via simulation	Static, predefined graph space
Network Size	Variable	Fixed or bounded
Biological Plausibility	High	Low
Interpretability	High (local rules → global form)	Medium
Reproducibility	Deterministic	Deterministic
Search Space Diversity	Very high	Limited by architecture schema

4. Benchmark Dataset and Evaluation Tasks

MorphoNAS-Bench includes a curated, developmentally-grounded benchmark dataset designed to characterize and uncover the search space of architectures generated from within the MorphoNAS framework. The benchmark is simply a starting point for search, discovery, and exploration of diversity within a biologically-inspired generative space, rather than for comparison of fitness value against predefined graph targets. This section describes the dataset construction, evaluation tasks, and supporting metrics to study emergent neural architectures.

4.1. Dataset Construction

The construction of the MorphoNAS-Bench dataset was accomplished via large numbers of simulated genomes, using stratified sampling [11], as well as filtering and post hoc analysis after simulated growth was complete. All genomes simulated through the morphogenetic growth process, generate a directed, weighted recurrent neural network, and, with it the structural and functional properties can then be analyzed.

4.2. Genome Generation

Genome sampling involved a combination of Latin Hypercube Sampling (LHS) methodology [9] and orthogonal array design to maximize coverage and statistical quality of core parameters. Two sampling strategies were used. The first was a fully stratified sampling of all queried parameter ranges, and the second, a biologically-informed Genome.random(), are available at GitHub https://github.com/sergemedvid/MorphoNAS-Bench. The stratified sampling maximizes space coverage by sampling parameters independently of each other, which could allow for some impossible combinations, while the biologically-informed sampling also stratifies core traits, while the additional fields are filled using domain-informed constraints (i.e., normalized diffusion and inhibition matrices, morphogen probabilities) to ensure biologically plausible combinations. Each Genome JSON is fully traceable in the generation_metadata.json file, where all parameters, random seed, CLI flags, and code version used to generate that genome are recorded.

4.3. In-Loop Quality Filtering

Real-time filtering occurs after the growth process is finished: only networks with a minimum of five neurons, at least three edges, and no less than 70% of neurons with an outgoing connection, are included in the dataset. We have additional thresholds on weak edge connectivity and density to eliminate trivial and degenerate topologies to ensure only structurally meaningful architectures enter the dataset. The pipeline will also keep track of the successful and failed generations to corroborate further analysis of the generative space within the framework.

By default, we use LHS over the parameter space with random morphogen-to-rule mappings (i.e., Genome.random()) set, while if the --no-genome-random flag is used, the mappings become deterministic with sampling from the rule table.

5. Toolkit and Implementation Resources

The MorphoNAS-Bench toolkit and resources, including genome generation scripts, morphogenetic development engine, visualization tools, and evaluation pipelines, is openly accessible at https://github.com/sergemedvid/MorphoNAS-Bench. Within the MorphoNAS-Bench codebase, the README contains important documentation, configuration examples, and Jupyter-compatible workflows for reproducing results and expanding the benchmark suite. We encourage users to explore the README and API documentation if they would like to integrate the MorphoNAS-Bench codebase into their own NAS pipelines or other developmental modeling experiments.

6. Baseline Results and Use Cases

To explore the structure, diversity, and functional viability of MorphoNAS-Bench, we completed baseline evaluations. This section outlines examples that highlight the diversity of the generative space and demonstrates the structural expressiveness of the underlying model, as well as provides summary use cases for researchers that work in the area of neural architecture search (NAS) and/or generative graph modeling.

6.1. Generation Metadata and Sample Efficiency

While both generation methods used the same post-growth filtering, requiring at least five neurons, weak connectivity, and 70% out-degree coverage, there are significant differences in sample efficiency for the two generation strategies. In the fully stratified method, for the generation of 1000 filtered genomes, there were 46822 rejected based on insufficient node count and 2610 rejected based on disconnectedness. In the Genome.random() method, for the generation of 1000 filtered genomes, there were 23817 rejected based on insufficient node count and another 2626 rejected based on disconnectedness. This indicates that the Genome.random() method resulted in a higher yield of valid architectures, and generated overall fewer genomes to arrive at each individual valid architecture.

This comparison shows the benefits of biologically inspired genome construction; it preserves diversity and expressiveness in the search space while being able to generate functionally plausible and structurally correct networks more quickly. The results, in the end, confirm both the generative richness of MorphoNAS, and the need for filtering to arrive at viable meaningful neural architectures.

6.2. Structural Metrics and Emergent Patterns

We conducted an extensive structural characterization of **1000 networks** sampled from the benchmark. The structural analysis shows the emergence of a range of topological properties and is shown in **Figures 1 and 2** (*advanced network metrics analyses*).

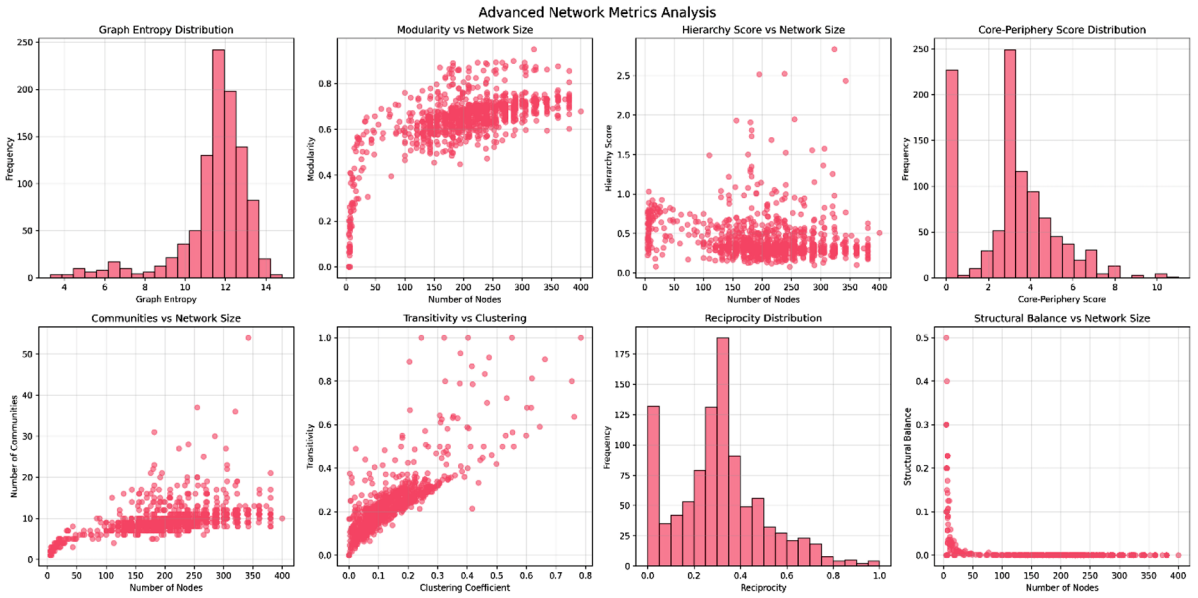


Figure 1. Advanced network metrics analysis for networks generated with Genome.random()

Both generation strategies produce networks that range in size and types of topology, although subtle structural distributional differences are indicated in Table 1.

Table 2. **Comparison of advanced network metrics for Genome.random() and fully stratified sampling (1,000 Samples)**

Metric / Pattern	Genome.random (biologically plausible)	Fully Stratified
Graph Entropy	More peaked, shifted higher (more complex)	Broader, more low-entropy outliers
Modularity (vs. Size)	Higher modularity, esp. at large sizes	More moderate modularity at large size
Hierarchy Score	More high-hierarchy outliers, esp. large	Mostly low-moderate, fewer outliers
Core-Periphery Score	More uniformly low-moderate	Pronounced secondary mid-range peak
Communities (vs. Size)	Tighter, higher at large sizes	More spread, fewer at large sizes
Transitivity vs. Clustering	More high-transitivity outliers	Tighter, lower overall
Reciprocity Distribution	Narrower, centered at mid-range	Broader, more low/high outliers
Structural Balance (vs. Size)	Mostly low, very few outliers	More outliers at small sizes

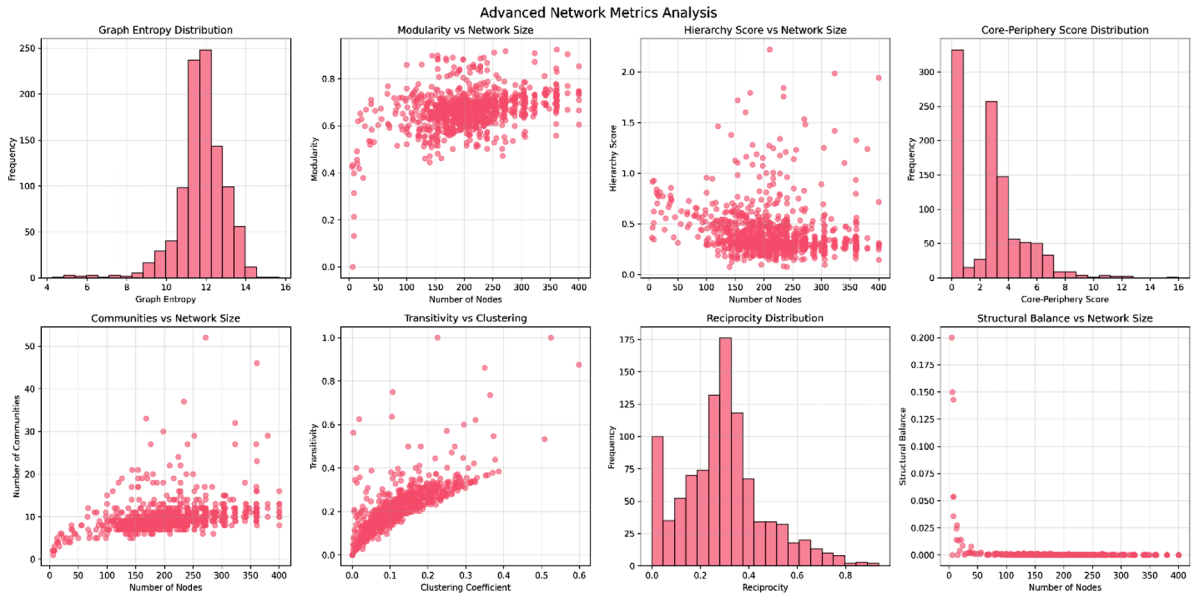


Figure 2. Advanced network metrics analysis for networks generated with the fully stratified method

6.3. Parameter Influence and Developmental Constraints

In order to understand how genome parameters influence the final architectures, we correlated the genome settings with network-level outcomes. **Figures 3 and 4** (genome parameters vs. network properties) depict these relationships.

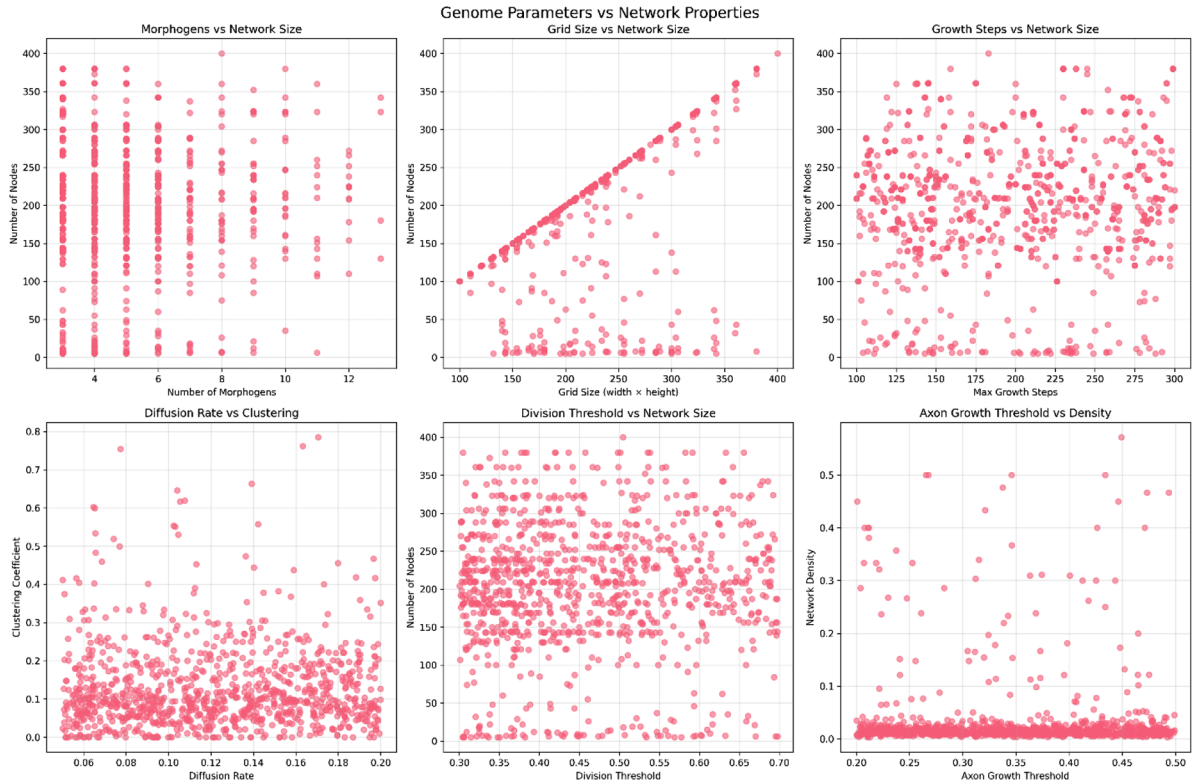


Figure 3. Genome parameters vs. network properties for genomes generated with Genome.random() method

Both generation strategies offer a broad exploration of how genome parameters shape network properties, although subtle differences in their variability and outlier behavior distinguish the two distinct methods, and are summarized further in Table 2.

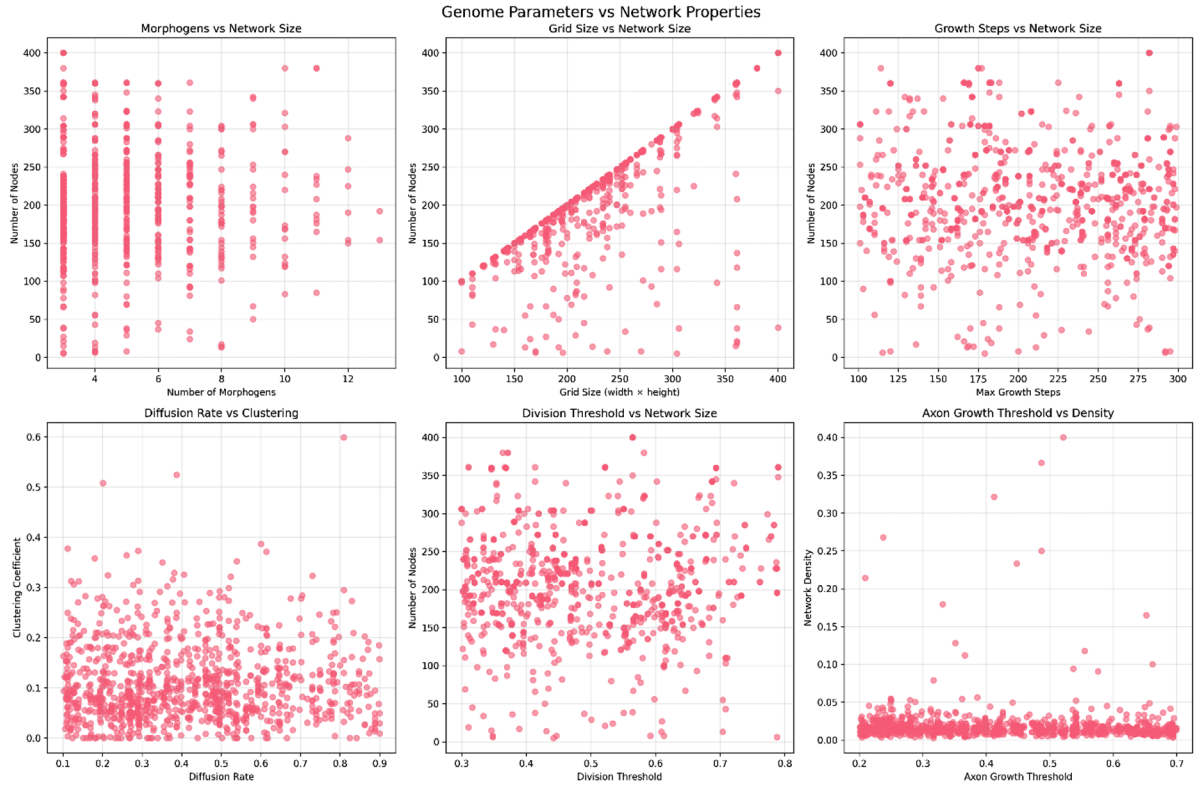


Figure 4. Genome parameters vs. network properties for genomes generated with the fully stratified method

Table 3. **Comparison of genome parameter effects on network properties for Genome.random() and fully stratified sampling (1,000 samples)**

Metric / Relationship	Genome.random (biologically plausible)	Fully Stratified
Morphogens vs. Network Size	Broad spread of network sizes at each morphogen count; more vertical variability across range	Similar broad spread, with slightly more high-size outliers at some counts
Grid Size vs. Network Size	Strong positive correlation, some dispersion; a few outlier networks above/below main diagonal	Very tight correlation, minimal dispersion; nearly all networks tightly follow grid area
Growth Steps vs. Network Size	Little direct correlation; wide vertical spread at all values	Same: step count does not predict size, wide spread across the range
Diffusion Rate vs. Clustering	Mostly low clustering; a few higher outliers up to 0.6	Mostly low clustering, but a few very high outliers (up to 0.8); wider spread at low diffusion rates
Division Threshold vs. Network Size	No strong trend; wide range of sizes at each threshold	No trend, but slightly broader vertical spread in network sizes at all thresholds
Axon Growth Threshold vs. Density	Density concentrated at low values, with a few moderate outliers; no strong relationship	Also concentrated at low density, but outliers reach higher density (up to ~0.5), especially at lower thresholds

Both methods provide a broad exploration of the parameter-to-architecture mapping of networks, although the **fully stratified sampling** indicates even more extreme outliers for clustering and density when compared to **Genome.random**, which produces a marginally more regular and reasonable distribution. Network size tended to dominate any reasonable number of genome parameters that modulated a range of structural properties, with considerable variability indicating the richness and complexity of the proposed generative developmental process.

6.4. Benchmark-Wide Structural Profiles

Figures 5 and 6 (*network structural analysis*) summarize general distributional properties across the 1,000-network benchmark.

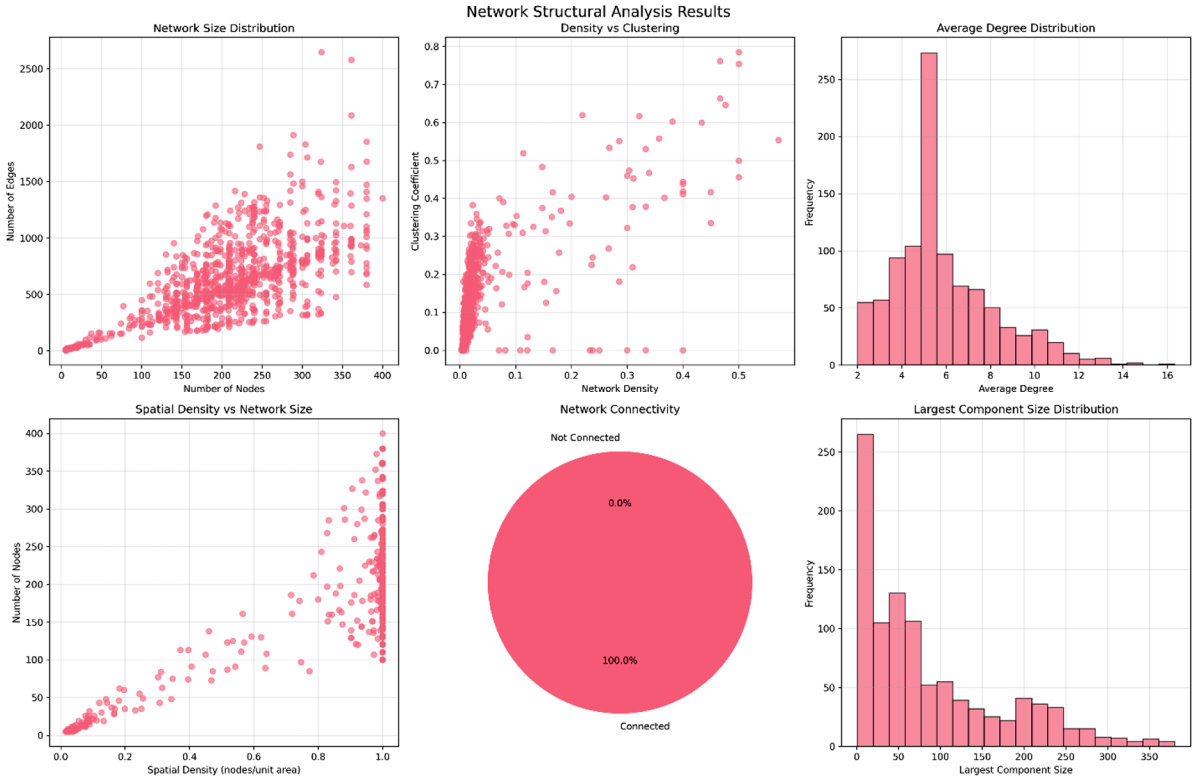


Figure 5. Network structural analysis results for genomes generated with Genome.random() method

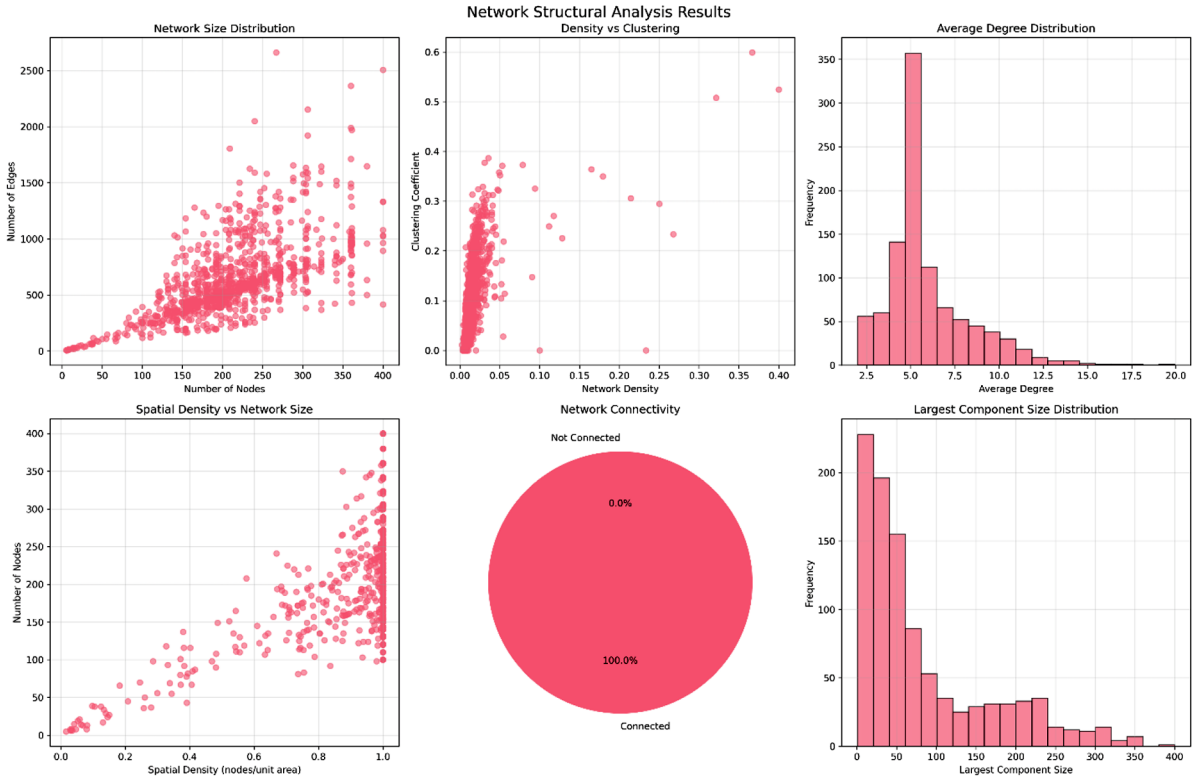


Figure 6. Network structural analysis results for genomes generated with the fully stratified method

In practical terms, **the Genome.random method presents optimal NAS research for applicable conditions**, as the search space is diverse and populated with functional, meaningful architectures requiring little filtering or alteration following generation.

In contrast, those wishing to assess structural diversity without bounds, or to benchmark NAS algorithms under the most extreme and unpredictable conditions, would likely **benefit from completely stratified sampling** since this allows exposure through the search process to a range of network topologies that included even the edge cases. For the purposes of establishing a recommendation, we suggest the Genome. random method as the primary approach for experimental evaluation, and use fully stratified sampling as a supplementary technique for assessing diversity and conducting ablation analyses.

6.5. Summary

This section has indicated that MorphoNAS-Bench is structurally rich, viable for assessing NAS search space, quantitatively diverse across genome and network features, and maintains transparent filtering and reproducibility. By bringing together developmental encoding, statistical sampling, and structural annotation, MorphoNAS-Bench provides a generative, biologically-based search space that can be investigated, analyzed, and benchmarked, filling an identifiable knowledge gap in NAS research.

7. Conclusion and Future Work

MorphoNAS-Bench presents a new methodology for neural architecture search benchmarks. In this regard, we have transitioned from static graph encodings to networks that are developed and shaped through a biologically inspired growth process. This enables us to produce a tight, generative, reproducible search space that is rich in structural and functional variation. Through careful parameter sampling, quality filtering, and analysis, we have shown that our benchmark samples form a larger, controllable architecture space, with sampled networks that are structurally complex and functionally able. We discovered that basic NAS and evolutionary algorithms are able to explore the architecture space without difficulties, establishing its accessibility, yet also its challenging nature. Our methodology differs from previous benchmarks, by not restricting the search space format to a set of cells or sequences of operations. The design of MorphoNAS-Bench opens a pathway for researchers to examine, among others, indirect encodings, morphogenetic processes, developmental constraints, evolvability, modularity, and variation in a biologically plausible manner, as well as apply metrics for evaluating search strategies on the basis of performance, coverage, structural novelty, and generative robustness.

Future Work

Future work is possible towards several promising opportunities to extend MorphoNAS-Bench. **Extending functional tasks** will be accomplished by adding new, reiterative reinforcement learning environments, supervised learning challenges, and evaluations of transfer or multi-task generalization. **Integrating a search strategy** is about developing benchmarks and competitions for differentiable, gradient-free and/or meta-learning NAS methods, with baseline comparisons of sample efficiency, diversity, and convergence. **Genotype-phenotype mapping** may involve studies related to redundancy, robustness, locality, and the neutral networks and mutational neighborhoods structure. We plan to **expand the benchmark** to become increasingly large with tiered datasets consisting of over 100,000 genomes, creating intentional parameters that allow only configurations for low-resource, diversity-based, or task-based evaluations. Finally, there is a rich opportunity for additional **tooling and visualization** within the structure of MorphoNAS-Bench; for example, bringing web-based viewers (that showcase a demographic of developmental simulations and network exploration) and surrogate predictors trained from the neural architecture benchmark data to support the genome-based NAS.

Список літератури

1. Bhatia J. Evolution gym: A large-scale benchmark for evolving soft robots / J. Bhatia, H. Jackson, Y. Tian, et al. // Advances in Neural Information Processing Systems. — 2021. — Vol. 34. — Pp. 2201–2214.
2. Deitke M. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation / M. Deitke, E. VanderBilt, A. Herrasti, et al. // Advances in Neural Information Processing Systems. — 2022. — Vol. 35. — Pp. 5982–5994.
3. Dong X. Nats-bench: Benchmarking nas algorithms for architecture topology and size / X. Dong, L. Liu, K. Musial, B. Gabrys // IEEE transactions on pattern analysis and machine intelligence. — 2021. — Vol. 44, no.7. — Pp. 3634–3646.
4. Dong X. Nas-bench-201: Extending the scope of reproducible neural architecture search / X. Dong, Y. Yang // arXiv preprint arXiv:2001.00326. — 2020.

5. Geng H. RoboVerse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning / H. Geng, F. Wang, S. Wei, et al. // arXiv preprint arXiv:2504.18904. — 2025.
6. Glybovets M. MorphoNAS: Embryogenic Neural Architecture Search Through Morphogen-Guided Development / M. Glybovets, S. Medvid // arXiv preprint arXiv:2507.13785. — 2025. — https://arxiv.org/abs/2507.13785.
7. Khalifa A. The Procedural Content Generation Benchmark: An Open-source Testbed for Generative Challenges in Games / A. Khalifa, R. Gallotta, M. Barthet, et al. — 2025.
8. Liu H. Darts: Differentiable architecture search / H. Liu, K. Simonyan, Y. Yang // arXiv preprint arXiv:1806.09055. — 2018.
9. McKay M. Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code / M. McKay, R. Beckman, W. Conover // Technometrics. — 1979. — Vol. 21, no. 2. — Pp. 239–245.
10. Migliore M. Parallel network simulations with NEURON / M. Migliore, C. Cannia, W. W. Lytton, et al. // Journal of computational neuroscience. — 2006. — Vol. 21, no. 2. — Pp. 119–129.
11. Neyman J. On the Two Different Aspects of the Representative Method: The Method of Stratified Sampling and the Method of Purposive Selection / J. Neyman // Journal of the Royal Statistical Society. — 1934. — Vol. 97, no. 4. — Pp. 558–625.
12. Stanley K. O. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks / K. O. Stanley, D. B. D'Ambrosio, J. Gauci // Artificial Life. — 2009. — https://doi.org/10.1162/ARTL.2009.15.2.15202.
13. Stanley K. O. Evolving Neural Networks through Augmenting Topologies / K. O. Stanley, R. Miikkulainen // Evolutionary Computation. — 2002. — https://doi.org/10.1162/106365602320169811.
14. Veenstra F. Evolution and Morphogenesis of Simulated Modular Robots: A Comparison Between a Direct and Generative Encoding / F. Veenstra, A. Faina, S. Risi, K. Stoy ; Squillero G., Sim K. — Cham : Springer International Publishing, 2017. — ISBN 978-3-319-55849-3.
15. Ying C. Nas-bench-101: Towards reproducible neural architecture search / C. Ying, A. Klein, E. Christiansen, et al. — 2019.

References

Bhatia, J., Jackson, H., Tian, Y., Xu, J., & Matusik, W. (2021). Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems*, *34*, 2201–2214.

Deitke, M., VanderBilt, E., Herrasti, A., Weihs, L., Ehsani, K., Salvador, J., Han, W., Kolve, E., Kembhavi, A., & Mottaghi, R. (2022). ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. *Advances in Neural Information Processing Systems*, *35*, 5982–5994.

Dong, X., Liu, L., Musial, K., & Gabrys, B. (2021). Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44* (7), 3634–3646.

Dong, X., & Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv Preprint arXiv:2001.00326*.

Geng, H., Wang, F., Wei, S., Li, Y., Wang, B., An, B., Cheng, C. T., Lou, H., Li, P., Wang, Y.-J., & others. (2025). RoboVerse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning. *arXiv Preprint arXiv:2504.18904*.

Glybovets, M., & Medvid, S. (2025). MorphoNAS: Embryogenic Neural Architecture Search Through Morphogen-Guided Development. *arXiv Preprint arXiv:2507.13785*. https://arxiv.org/abs/2507.13785

Khalifa, A., Gallotta, R., Barthet, M., Liapis, A., Togelius, J., & Yannakakis, G. N. (2025). The Procedural Content Generation Benchmark: An Open-source Testbed for Generative Challenges in Games. *Proceedings of the 20th International Conference on the Foundations of Digital Games*, 1–12.

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv Preprint arXiv:1806.09055*.

McKay, M., Beckman, R., & Conover, W. (1979). Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, *21* (2), 239–245.

Migliore, M., Cannia, C., Lytton, W. W., Markram, H., & Hines, M. L. (2006). Parallel network simulations with NEURON. *Journal of Computational Neuroscience*, *21* (2), 119–129.

Neyman, J. (1934). On the Two Different Aspects of the Representative Method: The Method of Stratified Sampling and the Method of Purposive Selection. *Journal of the Royal Statistical Society*, *97* (4), 558–625.

Stanley, K. O., D'Ambrosio, D. B., & Gauci, J. (2009). A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life*. https://doi.org/10.1162/ARTL.2009.15.2.15202.

Stanley, K. O., & Miikkulainen, R. (2002). Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*. https://doi.org/10.1162/106365602320169811.

Veenstra, F., Faina, A., Risi, S., & Stoy, K. (2017). Evolution and Morphogenesis of Simulated Modular Robots: A Comparison Between a Direct and Generative Encoding. In G. Squillero & K. Sim (Eds.), *Applications of Evolutionary Computation* (pp. 870–885). Springer International Publishing. https://doi.org/10.1007/978-3-319-55849-3_56.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., & Hutter, F. (2019). *Nas-bench-101: Towards reproducible neural architecture search*, 7105–7114. https://proceedings.mlr.press/v97/ying19a.html.

Медвідь С. О.

MORPHONAS-BENCH: БЕНЧМАРК ДЛЯ МОРФОГЕНЕТИЧНОЇ ГЕНЕРАЦІЇ НЕЙРОННИХ МЕРЕЖ

У роботі представлено MorphoNAS-Bench — бенчмарк і набір інструментів для пошуку нейронних архітектур (Neural Architecture Search, NAS), який оснований на генеративному, розвитково-натхненому просторі пошуку. На відміну від сучасних NAS-бенчмарків, які використовують статичні кодування графів, що представляють нейронні мережі, MorphoNAS-Bench

характеризується компактними геномами. Ці геноми контролюють процес морфогенетичного розвитку, що дозволяє створювати різні просторові рекурентні архітектури, які виникають внаслідок різних типів детермінованого зростання, які при цьому визначаються локальними правилами розвитку.

Початковий набір даних бенчмарку містить 1000 пар «геном-архітектура», які були обрані з більш ніж 50 000 спроб генерації. Цей набір був створений шляхом використання як повністю стратифікованого відбору параметрів, так і за біологічно-натхненним методом Genome.random(). Застосування випадкового підходу забезпечує адекватне охоплення площі пошуку та реалістичність результатів. Кожне знайдене рішення містить детальну анотацію структурних показників. Ми проводимо аналіз таких структурних характеристик, як розмір, модульність, групування та ефективність. Ми показуємо, що обидві стратегії генерації здатні утворювати як структуровані, так і нетривіальні мережі.

Наданий інструментарій на Python дозволяє вивчати процеси розвитку геномів нейронних мереж разом із відповідним структурним аналізом. Таким чином, MorphoNAS-Bench виступає як повторювана і біологічно обґрунтована платформа для досліджень різноманітності, еволюціонування, та емерджентні структури для NAS.

Ключові слова: нейронні мережі, розвиткове кодування, морфогенетичний розвиток, пошук нейронних архітектур, бенчмарк-інструментарій, емерджентна модульність, непряме кодування.

Матеріал надійшов 21.07.2025

УДК 004.8, 004.67
DOI: 10.18523/2617-3808.2025.8.15-27

Глибовець А. М., Дубовик А. В., Афонін А. О.

ПРОГРАМНА СИСТЕМА КЛАСИФІКАЦІЇ ТЕКСТІВ НА ОСНОВІ МАШИННОГО НАВЧАННЯ ТА РЕКУРЕНТНОЇ НЕЙРОННОЇ МЕРЕЖІ

У цій роботі описано побудову та результати тестування програмної системи автоматичної класифікації текстів, яка полягає в розподілі текстів за певними категоріями, зокрема текстів українською мовою. Наш застосунок побудований на використанні трьох моделей — Naive Bayes, Support Vector Machine, LSTM — архітектури рекурентної нейронної мережі Recurrent Neural Network (RNN) та їх комбінації. Він дає змогу доволі швидко і точно класифікувати тексти, надавати користувачу можливість зручним способом натренувати систему на власних даних і досить просто налаштувати параметри для оптимальних результатів.

Для ефективного опрацювання вхідних даних і реалізації алгоритму класифікації ми вибрали мову програмування Python. Основними бібліотеками реалізації функціоналу застосунку стали TensorFlow, scikit-learn (для надання простого та зрозумілого інтерфейсу), Natural Language Toolkit (nltk), NumPy, Pandas. Matplotlib і seaborn застосовували для візуалізації даних і побудови графіків. Розроблений графічний застосунок здатен розпізнавати тексти (англійською або українською мовою) чотирьох категорій (World, Sports, Science / Technology, Business) з точністю близько 92 %.

Для навчання моделей ми застосували AG News Classification Dataset із kaggle.com.

Тестування застосунку підтвердило припущення, що спеціалізовані моделі, крім того, що є значно ефективнішими в плані використання ресурсів, також можуть демонструвати кращий результат у класифікації текстів, ніж LLM. Система також може бути швидко адаптована й до задачі фільтрації спаму. За декілька секунд можна отримати SVM модель, яка зможе розпізнавати типові спам-повідомлення з точністю близько 99 %. Так само були протестовані можливості системи при розпізнаванні емоційної забарвленості тексту. Вдалося досягти точності 87,75 %.

Ключові слова: автоматична класифікація текстів, Naive Bayes, Support Vector Machine, LSTM, Recurrent Neural Network, машинне навчання, Python, TensorFlow, AG News Classification Dataset.

Вступ

Нині, з розвитком інформаційних технологій і зростанням обсягу доступної інформації, стає надзвичайно важливим ефективне управління аналізом даних. Одним із ключових завдань у цьому контексті є автоматична класифікація текстів, яка полягає в розподілі текстів за певними категоріями. Основні виклики, пов'язані з автоматичною класифікацією текстів, — це різноманітність форматів, структур текстів і семантична складність мови. Для вирішення цих проблем потрібні ефективні алгоритми та методи оброблення природної мови.

Програмні системи (ПС), що використовують технології машинного навчання (МН), такі як нейронні мережі та методи оброблення природної мови (NLP), демонструють високу ефективність у розв'язанні цієї задачі завдяки своїй здатності адаптуватися до нових даних і швидкості оброблення [1; 2; 4; 6].

У контексті оброблення текстів класифікація полягає у призначенні кожному текстовому документу або фрагменту відповідної категорії або класу, що допомагає в організації, розумінні та аналізі великих обсягів інформації [4]. Технології штучного інтелекту, зокрема методи МН, дали можливість значно покращити ефективність і точність класифікації текстів. Автоматична класифікація тексту є одним із фундаментальних завдань оброблення природної мови. Системи автоматичної класифікації текстів (САКТ) можна умовно поділити на такі три групи: системи на основі правил, системи на основі машинного навчання (Naive Bayes, Support Vector Machine (SVM), Recurrent Neural Network (RNN), Bidirectional Encoder Representations from Transformers (BERT), Large Language Models (LLM)), гібридні системи. Коротку їхню характеристику наведено у роботах [13; 15; 18].

Гібридні системи класифікації текстів можуть поєднувати переваги різних підходів і методів для створення більш точних та ефективних моделей [3]. Вони становлять потужний інструмент, який може бути налаштований та оптимізований для конкретних потреб. Такі моделі можуть досягати високої точності та ефективності у різних сценаріях та сферах застосування. Проте один із основних недоліків гібридних систем полягає в їхній складності. Поєднання різних методів і підходів може призвести до значного збільшення складності самої системи, зокрема щодо розробки, розуміння, підтримки та масштабування.

Тому ми вирішили розробити систему, яка зможе доволі швидко і точно класифікувати тексти та надавати користувачу можливість зручним способом натренувати систему на власних даних і налаштувати параметри для оптимальних результатів. У цій статті опишемо цю розробку.

Для ефективного опрацювання вхідних даних і реалізації алгоритму класифікації ми вибрали мову програмування Python, оскільки для цієї мови існує сформована екосистема бібліотек МН, оброблення даних, матричної математики тощо. Великий вибір бібліотек, простота і зручність використання, а також усі інші згадані фактори роблять Python стандартним вибором для проєктів, що стосуються МН і нейронних мереж.

Основними бібліотеками реалізації функціоналу нашої програмної системи класифікації текстів (ПСАКТ) за певними категоріями стали: TensorFlow — безкоштовна бібліотека для МН із відкритим кодом [16] (відома своєю масштабованістю та високою продуктивністю); scikit-learn — надає простий і зрозумілий інтерфейс для реалізації класичних алгоритмів МН [12]; Natural Language Toolkit (nltk) — для реалізації задач опрацювання природної мови [8]; NumPy [9] — забезпечує зручність доступу до великих і багатовимірних масивів, а також функцій високого рівня для роботи з такими масивами; Pandas [10] — для доступу до структури даних DataFrame, яка є швидкою, гнучкою та інтуїтивно зрозумілою; Matplotlib [7] і seaborn [14] — для візуалізації даних та побудови графіків.

1. Аналіз вхідних даних

Для навчання моделей ми застосували AG News Classification Dataset із kaggle.com [5]. Цей набір даних містить 120 тисяч коротких текстів, зібраних із різних джерел новин. Усі тексти рівномірно (по 30 тисяч) розподілені між такими чотирма категоріями:

- World (Світ) — тексти про події та новини з різних країн світу, міжнародні відносини, а також глобальні проблеми, як-от зміна клімату, міграція та інші;
- Sports (Спорт) — тексти про спортивні події, змагання, турніри, команди та виступи спортсменів у різних дисциплінах, таких як футбол, баскетбол, теніс тощо;
- Business (Бізнес) — тексти про компанії, фінансові новини, аналіз ринків, бізнес-стратегії, економічні тенденції та інші аспекти бізнесу, зокрема корпоративні злиття і поглиблений огляд економічних подій різного роду;
- Science / Technology (Наука / Технології) — тексти про нові відкриття в науці, технологічні досягнення, інновації у технологічній сфері, зокрема роботи над штучним інтелектом, космічними відкриттями, медичними технологіями тощо.

Щоб робота з даними була більш ефективною, ми вирішили розробити модуль, який виконував би попереднє оброблення текстових даних шляхом побудови функції, яка отримує на вхід текст і виконує такі дії: токенізація тексту, видалення стоп-слів, лематизація або стемінг токенів (допомагає значно зменшити розмір словника та покращити точність аналізу шляхом зведення різних форм одного слова до єдиного кореня). Всі ці операції мають допомогти зменшити розмірність текстових даних, виокремити важливі ключові слова та поліпшити якість аналізу.

Поділ набору даних також є дуже вагомим етапом для ефективного навчання та коректної оцінки продуктивності моделі. У більшості випадків достатньо дотримуватися рівномірних пропорцій класів. Тому ми розділили всі наявні екземпляри на три категорії: навчальний набір даних, який використовують для безпосереднього тренування й підбору оптимальних ваг та коефіцієнтів моделі; валідаційний набір даних, який використовують для перевірки моделі після кожного циклу тренування і який може допомогти підібрати оптимальні гіперпараметри для моделі; тестувальний набір даних, який необхідний для неупередженої оцінки кінцевих результатів моделі.

Тому використаний набір даних розбивали на категорії так: 96 тис. текстів (80 % загальної кількості) використано для безпосереднього тренування моделі; для валідації — 10 % загальної кількості текстів, тобто 12 тис.; решту текстів виділено для тестування фінальних результатів.

Щоби перевірити, наскільки ефективно моделі можуть навчатись, маючи досить обмежений обсяг даних для тренування, ми розбивали набір даних також на три категорії: 0,1 % (120 екземплярів) — для тренування; 0,1 % (120 екземплярів) — для валідації; 99,8 % (117 600 екземплярів) — для тестування.

Також ми обмежилися використанням комбінації трьох моделей: Naive Bayes, SVM і RNN.

Naive Bayes є швидким алгоритмом, який потребує найменше ресурсів серед усіх трьох згаданих. Якщо припущення про незалежність між ознаками справджується, то може демонструвати доволі ефективні результати, що робить його гарним вибором для таких задач, як, наприклад, фільтрація спаму. Бібліотека scikit-learn має клас MultinominalNB, який реалізує наївний баєсів класифікатор для багатьох номіальних моделей.

SVM також порівняно економний у плані використання ресурсів комп'ютера, але не настільки, як наївний баєсів класифікатор. Використання його може демонструвати доволі ефективні результати навіть після тренування на досить невеликому обсягу даних. Бібліотека scikit-learn також надає реалізацію опорно-векторної машини у формі класу LinearSVC.

RNN серед трьох вибраних алгоритмів найбільш затратний у плані ресурсів, проте в загальному випадку після тренування на достатній кількості даних демонструє найкращі результати. Підходить для завдань, де є важливим контекст, як-от аналіз настроїв тощо. Для того щоб уникнути проблеми затухання градієнта, буде використано LSTM.

BERT і LLM ми не використовували, оскільки вони потребують більш потужних обчислювальних ресурсів та великої кількості часу для тренування. Відкинули й підхід на основі правил, адже ми не зможемо адаптувати старі правила для нових категорій у випадку, якщо користувач натренує модель на власних текстах. Загалом згаданих трьох методів має бути достатньо, щоб переваги одного методу могли компенсувати недоліки іншого, не використовуючи при цьому надмірну кількість ресурсів комп'ютера.

Згідно з нашою задачею, ми передбачили можливість використовувати будь-яку комбінацію моделей для тренування на користувацьких текстах. Наприклад, для швидкого результату можна навчити тільки Naive Bayes, а для більш точних результатів натренувати всі три моделі. Так само й для класифікації має бути можливість використовувати будь-яку комбінацію натренованих моделей. Для цього отримували відсоткові передбачення категорії з кожного класифікатора, а фінальний результат за замовчуванням є середнім арифметичним усіх значень. Але у користувача повинна бути можливість зменшити вплив на результат класифікації тих класифікаторів, які, на його думку, показують гірший результат, і навпаки — збільшити для тих, які показують кращий результат. Тому остаточний результат класифікації тексту визначався такою формулою: $x = a \cdot w_c + b \cdot w_b + c \cdot w_c$ де: x — результат системи класифікації, a — результат класифікації з використанням Naive Bayes, w_c — «вага» результату класифікації Naive Bayes, ціле число, більше ніж 0, що визначається користувачем, b — результат класифікації з використанням SVM, w_b — «вага» результату класифікації SVM, c – результат класифікації з використанням RNN, w_c — «вага» результату класифікації RNN. Використання трьох коефіцієнтів дало можливість коригувати систему класифікації для більш оптимальних результатів.

2. Реалізація ПС класифікації

ПС реалізовано у вигляді застосунку.

2.1. Структура застосунку

Представлена на рис. 1 UML діаграма класів схематично демонструє основні класи, що реалізовані в застосунку, їхню взаємодію та функції, які вони зможуть виконувати.

Клас TextPreprocessor дає можливість вибрати мову та опції оброблення (стемінг і лематизація), а також надає метод preprocess, який прийматиме на вхід необроблений текст, а повертатиме вже оброблений.

TextClassifierNB — класифікатор, що використовує Naive Bayes. Має доступ до обробника тексту й містить список назв категорій, які здатен класифікувати. Метод train приймає на вхід або шлях до директорії, з якої будуть діставатись і оброблюватись текстові файли для тренування або ж уже оброблений набір даних у формі DataFrame. Після виконання тренування повертає прогнозоване

значення точності. Метод save виконує збереження моделі до вказаної директорії, а load — завантаження. Метод predict приймає на вхід необроблений текст і повертає результат класифікації.

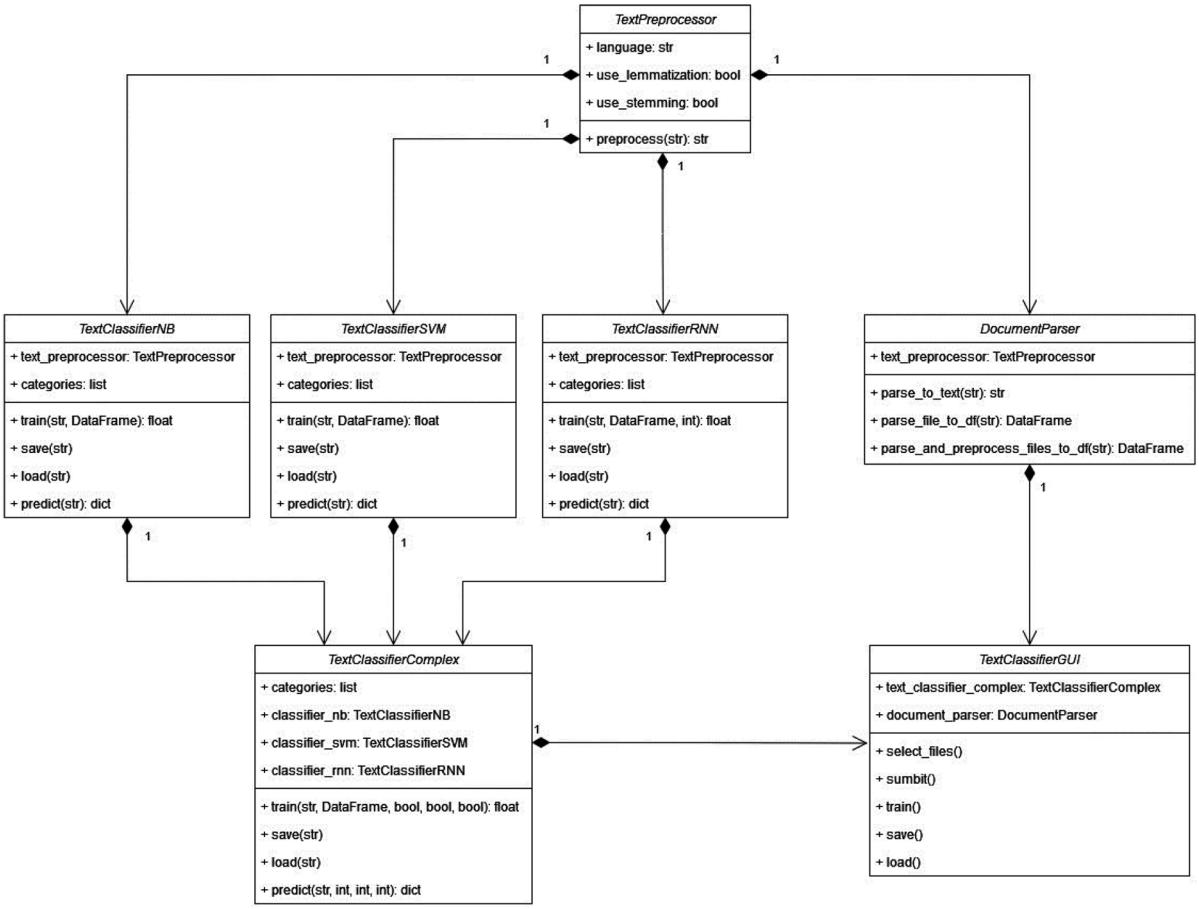


Рис. 1. Діаграма класів застосунку

TextClassifierSVM — клас, що працює схожим чином із попереднім класом, але використовує SVM. TextClassifierRNN — класифікатор, що використовує RNN модель. На відміну від інших класифікаторів, тут при тренуванні можна зазначити кількість епох.

DocumentParser — клас, що спрощує роботу з документами. Містить набір методів, які витягуються текст із txt, docx чи pdf файлів у потрібному форматі.

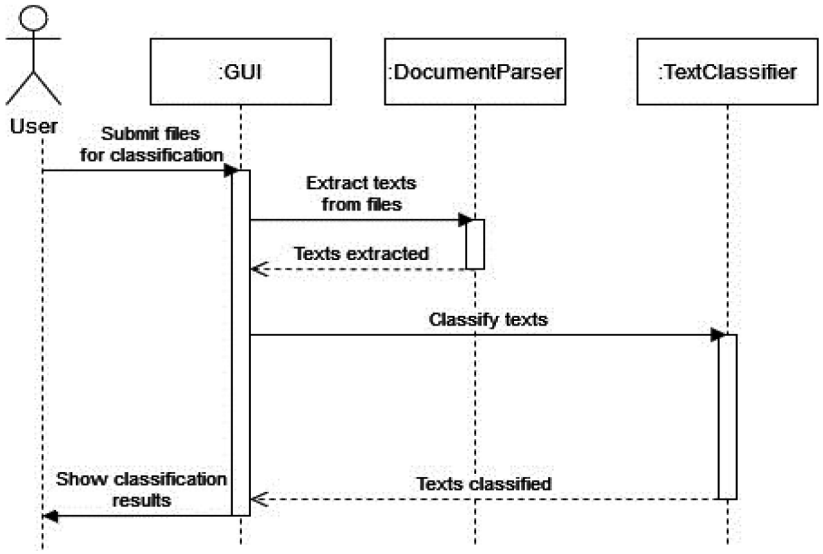


Рис. 2. Діаграма послідовності для процесу класифікації

TextClassifierComplex — має доступ до всіх трьох класифікаторів. При тренуванні є можливість обрати, які з моделей потрібно використовувати. Метод predict використовує коефіцієнти для кожного класифікатора при обчисленні результатів.

TextClassifierGUI — клас, який управляє графічним інтерфейсом. Працює з DocumentParser для отримання вмісту документів та з TextClassifierComplex для його класифікації.

Процес взаємодії графічного інтерфейсу з іншими сутностями описує UML діаграма послідовності (рис. 2), яка відображає простий випадок використання системи для класифікації файлів користувача.

Користувач обирає файли через GUI й запускає процес класифікації. DocumentParser витягує з файлів текстову інформацію, після чого вона передається в класифікатор тексту для визначення категорії. Результати класифікації повертаються й демонструються користувачеві у форматі, що відповідає обраним в графічному інтерфейсі опціям.

Рисунок 3 демонструє процес тренування моделі на файлах користувача та її збереження.

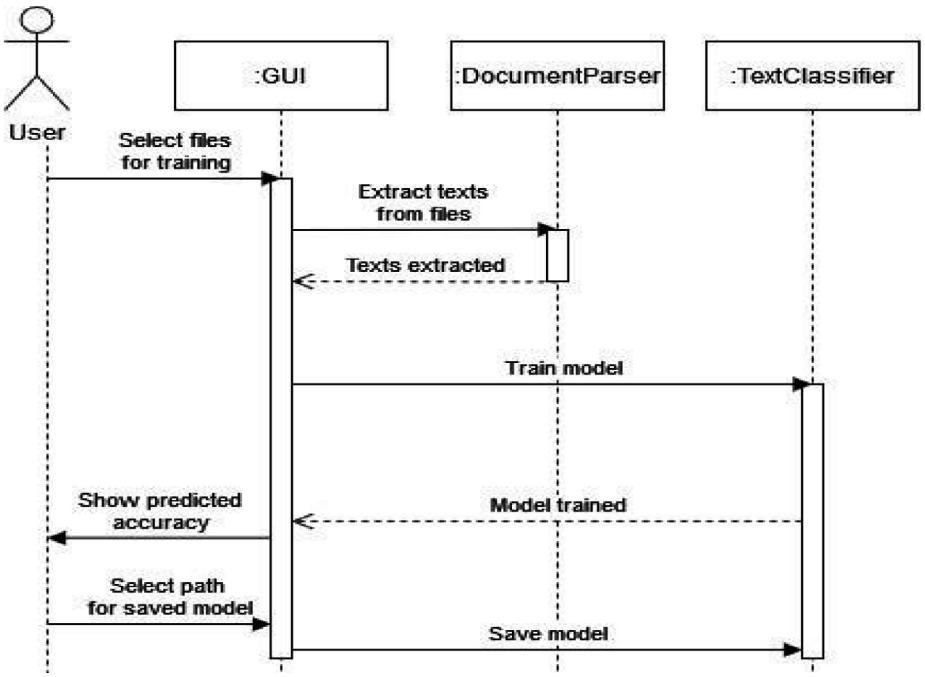


Рис. 3. Діаграма послідовності для процесу тренування моделі

Користувач вибирає файли для тренування (файли мають бути відсортовані за піддиректоріями відповідно до назв класів чи категорій). Текст із файлів витягується та передається до класифікатора для тренування. Після тренування користувач зможе побачити значення прогнозованої точності класифікації. Є можливість зберегти натреновану модель в обрану директорію для того, щоб була можливість завантажити її пізніше.

2.2. Оброблення тексту

Клас TextPreprocessor розроблено з використанням базових функцій бібліотеки NLTK. При ініціалізації є можливість обрати мову (доступна англійська та експериментальна реалізація для української), а також увімкнути опції стемінгу чи лематизації (рис. 4).

```
class TextPreprocessor:
    Andrii Dubovyk *
    def __init__(self, language='english', use_lemmatization=False, use_stemming=True):
        self.language = language
        self.use_stemming = use_stemming
        self.use_lemmatization = use_lemmatization
        if language == 'ukrainian':
            self.__setup_ukrainian_language()
        else:
            self.__setup_default_language()
```

Рис. 4. Ініціалізація класу TextPreprocssor

Для англійської мови застосовують PorterStemmer, WorldNetLemmatizer та список стоп-слів, який надає бібліотека NLTK. Для оброблення ж текстів української мови (рис. 5) використовують відповідний список стоп-слів із текстового файлу. Як стемер використано бібліотеку UkStemmer [17], а для лематизації — pymorhy2 [11].

```
def __setup_ukrainian_language(self):
    stopwords_ua = pandas.read_csv( filepath_or_buffer: "resources/stopwords_ua.txt", header=None, names=['stopwords'])
    self.stop_words = list(stopwords_ua.stopwords)
    if self.use_stemming:
        self.stemmer = uk_stemmer.UkStemmer()
        self.stem = self.stemmer.stem_word
    if self.use_lemmatization:
        self.lemmatizer = pymorphy2.MorphAnalyzer(lang='uk')
        self.lemmatize = lambda word: self.lemmatizer.parse(word)[0].normal_form
```

Рис. 5. Фрагмент коду, що демонструє налаштування для української мови

Для полегшення роботи з файлами створено модуль DocumentParser, який може одразу діставати текстову інформацію з файлів формату txt, pdf та docx, які відсортовано за директоріями відповідно до її класів (рис. 6). Текст оброблюється й повертається в зручному форматі (яким у цьому випадку є DataFrame з бібліотеки pandas).

```
@staticmethod
def parse_files_to_df(directory):
    classes = []
    for root, dirs, files in os.walk(directory):
        for d in dirs:
            classes.append(d)
    data = []
    for category in classes:
        category_dir = os.path.join(directory, category)
        for filename in os.listdir(category_dir):
            file_path = os.path.join(category_dir, filename)
            text = DocumentParser.parse_to_text(file_path)
            data.append({'label': category, 'text': text})
    return pd.DataFrame(data)

1 usage   Andrii Dubovyk
@staticmethod
def parse_and_preprocess_files_to_df(directory):
    df = DocumentParser.parse_files_to_df(directory)
    text_preprocessor = TextPreprocessor(language='english', use_stemming=False, use_lemmatization=True)
    df['text'] = df['text'].apply(lambda txt: text_preprocessor.preprocess(txt))
    return df
```

Рис. 6. Фрагмент коду, що демонструє процес завантаження та обробки текстових файлів

2.3. Реалізація алгоритмів класифікації

Для реалізації класифікатора Naive Bayes використано клас MutlinominalNB з бібліотеки scikit-learn (рис. 7).

```
# Define model
self.model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('nb', MultinomialNB()),
])

# Train model
self.model.fit(x_train['text'], x_train['label'])
```

Рис. 7. Визначення моделі Naive Bayes та її тренування

Схожим чином для SVM використовується функція LinearSVC з бібліотеки scikit-learn. Але звичайна реалізація LinearSVC всього лише може видавати результат у вигляді однієї найбільш вірогід-

ної категорії. Щоб отримувати відсоткове значення для кожної з категорій, ми робимо обгортку (рис. 8) за допомогою CallibratedClassifierCV.

```
# Define model
self.model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('svc', CalibratedClassifierCV(LinearSVC())),
])

# Train model
self.model.fit(x_train['text'], x_train['label'])
```

Рис. 8. Визначення моделі SVM та її тренування

Методи класифікації Naive Bayes та SVM доволі просто реалізовані за допомогою відповідних функцій бібліотеки scikit-learn, однак реалізація RNN за допомогою TensorFlow надає нам більше можливостей для модифікації та налаштувань задля покращення ефективності моделі.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization (TextVec  (None, None)            0
 torization)

 embedding (Embedding)       (None, None, 64)          640000

 bidirectional (Bidirectiona  (None, None, 128)        66048
 l)

 global_max_pooling1d (Globa  (None, 128)              0
 lMaxPooling1D)

 dense (Dense)               (None, 64)                8256

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 714,820
Trainable params: 714,692
Non-trainable params: 128
```

Рис. 9. Архітектура RNN моделі з використанням LSTM

Можемо детально розглянути архітектуру розробленої RNN (LSTM) моделі (рис. 9). Всього вона складається з 8 шарів.

TextVectorization перетворює оброблений текст на послідовність індексів токенів. Використовує словник розміром 10 000, що є більш ніж достатньо з урахуванням того, що текст проходить оброблення, яке передбачає лематизацію. Шар убудування Embedding зберігає один вектор на слово. При виклику він перетворює послідовності індексів слів на послідовності векторів. Ці вектори можна тренувати. Після навчання на достатній кількості даних слова зі схожими лексичними значеннями часто мають подібні вектори. Має параметр mask_zero=True, що дає змогу ефективно працювати з текстами різних розмірів. Bidirectional є обгорткою для RNN, що обробляє послідовний вхід у двох напрямках. У цьому випадку використовує LSTM шар із 64 нейронами для збереження контексту.

Рівень об'єднання GlovalMaxPooling1D зменшує чутливість до особливостей, створюючи більш узагальнені дані для кращих результатів тренування. Повнозв'язний шар Dense використовує функцію активації relu та містить 64 нейрони.

BatchNormalization використовують для підвищення швидкості та стабільності навчання штучних нейронних мереж. Це досягається шляхом нормалізації вхідних даних шарів через повторне центрування та масштабування. Dropout відкидає випадковим чином деякі нейрони, може допомогти частково уникнути надмірного тренування, яке відбувається, коли модель надто добре або занадто довго тренувалась на заданому наборі даних і їй не вдається демонструвати хороші результати при застосуванні до нових даних. У цьому випадку використовується значення rate 0,2, що означає, що 20 % випадкових нейронів будуть ігноруватись. Також варто зауважити, що цей шар використовується лише при тренуванні, тому під час справжніх передбачень моделі нейрони не будуть ігноруватись. Повнозв'язний шар Dense містить 4 нейрони, що відповідає кількості категорій у нашому наборі даних. Використовує функцію активації softmax та повертає розподіл ймовірностей визначених категорій.

Ця модель компілюється з такими параметрами: sparse_categorical_crossentropy, Adam, accuracy. Перший застосовується для обчислення втрат між прогнозованими та справжніми мітками. Це доцільний вибір для задачі класифікації з кількома класами, де мітки не кодуються як one-hot вектори, але представлені цілими числами. Дає можливість отримати список найбільш імовірних категорій. Оптимізатор Adam (зі швидкістю навчання 1e-4) ефективно пристосовує швидкість навчання для кожного параметра, зокрема на основі оцінок першого та другого моментів градієнта. Метрика accuracy використовується для оцінки точності моделі під час навчання. Вона вимірює відсоток правильно класифікованих зразків з усієї кількості екземплярів.

Використання функції зворотного виклику EarlyStopping покликано зупиняти тренування, коли точність класифікації моделі для валідаційного набору даних перестає зростати. В цьому випадку ця функція використовується з такими параметрами: val_accuracy, 2, 0.001. Перший параметр позначає значення, за яким потрібно робити нагляд, тут точність моделі на валідаційному наборі даних. Другий — визначає кількість епох без покращення, після яких навчання буде припинено. Третій — мінімальна зміна значення, що контролюється, яка буде вважатися покращенням (тобто абсолютна зміна менше ніж min_delta вважатиметься відсутністю покращення). Коли ця функція спрацьовуватиме, відповідне повідомлення виводиться на консоль. Після спрацювання функції буде відновлено той стан моделі, у якому вона мала найкраще значення точності. Ця функція в багатьох випадках відбирає в нас потребу підбирати оптимальну (для уникнення недостатнього або надмірного тренування) кількість епох вручну, дозволяючи просто запустити тренування на великій кількості епох і дочекатись, доки точність перестане зростати.

У підсумку після тренування можемо побачити графіки (рис. 10) зміни точності і втрат (синім кольором показані результати для тренувальних даних, оранжевим — для валідаційних). Тренування відбувалось впродовж п'яти епох, доки точність не перестала зростати, після чого було відновлено стан моделі, в якому вона демонструвала найкращу точність на валідаційних даних (у цьому випадку це третя епоха).

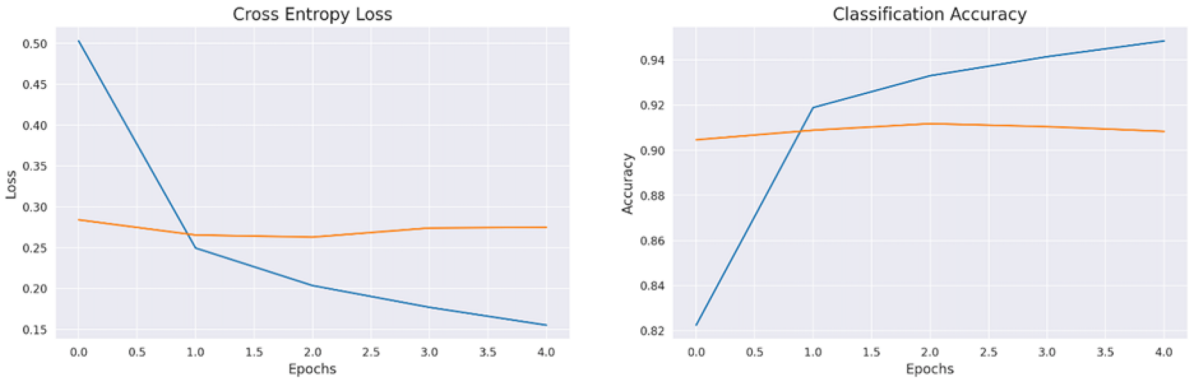


Рис. 10. Втрати та точність LSTM моделі впродовж тренування

Також були протестовані деякі інші часто використовувані оптимізатори, як-от Adagard, Adadelta і RMSprop, із метою визначити їхній вплив на ефективність натренованої моделі. З іншими оптимі-

заторами, використовуючи стандартну швидкість навчання зі значенням 0,001, після тренування максимально вдалось досягти таких значень: Adagard — 90,22 % точності та значення втрат 0,2933; Adadelta — 90,71 % точності та значення втрат 0,2764; RMSprop — 91,93 % точності та значення втрат 0,2524.

Тільки з оптимізатором RMSprop вдалося досягнути дещо кращого результату, ніж з Adam, натомість оптимізатори Adagard і Adadelta не змогли показати значення точності вище ніж 91 %, тому було вирішено далі використовувати саме цей оптимізатор. Також тестувалось тренування моделі з іншою (0,01 та 0,0001) швидкістю навчання, відмінною від стандартної 0,001, але якихось суттєвих змін в результатах помічено не було, тому швидкість навчання зі стандартним значенням 0.001 була залишена як оптимальна.

2.4. Система класифікації з використанням комбінації алгоритмів

Кожну реалізацію алгоритму класифікації з попереднього підрозділу було додано до окремих відповідних класів: TextClassifierNB, TextClassifierSVM, TextClassifierRNN. Ці класи мають такі методи, необхідні для роботи:

– train — приймає на вхід або директорію з текстами або DataFrame з обробленими текстами, для RNN моделі може також приймати кількість епох для навчання, тренує модель і повертає значення точності класифікації у відсотках;
– save — приймає на вхід директорію, до якої зберігає навчену модель (NB та SVM моделі зберігаються у форматі pkl, RNN — у форматі keras, окремо в json файл зберігаються назви категорій);
– load — приймає на вхід директорію, з якої завантажує попередньо збережену модель;
– predict — приймає на вхід текст користувача, для якого визначає категорії, результат повертає в форматі відсортованого Python словника, де ключі це назви категорій, а значення — ймовірність належності до категорії від нуля до одиниці.

Аналогічно розроблено клас TextClassifierComplex, що використовує всі вищезгадані класи та містить такі методи: train — схожий на методи окремих класів, за допомогою булевих параметрів use_nb, use_svm, use_rnn є можливість визначати, які моделі будуть тренуватись; save — зберігає всі натреновані моделі в обрану директорію; load — завантажує всі моделі, які збережені в обраній директорії; predict — аналогічний до методу окремих класів, має параметри nb_weight, svm_weight, rnn_weight, значення яких коригує вплив передбачення відповідної моделі на загальний результат.

2.5. Графічний інтерфейс

Було розроблено графічний інтерфейс, який дає змогу користувачу не тільки ефективно використовувати систему з натренованими моделями, які навчені на AG News Classification Dataset, а й натренувати систему на власних файлах, зберегти результати навчання та використовувати її надалі. Одразу при запуску застосунку завантажуються вже заздалегідь навчені моделі.

Вгорі інтерфейсного вікна містяться елементи, які стосуються тренування, збереження та завантаження моделі користувача. Прапорці Train NB, Train SVM, Train RNN відповідають за те, які моделі будуть тренуватись. Кнопка Train відкриває діалогове вікно, що дозволяє обрати директорію з текстовими документами (txt, pdf або docx) для тренування, потім запускається сам процес тренування обраних моделей. Після тренування в діалоговому вікні буде показано приблизне значення точності класифікації (середнє арифметичне для всіх використаних моделей на тренувальному наборі даних). Варто зазначити, що текстові документи в обраній директорії мають бути розміщені належним чином: кожен документ має бути в піддиректорії, назва якої відповідає визначеній категорії тексту. Кнопка Save дозволяє зберегти модель в одну з директорій комп'ютера, а кнопка Load — завантажити попередньо збережену модель. Текст Categories: [перелік категорій] демонструє, які категорії здатна розпізнавати модель, яка зараз використовується.

Далі розміщена група елементів графічного інтерфейсу, що стосуються завантаження документів для класифікації. Є можливість вибрати мову наданих текстів, що необхідно для правильної класифікації. За замовчуванням використовується англійська мова, але є можливість вибрати українську, в такому разі всі обрані тексти користувача будуть автоматично перекладатись через Google Translate на англійську мову перед тим, як модель буде робити передбачення їхньої категорії. Це дає змогу не тренувати модель заново для іншої мови, а використовувати ту саму модель для розпізнавання

текстів мов, відмінних від англійської. Документи для класифікації обирають у діалоговому вікні, що з'являється після натискання Select Files. Повний шлях до обраних файлів демонструється у відповідному текстовому полі.

Остання група елементів інтерфейсу відповідає за саму класифікацію і параметри системи, що при цьому використовуються. Поля NB Weight, SVM Weight, RNN Weight дозволяють коригувати вплив кожної окремої моделі на загальний результат. Прапорець Detailed Results впливає на формат відображення результатів; якщо він відмічений, то для кожного документа буде вказано відсоткове значення належності до категорій, якщо ні — лише одна найбільш вірогідна категорія (без значення відсотків). Кнопка Submit запускає процес визначення категорії, а Copy Results дозволяє швидко та зручно скопіювати результати в буфер обміну.

2.6. Оцінка результатів

На тестах класифікації новин найкращі результати для використаного набору даних демонструвала саме RNN модель, тому в цьому випадку детальніше розглянемо саме її результати (рис. 11). На тестових даних розроблена модель у остаточному її варіанті показала такі результати: точність — 91,92500114440918 %, втрати — 0,26782718300819397.

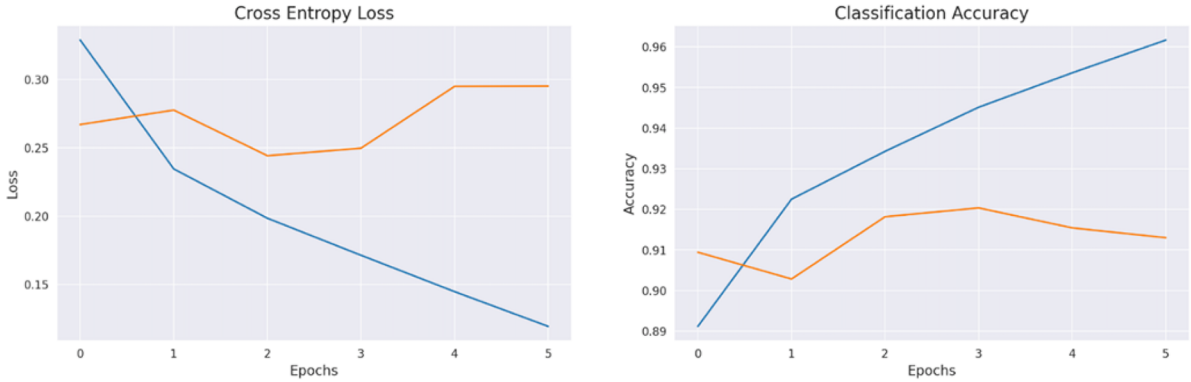


Рис. 11. Графіки втрат і точності RNN моделі в її остаточному вигляді

Також у звіті про класифікацію (рис. 12) можна побачити деякі показники для кожної з чотирьох категорій окремо: Precision — кількість справжніх позитивних результатів, поділена на кількість прогнозованих позитивних результатів; Recall — кількість справжніх позитивних результатів, поділена на загальну кількість фактичних позитивних результатів; F1-score — середнє гармонічне між precision та recall.

	precision	recall	f1-score	support
World	0.89	0.89	0.89	3000
Sports	0.90	0.89	0.89	3000
Business	0.96	0.98	0.97	3000
Science/Technology	0.93	0.91	0.92	3000
accuracy			0.92	12000
macro avg	0.92	0.92	0.92	12000
weighted avg	0.92	0.92	0.92	12000

Рис. 12. Звіт про класифікацію

У застосунку передбачено побудову і виведення матриці невідповідностей, яка демонструє, скільки екземплярів однієї категорії було визначено іншою, невідповідною категорією під час передбачень категорій. Тести показали, що найчастіше модель плутає між собою категорії World і Sports. Також порівняно часто плутає Science / Technology з World і Sports із Science / Techonlogy. Категорія ж Business визначається найбільш впевнено і плутається з усіма іншими доволі рідко.

Для порівняння ефективності навченої моделі категорії тексту для 200 екземплярів з цього ж набору даних (AG News Classification Dataset) було визначено через ChatGPT версії 3.5. В результаті правильно розпізнано було лише 176 категорій, що показує точність зі значенням близько 88 %. Реа-

лізована ж у цій роботі RNN (LSTM) модель демонструє майже 92 %. Отже, можна зробити припущення, що спеціалізовані моделі, крім того, що є значно ефективнішими в плані використання ресурсів, також можуть демонструвати кращий результат у класифікації текстів, ніж LLM.

Із другим поділом набору даних, який містить обмежений обсяг даних для тренування, моделі показали такий результат: Naive Bayes сягнула значення точності в 68,34 %, SVM досягла 70,25 % точності, LSTM модель продемонструвала точність передбачення всього лише 24,98 %. Це демонструє перевагу моделей SVM та Naive Bayes в ситуації з недостатньою кількістю даних для тренування.

Також було перевірено, наскільки ефективно система може навчатись на інших текстах. З kaggle.com із цією метою було завантажено набір даних із назвою (10)Dataset Text Document Classification. Цей набір даних містить по сотні текстів кожної з десяти категорій: «бізнес», «розваги», «їжа», «графіка», «історія», «медицина», «політика», «космос», «спорт» і «технології». На завантаження та оброблення текстів із цих файлів знадобилося 4,78 секунди. Моделі впорались із цією задачею так: Naive Bayes — демонструє 97 % точності, витративши 0,2 секунди на навчання, SVM — демонструє 98 % точності, витративши 0,59 секунди на навчання, RNN — демонструє 91 % точності, витративши 974 секунди на навчання, комбінація всіх моделей (з однаковими вагами) має точність 94,5 %.

Щоб перевірити, як система виконує таку задачу, як фільтрація спаму, було використано Spam Mails Dataset із kaggle.com, який містить 5171 зразок тексту з визначеними спам-повідомленнями. Моделі справляються з цією задачею з такою точністю та витраченим часом на навчання (без урахування 9,53 секунди на завантаження файлів і оброблення тексту): Naive Bayes — 92,07 % та 0,41 секунди, SVM — 99,23 % і 0,54 секунди, RNN — 98,07 % і 1153 секунди, комбінація всіх моделей (з однаковими вагами) має точність 99,13 %.

Отже, система може бути швидко адаптована й до такої задачі. Майже за 10 секунд можна отримати SVM модель, яка зможе розпізнавати типові спам-повідомлення з точністю близько 99 %.

Так само були протестовані можливості системи при розпізнаванні емоційної забарвленості тексту. Для цього використано Emotions dataset for NLP з kaggle.com, який містить 16 тисяч зразків тексту з визначеними емоціями (сум, злість, страх, радість, здивування, любов). Текст із файлів завантажується та оброблюється за 2,8 секунди. Моделі демонструють таку точність і час навчання: Naive Bayes — 56,99 % і 0,081 секунди, SVM — 83,09 % і 0,83 секунди, RNN — 85,56 % і 13,35 секунди, комбінація всіх моделей (з однаковими вагами) має точність 59,31 %. Але якщо проекспериментувати з коефіцієнтами, то можна отримати кращі результати, ніж при використанні кожної моделі окремо. Наприклад, вдалося досягти точності 87,75 %, використавши такі значення: 1 для Naive Bayes, 5 для SVM та 13 для RNN.

У результаті навіть якщо один або два алгоритми не можуть показати хороший результат для певної задачі, то майже завжди як мінімум одна модель може ефективно класифікувати тексти з точністю понад 85 %. Отже, система може застосуватись майже для всіх загальних випадків класифікації.

Висновки

У цій роботі описано побудову та результати тестування ПС автоматичної класифікації текстів, яка полягає в розподілі текстів за певними категоріями, зокрема текстів українською мовою. Наш застосунок побудований на використанні трьох моделей — Naive Bayes, Support Vector Machine (SVM), LSTM архітектури рекурентної нейронної мережі Recurrent Neural Network (RNN) та їх комбінації. Він дає можливість доволі швидко і точно класифікувати тексти, зручним способом натренувати систему на власних даних і досить просто налаштувати параметри для оптимальних результатів.

Для ефективного опрацювання вхідних даних і реалізації алгоритму класифікації ми обрали мову програмування Python. Основними бібліотеками реалізації функціоналу нашого застосунку стали: TensorFlow (за гарну масштабованість і високу продуктивність); scikit-learn (для надання простого та зрозумілого інтерфейсу); Natural Language Toolkit (nltk, для реалізації задач опрацювання природної мови); NumPy (що забезпечує зручність доступу до великих та багатовимірних масивів); Pandas (для простого доступу до структури даних DataFrame). Matplotlib та seaborn застосовувалися для візуалізації даних та побудови графіків.

Для навчання моделей ми застосували AG News Classification Dataset із kaggle.com. Розроблений графічний застосунок здатен розпізнавати тексти (англійською або українською мовою) чотирьох категорій (World, Sports, Science / Technology, Business) із точністю близько 92 %.

Тестування застосунку підтвердило припущення, що спеціалізовані моделі, крім того, що є значно ефективнішими в плані використання ресурсів, також можуть демонструвати кращий результат у класифікації текстів, ніж LLM. Система також може бути швидко адаптована й до задачі фільтрації спаму. За декілька секунд можна отримати SVM модель, яка зможе розпізнавати типові спам-повідомлення з точністю близько 99 %. Так само були протестовані можливості системи при розпізнаванні емоційної забарвленості тексту. Вдалось досягти точності 87,75 %.

Продовження досліджень та експериментів у цій сфері можуть призвести до підвищення ефективності створених моделей у розпізнаванні використаних у цій роботі або будь-яких інших категорій текстів. Також подальший розвиток може бути спрямований на покращення зручності та функціональності розробленого графічного застосунку.

Ця робота була підтримана грантом Фонду Саймонса (SFI-PD-Ukraine-00014577; T.S.)

Список літератури

1. Глибовець А. М. Програмна система перевірки на плагіат українських текстів [Електронний ресурс] / А. М. Глибовець, М. О. Бікчентаєв // Наукові записки НаУКМА. Комп'ютерні науки. — 2022. — Том 5. — С. 16–25. — Режим доступу: https://doi.org/10.18523/2617-3808.2022.5.16-25.
2. A Generative Learning Model for Heterogeneous Text Classification Based on Collaborative Partial Classifications [Electronic resource] / Zie Eya Ekolle, Ryuji Kohno // Applied Sciences. — 2023. — Vol. 13 (14). — P. 8211. — Mode of access: https://www.mdpi.com/2076-3417/13/14/8211.
3. A Hybrid Text Classification Approach for Analysis of Student Essays [Electronic resource] / C. P. Rosé, A. Roque, D. Bhembe, K. Vanlehn // Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing. — Pp. 79–86. — Mode of access: https://aclanthology.org/W03-0210.pdf.
4. A Survey on Text Classification Algorithms: From Text to Predictions [Electronic resource] / A. Gasparetto, M. Marcuzzo, A. Zangari, A. Albarelli // Information. — 2022. — Vol. 13 (2). — P. 3. — Mode of access: https://www.mdpi.com/2078-2489/13/2/83.
5. Kaggle [Electronic resource]. — Mode of access: https://www.kaggle.com.
6. Machine Learning Glossary [Electronic resource]. — Mode of access: https://developers.google.com/machine-learning/glossary.
7. Matplotlib [Electronic resource]. — Mode of access: https://matplotlib.org.
8. Natural Language Toolkit [Electronic resource]. — Mode of access: https://www.nltk.org.
9. NumPy [Electronic resource]. — Mode of access: https://numpy.org.
10. Pandas [Electronic resource]. — Mode of access: https://pandas.pydata.org.
11. Pymorphy2 [Electronic resource]. — Mode of access: https://github.com/pymorphy2/pymorphy2.
12. Scikit-learn [Electronic resource]. — Mode of access: https://scikit-learn.org/stable.
13. Scikit-learn: Naive Bayes [Electronic resource]. — Mode of access: https://scikit-learn.org/stable/modules/naive_bayes.html.
14. Seaborn [Electronic resource]. — Mode of access: https://seaborn.pydata.org.
15. Support vector machine [Electronic resource]. — Mode of access: https://en.wikipedia.org/wiki/Support_vector_machine.
16. TensorFlow [Electronic resource]. — Mode of access: https://www.tensorflow.org.
17. UkStemmer [Electronic resource]. — Mode of access: https://github.com/Desklop/Uk_Stemmer
18. Understanding Text Classification in Python [Electronic resource]. — Mode of access: https://www.datacamp.com/tutorial/text-classification-pythonpython.

References

Datacamp. (n. d.). *Understanding text classification in Python*. https://www.datacamp.com/tutorial/text-classification-pythonpython.

Ekolle, Z. E., & Kohno, R. (2023). A generative learning model for heterogeneous text classification based on collaborative partial classifications. *Applied Sciences*, *13* (14), 8211. https://www.mdpi.com/2076-3417/13/14/8211.

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). A survey on text classification algorithms: From text to predictions. *Information*, *13* (2), 83. https://www.mdpi.com/2078-2489/13/2/83.

Google Developers. (n. d.). *Machine learning glossary*. https://developers.google.com/machine-learning/glossary.

Hlybovets, A. M., & Bikchentayev, M. (2022). Prohramna systema perevirky na plahiat ukrainskykh tekstiv. *NaUKMA Research Papers. Computer Science*, *5*, 16–25. https://doi.org/10.18523/2617-3808.2022.5.16-25 [in Ukrainian].

Kaggle. (n. d.). https://www.kaggle.com.

Matplotlib. (n. d.). https://matplotlib.org.

Natural Language Toolkit (nltk). (n. d.). https://www.nltk.org.

NumPy. (n. d.). https://numpy.org.

Pandas. (n. d.). https://pandas.pydata.org.

Pymorphy2. (n. d.). https://github.com/pymorphy2/pymorphy2.

Rosé, C. P., Roque, A., Bhembe, D., & VanLehn, K. (2003). A hybrid text classification approach for analysis of student essays. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing* (pp. 79–86). https://aclanthology.org/W03-0210.pdf.

Scikit-learn. (n. d.). https://scikit-learn.org/stable.

Scikit-learn. (n. d.). *Naive Bayes*. https://scikit-learn.org/stable/modules/naive_bayes.html.

Seaborn. (n. d.). https://seaborn.pydata.org.

Support vector machine. (n. d.). *Wikipedia*. https://en.wikipedia.org/wiki/Support_vector_machine.

TensorFlow. (n. d.). https://www.tensorflow.org.

UkStemmer. (n. d.). https://github.com/Desklop/Uk_Stemmer.

A. Hlybovets, A. Dubovyk

TEXT CLASSIFICATION SOFTWARE SYSTEM BASED ON MACHINE LEARNING AND RECURRENT NEURAL NETWORK

In the context of rapid advancements in information technology and the exponential growth of accessible data, efficient data analysis management has become increasingly critical. One of the key challenges in this domain is automatic text classification — the process of assigning texts to specific categories. This task is complicated by the diversity of formats, structural variability, and the inherent semantic complexity of natural language. Addressing these challenges requires robust algorithms and effective natural language processing (NLP) techniques.

This paper describes the development and testing of a software system for automatic text classification, which involves categorizing texts into predefined classes, including texts written in Ukrainian. Our application is based on the use of three models: Naive Bayes, Support Vector Machine, and the LSTM-based architecture of Recurrent Neural Networks (RNN), as well as their combinations. It allows for fast and accurate text classification and provides users with a convenient way to train the system on their own datasets and easily configure parameters for optimal results.

To efficiently process input data and implement the classification algorithm, we chose the Python programming language. The core libraries used for the application's functionality include TensorFlow, scikit-learn (for a simple and intuitive interface), Natural Language Toolkit (nltk), NumPy, and Pandas. Matplotlib and Seaborn were used for data visualization and plotting. The developed graphical application is capable of recognizing texts (in English or Ukrainian) in four categories ("World", "Sports", "Science/Technology", "Business") with an accuracy of approximately 92%. For model training, we used the "AG News Classification Dataset" from Kaggle.com.

Testing confirmed the hypothesis that specialized models, in addition to being significantly more resource-efficient, can also outperform large language models (LLMs) in text classification tasks. The system can also be quickly adapted for spam filtering tasks. Within seconds, it is possible to obtain an SVM model capable of identifying typical spam messages with about 99 % accuracy. We also tested the system's capabilities in detecting emotional tone in texts, achieving an accuracy of 87.75 %.

This work was supported by a grant from the Simons Foundation (SFI-PD-Ukraine-00014577; T.S.).

Keywords: automatic text classification, Naive Bayes, Support Vector Machine, LSTM, Recurrent Neural Network, machine learning, Python, TensorFlow, AG News Classification Dataset.

Матеріал надійшов 18.06.2025

УДК 004.934.5:534.782
DOI: 10.18523/2617-3808.2025.8.28-37

D. Ivashchenko, O. Marchenko

MODERN APPROACHES TO CONTROLLABLE EMOTIONAL SPEECH SYNTHESIS

The generation of emotionally expressive and controllable speech is one of the most dynamic and technically demanding areas in the intersection of artificial intelligence, natural language processing, and speech synthesis. Recent progress in emotional text-to-speech (TTS) systems has enabled increasingly natural and emotionally nuanced voice generation, shifting from early concatenative methods to advanced neural models. This review provides a structured overview of the state of the art in controllable emotional TTS, highlighting key architectural paradigms. A special focus is placed on emotional control mechanisms, including discrete emotional tagging with categorical or dimensional labels, reference-based control which conditions synthesis on prosodic or stylistic exemplars, and prompt-based techniques that leverage the capabilities of large language models for flexible and intuitive emotional specification.

Despite substantial improvements in synthesis quality and emotional expressiveness, several critical challenges remain unresolved. These include the disentanglement of emotional, speaker, and prosodic features, the lack of standardized evaluation metrics for emotional clarity and naturalness, and the significant computational demands associated with training high-fidelity models. Furthermore, the scarcity of diverse and emotion-labeled speech data, especially for low-resource and morphologically rich languages, continues to limit the generalizability of current approaches. This review not only summarizes existing methods and their trade-offs but also outlines promising research directions, aiming to support the development of more robust, efficient, and emotionally expressive speech generation systems.

Keywords: deep learning, text-to-speech synthesis, natural language processing, speech emotion control, diffusion models.

Introduction

One of the most difficult but important tasks in developing natural language processing and artificial intelligence is generating speech with predefined characteristics and style. Generating natural, emotionally expressive speech has long been a central goal in artificial intelligence and natural language processing. Achieving this requires synthesizing speech that sounds humanlike and conveys emotional nuances, bridging the gap between human communication and machine-generated speech. Such advancements hold immense promise for applications ranging from assistive technologies and virtual assistants to entertainment and education. However, achieving this level of sophistication remains a formidable challenge, particularly in low-resource language contexts where labeled emotional datasets are limited.

Traditional text-to-speech (TTS) systems relied mainly on deterministic algorithms that frequently failed to produce dynamic, natural-sounding speech. Recent advances in deep learning have transformed TTS by allowing models to capture complex prosodic and emotive variations. While several systems have aimed to improve naturalness, few have investigated the explicit regulation of emotional expression in synthesized speech [34]. This feature is crucial for developing more engaging and context-appropriate human–computer interactions.

There are a number of basic obstacles that affect the capacity to produce emotionally expressive discourse. These include gathering enough emotion-labeled training data, effectively simulating the intricate link between text content and emotional expression, and providing intuitive controls for the emotional aspects of synthesized speech [5, 34]. These issues have been largely addressed by recent developments using innovative designs and training techniques.

This review examines the current state of emotional speech synthesis, focusing on recent advances in controllable and expressive TTS systems. We analyze the emotion control approaches that represent an important innovation in this domain. By examining its technical foundations, methodological approaches,

and evaluation results, we aim to provide a comprehensive overview of the current state of the art in emotional speech synthesis and identify promising directions for future research.

Text-to-Speech Synthesis Approaches

Non-Neural Approaches to Speech Synthesis

Text-to-speech conversion has historically progressed through several distinct paradigms. Early systems relied on concatenative synthesis, piecing together pre-recorded speech fragments to form complete utterances. While providing natural-sounding elements, these systems struggled with smooth transitions and had limited flexibility. The subsequent development of statistical parametric speech synthesis (SPSS), particularly Hidden Markov Model (HMM)-based approaches, offered greater control but often at the cost of naturalness [24].

Neural TTS Pipeline

The advent of deep learning has revolutionized TTS technology. Modern neural TTS systems can be categorized into several architectural approaches, each with distinct characteristics that affect their overall performance and application.

Mu, Yang, and Dong (2021) summarize TTS system construction approaches and methods. They articulate a modern speech generation pipeline as an end-to-end system, comprising three parts: a text-analysis front end, an acoustic model, and a vocoder [24]. The process begins with the text front end transforming the input text into a standardized format. Next, the acoustic model processes this standardized input into intermediate acoustic features, incorporating long-term structures from the speech. Common representations include spectrograms, vocoder features, and linguistic features. Lastly, the vocoder generates the final output by adding fine-grained signal details and converting the acoustic features into a time-domain waveform. Each system component has its unique architecture, which defines its application (Figure 1). Another category of methods follows a fully end-to-end philosophy, in which a single model performs all the necessary processing steps. It directly generates speech audio output from the input data, skipping the creation of intermediate features, such as mel-spectrograms, is skipped entirely [34].

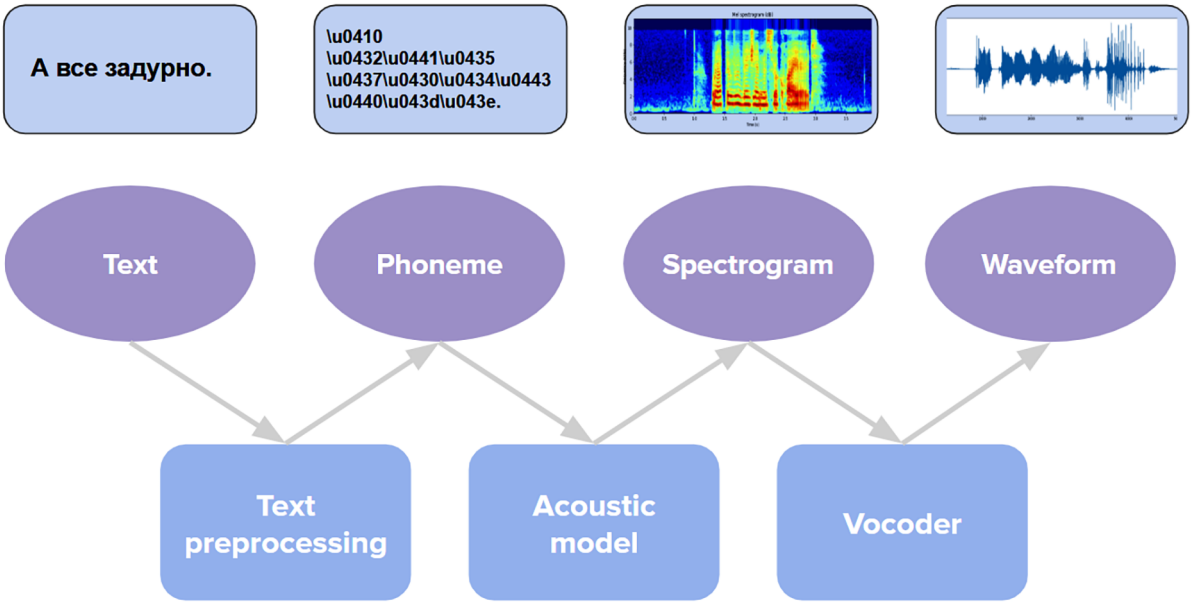


Figure 1. Mainstream TTS pipeline

There are a large volume of published studies describing a wide variety of approaches to neural speech generation. The systems can be conventionally divided into distinct segments based on the generation technology employed in a particular system. This review primarily considers architectures that are widely used for expressive speech synthesis.

Controllable TTS Systems Architectures

Autoregressive and Non-Autoregressive Generation Models

Autoregressive generation models have transformed speech synthesis by predicting speech representations sequentially, with each output based on all previously generated tokens, allowing for the capture of intricate temporal relationships inherent in human speech [24]. These models include a variety of acoustic architectures, such as transformer-based systems, convolutional neural networks (CNNs), recurrent neural networks (RNNs), flow-based models, and diffusion-based approaches, each with unique advantages for modeling speech characteristics and improving generation quality [3]. Autoregressive speech synthesis has made significant progress in producing genuine, expressive speech that closely resembles human voice patterns by learning directly from data rather than relying on substantial manual feature engineering.

Non-autoregressive speech synthesis models address the computational inefficiency of sequential generation by allowing parallel prediction of acoustic features, which reduces inference time from several seconds to real-time or faster production rates [24]. These architectures also encompass a variety of generative approaches, including feed-forward transformer networks, flow-based models, generative adversarial networks, variational autoencoders, and diffusion-based systems, all of which use parallel computation to eliminate sequential dependencies inherent in autoregressive methods [34]. While avoiding error propagation difficulties provides significant speedups and greater stability, non-autoregressive algorithms frequently require explicit duration prediction or external alignment mechanisms to handle the one-to-many mapping challenge between text and speech sequences [15].

Main Neural Architecture Types

Tacotron [33] presented the first acoustic model based on deep learning and remains used in various systems. A one-dimensional convolution- and bidirectional gated recurrent unit-based encoder, an attention-driven decoder, and a Griffin-Lim vocoder are the main parts of the proposed autoregressive architecture, which is a sequence-to-sequence (seq2seq) model. It accepts characters as input and generates spectrograms, which are subsequently transformed into waveforms by the vocoder.

Later, Ren et al. presented FastSpeech 2 (2020) [12], an enhanced iteration of the FastSpeech model. It extends parallel feed-forward transformer with length prediction to rectify the constraints of its predecessor, attaining accelerated training speed and providing control over speech variation information (e.g., pitch, energy, and more accurate duration) as conditional inputs with fully end-to-end text-to-waveform synthesis.

Another type of speech synthesizer models the complex speech distribution as a repetitive composition of simple distributions, which is called a generative flow (Glow) [24]. Previously, the concept was mainly used in vocoders of neural TTS, but now it is also applied to acoustic models [3]. Valle et al. introduced Flowtron, an autoregressive text-to-speech synthesis model based on autoregressive flow techniques [13]. This novel generative network achieves high-quality mel-spectrograms and enables the manipulation of speech variation and style transfer. Moreover, it possesses the capability to control various aspects of speech, including pitch, tone, speech rate, cadence, and accent.

Diffusion neural networks are probabilistic generative models, which operate by learning to reverse a gradual noise corruption process. This enables speech generation through iterative denoising of random noise conditioned on text input and control parameters [2]. The stochastic nature of diffusion processes allows for diverse synthesis outputs while maintaining stable training dynamics, effectively addressing the main issues that have historically challenged other controllable generative approaches in speech synthesis. StyleTTS 2 [32] is a non-autoregressive TTS model with differentiable duration modeling, leveraging style diffusion and adversarial training with large speech language models (SLMs) to achieve human-level synthesis quality. The system is capable of synthesizing contextually appropriate styles directly from reference speech in a zero-shot scenario.

Generative Adversarial Networks (GANs) have become an important technology in controllable TTS systems [34]. GANs consist of a generator that synthesizes speech from text and a discriminator that evaluates its realism, trained adversarially to produce high-fidelity outputs. GANs are widely used in vocoders, as well as acoustic models for end-to-end systems [3, 24, 34]. Among vocoders, HiFi-GAN [19] is a state-of-the-art GAN-based approach that achieves both high-fidelity speech synthesis and exceptional computational efficiency, which implies the diversity of applications in end-to-end speech systems. It employs in-

novative multi-scale and multi-period discriminators, combined with a generator incorporating multi-receptive field fusion modules, which effectively capture diverse temporal patterns in speech through dilated convolutions with different dilation rates and kernel sizes.

Classification of Emotional TTS Models

Emotional Tagging Control

Emotional tagging represents the most straightforward approach to expressiveness control, using explicit labels or encodings to condition synthesis models. This methodology has evolved from simple one-hot encodings to sophisticated multi-dimensional representations.

Older systems allow users to select from a predefined set of discrete emotion labels, such as happy, sad, angry, or neutral [5]. The model is usually trained on datasets where speech samples are annotated with these labels, such as ESD, RAVDESS, and IEMOCAP, enabling it to generate speech with the corresponding emotional tone.

The label-based method can be implemented in multiple modes. For instance, MsEmoTTS [23] employs a multi-scale emotional speech synthesis framework that can be conditioned with one-hot encoded vectors representing discrete emotions such as happiness, anger, sadness, surprise, fear, and disgust. Inoue et al. introduce a hierarchical emotion distribution (HED) model for continuous emotion intensity control across phonemes, words, and utterances [16]. Practically, the expressiveness is controlled via the vector of emotional intensities from 0 to 1. In contrast, StyleTagging-TTS (ST-TTS) [11] and Emo-DPO [7] systems represent an approach that utilizes a defined set of style tags written in natural language to control emotional expression, modeling the relationship between linguistic embedding and speaking emotion domain with a pre-trained language model.

A significant drawback of this approach is the limitations in emotional intensity control. Systems, such as MsEmoTTS, have implicit predictors, but no explicit control over that characteristic is provided. Other challenges include customization issues due to fixed categories set and the need for large labeled datasets for training [5].

A different method for emotion encoding refers to representing it as a vector of so-called basic (or fundamental) emotions. Zhou et al. [30] present a study on modeling and synthesizing mixed emotions in speech synthesis. The paper extensively references Plutchik's emotion wheel theory, stating eight primary emotions: anger, fear, sadness, disgust, surprise, anticipation, trust, and joy, and arranges them in a framework where all the emotional styles can be derived from those [26]. The framework allows users to manually control emotion rendering by defining an emotion attribute vector with specific percentages for each primary emotion, successfully synthesizing complex emotional states like delight (surprise + happy), outrage (surprise + angry), and disappointment (surprise + sad) [30]. Similarly, for the EmoMix model [8], the authors considered excitement (happy + surprise) and disappointment (sad + surprise) within the intensity range for evaluation.

The PAD (Pleasure-Arousal-Dominance) model, also known as the VAD (Valence-Arousal-Dominance) model, is a three-dimensional psychological framework developed by Albert Mehrabian and James A. Russell to describe and measure emotional states using numerical dimensions [4]. The model represents emotions through valence (the pleasantness or unpleasantness of an emotion), arousal (the intensity or energy level associated with the emotion), and dominance (the degree of control or power one feels in the emotional state).

This dimensional approach has been widely adopted in emotional speech synthesis systems because it provides more nuanced emotional rendering than discrete categorical emotion models [10]. Sivaprasad et al. [29] extended the FastSpeech2 TTS architecture, introducing a prosody control block that conditions phoneme-level pitch, energy, and duration on continuous arousal-valence values, enabling fine-grained, interpretable emotional prosody control. EmoSphere-TTS [10] synthesizes expressive speech by embedding valence-arousal-dominance representation into a spherical vector via transformation. This approach enables manual control of emotional style and intensity by assigning appropriate angles and lengths to the spherical vector. UDDETTS framework [20] introduces a neural codec language model that seamlessly integrates traditional discrete emotion labels with the continuous VAD model, enabling controllable, expressive TTS through joint optimization in a unified emotional space.

The main drawback of continuous vector emotional representations is the scarcity of annotated training data, as dimensional emotion labels (e.g., arousal, valence, dominance) are costly and subjective to obtain

at scale. Moreover, models that leverage these representations often require complex architectures and careful tuning to prevent overfitting and ensure interpretable, controllable outputs.

Reference-based Control

Reference-based methods extract style information from example utterances, enabling control without explicit annotation. The reference audio provides a direct example of the emotional tone, which the system mimics. Reference-based systems use architectures where a reference encoder extracts emotional style embeddings from the input sample. This paradigm supports both voice conversion and style transfer applications.

Mellotron [22] is a multispeaker voice synthesis model based on Tacotron 2 [6], which can make a voice emote without emotive training data by explicitly conditioning on rhythm and continuous pitch contours from an audio signal. It pioneered reference-based emotional control by incorporating global style tokens (GST) [31] as learned latent variables alongside reference-based speech conditioning, allowing it to transfer text, rhythm, and pitch characteristics from source audio to target speakers while maintaining fine-grained control over expressive speech characteristics. Building upon such a reference-based approach, GenerSpeech [14] proposes a generalizable text-to-speech model that decomposes speech variation into style-agnostic and style-specific parts through a multi-level style adaptor and Mix-Style Layer Normalization, enabling robust zero-shot style transfer for out-of-domain custom voices without requiring adaptation data.

StyleTTS [21] extends the approach by using style encoders that extract emotion-relevant features while disentangling them from speaker identity and linguistic content. The system employs adversarial training to ensure the sound quality of the reconstructed mel-spectrogram. SC VALL-E [18] adopts the VALL-E [25] neural codec language model, consideri speech synthesis as a conditional language modeling task, and uses reference audio for speaker cloning and style transfer, including speaking rate, pitch, voice intensity, and emotional styles. EmoSphere++ [9] extends the EmoSphere-TTS [10] model and introduces an emotion-adaptive spherical vector that extracts it directly from reference speech samples, using a multi-level style encoder to capture both high-level emotional categories and low-level nuanced expressions for zero-shot emotion transfer.

Reference-based TTS systems face significant challenges in their dependency on high-quality reference audio and struggle with generalizing to unseen emotions, speakers, or prosodic styles not represented in the training data, while also requiring effective disentanglement of emotional content from other style features like speaker identity and linguistic prosody. Additionally, these approaches encounter practical limitations, including computational overhead from processing reference audio during inference and the scarcity of fine-grained explicit emotion control with this method.

Prompt-based Control

With the rapid development of large language models (LLMs) in recent years, prompt-based approaches represent the frontier of controllable emotional TTS, using large language models to interpret and execute complex synthesis instructions. These systems go beyond simple emotional labels to understand contextual and situational factors and allow users to specify emotions using natural language descriptions or instructions, e.g., "speak happily" or "whisper fearfully." This approach is user-friendly, leveraging natural language processing (NLP) to interpret prompts.

PromptTTS [28] utilizes a BERT-based style encoder to extract semantic representations from text prompts, conditioning a FastSpeech 2-based non-autoregressive acoustic model for mel-spectrogram generation, paired with a HiFi-GAN vocoder for waveform synthesis. Its successor, PromptTTS 2 [27], integrates LLMs for enhanced prompt understanding and a diffusion-based variation network to model diverse emotional styles, improving expressiveness and voice variability. InstructTTS [17] employs a VQ-VAE to discretize speech into latent codes, using a diffusion transformer to align text prompts with these codes, enabling fine-grained emotional control via a BERT-like approach, and a RoBERTa encoder for instruction processing. CosyVoice [6] combines an LLM to enable fine-grained freestyle natural language emotion control, using Qwen2.5-0.5B as its backbone with supervised semantic tokens, feeding into a non-autoregressive transformer decoder for multilingual zero-shot synthesis, with a style encoder for prompt-driven emotion control.

These systems demonstrate the field's convergence toward transformer-based architectures for interpreting emotional descriptions. However, challenges around prompt ambiguity and computational complexity

persist. Yet, translating natural language emotion descriptions to consistent acoustic realizations remains challenging due to the inherent subjectivity in emotional expressions and data availability for a prompt-to-style alignment system.

Quality Evaluation

Evaluating emotional TTS systems requires assessing both synthesis quality and emotional appropriateness, presenting unique challenges compared to neutral TTS evaluation.

Objective Metrics

Mel Cepstral Distortion (MCD) quantifies the spectral distance between synthesized and reference emotional speech [34]. Better speech synthesis quality is shown by a lower MCD value, which also indicates a higher resemblance between the reference and synthetic speech. Good quality is often indicated by an MCD value less than 4; however, substantial distortion may be indicated by values greater than 6. It can be calculated using the following formula:

$$MCD = \frac{10}{\ln 10}\sqrt{2\sum_{i=1}^{M}\left(c_i^{gen} - c_i^{ref}\right)^2}$$

where c_i^{gen} and c_i^{ref} are the i-th mel cepstral coefficient (MCC) of generated and reference speech, respectively, and M is the total number of MCCs. While widely used and applicable in reference-based systems, MCD may not capture perceptually important emotional characteristics.

As pitch level is vastly impacted by emotional state, Gross Pitch Error (GPE) is considered for expressive TTS [5]. It determines the proportion of voiced frames with a pitch deviation of greater than a specific threshold (often 20%) [1]. Emotion and style classifiers and speech emotion recognition (SER) models are widely used to measure classification accuracy, reflecting the efficiency of the proposed model in generating emotional speech [5]. It is worth mentioning that this evaluation relies on appropriate SER method selection.

Cosine similarity between emotional embeddings quantifies how similar the synthesized and reference speech is in terms of emotional expression [34]. It can be used to evaluate emotion-controllable TTS methods, where higher values indicate better emotional similarity. Emotional embeddings can be extracted using pre-trained emotion recognition models, such as x-vector-based systems trained on emotional speech datasets, to quantify how well the synthesized speech matches the intended emotional expression of the reference audio.

The Word Error Rate (WER) [34] is used to ensure intelligibility. By calculating the amount of transcription errors, it measures the difference between the reference transcript and the recognized transcript. WER is calculated as follows:

$$WER = \frac{W + M + E}{N}$$

where W is the number of wrong words in place of the correct word, M is the number of missed words, E is the number of extra words added, and N is the total number of words in the reference transcript. However, this metric is highly dependent on the choice of transcription method.

Subjective metrics

The most widely used subjective statistic is the Mean Opinion Score (MOS) [34]. On a scale of 1 to 5, listeners judge many characteristics of synthesized speech, including naturalness and expressiveness [10], and the result is the average of these scores. Higher ratings denote higher quality. Although MOS is costly for extensive assessments, it successfully captures the human perspective [34]. Two TTS audio samples are

compared for relative quality differences using the Comparison Mean Opinion Score (CMOS) [5]. After hearing paired samples, participants score their preference on a scale, which usually includes both negative and positive values (e.g., from -3 to 3) [34], and then it is averaged as in MOS.

Various researchers have employed these metrics in various ways to evaluate expressivity-related characteristics. Specifically, the metrics have been tested on audio samples that encompass diverse emotional expressions, distinct speaking mannerisms, and different degrees of intensity [5]. Additionally, these samples span multiple speech synthesis contexts, including scenarios involving parallel versus non-parallel style conversion, as well as familiar versus unfamiliar speaking styles and voices.

Discussion

Emotional TTS systems pose considerable technological problems in terms of feature disentanglement and representation, making it difficult to distinguish emotional aspects from other speech features such as speaker identification or linguistic content. This can, however, be handled using specialized disentanglement architectures or adversarial training strategies that expressly require the separation of emotional and speaker-specific information. Development of interpretable latent space manipulation techniques and controllable generation methods using semantic emotion embeddings could potentially improve fine-grained control and interpretability.

Evaluation metrics lack objectivity and standardization, making it difficult to compare different approaches or accurately measure progress across studies, an issue which cannot be fully eliminated. The significant gap in generating computationally efficient models can be bridged using knowledge distillation approaches, in which lightweight student models learn from bigger teacher networks, or architectural improvements such as separable convolutions and parameter sharing algorithms.

Low-resource languages face even greater obstacles due to minimal emotional speech data availability, though cross-lingual transfer learning and multilingual pre-training approaches offer promising solutions to explore. Given the daunting challenge of building comprehensive emotional speech datasets spanning diverse emotions, styles, speakers, and languages, data augmentation and knowledge transfer techniques become essential for addressing data scarcity in expressive speech synthesis.

These challenges collectively limit practical deployment, but the convergence of these solution strategies creates opportunities for researchers to make meaningful contributions that advance multiple aspects of emotional TTS simultaneously.

Conclusion

The swift progressions in emotional text-to-speech (TTS) synthesis have transitioned from rudimentary concatenative methodologies to sophisticated neural architectures capable of generating emotionally resonant and high fidelity speech. This evolution has been significantly propelled by deep learning, which enables intricate management of emotional tone and stylistic variation.

In contemporary systems, transformer-based and diffusion-based architectures are increasingly prevalent. Transformers facilitate adjustable synthesis and parallel generation, while diffusion models employ iterative refinement to enhance quality. The previously utilized fixed emotive tags have been refined by reference-based and prompt-driven control mechanisms, and the incorporation of LLMs has created novel strategies for adaptable, and user-centric interaction.

Notwithstanding these advancements, several challenges persist. It remains an obstacle to disentangling emotion from speaker identification and content. The acquisition of high-quality emotional data continues to be a daunting endeavor, particularly within low-resource languages, and the lack of standardized evaluation leads to inconsistencies in comparative analyses.

Список літератури

1. A comparative study of pitch extraction algorithms on a large variety of singing sounds / Onur Babacan [et al.] // ICASSP 2013 - proceedings. – Vancouver, Canada, 2013. — Pp. 1–5.
2. A survey on audio diffusion models: text to speech synthesis and enhancement in generative AI [Electronic resource] / Chenshuang Zhang [et al.]. — Mode of access: arXiv.2303.13336.
3. A survey on neural speech synthesis [Electronic resource] / Xu Tan [et al.]. — Mode of access: arXiv.2106.15561.
4. Bales R. F. Social interaction systems: theory and measurement / Robert Freed Bales. — New Brunswick, NJ : Transaction Publishers, 2017.

5. Barakat H. Deep learning-based expressive speech synthesis: a systematic review of approaches, challenges, and resources / Huda Barakat, Oytun Turk, Cenk Demiroglu // EURASIP journal on audio, speech, and music processing. — 2024. — Vol. 2024, no. 1. — P. 11.
6. CosyVoice: a scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens [Electronic resource] / Zhihao Du [et al.]. — Mode of access: arXiv.2407.05407.
7. Emo-DPO: controllable emotional speech synthesis through direct preference optimization [Electronic resource] / Xiaoxue Gao [et al.]. — Mode of access: arXiv.2409.10157.
8. EmoMix: emotion mixing via diffusion models for emotional speech synthesis / Haobin Tang [et al.] // Interspeech 2023. — [S. l.], 2023. — Pp. 12–16.
9. EmoSphere++: Emotion-Controllable Zero-Shot Text-to-Speech via Emotion-Adaptive Spherical Vector / Deok-Hyeon Cho [et al.] // IEEE transactions on affective computing. — 2025. — Pp. 1–16.
10. EmoSphere-TTS: emotional style and intensity modeling via spherical emotion vector for controllable emotional text-to-speech / Deok-Hyeon Cho [et al.] // Interspeech 2024. — [S. l.], 2024. — Pp. 1810–1814.
11. Expressive text-to-speech using style tag / Minchan Kim [et al.] // Interspeech 2021. — [S. l.], 2021. — Pp. 4663–4667.
12. FastSpeech 2: fast and high-quality end-to-end text to speech [Electronic resource] / Yi Ren [et al.]. — Mode of access: arXiv.2006.04558.
13. Flowtron: an autoregressive flow-based generative network for text-to-speech synthesis [Electronic resource] / Rafael Valle [et al.] // International conference on learning representations. — [S. l.], 2021. — Mode of access: https://openreview.net/forum?id=Ig53hpHxS4.
14. GenerSpeech: towards style transfer for generalizable out-of-domain text-to-speech / Rongjie Huang [et al.] // Proceedings of the 36th international conference on neural information processing systems. — Red Hook, NY, USA, 2022.
15. Glow-TTS: a generative flow for text-to-speech via monotonic alignment search / Jaehyeon Kim [et al.] // Advances in neural information processing systems. — [S. l.], 2020. — Pp. 8067–8077.
16. Hierarchical emotion prediction and control in text-to-speech synthesis / Sho Inoue [et al.] // ICASSP 2024 - 2024 IEEE international conference on acoustics, speech and signal processing (ICASSP). — [S. l.], 2024. — Pp. 10601–10605.
17. InstructTTS: modelling expressive TTS in discrete latent space with natural language style prompt / Dongchao Yang [et al.] // IEEE/ACM transactions on audio, speech, and language processing. — 2024. — Vol. 32. — Pp. 2913–2925.
18. Kim D. SC VALL-E: Style-Controllable Zero-Shot Text to Speech Synthesizer [Electronic resource] / Daegyeom Kim, Seongho Hong, Yong-Hoon Choi. — Mode of access: arXiv.2307.10550.
19. Kong J. HiFi-GAN: generative adversarial networks for efficient and high fidelity speech synthesis / Jungil Kong, Jaehyeon Kim, Jaekyoung Bae // Advances in neural information processing systems. — [S. l.], 2020. — Pp. 17022–17033.
20. Liu J. UDDETTS: unifying discrete and dimensional emotions for controllable emotional text-to-speech [Electronic resource] / Jiaxuan Liu, Zhenhua Ling. — Mode of access: arXiv.2505.10599.
21. Li Y. A. StyleTTS: a style-based generative model for natural and diverse text-to-speech synthesis / Yinghao Aaron Li, Cong Han, Nima Mesgarani // IEEE journal of selected topics in signal processing. — 2025. — Vol. 19, no. 1. — Pp. 283–296.
22. Mellotron: multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens / Rafael Valle [et al.] // ICASSP 2020 - 2020 IEEE international conference on acoustics, speech and signal processing (ICASSP). — [S. l.], 2020. — Pp. 6189–6193.
23. MsEmoTTS: multi-scale emotion transfer, prediction, and control for emotional speech synthesis / Yi Lei [et al.] // IEEE/ACM trans. audio, speech and lang. proc. — 2022. — Vol. 30. — Pp. 853–864.
24. Mu Z. Review of end-to-end speech synthesis technology based on deep learning [Electronic resource] / Zhaoxi Mu, Xinyu Yang, Yizhuo Dong. — Mode of access: arXiv.2104.09995.
25. Neural codec language models are zero-shot text to speech synthesizers [Electronic resource] / Chengyi Wang [et al.]. — Mode of access: arXiv.2301.02111.
26. Plutchik R. Theories of emotion / Robert Plutchik, Henry Kellerman. — Burlington : Elsevier Science, 1980.
27. PromptTTS 2: describing and generating voices with text prompt / Yichong Leng [et al.] // The twelfth international conference on learning representations. — [S. l.], 2024.
28. Promptttts: controllable text-to-speech with text descriptions / Zhifang Guo [et al.] // ICASSP 2023 - 2023 IEEE international conference on acoustics, speech and signal processing (ICASSP). — [S. l.], 2023. — Pp. 1–5.
29. Sivaprasad S. Emotional prosody control for speech generation / Sarath Sivaprasad, Saiteja Kosgi, Vineet Gandhi // Interspeech 2021. — [S. l.], 2021. — Pp. 4653–4657.
30. Speech synthesis with mixed emotions / Kun Zhou [et al.] // IEEE transactions on affective computing. — 2023. — Vol. 14, no. 4. — Pp. 3120–3134.
31. Style tokens: unsupervised style modeling, control and transfer in end-to-end speech synthesis / Yuxuan Wang [et al.] // Proceedings of the 35th international conference on machine learning. — [S. l.], 2018. — Pp. 5180–5189.
32. StyleTTS 2: towards human-level text-to-speech through style diffusion and adversarial training with large speech language models / Yinghao Aaron Li [et al.] // Advances in neural information processing systems / ed. by A. Oh [et al.]. — [S. l.], 2023. — Pp. 19594–19621.
33. Tacotron: towards end-to-end speech synthesis / Yuxuan Wang [et al.] // Interspeech 2017. — [S. l.], 2017. — Pp. 4006–4010.
34. Towards controllable speech synthesis in the era of large language models: a survey [Electronic resource] / Tianxin Xie [et al.]. — Mode of access: arXiv.2412.06602.

References

Babacan, O., Drugman, T., d'Alessandro, N., Henrich Bernardoni, N., & Dutoit, T. (2013). A comparative study of pitch extraction algorithms on a large variety of singing sounds. *ICASSP 2013 - Proceedings*, 1–5. https://hal.science/hal-00923967.

Bales, R. F. (2017). *Social interaction systems: Theory and measurement*. Transaction Publishers. https://doi.org/10.4324/9781315129563.

Barakat, H., Turk, O., & Demiroglu, C. (2024). Deep learning-based expressive speech synthesis: A systematic review of approaches, challenges, and resources. *EURASIP Journal on Audio, Speech, and Music Processing*, *2024* (1), 11. https://doi.org/10.1186/s13636-024-00329-7.

Cho, D.-H., Oh, H.-S., Kim, S.-B., Lee, S.-H., & Lee, S.-W. (2024). EmoSphere-TTS: Emotional Style and Intensity Modeling via Spherical Emotion Vector for Controllable Emotional Text-to-Speech. *Interspeech 2024*, 1810–1814. https://doi.org/10.21437/Interspeech.2024-398.

Cho, D.-H., Oh, H.-S., Kim, S.-B., & Lee, S.-W. (2025). EmoSphere++: Emotion-Controllable Zero-Shot Text-to-Speech via Emotion-Adaptive Spherical Vector. *IEEE Transactions on Affective Computing*, 1–16. https://doi.org/10.1109/TAFFC.2025.3561267.

Du, Z., Chen, Q., Zhang, S., Hu, K., Lu, H., Yang, Y., Hu, H., Zheng, S., Gu, Y., Ma, Z., Gao, Z., & Yan, Z. (2024). *CosyVoice: A Scalable Multilingual Zero-shot Text-to-speech Synthesizer based on Supervised Semantic Tokens* (No. arXiv:2407.05407). arXiv. https://doi.org/10.48550/arXiv.2407.05407.

Gao, X., Zhang, C., Chen, Y., Zhang, H., & Chen, N. F. (2024). *Emo-DPO: Controllable Emotional Speech Synthesis through Direct Preference Optimization* (No. arXiv:2409.10157). arXiv. https://doi.org/10.48550/arXiv.2409.10157.

Guo, Z., Leng, Y., Wu, Y., Zhao, S., & Tan, X. (2023). Prompttts: Controllable Text-To-Speech With Text Descriptions. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10096285.

Huang, R., Ren, Y., Liu, J., Cui, C., & Zhao, Z. (2022). GenerSpeech: Towards style transfer for generalizable out-of-domain text-to-speech. *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Inoue, S., Zhou, K., Wang, S., & Li, H. (2024). Hierarchical Emotion Prediction and Control in Text-to-Speech Synthesis. *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 10601–10605.

Kim, D., Hong, S., & Choi, Y.-H. (2023). *SC VALL-E: Style-Controllable Zero-Shot Text to Speech Synthesizer* (No. arXiv:2307.10550). arXiv. https://doi.org/10.48550/arXiv.2307.10550.

Kim, J., Kim, S., Kong, J., & Yoon, S. (2020). Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search. *Advances in Neural Information Processing Systems*, *33*, 8067–8077.

Kim, M., Cheon, S. J., Choi, B. J., Kim, J. J., & Kim, N. S. (2021). Expressive Text-to-Speech Using Style Tag. *Interspeech 2021*, 4663–4667. https://doi.org/10.21437/Interspeech.2021-465.

Kong, J., Kim, J., & Bae, J. (2020). HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. *Advances in Neural Information Processing Systems*, *33*, 17022–17033.

Lei, Y., Yang, S., Wang, X., & Xie, L. (2022). MsEmoTTS: Multi-Scale Emotion Transfer, Prediction, and Control for Emotional Speech Synthesis. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, *30*, 853–864. https://doi.org/10.1109/TASLP.2022.3145293.

Leng, Y., Guo, Z., Shen, K., Tan, X., Ju, Z., Liu, Y., Liu, Y., Yang, D., Zhang, L., Song, K., He, L., Li, X.-Y., Zhao, S., Qin, T., & Bian, J. (2024). PromptTTS 2: Describing and Generating Voices with Text Prompt. *The Twelfth International Conference on Learning Representations*.

Li, Y. A., Han, C., & Mesgarani, N. (2025). StyleTTS: A Style-Based Generative Model for Natural and Diverse Text-to-Speech Synthesis. *IEEE Journal of Selected Topics in Signal Processing*, *19* (1), 283–296. https://doi.org/10.1109/JSTSP.2025.3530171.

Li, Y. A., Han, C., Raghavan, V., Mischler, G., & Mesgarani, N. (2023). StyleTTS 2: Towards Human-Level Text-to-Speech through Style Diffusion and Adversarial Training with Large Speech Language Models. B A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in Neural Information Processing Systems* (Vol. 36, pp. 19594–19621). Curran Associates, Inc.

Liu, J., & Ling, Z. (2025). *UDDETTS: Unifying Discrete and Dimensional Emotions for Controllable Emotional Text-to-Speech* (No. arXiv:2505.10599). arXiv. https://doi.org/10.48550/arXiv.2505.10599.

Mu, Z., Yang, X., & Dong, Y. (2021). *Review of end-to-end speech synthesis technology based on deep learning* (No. arXiv:2104.09995). arXiv. https://doi.org/10.48550/arXiv.2104.09995.

Plutchik, R., & Kellerman, H. (1980). *Theories of Emotion*. Elsevier Science.

Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., & Liu, T.-Y. (2021). FastSpeech 2: Fast and High-Quality End-to-End Text to Speech. *International Conference on Learning Representations*. https://openreview.net/forum?id=piLPYqxtWuA.

Sivaprasad, S., Kosgi, S., & Gandhi, V. (2021). Emotional Prosody Control for Speech Generation. *Interspeech 2021*, 4653–4657. https://doi.org/10.21437/Interspeech.2021-307.

Tan, X., Qin, T., Soong, F., & Liu, T.-Y. (2021). *A Survey on Neural Speech Synthesis* (No. arXiv:2106.15561). arXiv. https://doi.org/10.48550/arXiv.2106.15561.

Tang, H., Zhang, X., Wang, J., Cheng, N., & Xiao, J. (2023). EmoMix: Emotion Mixing via Diffusion Models for Emotional Speech Synthesis. *INTERSPEECH 2023*, 12–16. https://doi.org/10.21437/Interspeech.2023-1317.

Valle, R., Li, J., Prenger, R., & Catanzaro, B. (2020). Mellotron: Multispeaker Expressive Voice Synthesis by Conditioning on Rhythm, Pitch and Global Style Tokens. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6189–6193. https://doi.org/10.1109/ICASSP40776.2020.9054556.

Valle, R., Shih, K. J., Prenger, R., & Catanzaro, B. (2021). Flowtron: An Autoregressive Flow-based Generative Network for Text-to-Speech Synthesis. *International Conference on Learning Representations*. https://openreview.net/forum?id=Ig53hpHxS4.

Wang, C., Chen, S., Wu, Y., Zhang, Z., Zhou, L., Liu, S., Chen, Z., Liu, Y., Wang, H., Li, J., He, L., Zhao, S., & Wei, F. (2023). *Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers* (No. arXiv:2301.02111). arXiv. https://doi.org/10.48550/arXiv.2301.02111.

Wang, Y., Stanton, D., Zhang, Y., Ryan, R.-S., Battenberg, E., Shor, J., Xiao, Y., Jia, Y., Ren, F., & Saurous, R. A. (2018). Style Tokens: Unsupervised Style Modeling, Control and Transfer in End-to-End Speech Synthesis. *Proceedings of the 35th International Conference on Machine Learning*, 5180–5189. https://proceedings.mlr.press/v80/wang18h.html.

Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., & Saurous, R. A. (2017). Tacotron: Towards End-to-End Speech Synthesis. *Interspeech 2017*, 4006–4010. https://doi.org/10.21437/Interspeech.2017-1452.

Xie, T., Rong, Y., Zhang, P., Wang, W., & Liu, L. (2025). *Towards Controllable Speech Synthesis in the Era of Large Language Models: A Survey* (No. arXiv:2412.06602; arXiv. https://doi.org/10.48550/arXiv.2412.06602.

Yang, D., Liu, S., Huang, R., Weng, C., & Meng, H. (2024). InstructTTS: Modelling Expressive TTS in Discrete Latent Space With Natural Language Style Prompt. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *32*, 2913–2925. https://doi.org/10.1109/TASLP.2024.3402088.

Zhang, C., Zhang, C., Zheng, S., Zhang, M., Qamar, M., Bae, S.-H., & Kweon, I. S. (2023). *A Survey on Audio Diffusion Models: Text To Speech Synthesis and Enhancement in Generative AI* (No. arXiv:2303.13336). arXiv. https://doi.org/10.48550/arXiv.2303.13336.

Zhou, K., Sisman, B., Rana, R., Schuller, B. W., & Li, H. (2023). Speech Synthesis With Mixed Emotions. *IEEE Transactions on Affective Computing*, *14* (4), 3120–3134. https://doi.org/10.1109/TAFFC.2022.3233324.

Іващенко Д. С., Марченко О. О.

СУЧАСНІ ПІДХОДИ ДО КОНТРОЛЬОВАНОГО СИНТЕЗУ ЕМОЦІЙНОГО МОВЛЕННЯ

У статті представлено комплексний огляд сучасних технологій керованих систем для емоційного синтезу мовлення. Проаналізовано еволюцію нейронних архітектур, систематизовано підходи за технологіями та методами емоційного контролю. Визначено ключові виклики галузі, що охоплюють відокремлення мовленнєвих ознак та дефіцит даних для мов з обмеженими ресурсами. Окреслено перспективні напрями розвитку систем емоційно контрольованого синтезу мовлення.

Ключові слова: глибоке навчання, синтез мовлення з тексту, обробка природної мови, емоційний контроль мовлення, дифузійні моделі.

Матеріал надійшов 25.06.2025

УДК 004.855.91:519.2:336.76
DOI: 10.18523/2617-3808.2025.8.38-42

M. Voitishyn, D. Kuzmenko

A HYBRID AI MODEL FOR FINANCIAL MARKET PREDICTION

Financial time series modeling is increasingly complex due to volatility, unexpected breakouts, and the impact of external factors, such as macroeconomic indicators, investor sentiment, company fundamentals, and extreme shocks, like geopolitical events or market manipulations. This paper introduces a hybrid artificial intelligence framework that integrates traditional statistical methods, machine learning models, and Bayesian neural networks (BNNs) to improve predictive performance and uncertainty quantification in financial forecasting. The model leverages a variety of engineered features, including rolling statistics, technical indicators, anomaly scores, interpolated macroeconomic data, and transformer-based sentiment scores.

A complete ablation study compares various architectures, including ARIMA, SARIMA, MLR, SNN, and BNN, across multiple prediction windows (1, 3, 5 days) and feature combinations. Results show that while linear models yield the lowest MSE for short-term predictions, they fail to capture non-linear dependencies and uncertainty. In contrast, BNNs offer more reliable mid-term predictions by estimating predictive distributions. The best BNN configuration (Normal distribution, constant variation, TanH activation, 1 hidden layer) achieved an MSE of 0.00022, confirming the advantage of uncertainty-adjusted modeling. Sentiment analysis and anomaly detection were especially impactful when combined with macroeconomic indicators, improving signal reliability and behavioral insight.

Our findings highlight the importance of integrating diverse data sources and accounting for predictive uncertainty in financial applications. Additionally, the experiments revealed that compact network architectures often outperform deeper ones when paired with engineered features. All experiments were systematically tracked to ensure reproducibility and facilitate future model benchmarking.

Keywords: probability theory, Bayesian neural networks, financial analysis, uncertainty quantification, anomaly detection, time-series forecasting.

Introduction

In this study, we conduct an ablation analysis to evaluate how different model architectures and feature sets impact financial time series prediction. The objective is to better understand the contributions of various components under real-world financial uncertainty.

We experiment with two key assumptions:

1. Modeling uncertainty improves prediction robustness – by comparing Bayesian Neural Networks with traditional models, we explore how estimating predictive distributions, including location and scale, enhances forecast reliability.
2. Behavioral signals embedded in sentiment data are predictive – by integrating sentiment analysis extracted from Twitter using transformer models, we assess whether these features improve financial forecasts.

Methods

Feature engineering. We developed a diverse feature set grouped into: market data, macroeconomic indicators, technical features, datetime variables, and anomaly sentiment signals. For baseline data: raw OHLCV was used, where prices were adjusted for stock splits. Macroeconomic Indicators were parsed from Federal Reserve Economic Data(FRED) database and included variables such as CPI, GDP, and Unemployment Rate, interpolated to daily frequency, We focused heavily on extra features: Technical indicators (SMA, EMA, RSI), anomaly scores (IQR and Isolation Forest), sentiment signals, datetime features. Each feature type was normalized and merged into a comprehensive dataset with a unified target — Close Price Daily Return.

Anomaly detection. Continuous anomaly scores were computed using the Isolation Forest algorithm, which isolates outliers by random partitioning data. Discrete anomaly scores detect unusual market behavior using interquartile range (IQR) filtering.

Sentiment extraction. Sentiment scores were derived from Twitter financial text sources and incorporated as predictive features. These scores provide insights into market psychology, capturing reactions to news, earnings, or macroeconomic developments, and offer a complementary perspective to purely numerical indicators.

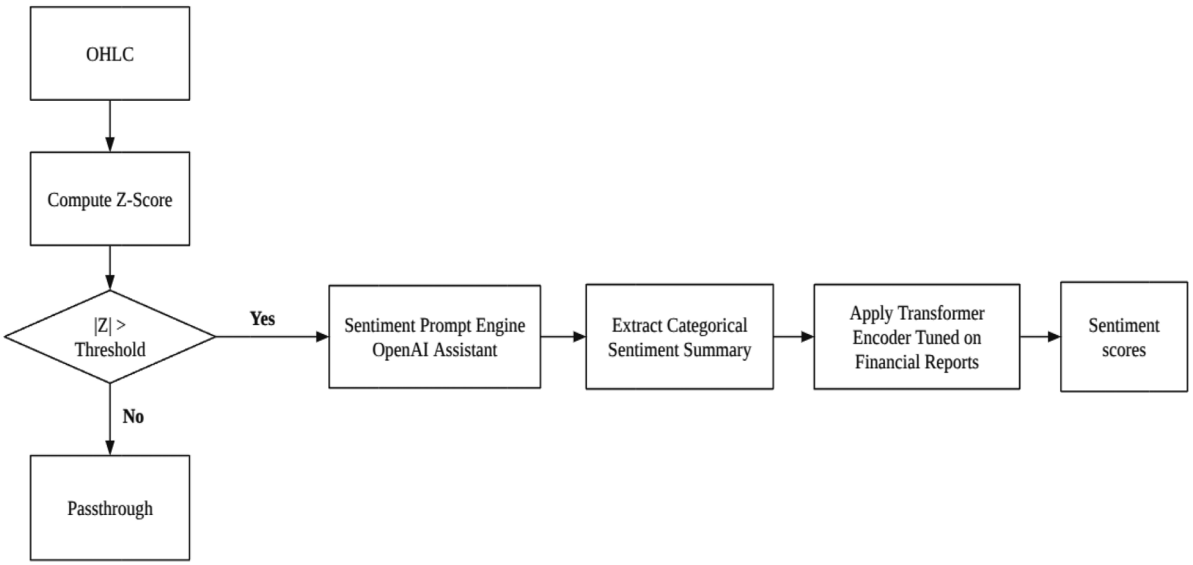


Figure 1. Sentiment Extraction Flowchart

Based on the sentiment extraction flow chart above, the following example, generated using an OpenAI Assistant, illustrates how social media sentiment was quantified and integrated into the forecasting pipeline:

"On November 5, 2015, Nvidia (ticker: NVDA) experienced a notable price jump, which coincided with a surge in Twitter activity. The sentiment on social media was mixed, with many users expressing excitement over Nvidia's strong earnings report and its advancements in gaming and AI technologies. This generated a wave of optimism among investors. At the same time, some users voiced concerns about market volatility and potential overvaluation, reflecting a degree of skepticism."

To quantify this sentiment, the tweet content was processed by a specialized transformer model named FitTwitterRoberta. The model outputs the following sentiment probabilities: positive: 0.9345, neutral: 0.0608, and negative: 0.0047. This resulted in a strong positive sentiment signal for this data point, which served as a valuable input for the main forecasting model.

Model tracking. To ensure reproducibility and efficient experiment management, we have implemented an MLflow Tracking Server to log and monitor all modeling experiments. The setup captures key details, including model configurations, training and validation losses, evaluation metrics, and other relevant parameters throughout the experimentation process.

Experiments

We compared several model architectures across different prediction horizons of 1, 3, and 5 days. The models included ARIMA and SARIMA as baseline statistical approaches to begin with, then Multiple Linear Regression (MLR) as a linear baseline suitable for structured inputs, SNN as the deep learning baseline, and BNNs, which are particularly promising for modeling uncertainty. These comparisons were conducted on five semiconductor stock assets.

The experimental setup involved seven types of processed data and a total of 87 engineered features. Each of the five model architectures was evaluated with different configurations across three types of predictive tasks.

Within the BNN framework specifically, we tested variations in distributional assumptions (Normal vs. Student's-t) as well as output configurations, comparing models that estimate only the meaning with those that estimate both the mean and scale parameters. In addition to these core variations, we explored different

hidden layer complexities, allowing us to assess the effect of model capacity on predictive performance and uncertainty quantification. We also experimented with different prior distributions over weights to evaluate their influence on posterior estimates. Furthermore, we tested multiple activation functions — including ReLU and TanH.

To evaluate the contribution of various data sources, features were grouped and tested incrementally. The experiments began with a baseline feature set, then progressively incorporated macroeconomic indicators, and finally included additional external signals, allowing us to assess the marginal impact of each feature group on model performance.

Results

The results show that linear models without non-linear transformations and using all features (Baseline & Macro & Extra) perform the best, especially for short-term 1-day forecasts, the lowest test MSE is 2.46e-05. In contrast, adding non-linear transformations leads to huge overfitting, with test MSEs exploding.

Table 1. *MLR Results with an emphasis on insightful findings*

MLR_INTC model configuration	Features	MSE
non-linear transformation: False, target: 1 day	Baseline + Macro + Extra	2.46e-05
non-linear transformation: False, target: 3 days	Baseline + Macro + Extra	9.94e-05
non-linear transformation: False, target: 1 day	Baseline	1.15e-04
non-linear transformation: True, target: 1 day	Baseline + Macro + Extra	2998.1

The simple neural network results show consistent performance across different feature sets and horizons, with the best test MSE of 0.000167 observed for the 3-day forecast using only baseline features. Interestingly, adding macro and extra features slightly increased test MSE.

Table 2. *SNN Results with an emphasis on insightful findings*

SNN_INTC model configuration	Features	MSE
activation layer: TanH, hidden neurons: 1, target: 3 days	Baseline	0.000167
activation layer: TanH, hidden neurons: 1, target: 3 days	Baseline + Macro + Extra	0.000168
activation layer: TanH, hidden neurons: 1, target: 5 days	Baseline + Macro + Extra	0.000169
activation layer: TanH, hidden neurons: 1, target: 1 day	Baseline + Macro + Extra	0.000179

Focusing on BNN experiments, we have extracted several key insights. Firstly, there is no consistent evidence that the Student's-t distribution, often expected to handle financial return outliers more robustly, outperforms the Normal distribution across different configurations. While the Student's-t models demonstrated higher robustness in some setups, they also showed greater variance in performance, particularly when both location and scale parameters are estimated.

The best-performing models were based on the Normal distribution estimating only the location parameter. Notably, the top result MSE of 0.00022 was achieved with a Normal(loc) setup using the TanH activation function, a minimal architecture (1 hidden neuron, 3-day prediction window), and the full feature set including macroeconomic and extra variables.

Table 3. *BNN Results with an emphasis on insightful findings*

BNN_INTC model configuration	Features	MSE
Normal(loc,scale), activation layer: ReLU, hidden neurons: 9, target: 3 days	Baseline	0.0008
Normal(loc,scale), activation layer: TanH, hidden neurons: 9, target: 1 day	Baseline	0.00090
Normal (loc, scale = const), activation layer: ReLU, hidden neurons: 1, target: 3 days	Baseline + Macro + Extra	0.00023
Normal(loc,scale=const), activation layer: TanH, hidden neurons: 1, target: 3 days	Baseline + Macro + Extra	0.00022
Student's-t(df=4, loc, scale), activation layer: ReLU, hidden neurons: 9, target: 5 days	Baseline + Macro + Extra	0.00145
Student's-t(df=4,loc, scale), activation layer: TanH, hidden neurons: 9, target: 1 day	Baseline + Macro + Extra	0.00121
Student's-t(df=4, loc, scale = const), activation layer: ReLU, hidden neurons: 1, target: 5 days	Baseline + Macro + Extra	0.00029
Student's-t(df=4, loc, scale = const), activation layer: TanH, hidden neurons: 1, target: 3 days	Baseline + Macro + Extra	0.0036

Regarding activation functions, TanH showed slightly better performance in several configurations compared to ReLU, especially in smaller models. However, the advantage was not universal. Lastly, increasing hidden neuron complexity did not guarantee improved performance. Smaller models often achieved better results, emphasizing the importance of architectural simplicity in combination with rich features for financial time series forecasting.

The plot below corresponds to a BNN where both the location (mean) and scale (variance) parameters are estimated. As a result, the model not only forecasts the expected return but also adjusts its level of uncertainty dynamically, depending on the volatility of the input data. This behavior is particularly valuable in financial contexts where risk varies across time.

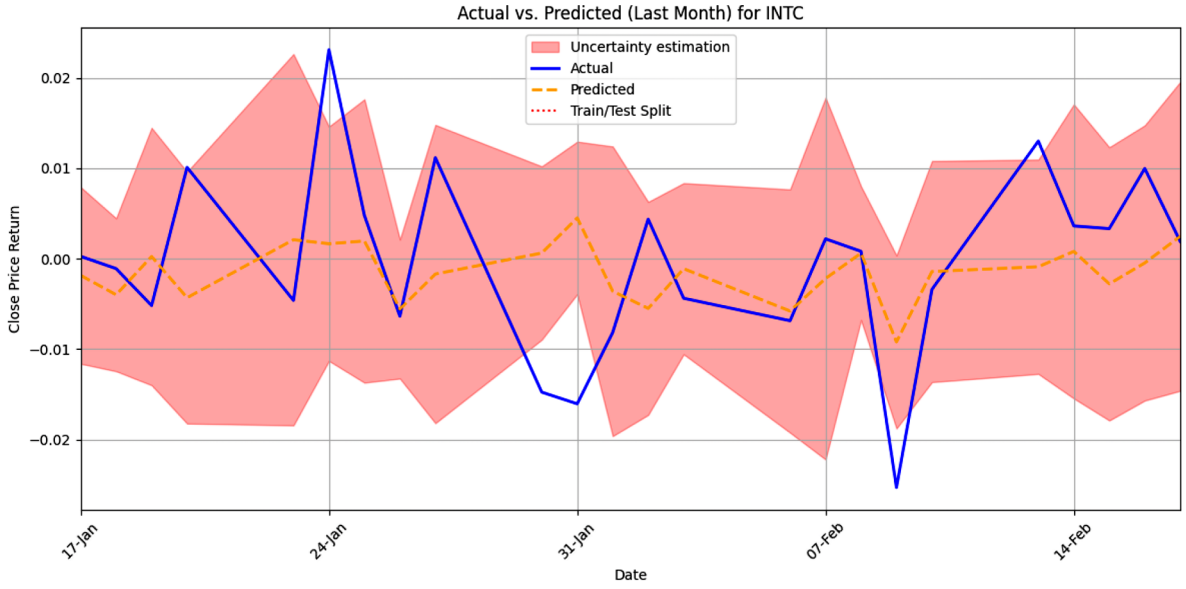


Figure 2. Ground Truth vs. Predicted Return and Uncertainty Estimation with MLFlow

Conclusion

While the MLR model achieved the lowest MSE of 2.46e-05, this result holds limited value when compared with other models in the ablation study, as the model itself lacks the capacity to capture the nonlinear dynamics present in real-world financial data. Therefore, despite its slightly higher MSE of 0.00022, the BNN is more reliable for real-world financial forecasting tasks, validating the first assumption that accounting for uncertainty offers more robust predictive performance. Moreover, both SNN and BNN returned their best results when all three feature sets — baseline, macroeconomic, and extra features — were included with a target window of 3 days. This clearly supports the second assumption, confirming that the feature engineering pipeline we developed significantly enhances model performance.

Table 4. *Best results over all architectures*

Model configuration	Features	MSE
Linear regression, scaled: True, nonlinear: False, target: 1 day	Baseline + Macro + Extra	2.46e-05
Simple Neural Network: scaled: True, target: 3 days	Baseline + Macro + Extra	0.000168
Bayesian Neural Network: scaled: True, Normal (loc, scale=const), TanH, target: 3 days	Baseline + Macro + Extra	0.00022

Список літератури

1. Barbieri F. Twitter-roBERTa-base for Sentiment Analysis [Electronic resource] / F. Barbieri, J. Camacho-Collados, L. Espinosa Anke, L. Neves // HuggingFace.co. — Mode of access: https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment (date of access: 25.05.2025).
2. Borch: Tutorials and contribution guides [Electronic source] // borch.readthedocs.io. — Mode of access: https://borch.readthedocs.io/en/latest/index.html (date of access: 25.05.2025).
3. Federal Reserve Bank of St. Louis. Federal Reserve Bank of St. Louis [Electronic resource] / Federal Reserve Bank of St. Louis // StLouisFed.org. — Mode of access: https://www.stlouisfed.org/ (date of access: 25.05.2025).
4. Hauzenberger N. Enhanced Bayesian Neural Networks for Macroeconomics and Finance [Electronic resource] / N. Hauzenberger et al. // arXiv.org. — Mode of access: https://arxiv.org/abs/2211.04752v2 (date of access: 25.05.2025).

5. Kirtac K. Sentiment Trading with Large Language Models [Electronic resource] / K. Kirtac, G. Germano // arXiv.org. — Mode of access: https://arxiv.org/abs/2412.19245 (date of access: 25.05.2025).
6. Lewis B. A Deep Universal Probabilistic Programming Language [Electronic resource] / L. Belcher, J. Gudmundsson, M. Green Borch // arXiv.org. — Mode of access: https://arxiv.org/abs/2209.06168 (date of access: 25.05.2025).
7. MLflow: An Open Source Platform for the Machine Learning Lifecycle [Electronic source] // MLflow.org. — Mode of access: https://mlflow.org/docs/latest/index.html (date of access: 25.05.2025).
8. OpenAI. Assistants API Overview [Electronic resource] / OpenAI // OpenAI Platform. — Mode of access: https://platform.openai.com/docs/assistants/overview (date of access: 25.05.2025).

References

Barbieri F., Camacho-Collados, J., Espinosa Anke, L., & Neves, L. (2025). *Twitter-roBERTa-base for sentiment analysis*. Hugging Face. Retrieved May 25, 2025. https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment.

Borch. (2025). *Tutorials and contribution guides*. Borch Documentation. Retrieved May 25, 2025. https://borch.readthedocs.io/en/latest/index.html.

Federal Reserve Bank of St. Louis. (2025). *Federal Reserve Bank of St. Louis website*. Retrieved May 25, 2025. https://www.stlouisfed.org/.

Hauzenberger, N., Huber, F., Klieber, K., & Marcellione, M. (2022). *Enhanced Bayesian neural networks for macroeconomics and finance* [Preprint]. arXiv. https://arxiv.org/abs/2211.04752v2.

Kirtac, K., & Germano, G. (2024). *Sentiment trading with large language models* [Preprint]. arXiv. https://arxiv.org/abs/2412.19245.

Lewis, B., Gudmundsson, J., & Green Borch, M. (2022). *A deep universal probabilistic programming language* [Preprint]. arXiv. https://arxiv.org/abs/2209.06168.

MLflow. (2025). *An open source platform for the machine learning lifecycle*. Retrieved May 25, 2025. https://mlflow.org/docs/latest/index.html.

OpenAI. (2025). *Assistants API overview*. OpenAI Platform. Retrieved May 25, 2025. https://platform.openai.com/docs/assistants/overview.

М. Є. Войтішин, Д. О. Кузьменко

РОЗРОБКА ГІБРИДНОЇ МОДЕЛІ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПРОГНОЗУВАННЯ ФІНАНСОВИХ РИНКІВ

У статті досліджено можливості гібридного підходу до прогнозування фінансових ринків із застосуванням методів штучного інтелекту. Основну увагу приділено аналізу впливу різних архітектур моделей та наборів ознак на якість прогнозування часових рядів у фінансовому середовищі з високою невизначеністю. Запропоновано поєднання традиційних статистичних моделей, простих нейронних мереж та баєсівських нейронних мереж для моделювання як прогнозного значення, так і масштабної невизначеності. Особливу увагу приділено інженерії ознак, зокрема інтеграції макроекономічних індикаторів, технічних показників, а також поведінкових сигналів на основі аналізу настроїв з Twitter. Результати експериментів показують, що хоча лінійні моделі досягають найменшої середньоквадратичної помилки, саме баєсівські нейронні мережі забезпечують надійніші прогнози завдяки врахуванню невизначеності. Аналіз підтверджує ефективність нашої інженерії ознак та демонструє потенціал поєднання кількісних і якісних даних у фінансовому прогнозуванні.

Ключові слова: прогнозування часових рядів, баєсівські нейронні мережі, фінансовий аналіз, невизначеність, пошук аномалій, аналіз настроїв.

Матеріал надійшов 31.05.2025

УДК 004.8:004.93:004.89
DOI: 10.18523/2617-3808.2025.8.43-49

Глибовець М. М., Сидорова Є. О.

ЗАСТОСУВАННЯ ГРАФОВИХ БАЗ ДАНИХ І EXPLAINABLE AI У СТВОРЕННІ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ З ГІБРИДНИМ СПОСОБОМ ФІЛЬТРАЦІЇ

У дослідженні описано побудову рекомендаційної системи з гібридним способом фільтрації, що використовує гетерогенні графові структури, нейронні мережі та Explainable AI заради підвищення персоналізації та прозорості рішень. Базова методологія оптимізації рекомендацій ґрунтується на сучасних методах машинного навчання, які здатні ефективно моделювати залежності між об'єктами. Предметна область задачі охоплює мультимедійні рекомендації, зосереджені на персоналізованому підході. Представлена архітектура системи дає можливість ефективно обробляти множину вузлів різних типів і зв'язків (набір взаємодій, належність елементів до жанру, подібність), що відображають комплексні ознаки поведінкових і контекстуальних даних.

Результати експериментального тестування підтвердили ефективність запропонованого підходу: система демонструє високі показники персоналізації, новизни рекомендацій та здатності виявляти подібних користувачів.

Ключові слова: рекомендаційна система; графова нейронна мережа; гетерогенний граф; гібридна фільтрація; мультимодальні дані; персоналізація; Explainable AI; Neo4j; Python.

Вступ

У сучасному інформаційному просторі обсяг цифрового контенту постійно зростає, що одночасно і спрощує, і ускладнює для користувачів процес пошуку цікавої та корисної інформації, зокрема у вигляді книжок, ігор, фільмів і безлічі інших мультимедійних об'єктів. З огляду на це рекомендаційні системи (далі — РС) мають неабияке значення для отримання релевантного контенту в процесі взаємодії з інформаційними платформами. Це дослідження спрямоване на створення РС із гібридним способом фільтрації, залученням гетерогенних графових структур і підходів Explainable AI для покращення персоналізації наданих порад.

Зазвичай подібні системи стикаються з низкою обмежень, зокрема пов'язаних із класичними підходами у формі колаборативної і контентної фільтрації, відсутністю прозорості у прийнятих рішеннях і вузьким спектром мультимодальних ознак.

Отже, основною метою цієї роботи є побудова такої архітектури, яка б надавала можливість застосовувати ефективний і більш розширений підхід для досягнення персоналізації і прозорості рекомендацій. Запропоноване у цій статті програмне рішення враховує мультимодальні ознаки, об'єднує атрибутивну та поведінкову інформацію, а також обґрунтовує причинові зв'язки обраних рекомендацій, орієнтуючись на реальні патерни користувача.

Аналіз сучасних підходів

Найпоширенішими парадигмами РС є колаборативна фільтрація (CF) і контентна фільтрація (CBF), однак їхнє впровадження дуже залежить від структури і типів даних для контенту [2]. Для першого типу значною вразливістю є проблема «холодного старту» за наявності вузького спектра початкових даних; для другого ж значним викликом є складність даних у формі мультимодальності та різноманітності типів контенту, тоді як інтеграція гібридного способу дає можливість використовувати обидва підходи одночасно [12].

Також слід зазначити, що запровадження такої гібридної архітектури часто підкріплено використанням графових структур і нейронних мереж для моделювання зв'язків між об'єктами різних типів. Таким чином, побудована програмна реалізація відображає комплексні ознаки поведінкових і кон-

текстуальних даних. Також слід сказати, що впровадження й застосування таких моделей, як GraphSAGE, забезпечує індуктивне навчання з високою здатністю до узагальнення [5].

Крім того, зі стрімким розвитком технологій усе більшої актуальності набувають інтеграції компонентів Explainable AI (XAI) до такого роду моделей. Через архітектуру «чорної скриньки» взаємозв'язок між результатом і системою часто потребує пояснень, тому рішення з їхнім використанням дають змогу зрозуміти, які елементи стали найбільш вагомими [4].

Порівняно з наявними рішеннями, які використовують лише текстові чи табличні ознаки, у цій роботі система враховує інтегровані векторні ознаки з різних медіамодальностей — текстові, візуальні та жанрові ознаки; забезпечує крос-типову подібність (наприклад, книжки, подібні до фільмів за жанровим профілем); об'єднує поведінкові взаємодії, доповнені атрибутами об'єктів; використовує пояснення XAI для підвищення довіри до рекомендацій.

Архітектура графової системи

В основі архітектури лежить гетерогенний граф G=(V, E), у якому вузли й ребра мають різні типи й категорії. Така структура дає змогу одночасно моделювати поєднання різнорідних об'єктів і їхніх відношень, що є основною складовою для рекомендаційних задач у складному медіасередовищі [4]. Предметна область персоналізованих рекомендацій мультимедійного контенту охоплює такі типи ресурсів, як книжки, фільми та відеоігри. Вона характеризується високою різнорідністю об'єктів, множинним вибором взаємодій і потребою в глибокій персоналізації результатів.

Множина вузлів V для цього дослідження охоплює користувачів (User), книжки (Book), фільми (Movie), ігри (Game) та жанри (Genre). Множина ребр E описує такі зв'язки: належність медіаелементу до певного жанру, взаємодія користувачів з об'єктами (перегляд, оцінка, ігнорування, купівля), а також залежності подібності.

Отже, кожен тип вузла та зв'язку має окрему семантику у формі типізованих ознак, відповідно до моделі гетерогенних графів.

Набір даних

Було сформовано синтетичний набір даних, проте з реалістичними показниками, який містить об'єкти трьох доменів, а також штучно згенеровану популяцію користувачів із симульованою поведінкою. Дані для книжок було отримано за допомогою API Google Books [3], де для кожного жанру вказано конфігурації належності саме до типу книжок, оскільки категоризація відповідає саме початковому типу об'єкта. Для ігор було використано API RAWG [9], яке надає доступ до великої бази даних ігровою класифікацією, а для фільмів — сервіс RapidAPI з API Advanced Search [1], за допомогою яких аналогічно до попередніх відбувався процес класифікації за жанровою належністю. Кожному типу об'єкта ми зберегли набір атрибутів, який складається з унікального ідентифікатора, назви, жанру, короткого опису та посилання до відповідного зображення.

Користувачів було створено як незалежних агентних сутностей із відповідними атрибутами віку (15–65 років), статі, розташування, дати реєстрації на визначеному проміжку й рівня активності, який обраховується як відношення кількості прикладів взаємодії з моменту реєстрації дотепер (нормований коефіцієнт). Усі 1000 сутностей були відмічені унікальними ідентифікаторами для імітації гетерогенної популяції з різними стилями споживання контенту. За поведінкові зв'язки відповідають ребра, типізовані за типами взаємодій, згаданих до цього. Кількість таких взаємодій було згенеровано пропорційно до рівня активності кожного користувача.

Усі зібрані дані нормалізувалися для використання єдиного класу жанру та відповідного універсального зв'язку. Збережені об'єкти інтегрувалися до графової бази даних Neo4j.

Векторні репрезентації та матриці подібності

Для представлення об'єктів використано мультимодальні ембедінги, які формуються шляхом поєднання текстових, візуальних і жанрових ознак. Для побудови векторних подань словесних матеріалів медіаоб'єктів було використано модель Sentence-BERT, що відображає текстові описи у векторний простір сталої розмірності [10]. За жанровими мітками було збудовано one-hot репре-

зентації, які дозволяють створювати належність до конкретного набору жанрів. Для візуальних прив'язок було використано попередньо навчену модель ResNet-50 з обробленням фотоданих у контексті масштабування і нормалізації. Оброблення виконується для кожного типу медіа окремо. Формування векторів для користувачів ґрунтується на метаданих, тому для числових змінних застосовано стандартизацію, а для категоріальних — one-hot.

На цій основі обчислюються матриці косинусної подібності з подальшим формуванням симетричних графових відношень. Тоді для кожного з типів медіа розраховується top-k (у цьому випадку 5) найбільш подібних об'єктів за косинусною відстанню між ембедінгами. Оцінюється міжтипова жанрова подібність за one-hot репрезентаціями жанрів, де схожість вище, ніж заданий поріг. На основі інформації про користувачів будується top-k (у цьому випадку 5) найближчих користувачів для колаборативних рекомендацій. Усі обчислені відношення імпортуються до графової бази даних Neo4j з відповідними ваговими коефіцієнтами. Приклади запитів для наочної демонстрації зв'язків у графовій структурі подано на рис. 1 і 2.

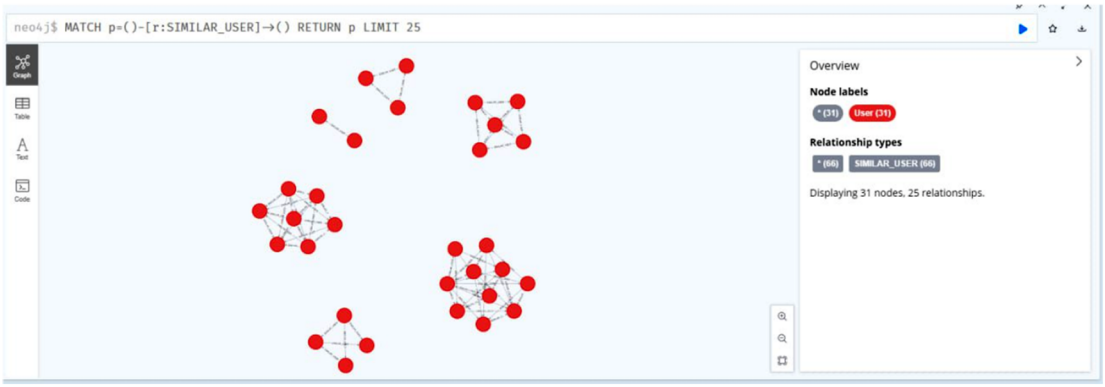


Рис. 1. Приклад зв'язків схожості користувачів у графовій системі

Рис. 2. Приклад зв'язків міжтипової схожості у графовій системі

Таким чином було сформовано систему з 4150 вузлів, які описують об'єкти різних типів і 122 862 ребра для відображення стосунків між ними.

Надання рекомендацій

Для побудови персоналізованих рекомендацій ми застосували графові нейронні мережі, що дає можливість досить просто навчити спільний простір представлень для різноманітних сутностей і їхніх взаємодій [6].

Розроблений гетерогенний граф слугує початком для створення одиничної ознаки, всі зв'язки серіалізуються та оброблюються за допомогою фреймворку PyTorch Geometric. Архітектура запропонованої моделі передбачає агрегацію інформації, використання двох рівнів конвулюційних шарів і оброблення лінійними шарами для обчислення функції подібності. Навчання триває 50 епох, містить метод negative sampling і створює ітеративний процес оновлення ваг шарів.

Для процесу пояснення рекомендацій до використання цієї моделі використовується кілька важливих етапів. По-перше, для прозорості результатів пошуку подібних користувачів використовується градієнт косинусної подібності між векторами схожості користувачів і ідентифікація перетину їхніх взаємодій. Генерація інтерпретованих пояснень передбачає обраховану градієнтну оцінку, спільні демографічні атрибути та дії користувачів. Приклад продемонстровано на рис. 3.

```
🔍 Similar Users:
👤 217 | Similarity: 0.993
   🧠 Age: 26 | Gender: Female | Location: Australia
   📌 Why similar? High feature similarity (importance score: 3.953); Same gender: Female
👤 715 | Similarity: 0.992
   🧠 Age: 63 | Gender: Other | Location: USA
   📌 Why similar? High feature similarity (importance score: 2.658)
👤 668 | Similarity: 0.992
   🧠 Age: 46 | Gender: Male | Location: India
   📌 Why similar? High feature similarity (importance score: 6.957); Shared interactions: movie ID 2702
👤 12 | Similarity: 0.992
   🧠 Age: 47 | Gender: Female | Location: France
   📌 Why similar? High feature similarity (importance score: 4.785); Same gender: Female; Same location: France
👤 587 | Similarity: 0.991
   🧠 Age: 57 | Gender: Male | Location: Ukraine
   📌 Why similar? High feature similarity (importance score: 4.646)
```

Рис. 3. Приклад рекомендації схожих користувачів

По-друге, процес обґрунтування рекомендацій об'єктів відбувається шляхом градієнтної оцінки важливості за сумністю пари «користувач — елемент». Таким чином для кожного типу вузлів обираються найбільш впливові, що дозволяє сформувати список представників, які відіграли значущу роль у формуванні рекомендації. Приклад продемонстровано на рис. 4.

```
🎯 Recommended Items:

📌 The Encyclopedia of Animals (ID: 1337, Type: Book)
   🔍 Why this recommendation?
   - Influential user: ID 566 (Importance: 230.473)
   - Influential user: ID 337 (Importance: 247.173)
   - Influential user: ID 7 (Importance: 946.939)
   - Influential book: ID 1337 (Importance: 456.681)

📌 VRChat (ID: 3411, Type: Game)
   🔍 Why this recommendation?
   - Influential user: ID 566 (Importance: 472.993)
   - Influential user: ID 337 (Importance: 509.165)
   - Influential user: ID 7 (Importance: 1924.821)
   - Influential book: ID 1337 (Importance: 456.681)
   - Influential game: ID 3411 (Importance: 241.711)

📌 Minecraft (ID: 3282, Type: Game)
   🔍 Why this recommendation?
   - Influential user: ID 566 (Importance: 741.872)
   - Influential user: ID 337 (Importance: 799.964)
   - Influential user: ID 7 (Importance: 3009.730)
   - Influential book: ID 1337 (Importance: 456.681)
   - Influential game: ID 3282 (Importance: 235.309)
   - Influential game: ID 3411 (Importance: 241.711)
```

Рис. 4. Приклад рекомендації медіаелементів

Програмна реалізація

Для програмної реалізації (далі — ПР) було використано мову програмування Python із залученням таких основних бібліотек: PyTorch Geometric (для побудови графових нейронних мереж), scikit-learn, pandas (для кодування й попереднього оброблення даних) і matplotlib (для візуалізації).

РС реалізовано у формі модульного застосунку з окремими компонентами для побудови графа, тренування моделі, генерації рекомендацій, XAI-компонентами (пояснення схожості користувачів і важливості вузлів за градієнтними характеристиками), а також тестування.

Експериментальне тестування

У межах експериментального тестування було обрано набір нееталонних метрик (reference-free, без ground-truth) для оцінки, оскільки система не оперує завчасно готовими правильними відповідями для рекомендацій. Вони містять Intra-List Similarity і Novelty для оцінки рекомендованих елементів; User Similarity, Shared Interaction Overlap, Attribute Alignment, User Diversity для оцінки схожості користувачів; а також Coverage для аналізу спектра доступних релевантних елементів і Personalization для оцінки унікальності рекомендацій [7; 8; 11].

Загалом результати експериментального дослідження засвідчили якісну здатність знаходити схожих користувачів і пропонувати нові унікальні медіаоб'єкти. Це підтверджується високими показниками новизни рекомендацій і внутрішньої різноманітності списків. Нормалізоване отримане значення спільних об'єктів взаємодії між цільовими й подібними користувачами, а також рівень різноманітності схожих користувачів не завжди були оптимальними, що зумовлено великим каталогом медіапродуктів порівняно з кількістю користувачів та обсягом їхньої активності. Помірна відповідність атрибутивної схожості між користувачами найймовірніше пов'язана з вузьким наповненням метаданих профілів, аналогічно до показників персоналізації.

Слід зазначити, що результати тестування, показані на рис. 5, окреслюють ключові напрями подальшого вдосконалення, зокрема збагачення атрибутивної моделі користувачів.

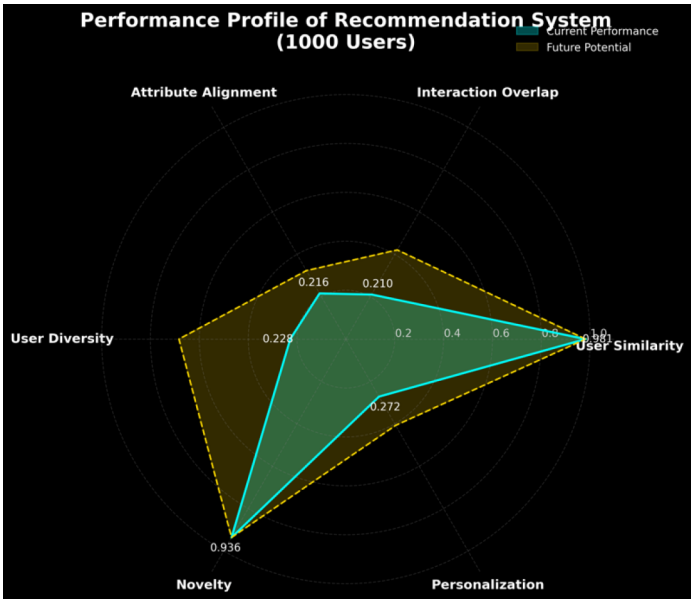


Рис. 5. Результати роботи системи

Висновок

У статті описано процес розроблення гібридної архітектури РС і продемонстровано підхід до її побудови із залученням графових нейронних мереж і Explainable AI. Програмна реалізація РС представлена у формі застосунку. Запропонована архітектура програмної системи включає модульну структуру, яка позначає свої результати роботи на відповідних запитах.

Тестування показало здатність системи знаходити нові релевантні рекомендації з високим рівнем різноманітності й персоналізації, а також формувати логічні пояснення на основі обґрунтованих порад.

Варто зазначити, що під час розроблення було враховано низку технічних викликів, зокрема обчислювальну складність, вплив шуму у даних, складність масштабування графових моделей та необхідність зрозумілості XAI-рішень.

У результаті запропонована архітектура РС дає можливість реалізувати розширений підхід до рекомендацій, здатний адаптуватися до різних сценаріїв використання, який можна удосконалювати згідно з проведеним аналізом.

Список літератури

1. Advanced Movie Search API [Electronic resource]. — Mode of access: https://rapidapi.com/rapidapi/api/advanced-movie-search.
2. Darban Z. Z. GHRS: Graph-based Hybrid Recommendation System with Application to Movie Recommendation [Electronic resource] / Z. Z. Darban, M. H. Valipour // Faculty of Information Technology, Monash University, Melbourne, Australia, Department of Engineering and Product, Ostadkar Company, Tehran, Iran. — 2021. — Mode of access: https://arxiv.org/pdf/2111.11293.
3. Google Books API [Electronic resource]. — Mode of access: https://developers.google.com/books.
4. Gupta S. K. Multimodal Graph-based Recommendation System using Hybrid Filtering Approach [Electronic resource] / S. K. Gupta, A. Kumar, D. Prasad // International Journal of Computing and Digital Systems. — 2025. — Mode of access: https://doi.org/10.12785/ijcds/1571047857.
5. Hamilton W. Inductive representation learning on large graphs [Electronic resource] / W. Hamilton, Z. Ying, J. Leskovec // NIPS. — 2017. — Mode of access: https://arxiv.org/pdf/1706.02216.
6. Hu W. Open Graph Benchmark: Datasets for Machine Learning on Graphs [Electronic resource] / W. Hu et al. NeurIPS, 2020. — Mode of access: https://arxiv.org/pdf/2005.00687.
7. Jesse M. W. Intra-list similarity and human diversity perceptions of recommendations: the details matter [Electronic resource] / M. W. Jesse, C. Bauer, D. Jannach // User Modeling and User-Adapted Interaction. — 2022. — Vol. 33 (5). — Mode of access: https://doi.org/10.1007/s11257-022-09351-w.
8. Liu H. A new user similarity model to improve the accuracy of collaborative filtering [Electronic resource] / H. Liu, Z. Hu, A. U. Mian // Knowledge-Based Systems. — 2014. — Vol. 56 (6). — Pp. 156–166. — Mode of access: https://doi.org/10.1016/j.knosys.2013.11.006.
9. RAWG Video Games Database API [Electronic resource]. — Mode of access: https://rawg.io/apidocs.
10. Reimers N. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks [Electronic resource] / N. Reimers, I. Gurevych // Proceedings of EMNLP-IJCNLP. — 2019. — Mode of access: https://aclanthology.org/D19-1410/.
11. Vargas S. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems [Electronic resource] / S. Vargas, P. Castells // Proceedings of the 2011 ACM Conference on Recommender Systems (RecSys 2011), Chicago, IL, USA, October 23–27, 2011. — Mode of access: https://doi.org/10.1145/2043932.2043955.
12. Wu L. A survey on neural recommendation: From collaborative filtering to content and knowledge enhanced recommendation [Electronic resource] / L. Wu, P. Sun, Y. Fu, R. Hong // IEEE Transactions on Knowledge and Data Engineering. — 2020 — Mode of access: https://www.researchgate.net/publication/351120123_A_Survey_on_Neural_Recommendation_From_Collaborative_Filtering_to_Content_and_Context_Enriched_Recommendation.

References

Advanced Movie Search API. https://rapidapi.com/rapidapi/api/advanced-movie-search.

Darban, Z. Z., & Valipour, M. H. (2021). *GHRS: Graph-based Hybrid Recommendation System with Application to Movie Recommendation*. Faculty of Information Technology, Monash University, Melbourne, Australia, Department of Engineering and Product, Ostadkar Company, Tehran, Iran. https://arxiv.org/pdf/2111.11293.

Google Books API. https://developers.google.com/books.

Gupta, S. K., Kumar, A., & Prasad, D. (2025). Multimodal Graph-based Recommendation System using Hybrid Filtering Approach. *International Journal of Computing and Digital Systems*. https://doi.org/10.12785/ijcds/1571047857.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*. https://arxiv.org/pdf/1706.02216.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., & Leskovec, J. (2020). Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Proceedings of NeurIPS*. https://arxiv.org/pdf/2005.00687.

Jesse, M. W., Bauer, C., & Jannach, D. (2022). Intra-list similarity and human diversity perceptions of recommendations: the details matter. *User Modeling and User-Adapted Interaction*, *33* (5). https://doi.org/10.1007/s11257-022-09351-w.

Liu, H., Hu, Z., & Mian, A. U. (2014). A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems*, *56*(6), 156–166. https://doi.org/10.1016/j.knosys.2013.11.006.

RAWG Video Games Database API. https://rawg.io/apidocs.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of EMNLP-IJCNLP*. https://aclanthology.org/D19-1410/.

Vargas, S., & Castells, P. (2011). Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. In *Proceedings of the 2011 ACM Conference on Recommender Systems (RecSys 2011)*. Chicago, IL, USA, October 23–27, 2011. https://doi.org/10.1145/2043932.2043955.

Wu, L., Sun, P., Fu, Y., & Hong, R. (2020). A survey on neural recommendation: From collaborative filtering to content and knowledge enhanced recommendation. *IEEE Transactions on Knowledge and Data Engineering*. https://www.researchgate.net/publication/351120123_A_Survey_on_Neural_Recommendation_From_Collaborative_Filtering_to_Content_and_Context_Enriched_Recommendation.

M. Hlybovets, Y. Sydorova

GRAPH-BASED MULTIMODAL RECOMMENDATION SYSTEM WITH EXPLAINABLE AI

The article presents the development process of a hybrid recommendation system (RS) based on heterogeneous graph data structures and graph neural networks (GNN), incorporating Explainable AI (XAI) approaches. The target domain is personalized multimedia recommendations, encompassing books,

movies, and games, characterized by diverse item modalities and high demand for personalization.

The system architecture is produced upon a heterogeneous graph G=(V, E), where nodes and edges represent different entity types (users, items, genres) and relationships (interactions, similarity, genre associations). Multimodal embeddings are defined by using textual features (via Sentence-BERT), visual data (via ResNet-50), and categorical labels (via one-hot genre encoding). These embeddings are used to generate similarity-based connections across item types and users.

For recommendations, a neural model based on the PyTorch Geometric framework is trained with negative sampling. The model predicts user-item relevance and generates recommendations. To support clarity, XAI modules provide explanations via gradient-based influence analysis, shared interactions, and demographic attribute comparison.

A set of reference-free evaluation metrics is used, including Intra-List Similarity, Novelty, User Similarity, Attribute Alignment, and Coverage. Experimental results demonstrate the model's capability to generate novel, diverse, and personalized recommendations while offering interpretable justifications.

The implementation is based on Python with modular components for data preprocessing, training, evaluation, and explanation outcomes, and leverages the Neo4j graph database for storage and analysis.

Keywords: recommendation system, graph neural network, heterogeneous graph, hybrid filtering, explainable AI, personalization, multimodal embeddings, Neo4j, Python.

Матеріал надійшов 25.06.2025

УДК 004.8:004.93
DOI: 10.18523/2617-3808.2025.8.50-56

A. Mykytyshyn, N. Shvai

VALIDATING ARCHITECTURAL HYPOTHESES IN NEURAL DECISION TREES WITH NEURAL ARCHITECTURE SEARCH

This article introduces an automated and unbiased framework for validating architectural hypotheses for neural network models, with a particular focus on Neural Decision Trees (NDTs). The proposed methodology employs Neural Architecture Search (NAS) as an unbiased tool to explore architectural variations and empirically assess theoretical claims. To demonstrate this framework, we investigate a hypothesis found in the literature: that the complexity of decision nodes in NDTs decreases monotonically with tree depth. This assumption, initially motivated by the task of monocular depth estimation, suggests that deeper nodes in the tree require fewer parameters due to simpler split functions.

To rigorously test this hypothesis, we conduct a series of NAS campaigns over the CIFAR-10 image classification dataset. The search space parameterizes each node by the number of convolutional blocks and fully connected layers, while all other architectural components are held constant to isolate the effect of node depth. By applying Tree-structured Parzen Estimator (TPE)-based NAS and evaluating over 300 architectures, we quantify complexity metrics across tree levels and analyze their correlations using Spearman's rank coefficient.

The results provide no statistical or visual evidence supporting the hypothesized trend: node complexity does not decrease with depth. Instead, complexity remains nearly constant across levels, regardless of tree depth or search space size. These results suggest that assumptions derived from specific applications may not generalize to other domains, underscoring the importance of empirical validation and careful search-space design. The presented framework may serve as a foundation for verifying other structural assumptions across various neural network families and applications.

Keywords: Neural Architecture Search (NAS), Neural Decision Trees (NDTs), Automated Machine Learning (AutoML), Computer Vision, Node Complexity.

Introduction

Automated Machine Learning (AutoML) has significantly streamlined the development and deployment of complex machine learning models by automating various aspects of the machine learning pipeline, including data preprocessing, feature engineering, model selection, and hyperparameter optimization [4]. A major subset of AutoML is Neural Architecture Search (NAS), a technique focused specifically on automating the design of neural network architectures. NAS has demonstrated remarkable success across various domains, consistently delivering architectures that match or even outperform those designed by human experts [3, 5].

At the same time, Neural Decision Trees (NDTs) have emerged as an innovative class of hybrid machine learning models, integrating the interpretability and intuitive decision logic of classical decision trees with the expressive capabilities of neural networks [2]. Despite their practical benefits, several underlying theoretical assumptions about their structure remain untested. In particular, a hypothesis proposed by Roy and Todorovic (2016) states that nodes deeper within an NDT should be structurally simpler, since learning split functions presumably becomes easier deeper within the tree [9].

However, this assumption has not been validated beyond the original context in which it was proposed (monocular depth estimation). Unverified theoretical assumptions pose a significant risk, potentially leading researchers and practitioners towards suboptimal architectural decisions and incorrect generalizations across different applications. Therefore, rigorous empirical testing of such architectural hypotheses is necessary to ensure reliable design decisions for neural decision trees in diverse tasks.

The objective of this research is to empirically verify the hypothesis that the complexity of neural decision tree nodes decreases with increasing depth using Neural Architecture Search as an unbiased experimental tool.

The scientific novelty of the obtained results includes:

- The first systematic, NAS-based empirical validation of an architectural hypothesis regarding the complexity of neural decision tree nodes.
- Empirical evidence demonstrating that the assumption of decreasing complexity with increasing depth, proposed by Roy and Todorovic, does not generalize across tasks.

The practical value of the obtained results lies in providing a validated methodological approach to critically evaluate theoretical claims about NDT architectures. Researchers and practitioners can leverage these findings to guide more reliable and empirically grounded architectural decisions in neural decision tree design.

Problem Definition

Roy and Todorovic [9] introduce the Neural Regression Forest (NRF) for monocular depth estimation, in which each split node is implemented by a "shallow" CNN – specifically, one with only one or two convolutional layers followed by one or two fully-connected layers – and the overall network depth emerges from the stacking of these modules. They conjecture that "it becomes easier to learn the split functions as we go down the tree," suggesting that nodes at lower levels should require progressively fewer layers to achieve the best possible predictive performance.

As a result of this conjecture, they propose an architecture where the nodes in a tree get "simpler" as they get closer to the leaves. Specifically, they split the tree into three equally deep layers. "For the top one third of the tree height, we use CNNs with 2 convolution + pooling layers, and 2 fully connected perceptron layers. For the lower one third of the tree height (closer to the leaf nodes), we use CNNs with 2 convolution + pooling layers and 1 fully connected perceptron layer. Finally, for the bottom third of the tree height, we use CNNs with 1 convolution + pooling layer and 1 connected perceptron layer" [9].

Drawing directly on their assumption and generalizing it, we formulate our central hypothesis for neural decision trees (NDTs):

Hypothesis. *In neural decision trees, the complexity of each split node, quantified by the number of convolutional blocks and fully connected layers, decreases monotonically with increasing tree depth.*

Proposed Approach

To verify the hypothesis that node complexity in neural decision trees decreases with tree depth, we employ neural architecture search (NAS) solely as an empirical tool. NAS systematically explores a predefined set of discrete architectural choices by optimizing for a target metric (in our case, accuracy), thus revealing which per-node configurations best support the task under identical training conditions [1, 3]. By automating the search, we eliminate human bias in selecting convolutional and fully connected layers, thus obtaining an unbiased measurement of how complexity varies across different levels.

All candidate architectures are trained and evaluated on the CIFAR-10 dataset, which comprises 60,000 color images of size 32×32 across 10 classes [6]. We choose CIFAR-10 for its status as a standard benchmark. Another benefit is that its modest image resolution and well-established augmentation protocols enable rapid NAS iterations. It is also worth noting that the decision to select a dataset different from the one used in the paper where this idea was initially introduced ([9]) is deliberate, as this allows us to test the general applicability of the hypothesis.

We define our search space by parameterizing each split node at tree level ℓ with two integer hyperparameters:

- b_ℓ – the number of convolutional blocks at level ℓ
- f_ℓ – the number of fully connected layers at level ℓ

Each convolutional block replicates the module from [2] – namely, a 3×3 convolution with 256 channels, followed by batch normalization and ReLU activation. We consider two value ranges for (b_ℓ, f_ℓ): {1, 2} (matching the original setup from [9] at depth ten), and {1, 2, 3} (to probe a broader spectrum of complexity).

All other architectural and optimization parameters remain fixed across experiments: the overall tree depth d is set per experiment, a double-block stem precedes branching, channel widths are constant at 256, and the optimizer, learning rate schedule, and augmentation pipeline are identical. This ensures that varia-

tions in NAS outcomes reflect only the relative efficacy of different (b_t, f_t) choices, allowing for a clear test of the monotonic complexity hypothesis.

We employ the Tree-structured Parzen Estimator (TPE) algorithm. Each candidate model undergoes 5 epochs of training ("proxy evaluation"), after which the top 10% of architectures (by validation accuracy) are retrained for 15 epochs to refine the final rankings.

All experiments employ mixed-precision training and a gradient scaler to speed up training. Input images are augmented with random horizontal flips and random crops to 32×32 (with four-pixel padding), followed by tensor conversion and normalization using the CIFAR-10 channel means and standard deviations. Optimization uses the AdamW optimizer with a fixed initial learning rate of 1×10^{-3}.

Architectures are ranked by accuracy on a held-out CIFAR-10 validation set. Accuracy directly measures the quality of learned split functions in a classification context, providing a principled and unbiased basis for comparing node complexities [3].

Implementation Details and Reproducibility

All experiments were conducted on Google Colab using an NVIDIA A100 GPU with 32 GB of RAM, subject to Colab's 12-hour session limit. The code was written in Python 3 and relied on the PyTorch deep learning framework for model definition and training [8], as well as Microsoft's NNI library for the implementation of NAS [7]. To ensure reproducibility, a fixed random seed of 42 was used for all data splits, network initialisations, and sampling in the NAS algorithm.

Experimental Campaigns

To test the node complexity hypothesis, we ran six NAS campaigns, each fixing tree depth d and the discrete knob set for (b_t, f_t) using TPE with median-stop early stopping after 3 epochs. Each trial required approximately 1 minute for 5 epochs (shorter if stopped early), and all campaigns ran no longer than the 12-hour Colab limit (approximately 700 minutes). All campaigns used a 5-epoch proxy evaluation and 15-epoch retraining of the top 10 % of candidates.

Table 1. **Experimental runs that were performed**

Run ID	Depth d	$(b_t, f_t) \in$	Search space size
A	3	{1, 2}	$2^3 \times 2^3 = 64$
B	4	{1, 2}	$2^4 \times 2^4 = 256$
C	6	{1, 2}	$2^6 \times 2^6 = 4{,}096$
D	3	{1, 2, 3}	$3^3 \times 3^3 = 729$
E	4	{1, 2, 3}	$3^4 \times 3^4 = 6{,}561$
F	6	{1, 2, 3}	$3^6 \times 3^6 = 531{,}441$

- Runs A-C probe the original Roy and Todorovic complexity range {1, 2} at depths 3, 4, and 6.
- Runs D-F expand the search to {1, 2, 3} at the same depths.

Each trial sampled a full configuration of $\{b_0, ..., b_{d-1}, f_0, ..., f_{d-1}\}$ and returned validation accuracy after 5 epochs; the top 10 % of trials per run were then retrained for 15 epochs for final evaluation.

Run-wise accuracy summaries

Figure 1 (next page) displays a table listing the top-five architectures returned by each campaign after the 15-epoch retraining phase. Three regularities are evident:

- **Tight within-run spread.** All five models in a given run differ by ≤ 0.01 in top 1 accuracy, showing that the parameters we searched over most likely do not influence the accuracy that much.
- **Depth penalty.** Mean accuracy declines as depth increases—from ~0.83 for depth 3 (Runs A, D) to ~0.79 for depth 6 (Runs C, F). Deeper NDTs entail more parameters and longer inference paths, making them harder to optimize given the fixed 15-epoch budget and 256-channel bottleneck.
- **Effect of knob range.** For each depth, the runs with the broader option set {1, 2, 3} (D–F) surpass their {1, 2} counterparts (A–C) by ~0.01. Allowing an extra layer choice evidently enables NAS to fine-tune node expressiveness without overfitting, hence the modest but consistent gain.

run	accuracy	rank	fc_l0	stg_l0	fc_l1	stg_l1	fc_l2	stg_l2	fc_l3	stg_l3	fc_l4	stg_l4	fc_l5	stg_l5
	0.83	1	1	2	2	1	2	2						
	0.828	2	2	2	2	2	2	1						
A	0.826	3	1	2	1	1	1	1						
	0.824	4	1	2	1	2	2	1						
	0.822	5	1	2	2	2	2	1						
	0.81	1	1	1	2	2	2	1	1	1				
	0.808	2	1	1	2	1	1	2	1	1				
B	0.806	3	1	1	1	2	2	1	1	1				
	0.804	4	1	1	2	1	1	2	1	1				
	0.802	5	1	1	2	1	2	2	2	1				
	0.79	1	2	1	2	1	2	2	1	2	2	1	1	2
	0.788	2	1	2	2	2	2	2	1	2	1	2	2	2
C	0.786	3	2	1	2	1	1	2	2	1	2	1	2	1
	0.784	4	1	1	1	1	2	2	1	1	1	2	2	1
	0.782	5	2	1	1	2	1	2	2	2	2	2	2	1
	0.84	1	1	1	2	3	1	1						
	0.838	2	1	1	3	2	1	1						
D	0.836	3	3	1	3	2	1	2						
	0.834	4	2	1	1	3	1	3						
	0.832	5	1	1	1	3	3	2						
	0.82	1	3	3	1	3	2	1	2	3				
	0.818	2	1	1	3	1	1	1	1	1				
E	0.816	3	1	3	1	3	2	2	2	1				
	0.814	4	2	2	2	1	2	3	1	2				
	0.812	5	3	1	3	1	2	3	3	2				
	0.8	1	1	2	2	1	3	3	3	3	2	3	3	3
	0.798	2	3	3	1	2	3	3	2	3	2	1	3	3
F	0.796	3	1	3	1	1	3	1	3	3	3	1	1	1
	0.794	4	2	3	1	2	3	3	3	2	1	1	1	1
	0.792	5	3	3	1	1	1	2	3	1	1	2	1	3

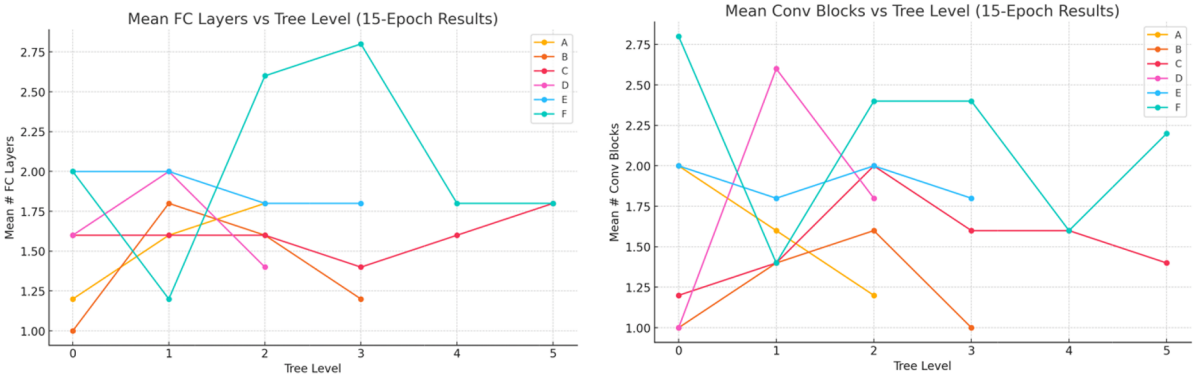
Figure 1. Table, depicting the top 5 results for each run, measured by validation accuracy after 15 epochs

Taken together, the accuracies fall in the narrow band 0.78 – 0.84, confirming that all architectures are viable classifiers and that subsequent analyses can focus on node complexity rather than gross performance differences.

Complexity profiles visualized

Figure 2 (left, 1) plots the mean number of convolutional blocks per level ($\bar{b}_\ell$) for every run, while Figure 2 (right, 2) shows the analogous curve for fully-connected layers ($\bar{f}_\ell$). Shaded markers denote tree depth. Below are the observations we draw from visually examining the plots:

- The curves are essentially flat; no run exhibits a consistent downward slope.
- Local fluctuations are noise-like, reflecting the stochastic nature of NAS sampling rather than a systematic preference for simpler nodes at deeper levels.
- Runs with the larger knob range (D–F) unsurprisingly have higher absolute means, yet their level-wise profiles are again flat.



1 *2*

Figure 2. Plots showing the mean FC layers (*1*) and convolution blocks (*2*) vs tree level

Visually, therefore, the data do not support the conjectured monotonic decrease in node complexity.

Statistical test of the monotonic-simplicity hypothesis

To formalize the visual impression, we apply Spearman's rank correlation coefficient ρ, a non-parametric measure of monotone association between two variables [10]:

$$c = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)},$$

$$d_i = \mathrm{rank}(x_i) - \mathrm{rank}(y_i),$$

where x_i is the tree level and y_i is the corresponding complexity metric (either b_ℓ or f_ℓ), and n is the total number of node observations pooled across runs.

We define the null hypothesis and the alternative hypothesis as follows:

- Null hypothesis H_0: c = 0 (no monotonic relationship between level and complexity).
- Alternative hypothesis H_1: c < 0 (complexity decreases with depth).

Using the pooled dataset, we obtain:

Table 2. **The numeric results of the Spearman's rank-correlation test**

Metric	ρ	p-value
Conv-blocks b_ℓ	+0.03	0.75
FC-layers f_ℓ	+0.11	0.21

Because both p-values are $\gg 0.05$, we fail to reject H_0. Moreover, the positive signs of c, although small, run counter to H_1, reinforcing the qualitative conclusion: there is no statistical evidence that node complexity diminishes with tree depth under the examined search space and training regime.

Together, the near-identical accuracies, flat complexity profiles, and non-significant Spearman coefficients collectively refute the monotonic-simplicity hypothesis for neural decision trees on CIFAR-10, at least within the constraints of the present experimental design.

Hypothesis Evaluation

The central hypothesis under investigation posited that, in neural decision trees, node complexity, measured by the number of convolutional blocks b_ℓ and fully connected layers f_ℓ, would decrease monotonically with increasing tree depth. In other words, deeper split nodes should require fewer layers to achieve comparable classification accuracy.

However, the empirical evidence fails to support this conjecture. As shown in the previous section, the mean complexity profiles $\bar{b}_\ell$ and $\bar{f}_\ell$ are essentially constant across levels, without any discernible downward trend. The formal Spearman rank test further confirms that the observed correlations are small and positive ($\rho \approx +0.03$ for b_ℓ, $c \approx +0.11$ for $\bar{f}_\ell$) and statistically non-significant ($p \gg 0.05$), leading us to retain the null hypothesis of no monotonic association.

Several factors may explain this departure from the original expectation of Roy and Todorovic. First, their depth-10 architecture was tailored to monocular depth estimation, a regression task with spatial continuity, whereas our CIFAR-10 classification problem may impose different representational requirements at all tree levels. Second, our fixed 256-channel convolutional blocks and limited training budget (15 epochs plus early stopping) may attenuate any subtle benefits of reduced complexity in lower nodes. Finally, NAS optimizes for overall accuracy, not explicitly for per-node efficiency, so it may favor uniformly expressive nodes to maximize global performance under the given constraints.

In sum, within the confines of our search space and training regime, there is no evidence that node complexity in neural decision trees decreases with depth. This negative result suggests that the "easier-to-learn" assumption articulated by Roy and Todorovic does not generalize straightforwardly from depth estimation to image classification, or that more tailored search strategies are required to expose such a trend.

Conclusion

This thesis provided an empirical evaluation of a critical architectural hypothesis within Neural Decision Trees (NDTs), specifically the claim by Roy and Todorovic (2016) that nodes deeper within the tree require

progressively simpler neural structures. Employing Neural Architecture Search (NAS) as an unbiased methodological tool, we systematically explored variations in node complexity, measured by the number of convolutional blocks and fully connected layers, across multiple tree depths using the CIFAR-10 image classification dataset.

Our comprehensive experiments and subsequent statistical analyses revealed no support for the monotonic simplicity hypothesis. Contrary to expectations, node complexity remained effectively constant across all tree levels, and no statistically significant correlation was observed between node depth and complexity. This finding suggests that architectural assumptions derived from specialized tasks, such as monocular depth estimation, may not generalize across different domains and datasets, underscoring the importance of empirically validating theoretical claims in neural architecture research.

While our experiments provide strong evidence refuting the generalized hypothesis, several limitations remain. The search space, though carefully chosen, was constrained by practical computational considerations, including limited training epochs and fixed convolutional channel widths. Future research could explore expanded or more nuanced search spaces, including varying parameters per node rather than per level, different datasets and tasks, and incorporating more advanced NAS strategies or evaluation metrics that explicitly target node-level complexity.

In summary, this study underscores the necessity for rigorous empirical testing of architectural assumptions in neural network research. The negative result regarding node simplicity in NDTs provides important insights for future architecture design and highlights the critical role of NAS methodologies in facilitating unbiased, systematic explorations.

Список літератури

1. A comprehensive survey of neural architecture search: Challenges and solutions [Electronic resource] / P. Ren et al. // ACM Computing Surveys (CSUR). — 2021. — Vol. 54, no. 4. — Pp. 1–34. — Mode of access: https://doi.org/10.1145/3447582.
2. Deep neural decision forests [Electronic resource] / P. Kontschieder et al. // Proceedings of the IEEE international conference on computer vision. — 2015. — Pp. 1467–1475. — Mode of access: https://openaccess.thecvf.com/content_iccv_2015/papers/Kontschieder_Deep_Neural_Decision_ICCV_2015_paper.pdf.
3. Elsken T. Neural architecture search: A survey [Electronic resource] / T. Elsken, J. H. Metzen , F. Hutter // Journal of Machine Learning Research. — 2019. — Vol. 20, no. 55. — Pp. 1–21. — Mode of access: https://www.jmlr.org/papers/volume20/18-598/18-598.pdf.
4. He X. AutoML: A survey of the state-of-the-art [Electronic resource] / X. He, K. Zhao, X. Chu // Knowledge-based systems. — 2021. — Vol. 212. — Pp. 106622. — Mode of access:https://doi.org/10.1016/j.knosys.2020.106622.
5. Hutter F. Automated machine learning: methods, systems, challenges [Electronic resource] / F. Hutter, L. Kotthoff, J. Vanschoren. — Springer Nature, 2019. — Mode of access: https://doi.org/10.1007/978-3-030-05318-5.
6. Krizhevsky A. Learning multiple layers of features from tiny images [Electronic resource] / A. Krizhevsky, G. Hinton, et al. — 2009. — Mode of access: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.
7. NNI: Neural Network Intelligence [Electronic resource] / H. Liu et al. — 2019. — Mode of access: https://github.com/microsoft/nni.
8. Pytorch: An imperative style, high-performance deep learning library [Electronic resource] / A. Paszke et al. // Advances in neural information processing systems. — 2019. — Vol. 32. — Mode of access: https://pytorch.org.
9. Roy A. Monocular depth estimation using neural regression forest [Electronic resource] / A. Roy, S. Todorovic // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — Pp. 5506–5514. — Mode of access:https://doi.org/10.1109/cvpr.2016.594.
10. Spearman C. The proof and measurement of association between two things [Electronic resource] / C. Spearman // International journal of epidemiology. — 2010. — Vol. 39, no. 5. — Pp. 1137–1150. — Mode of access:https://doi.org/10.1093/ije/dyq191.

References

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, *20* (55), 1–21.

He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. *Knowledge-based systems*, *212*, 106622.

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated machine learning: methods, systems, challenges*. Springer Nature.

Kontschieder, P., Fiterau, M., Criminisi, A., & Bulo, S. R. (2015). Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision* (pp. 1467–1475).

Krizhevsky, A., Hinton, G., & others. (2009). *Learning multiple layers of features from tiny images*.

Liu, H., Li, Y., Shen, Y., Zhao, D., & Xie, C. (2019). *NNI: Neural Network Intelligence*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & others. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., & Wang, X. (2021). A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, *54* (4), 1–34.

Roy, A., & Todorovic, S. (2016). Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5506–5514).

Spearman, C. (2010). The proof and measurement of association between two things. *International journal of epidemiology*, *39* (5), 1137–1150.

Микитишин А. П., Швай Н. О.

ВАЛІДАЦІЯ АРХІТЕКТУРНИХ ГІПОТЕЗ У НЕЙРОННИХ ДЕРЕВАХ РІШЕНЬ ЗА ДОПОМОГОЮ ПОШУКУ НЕЙРОННИХ АРХІТЕКТУР

У цій роботі запропоновано автоматизовану та об'єктивну методику перевірки архітектурних гіпотез за допомогою пошуку нейронних архітектур (Neural Architecture Search, NAS). Основна ідея полягає в застосуванні NAS як інструменту для оцінки теоретичних припущень щодо структури моделей без ручного налаштування архітектур або впливу суб'єктивних рішень дослідника. Для демонстрації підходу було перевірено гіпотезу про те, що складність вузлів у нейронних деревах рішень (Neural Decision Trees, NDTs) зменшується зі збільшенням глибини дерева. Це припущення зустрічається в науковій літературі та використовується як обґрунтування для побудови спеціалізованих архітектур, однак раніше не було перевірене на систематичній експериментальній основі.

У межах дослідження було розроблено повністю автоматизований експериментальний фреймворк для генерації, навчання та оцінювання сотень архітектур NDT з різними конфігураціями вузлів. Для пошуку ефективної структури дерев було використано метод баєсівської оптимізації (Tree-structured Parzen Estimator, TPE). Складність вузлів оцінювали за кількома метриками: кількістю параметрів, кількістю обчислювальних операцій, кількістю нейронів у шарі та глибиною шару. Для аналізу зв'язку між глибиною вузлів і їхньою складністю застосовували коефіцієнт рангової кореляції Спірмена (Spearman's rank correlation coefficient).

За результатами обчислювального експерименту, що охопив понад 300 згенерованих моделей на синтетичному класифікаційному датасеті, не було виявлено жодної стабільної або статистично значущої залежності між глибиною вузла та його складністю. Отримані результати свідчать про те, що припущення, сформовані на основі окремих прикладів або інтуїції, можуть не узагальнюватися на інші задачі або домени. Це підкреслює важливість емпіричної перевірки теоретичних архітектурних міркувань, а також необхідність уважного проєктування простору пошуку в NAS.

Запропонований підхід може бути використаний для перевірки інших архітектурних гіпотез у різноманітних типах нейронних мереж, що робить його перспективним інструментом у дослідженнях у сфері автоматизованого машинного навчання.

Ключові слова: пошук нейронних архітектур, нейронні дерева рішень, автоматизоване машинне навчання, комп'ютерний зір, складність вузлів.

Матеріал надійшов 16.06.2025

УДК 004.8:004.93
DOI: 10.18523/2617-3808.2025.8.57-67

M. Mokryi, N. Shvai

ROBUSTNESS OF NEURAL DECISION TREES TO NOISE IN INPUT DATA FOR IMAGE CLASSIFICATION TASKS

Neural networks, particularly convolutional neural networks (CNNs), have demonstrated high effectiveness in image classification tasks. However, they are known to be vulnerable to input data perturbations and have weak interpretability due to their black-box nature. In contrast, traditional decision trees (DTs) provide transparent decision-making processes, but are limited to low-dimensional or tabular data, restricting their field of application in computer vision tasks such as image classification. To address this gap, a hybrid architecture known as Neural Decision Trees (NDTs) has emerged, combining strong generalization and learning capabilities of neural networks, with transparent hierarchical inference and interpretability of DTs.

The article investigates the robustness of NDTs to noise in input data for image classification tasks. Despite the extensive studies covering the robustness of both CNNs and traditional DTs against various forms of input perturbations, the robustness of NDT models remains a largely underexplored area. This study provides two robust training methods to improve robustness: constant noise learning and incremental noise learning, originally developed for CNNs, but which can be effectively applied to NDT-based architectures and significantly improve the robustness to noisy images for models. These methods involve adding perturbed samples via a Gaussian blur during the training stage. The noisy test set consists of images perturbed by a Gaussian blur and is used to evaluate the robustness performance.

A series of experiments were conducted on the CIFAR-10 dataset using the original training baseline and robust training methods. The results demonstrate that constant and incremental noise learning significantly improve the robustness of all tested NDT models to noisy images compared to their original training performance. While the ResNet18 baseline model demonstrates higher overall performance, the NDT models show comparable robustness improvements using the proposed robust training strategies. Constant noise learning offered an adjustable trade-off between performance on clean and noisy images, while incremental noise learning provided a more stable training process. The first method is considered preferable due to the simplicity of implementation.

This study empirically confirms that NDT models can effectively use methods adapted from CNNs to improve their robustness against perturbations in input data. An NDT framework was developed to conduct training and validation using a standardized shared pipeline. It is available via the link: github.com/MikhailoMokryy/NDTFramework.

Keywords: Neural Decision Trees, machine learning, robustness, image perturbations, image classification, computer vision, convolutional neural networks.

Introduction

The field of computer vision, particularly image classification tasks, has advanced significantly with the introduction of convolutional neural networks (CNNs), which demonstrate remarkable generalization abilities for high-dimensional input data. However, they are not ideal, suffer from a lack of interpretability due to their black-box architecture, and are vulnerable to perturbations in input data [3, 7, 9, 14]. Such images can create misclassifications by a neural network (NN), which leads to poor performance. The robustness of traditional decision trees (DTs) to noise has been studied only for low-dimensional or tabular data due to their limited field of application. To extend this field, a hybrid architecture referred to as Neural Decision Trees (NDTs) has been proposed [1, 4, 6, 8, 11, 13]. The NDT model combines two distinct architectures – CNN and DTs, resulting in a tree-based architecture capable of solving image classification tasks by combining a strong generalization ability acquired from CNNs with the interpretability of a DT hierarchy. While the robustness of CNNs and DTs to noise has been extensively studied, NDT models remain largely under-

explored. This study aims to investigate methods that can be incorporated to improve the robustness of NDTs.

Previous efforts to solve the challenge of improving the robustness of NNs, including defensive strategies, have drawbacks and do not cover all possible input perturbation forms. Robust image classification, focused on real-world alterations such as hardware defects or environmental distortions, uses different training strategies, including constant noise learning, incremental noise learning, and architecture modification [14]. Robust training methods for traditional DT models are also proposed, but they primarily target low-dimensional or tabular data and do not cover the field of image classification tasks [2, 12, 16]. The robustness of existing NDT models to input data noise is largely unexplored. Given the limited understanding of NDT robustness to noisy images, this study investigates whether learning strategies developed to improve the robustness of CNN models can be effectively applied to NDT-based architectures and demonstrate comparable robustness improvement.

To validate the proposed approach, a series of experiments were conducted on the CIFAR-10 dataset, which is a widely used benchmark for image classification tasks. Image perturbations are created by applying a Gaussian blur to input samples. These perturbed images are used both to evaluate model robustness in experiments and as a part of the robust training process. The experiments involve evaluating the robustness of various models using the original training baseline, constant noise learning, and incremental noise learning strategies. The constant noise learning method incorporates a fixed proportion of perturbed images during the training phase, with noise applied at varying probabilities: 0.05, 0.1, 0.2, 0.4, 0.6, 0.8. In contrast, the incremental noise learning approach gradually increases the proportion of noisy images during the training phase, starting from a low initial level, eventually almost reaching the size of the original training set in the final stage of training. In total, 40 models were trained under various conditions to evaluate their performance on clean and noisy test sets. The trained models include NDT models: Deep Neural Decision Tree, Deep Neural Decision Forest, and Neural Backed Decision Trees variations with hard and soft inferences, as well as the baseline ResNet18 model for comparison.

Main contributions of this study:

1. A comprehensive investigation of the robustness of NDT models for image classification tasks, focusing on the impact of perturbed images on these models' performance.
2. Demonstration that methods to improve robustness, originally developed for CNNs, specifically constant and incremental noise learning, can be successfully applied and significantly improve the robustness of NDT models against input perturbations.
3. Empirically determined these findings through a series of experiments under various conditions on the CIFAR-10 dataset, providing detailed performance metrics including model accuracy on clean and noisy test sets, accuracy drop, and robustness gain.

The rest of the paper is organized as follows: the Related Work section provides a general literature overview of NDT models and studies about improving robustness to noisy input samples for CNNs, traditional DTs, and the current state of NDT research. The Methodology section describes the NDT architectures and strategies to improve robustness. The Experiments section outlines the CIFAR-10 dataset, the used models, including their hyperparameters, and the overall training workflow. In the Results and Discussion section, results of experiments are presented in tabular and graphical form, with a detailed comparison between models and methods to improve robustness. Finally, the Conclusion section provides a comprehensive summary of this study.

Related Work

Early attempts to combine neural networks and decision trees, known as Hierarchical Mixtures of Experts (HMoE), were introduced by Jordan and Jacobs [10]. The HMoE architecture has a routing function: a linear classifier in each tree node, which decides where to send an input sample: to the left or right branch, passing it down a fixed tree structure. A more advanced, Soft Decision Trees (SDT), the base part of many NDT architectures, which is a fuzzy DT used for classification and regression tasks, appeared in Suarez and Lutsko [15] work. It is built with consideration of data partial membership in the tree nodes that form the tree structure. It was a key part in the future development of NDTs, allowing the use of the back-propagation method for training models. The next significant improvement of NDT was made by Kontschieder et al. [4] by enhancing soft DTs with an updated routing function in each node that contains a neural linear layer and sigmoid activation function. Their neural decision forest model (dNDF) is an ensemble of DTs in which the

whole CNN architecture, excluding the final linear layer, is used as the root transformer, which extracts features and passes them to tree-structured classifiers. Another vision of NDTs development presented tree-like structures of NNs with a routing mechanism. Ioannou et al. [6] introduced a Conditional Network model that reduces the computational load and the number of CNN architecture parameters by distributing computations through a hierarchical structure: a directed acyclic graph. This model uses an MLP-based route and achieves the same level of accuracy on image classification tasks with lower computational cost. Frosst et al. [8] utilized a NN to extract knowledge from it and use it in the SDT model training process. Thus, a NN provides a more informative soft target for training. The soft DT has learning filters to make hierarchical decisions, based on input targets in every internal node and a static probability distribution over the output classes for every leaf node. Tanno et al. [1] proposed adding adaptive architecture growth support to NDT, a feature of DTs. The Adaptive Neural Trees (ANT) model is constructed using a greedy algorithm that selects the best option between increasing the tree's depth and partitioning the input space before the model training phase. Unlike previous works, the Neural-Backed Decision Trees (NBDT) model introduced by Wan et al. [11] addresses the limitation of NDT models: the trade-off between accuracy and interpretability. It employs a WordNet lexical database to assign a selected concept to each tree node to improve model interpretability by labeling. NBDTs replace the final linear layer of the NN with a differentiable sequence of decisions and uses a hierarchical tree loss for model training. A novel approach to learning trees from deep NN (DNN) architecture, called Self-born Wiring (SeBoW), was proposed by Chen et al. [13]. It extended the ANT architecture growth ideas by using self-born neural trees that evolve themselves from a user-designed mother DNN architecture, instead of growing trees progressively or by using greedy algorithms. This self-born learning procedure allows for global-level tree-architecture parameter optimization over the neural tree search.

The generation of perturbed images, designed to cause a misclassification in a NN, has been extensively studied. These images are commonly referred to as adversarial examples. The early foundational work on adversarial example misclassification was covered by Goodfellow et al. [9]. Their work revealed that the linear nature of NNs is one of the main reasons why NNs are prone to adversarial examples. Input data is made by adding a selected adversarial perturbation to an original sample. Subsequent research by Papernot et al. [7] introduced a defense mechanism called defensive distillation to reduce the negative impact of adversarial examples on image classification, highlighting the fundamental nature of NN vulnerabilities. Carlini and Wagner [3] demonstrated that defensive distillation has some drawbacks and does not cover all adversarial examples, and created a set of attacks that can be used to improve the robustness of NNs. Stock et al. [14] proposed comprehensive strategies for robust image classification, examining defensive training techniques and architecture modifications that improve robustness to noisy input data for NN models. Their study focuses on real-world alterations that occur when an image is captured and can be caused by hardware defects or environmental distortions. Different techniques were used for altering input data to create digital augmentations, such as single-pixel modification, noise, and blur.

Research on adversarial examples and model robustness has been extensively conducted for linear models and NNs. However, the impact of adversarial examples on tree-based model robustness remains poorly studied. Unlike NNs, tree-based models are not differentiable, have a hierarchical structure, and are interpretable due to their nature, which can lead to the assumption that they are more robust than CNNs. However, Chen et al. [12] show that tree-based models can also struggle against adversarial examples. They investigate the robustness of tree-based models and the impact of adversarial examples on both classical DTs and more advanced ensemble boosting methods. A novel robust DT training framework based on a robust splitting function was proposed to improve robustness. Further ideas were presented by Andriushchenko and Hein [2]. This paper identified a drawback of the previously proposed method: a lack of robustness guarantee. The authors proposed robust training methods that achieve a provable robustness for boosted trees, and the results are comparable with CNN-based methods. Vos and Verwer [16] proposed a novel method for training robust DTs called growing robust trees, or GROOT for short, which adds a parameter to control a trade-off between model accuracy and robustness against adversarial examples.

All existing tree-based methods that improve robustness focus primarily on low-dimensional and tabular data. NDT robustness against adversarial examples remains underexplored. To address this gap, this study aims to investigate the impact of images perturbed by noise on the robustness of NDT models and determine whether robust training methods originally developed for CNN models can be effectively applied to NDTs, demonstrating improvements in robustness.

Methodology

DTs are a tree-structured machine learning method with internal decision nodes and prediction nodes, known as leaf nodes, used for low-dimensional or tabular data for classification or regression tasks. In contrast, NNs, including CNNs, are powerful models with strong generalization capabilities, widely used for computer vision tasks involving image classification. NDT models aim to combine these two distinct architectures to provide high performance on high-dimensional data, strong generalization and learning features, abilities obtained from NNs, with interpretability and transparent inference of a DT hierarchy. To achieve this, NNs should be integrated into a tree structure by implementing a differentiable routing function that controls how samples traverse down the tree. It unlocks the usage of gradient descent-based optimization methods with back-propagation during the training stage.

Generic NDT can be divided into three main modules:

1. *Router*. This decision module is responsible for sending input data to the child nodes. Each internal (decision) tree node contains a router module, which performs a routing function and partitions the input space. This mandatory module of NDTs is closely related to the reasoning mechanism of the model.
2. *Solver*. This prediction module is essential in NDT architecture. Each leaf (prediction) node contains a solver module that produces the outcomes. Depending on the implementation, it can provide a final output of both the class hierarchy and the static probability distribution over these classes.
3. *Learner*. A transformer module is assigned to every edge of the tree. The learner module transforms input samples from the parent node and passes them down to the child nodes. However, it is an optional module in NDTs. While some architectures, like ANT, incorporate it for representation learning, most NDTs use an identity function, passing features down the tree without data transformation.

Different NDT models are used in this paper to evaluate robustness against noise in input data, alongside a baseline ResNet18 model [5] for comparison.

Deep Neural Decision Forest (dNDF) is the first model considered for evaluating robustness to noise in input data. The model structure is quite straightforward. It uses a CNN architecture without the last fully connected linear layer to extract features, while a decision forest produces final predictions. The representations obtained from the trained CNN are passed to an ensemble of DTs with soft inference. These representations provide routing functions for all nodes in the forest of trees.

Unlike a standard decision forest with binary and deterministic routing, the dNDF model uses probabilistic routing. Routing functions, which determine the routing direction decisions for each internal node, are an output of Bernoulli random variables and are defined by a sigmoid activation function. When a sample traverses down a tree to a leaf node, the corresponding tree predictor gives the distribution over output classes. With this stochastic routing, each leaf node predictor provides a result averaged according to the probability of a sample reaching a leaf. The final prediction from a single tree is computed as a sum over all leaves of the learned class distribution multiplied by its routing probability. Then, the final dNDF model prediction for a sample is obtained by averaging the predictions of each tree in the ensemble. A learning procedure requires estimating both internal node parameters, responsible for decisions, and the leaf node parameters, responsible for the output predictions. The dNDF model uses a log-loss function and a two-step optimization strategy. Internal node parameters are randomly initialized and iterated during the learning procedure for a predefined number of training epochs. During each training epoch, the prediction parameters of all leaf nodes are updated independently for each tree by retrieving the current parameters from internal nodes, using a specific iterative scheme that solves a convex optimization problem.

Next, the training set is split into random mini-batches. The Stochastic Gradient Descent (SGD) optimization algorithm updates internal node parameters for each mini-batch. The original model structure is designed as an ensemble of DTs with soft inference, but it can also be converted into a single DT. This model is referred to as Deep Neural Decision Tree (dNDT). It has an identical structure but uses one tree instead of an ensemble of DTs.

At first glance, the next model structure looks similar to the previous model, utilizing a ResNet18 architecture without the final layer, which is replaced by a DT. In addition, much attention is given to model interpretability.

NBDT model implementation employs a differentiable oblique DT and incorporates several key design choices:

1. *Path probabilities for inference.* They help the model to tolerate highly uncertain intermediate decisions.
2. *Induced hierarchies*. Hierarchies are built from pre-trained NN weights to decrease the impact of overfitting.
3. *Tree Supervision Loss*. Training with surrogate hierarchy loss helps the model to make significantly better high-level decisions, leading to improved model generalization and performance.
4. *Pure leaves*. Each leaf corresponds to one pure class, allowing the model to select one path from the root to the leaf.

Similar to the dNDF implementation, the learning procedure shares many similarities. Input samples are transformed into feature representations from the ResNet18 NN backbone. After the feature extraction process, a final fully-connected layer is replaced by an oblique DT with soft inference. For each input sample and tree node, the probability of traversing to each child is computed by applying a softmax function of the inner product between the extracted features and the child node weight. The path probability for a leaf node representing a selected class is computed as a product of the traversal probabilities of each node along the unique path from the root to that leaf.

The next step is to build an induced hierarchy, which is essential for the NBDT model. Using data-based hierarchies like information gain or existing hierarchies like WordNet has notable drawbacks. The former is prone to input data overfitting, and the latter emphasizes conceptual rather than visual similarities.

NBDT is constructed using the hierarchy derived from pre-trained model weights to address this limitation. The process of the induced hierarchy creation starts with the final fully-connected layer weights from the ResNet18 backbone, viewing each row vector as a class representation. Next, the hierarchical agglomerative clustering is performed on normalized class representatives, iteratively pairing tree nodes and groups of nodes. For leaf nodes, the weights are normalized row vectors, while for internal nodes, the weights are the average of the weights of all leaf nodes within the corresponding subtree.

After the induced hierarchy generation, the decision nodes are labeled using the WordNet lexical database hierarchy of nouns. The earliest common ancestor for all leaf nodes in a subtree is identified to assign a corresponding WordNet noun to an internal node.

A hierarchical loss named Tree Supervision Loss is proposed to improve the NBDT training process. Being a modified cross-entropy loss, it is calculated over the class distribution of path probabilities. The total model loss is a weighted sum of the original cross-entropy loss and the Tree Supervision Loss. Tree Supervision Loss includes two variants: the Hard Tree Supervision Loss, which applies a cross-entropy at each node, and the Soft Tree Supervision Loss, which computes a cross-entropy loss over the distribution of leaf probabilities. Accordingly, based on the Tree Supervision Loss function variant, the NBDT inference can be either soft or hard. Despite hard inference being more intuitive and improving model interpretability, starting at the root node, each sample is sent to the child node with the most similar representative and traverses down the tree until a leaf node is reached. In the original paper, an NBDT model with hard inference, referred to as Hard NBDT, underperforms the NBDT model with soft inference. It was decided to evaluate both models' robustness to input perturbations.

The Gaussian blur method is used to create image perturbations. It is based on 2D Gaussian filtering, achieved by shifting a kernel filter over the image and performing a convolution based on the kernel size for each pixel in the image, without changing its dimensions. The kernel size depends on the value of σ, the standard deviation of the distribution. As σ increases, the kernel size increases accordingly, resulting in a more blurry image. The Gaussian function with (x, y) coordinates relative to the kernel center is shown below:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Constant noise learning is a straightforward method for training models that uses a predefined part of the perturbed data during training. The perturbed data remains fixed during training, generalizing from the original data with a certain proportion of modified samples during representation learning. Images perturbed by a Gaussian blur are used, with a certain part of the noisy images selected for each experiment. This method aims to achieve both high performance on clean images and noisy images, based on a proportion of perturbed samples in a training set. With an adjustable noise application probability, the constant noise learning method can significantly enhance the robustness of NDTs and the ResNet18 model to input noise, while remaining simple to integrate into the model training pipeline. More details are provided in Algorithm 1.

Algorithm 1 Constant noise learning

Require: Training set T, probability of applying noise p, $nEpochs$

for all $i \in \{1, \ldots, nEpochs\}$ **do**
  Shuffle T and divide into mini-batches $\{B_j\}$
  for each mini-batch B_j **do**
    for each image x in B_j **do**
      $r \sim \mathcal{U}(0, 1)$
      if $r < p$ **then**
        $x \leftarrow \text{GaussianBlur}(x)$
      end if
    end for
    Compute loss using current batch B_j
    Update model parameters
  end for
end for

Algorithm 1. Constant noise learning algorithm

The incremental noise learning method is a more complex method for improving the model's robustness. This strategy gradually adds perturbed data to the training set, progressively increasing the proportion with each training epoch. Training begins with a small proportion of perturbed data, and in the final stage of training, the proportion of perturbed data can reach almost the same size as the original training set. The proportion of noisy images to be added is computed based on the current epoch number and batch size, while the set of noisy images from previous epochs remain unaltered. It is assumed that this method is more effective in improving the robustness of NDT models, as it gradually introduces perturbed images based on the current stage of training, which can help the model learn sample features more naturally and generalize them better. A detailed implementation is provided in Algorithm 2.

Algorithm 2 Incremental noise learning

Require: Training set T, indices set S, initial noise percentage p_0, max noise percentage p_{max}, $nEpochs$

Initialize noisy indices set $S \leftarrow \emptyset$
Compute noise increment $\delta = \frac{p_{max} - p_0}{nEpochs - 1}$
for all $i \in \{1, \ldots, nEpochs\}$ **do**
  Compute target noise percentage $p_i = \min(p_0 + (i-1)\delta,\ p_{max})$
  Get number of new noisy indices $k = \lfloor p_i \cdot |T| \rfloor$
  Select random indices from non-noisy samples $M \leftarrow k - |S|$
  Update $S \leftarrow S \cup M$
  Shuffle T and divide into mini-batches $\{B_j\}$
  for each mini-batch B_j **do**
    for each image x with index idx in B_j **do**
      if $idx \in S$ **then**
        $x \leftarrow \text{GaussianBlur}(x)$
      end if
    end for
    Compute loss using current batch B_j
    Update model parameters
  end for
end for

Algorithm 2. Incremental noise learning algorithm

Experiments

The experiments are conducted on the CIFAR-10 dataset. It contains 60,000 images, including a training set of 50,000 images and a validation set of 1,000 images. The dataset is labeled with 10 different classes, with an equal number of images per class. Each sample is a 3-channel RGB image with square dimensions of 32 pixels. The CIFAR-10 dataset is widely used to validate computer vision tasks, particularly image

classification. It is well-suited for the needs of this study and provides sufficient input data complexity, making it relevant for robustness experiments.

Common hyperparameters are used for training and shared across all models. The number of training epochs is set to 40. A fixed random seed of 1 is applied to make all experiments more consistent and to provide fair results. A Gaussian blur σ value is set to 0.8, with the corresponding kernel filter size of 5 for image perturbation. An example of a perturbed image with Gaussian blur and clean images is demonstrated in Fig. 1.

Figure 1. Clean and noisy images from CIFAR-10

Training hyperparameters, including learning rate, input batch size, and others, vary according to the model.

Both dNDF and dNDT models use a batch size of 64. The tree structure has a depth of 8. In the case of the dNDF model, the ensemble contains 20 trees. This implementation employs the Adam optimization algorithm instead of the original SGD optimization used in the paper, with weight decay of 1^{-5} and a learning rate of 1^{-4}. For training models, a negative log-likelihood loss is applied. As a NN module, a 3-block CNN is used, each composed of 2 convolutional layers and batch normalization layers, the second layer in each block is followed by max pooling and dropout.

A batch size of 128 is used for training NBDTs and ResNet18 models. Two types of NBDT models are used in experiments: one with soft inference and another with hard inference.

Although NBDTs and ResNet18 models share the same ResNet18 backbone architecture and have similar training pipelines, their loss functions are different.

A NBDT model (with soft inference) uses a Soft Tree Supervision Loss during training, Hard NBDT uses a Hard Tree Supervision Loss, and ResNet18 uses a cross-entropy loss widely used for classification tasks. The NBDT, Hard NBDT, and ResNet18 models are trained with the SGD optimization algorithm with 0.9 momentum, 5^{-4} weight decay, and starting learning rate of 0.1 with a cosine annealing schedule where a maximum number of iterations is set to total epochs.

The constant and incremental noise learning training pipeline includes several new hyperparameters. Constant noise learning experiments were conducted by applying perturbed images to the original training set with varying probabilities denoted as ρ. The values of p are set to 0.05, 0.1, 0.2, 0.4, 0.6, and 0.8. For incremental noise learning experiments, the training process is configured to start with an initial noise level of 5 %, which gradually increases to a maximum of 95 % on the final training epoch.

Overall, 5 selected models proceeded through 8 learning procedures, resulting in 40 models trained under varying conditions on the CIFAR-10 dataset. Each model was trained using a regular training method with the original training set, a constant noise learning method with 6 different noise application probabilities, and an incremental noise learning method.

Experimental results are presented in a subsequent section.

Results and Discussion

The results of constant noise learning and the original training baseline are presented in Table 1, Table 2, and Table 3, evaluating the effect of the robust training strategy on model performance on the image classification tasks with validation on both clean and noisy images.

In Table 1, the performance of the model trained using the original baseline is compared against robustness to Gaussian blur. The performance is evaluated on both the original CIFAR-10 test set, which consists of clean images, as well as a noisy test set, containing images perturbed by Gaussian blur. The evaluation metrics are referred to as clean accuracy and noise accuracy, respectively. Additionally, the accuracy drop evaluation metric is used to demonstrate the performance drop on images with noise. Each model shows a significant performance degradation on the Gaussian blur test set, with an accuracy drop from 27.81% on the dNDT model to 36.86% on NBDT. The difference between NBDT and Hard NBDT models is also notable at 5.34%, indicating that NBDT with hard inference is less affected by noisy images than NBDT with soft inference. The original training baseline performance results are compared to robust training methods that should improve robustness across all models.

Table 1. **Original training baseline performance**

Model	Clean accuracy	Noise accuracy	Accuracy drop
dNDT	85.92%	58.11%	-27.81%
dNDF	86.21%	58.09%	-28.12%
ResNet18	93.75%	60.63%	-33.12%
NBDT	93.48%	56.62%	-36.86%
Hard NBDT	93.94	62.42%	-31.52%

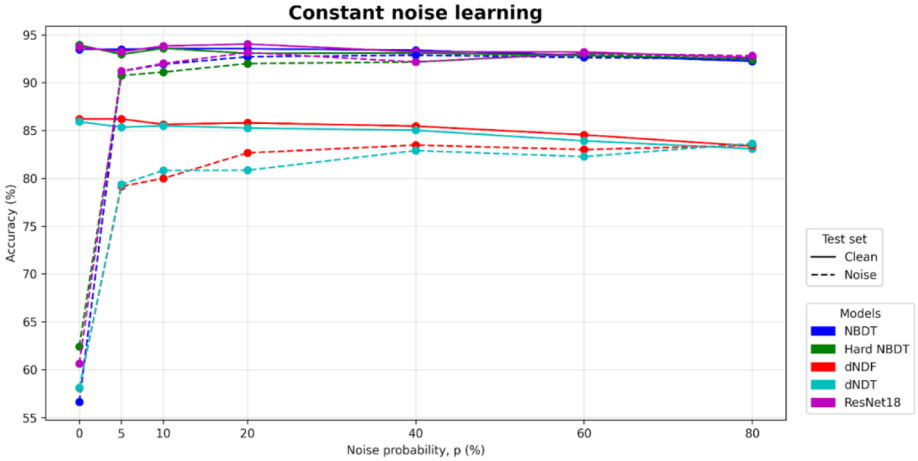


Figure 2. Constant noise learning performance on models

The results of the constant noise learning method using various application probabilities of perturbed images ρ to improve robustness, are presented in Table 2 and Table 3. Table 2 demonstrates the performance of each model on both clean and noisy images. Based on these results, Fig. 2 visualizes a performance change of each model depending on the noise probability ρ.

Table 2. **Constant noise learning performance**

Model	Acc.	p=0.05	p=0.1	p=0.2	p=0.4	p=0.6	p=0.8
dNDT	Clean	85.35%	85.49%	85.26%	85.03%	83.92%	83.09%
dNDT	Noise	79.37%	80.83%	80.85%	82.90%	82.27%	83.64%
dNDF	Clean	86.21%	85.64%	85.80%	85.46%	84.55%	83.37%
dNDF	Noise	79.16%	80.00%	82.66%	83.48%	83.01%	83.44%
ResNet18	Clean	93.23%	93.83%	94.04%	93.23%	93.21%	92.62%
ResNet18	Noise	91.22%	92.01%	93.15%	92.51%	93.07%	92.81%
NBDT	Clean	93.49%	93.58%	93.55%	93.39%	92.85%	92.24%
NBDT	Noise	91.19%	91.92%	92.71%	92.88%	92.63%	92.47%
Hard NBDT	Clean	93.60%	92.96%	93.06%	93.10%	92.82%	92.46%
Hard NBDT	Noise	90.75%	91.11%	92.00%	92.16%	93.05%	92.41%

Table 3 presents the accuracy drop for each model at different ρ values, along with a robustness gain metric. This metric indicates the model accuracy change from baseline to robust training on a noisy test set.

Table 3. **Constant noise learning vs Original baseline performance**

Model	Metric	p=0.05	p=0.1	p=0.2	p=0.4	p=0.6	p=0.8
dNDT	Acc. Drop	-5.98%	-4.66%	-4.41%	-2.13%	-1.65%	+0.55%
dNDT	R. Gain	+21.83%	+23.15%	+23.40%	+25.68%	+26.16%	+28.36%
dNDF	Acc. Drop	-7.05%	-5.64%	-3.14%	-1.98%	-1.54%	+0.07%
dNDF	R. Gain	+21.07%	+22.48%	+24.98%	+26.14%	+26.58%	+28.19%
ResNet18	Acc. Drop	-2.01%	-1.82%	-0.89%	-0.72%	-0.14%	+0.19%
ResNet18	R. Gain	+31.11%	+31.30%	+32.23%	+32.40%	+32.98%	+33.31%
NBDT	Acc. Drop	-2.30%	-1.66%	-0.84%	-0.51%	-0.22%	+0.23%
NBDT	R. Gain	+34.56%	+35.20%	+36.02%	+36.35%	+36.64%	+37.09%
Hard NBDT	Acc. Drop	-2.85%	-1.85%	-1.06%	-0.94%	+0.23%	-0.05%
Hard NBDT	R. Gain	+28.67%	+29.67%	+30.46%	+30.58%	+31.75%	+31.47%

The dNDT and dNDF models show higher volatility on the noisy test set at lower ρ values. However, from a ρ value of 0.2, the performance drop becomes closer to NBDTs and ResNet18 models. Based on the performance of models trained with constant noise learning, NBDTs and ResNet18 models are less dependent on varying probabilities of noisy images. The optimal performance, showing high accuracy on both clean and noisy test sets, is achieved at ρ values of 0.2 and 0.4. The best robustness results are obtained with ρ values of 0.6 and 0.8, causing a slight decrease in performance on clean images. The robustness gain compared to the original baseline performance for each model is significant, even the smallest ρ of 0.05 provides a robustness gain from 21.83% on NDT to 34.56% on NBDT.

Incremental noise learning provides excellent robustness results with minimal performance drop on the clean test set for dNDT, ResNet18, and Hard NBDT models, and a small performance improvement for dNDF and NBDT models. A trade-off between accuracy on the clean and noisy test sets is minimal for every model. Robustness gain performance ranges from 27.40% on NDT to 37.0% on the NBDT model. Results are presented in Table 4.

Table 4. **Incremental noise learning vs Original baseline performance**

Model	Clean acc.	Noise acc.	Acc. Drop	Robust. Gain
dNDT	83.00%	82.59%	-0.41%	+27.40%
dNDF	83.86%	83.92%	+0.06%	+28.18%
ResNet18	93.25%	93.09%	-0.16%	+32.96%
NBDT	92.71%	92.84%	+0.13%	+37.00%
Hard NBDT	92.28%	92.23%	-0.05%	+31.47%

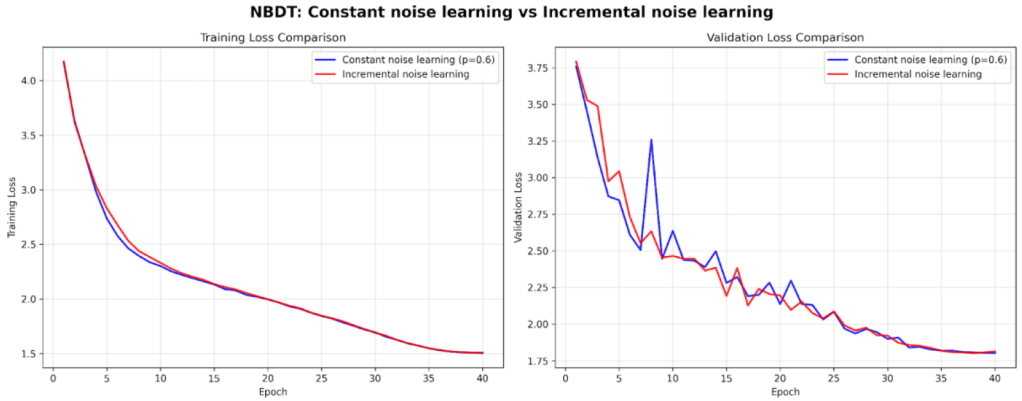


Figure 3. Loss changes of NBDT for constant and incremental noise learning

The constant learning method offers an adjustable trade-off between clean performance and robustness on noisy images, while incremental noise learning provides more stable results without relying on fine-tuning the application probability. Both methods can improve robustness to image perturbations without compromising performance on clean images. Although the dNDT and dNDF models have lower baseline performance, they achieve comparable robustness improvements using constant and incremental noise learning training strategies. For a detailed comparison of constant noise learning with a ρ value of 0.6 and

incremental noise learning, an NBDT model is selected. The training and validation loss for each training epoch for both approaches is demonstrated in Fig. 3.

Training losses are almost identical across 40 epochs, starting with very similar initial loss values, then steadily decreasing, showing a healthy training curve without overfitting. Validation losses have minor changes, indicating that the two models generalize differently. A constant noise learning method achieves slightly better final performance due to a lower loss with a notable spike at epoch 8, while incremental noise learning demonstrates a more stable curve with fewer sharp spikes. Thus, the NBDT model trained with incremental noise learning has a more stable validation loss progression. Although incremental noise learning demonstrates better stability during training and notable improvements, constant noise learning provides a simpler implementation, showing identical results. It is determined to be the preferred method for improving the robustness of the models to noise in input data.

Overall, a ResNet18 model shows slightly better results across all experiments. Moreover, it significantly outperforms dNDT and dNDF models in both training strategies, alongside the original training baseline. This confirms the assumption that, while the NDT models offer better interpretability, they involve some performance trade-offs. However, NDT models demonstrate comparable robustness improvements with robust training methods, demonstrating that they can benefit from the CNN-based method to improve robustness.

Conclusion

This work investigates the robustness of various NDT models, including dNDT, dNDF, NBDT, and Hard NBDT, to perturbations applied by a Gaussian blur in input data. The CIFAR-10 dataset is used for training models and validating experimental results. A ResNet18 model is used as a baseline for comparison with NDT models. Using the original training baseline, all models experienced a significant performance drop on images perturbed by a Gaussian blur. Two methods, originally developed to improve the robustness of CNN models, are applied to enhance robustness: constant noise learning and incremental noise learning. Experimental results demonstrate that both methods significantly improve robustness to noise for all models compared to the original model training baseline. Outcomes of the two robust training methods are nearly identical, showing no noticeable differences. Constant noise learning method offered an adjustable trade-off between performance on clean and perturbed images, while incremental noise learning provided more stable training results. However, constant noise learning is preferable for improving model robustness to noise in input data due to the simplicity of implementation. The experiments show that even a small part of noisy images significantly improves the robustness of the model.

Overall, NDT models demonstrate improvements in robustness to noise in input data comparable to the baseline ResNet18 model using both constant and incremental noise learning methods, indicating that NDTs can effectively benefit from adapted methods originally developed for CNNs.

Список літератури

1. Adaptive neural trees / Ryutaro Tanno [et al.] // International conference on machine learning. — 2019. — Pp. 6166–6175.
2. Andriushchenko M. Provably robust boosted decision stumps and trees against adversarial attacks / Maksym Andriushchenko, Matthias Hein // Advances in neural information processing systems. — 2019. — Vol. 32.
3. Carlini N. Towards evaluating the robustness of neural networks / Nicholas Carlini, David Wagner // 2017 IEEE symposium on security and privacy (SP). — 2017. — Pp. 39–57.
4. Deep neural decision forests / Peter Kontschieder [et al.] // IEEE international conference on computer vision. — 2015. — Pp. 1467–1475.
5. Deep residual learning for image recognition / Kaiming He [et al.] // IEEE conference on computer vision and pattern recognition. — 2016. — Pp. 770–778.
6. Decision forests, convolutional networks and the models in-between [Electronic resource] / Yani Ioannou [et al.] // ArXiv preprint arxiv:1603.01250. — 2016. — Mode of access: https://doi.org/10.48550/arXiv.1603.01250.
7. Distillation as a defense to adversarial perturbations against deep neural networks / Nicolas Papernot [et al.] // 2016 IEEE symposium on security and privacy (SP). — 2016. — Pp. 582–597.
8. Frosst N. Distilling a neural network into a soft decision tree [Electronic resource] / Nicholas Frosst, Geoffrey Hinton // ArXiv preprint arxiv:1711.09784. — 2017. — Mode of access: https://doi.org/10.48550/arXiv.1711.09784.
9. Goodfellow I. J. Explaining and harnessing adversarial examples [Electronic resource] / Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy // ArXiv preprint arxiv:1412.6572. — 2014. — Mode of access: https://doi.org/10.48550/arXiv.1412.6572.
10. Jordan M. I. Hierarchical mixtures of experts and the EM algorithm / Michael I. Jordan, Robert A. Jacobs // Neural computation. — 1994. — Vol. 6, no. 2. — Pp. 181–214.
11. NBDT: Neural-backed decision trees [Electronic resource] / Alvin Wan [et al.] // ArXiv preprint arxiv:2004.00221. — 2020. — Mode of access: https://doi.org/10.48550/arXiv.2004.00221.

12. Robust decision trees against adversarial examples / Hongge Chen [et al.] // International conference on machine learning. — 2019. — Pp. 1122–1131.
13. Self-born wiring for neural trees / Ying Chen [et al.] // Proceedings of the IEEE/CVF international conference on computer vision. — 2021. — Pp. 5047–5056.
14. Stock J. Strategies for Robust Image Classification [Electronic resource] / Jason Stock, Andy Dolan, Tom Cavey // ArXiv preprint arxiv:2004.03452. — 2020. — Mode of access: https://doi.org/10.48550/arXiv.2004.03452.
15. Suárez A. Globally optimal fuzzy decision trees for classification and regression / Alberto Suárez, James F. Lutsko // IEEE transactions on pattern analysis and machine intelligence. — 1999. — Vol. 21, no. 12. — Pp. 1297–1311.
16. Vos D. Efficient training of robust decision trees against adversarial examples / Daniel Vos, Sicco Verwer // International conference on machine learning. — 2021. — Pp. 10586–10595.

References

Andriushchenko, M., & Hein, M. (2019). Provably robust boosted decision stumps and trees against adversarial attacks. *Advances in Neural Information Processing Systems*, *32*.

Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (SP)* (pp. 39–57). IEEE.

Chen, H., Zhang, H., Boning, D., & Hsieh, C. J. (2019). Robust decision trees against adversarial examples. In *International Conference on Machine Learning* (pp. 1122–1131). PMLR.

Chen, Y., Mao, F., Song, J., Wang, X., Wang, H., & Song, M. (2021). Self-born wiring for neural trees. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 5047–5056).

Frosst, N., & Hinton, G. (2017). Distilling a neural network into a soft decision tree. *arXiv preprint arXiv*:1711.09784.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv*:1412.6572.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Ioannou, Y., Robertson, D., Zikic, D., Kontschieder, P., Shotton, J., Brown, M., & Criminisi, A. (2016). Decision forests, convolutional networks and the models in-between. *arXiv preprint arXiv*:1603.01250.

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, *6* (2), 181–214.

Kontschieder, P., Fiterau, M., Criminisi, A., & Bulo, S. R. (2015). Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision* (pp. 1467–1475).

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016, May). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)* (pp. 582–597). IEEE.

Stock, J., Dolan, A., & Cavey, T. (2020). Strategies for Robust Image Classification. *arXiv preprint arXiv*:2004.03452.

Suárez, A., & Lutsko, J. F. (1999). Globally optimal fuzzy decision trees for classification and regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21* (12), 1297–1311.

Tanno, R., Arulkumaran, K., Alexander, D., Criminisi, A., & Nori, A. (2019). Adaptive neural trees. In *International Conference on Machine Learning* (pp. 6166–6175). PMLR.

Vos, D., & Verwer, S. (2021). Efficient training of robust decision trees against adversarial examples. In *International Conference on Machine Learning* (pp. 10586–10595). PMLR.

Wan, A., Dunlap, L., Ho, D., Yin, J., Lee, S., Jin, H., ... & Gonzalez, J. E. (2020). NBDT: Neural-backed decision trees. *arXiv preprint arXiv*:2004.00221.

Мокрий М. В., Швай Н. О.

СТІЙКІСТЬ НЕЙРОННИХ ДЕРЕВ РІШЕНЬ ДО ШУМУ У ВХІДНИХ ДАНИХ ДЛЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

У роботі досліджується стійкість моделей нейронних дерев рішень, які об'єднують архітектуру нейронних мереж і дерев рішень, до шуму у вхідних даних для класифікації зображень. Було запропоновано використати два методи навчання для підвищення стійкості моделей, які початково використовувалися в згорткових нейронних мережах. Зашумлення зображень з набору даних CIFAR-10 відбувається за допомогою методу гаусівського розмиття. Було розглянуто вплив методів підвищення стійкості на моделей нейронних дерев рішень і показано, що стійкість моделей до шуму у вхідних даних значно покращується.

Ключові слова: нейронні дерева рішень, машинне навчання, стійкість, збурення зображень, класифікація зображень, комп'ютерний зір, згорткові нейронні мережі.

Матеріал надійшов 21.06.2025

УДК 004.8:004.94:629.113.018.7
DOI: 10.18523/2617-3808.2025.8.68-75

V. Beimuk, D. Kuzmenko

ENERGY CONSERVATION FOR AUTONOMOUS AGENTS USING REINFORCEMENT LEARNING

Reinforcement learning (RL) has shown strong potential in autonomous racing for its adaptability to complex and dynamic driving environments. However, most research prioritizes performance metrics such as speed and lap time. Limited consideration is given to improving energy efficiency, despite its increasing importance in sustainable autonomous systems. This work investigates the capacity of RL agents to develop multi-objective driving strategies that balance lap time and fuel consumption by incorporating a fuel usage penalty into the reward function. To simulate realistic uncertainty, fuel usage is excluded from the observation space, forcing the agent to infer fuel consumption indirectly. Experiments are conducted using the Soft Actor-Critic algorithm in a high-fidelity racing simulator, Assetto Corsa, across multiple configurations of vehicles and tracks.

We compare various penalty strengths against the non-penalized agent and evaluate fuel consumption, lap time, acceleration and braking profiles, gear usage, engine RPM, and steering behavior. Results show that mild to moderate penalties lead to significant fuel savings with minimal or no loss in lap time. Our findings highlight the viability of reward shaping for multi-objective optimization in autonomous racing and contribute to broader efforts in energy-aware RL for control tasks. Results and supplementary material are available on our project website.

Keywords: reinforcement learning, autonomous driving, energy efficiency, multi-objective optimization, Soft Actor-Critic, racing simulation.

Introduction

As autonomous driving technology advances, it has the potential to reshape mobility, offering benefits ranging from reduced traffic congestion to fewer accidents [5]. Yet, developing autonomous agents involves complex challenges in perception, planning, control, and decision-making in unpredictable environments [8].

Within this broader field, autonomous racing has emerged as an insightful research area [1]. Like traditional motorsport, it pushes systems to operate at their limits, making it a powerful testbed for high-performance, safe, and efficient algorithms [7]. RL has become a popular approach in this domain due to its ability to learn from high-dimensional inputs [3, 9]. However, most current RL applications in racing focus solely on maximizing speed or minimizing lap time [1]. Energy efficiency is rarely addressed, despite its growing societal and environmental impact.

Our work addresses this gap by investigating how penalizing fuel consumption in the reward function affects RL agent behavior. We aim to encourage the agent to balance speed and energy efficiency, forcing it to learn non-trivial trade-offs. Our key hypothesis is that shaping the reward function to penalize fuel use and incentivize speed leads to more energy-efficient strategies without significantly affecting lap time.

We evaluate agents across multiple vehicle-track combinations and penalty strengths. Results show that even mild penalties can lead to significant fuel savings with minimal performance loss, in some cases even outperforming baselines.

The results highlight the importance of reward design in multi-objective RL and contribute to the broader efforts in energy-aware autonomous systems. Supplementary material and additional results are available at https://nomadflamingo.github.io/assetto_corsa_gym/.

Related Work

Autonomous driving systems have traditionally followed a perception-planning-control pipeline, widely used in both industry and research [1]. More recently, end-to-end systems using RL have gained popularity due to their ability to learn complex behaviors through interaction with the environment [3]. RL has been successfully applied to tasks ranging from highway driving [11] to aggressive maneuvers like overtaking [10].

RL has also shown potential in reducing fuel consumption. For instance, Kim et al. [6] trained a neural network to predict the most fuel-efficient speeds based on road data and trip constraints. Yet, few studies examine how RL agents adapt when fuel usage is treated as a direct constraint rather than a prediction target. This leaves open questions regarding agent behavior under explicit fuel usage penalties.

Another limitation lies in the continuous reliance on simplified simulators that often lack realistic vehicle dynamics; this limits the generalizability of learned policies to real conditions. Additionally, most RL work assumes full observability, despite real-world agents operating without access to direct sensor information [3].

Our work addresses these gaps by applying the Soft Actor-Critic (SAC) algorithm in a high-fidelity simulator, compatible with the OpenAI Gym interface [2]. We introduce fuel efficiency objectives via reward shaping, while excluding fuel level from the observation space to simulate real-world constraints.

Methods

We use the AssettoCorsaGym interface developed by Remonda et al. [7], which integrates a high-fidelity racing simulator, Assetto Corsa, with the OpenAI Gym environment. Figure 1 shows an overview of the AssettoCorsaGym platform.

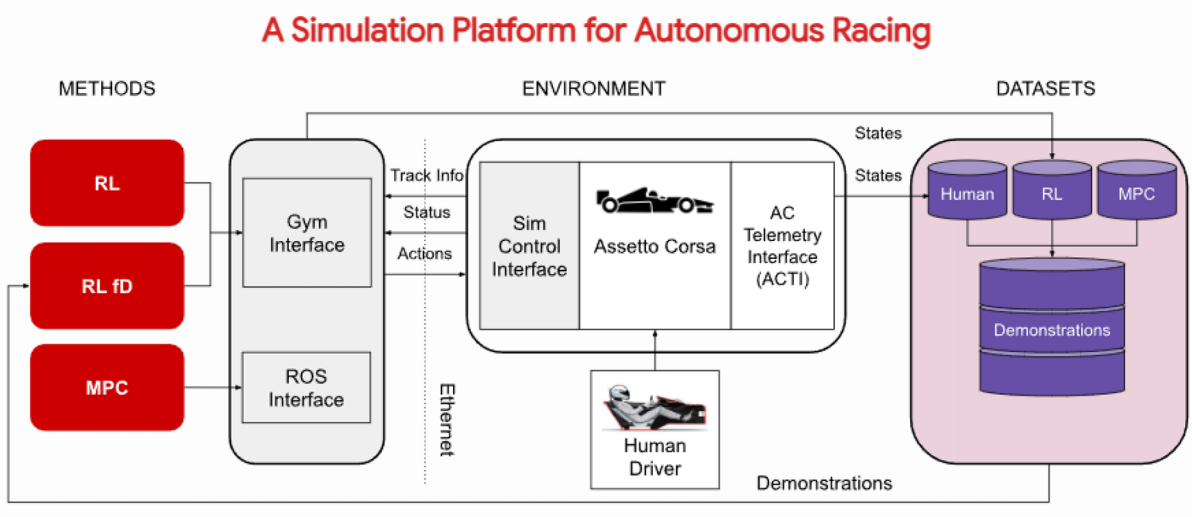


Figure 1. The architecture of the AssettoCorsaGym platform [8]

We used the SAC RL algorithm designed specifically for continuous control tasks [4]. It offers strong performance in autonomous racing benchmarks [7], as well as robustness in tasks that require balance between speed, control, and long-term planning [4, 7].

The setup included two tracks: Track A (Austria) and Track B (Monza) (Figure 2), and two vehicle models: a lightweight Formula 3 series car (F317) and a heavier GT3 series car (BMW Z4 GT3). These selections were made from the Assetto Corsa environment to align with the original AssettoCorsaGym dataset.

Track A offers a balanced layout for general driving evaluation, while Track B, containing tighter corner sequences, tests performance in more challenging conditions.

The two vehicles differ in dynamics. The F3 series car is a lightweight and high-downforce car that is nimble and agile, while the GT3 is a heavier, high-power vehicle that requires more cautious driving strategies, especially during sharp turns.

We extended the AssettoCorsaGym platform to include fuel usage data in the reward calculation. The reward function provided in AssettoCorsaGym is based on the car's velocity and penalizes deviation from the optimal driving line. It is computed as:

$$r = v \cdot (1 - a \cdot d)$$

where v is the car's current speed, d is the L2 distance from the optimal path (as determined by the simulator), and a is a penalty coefficient [7].

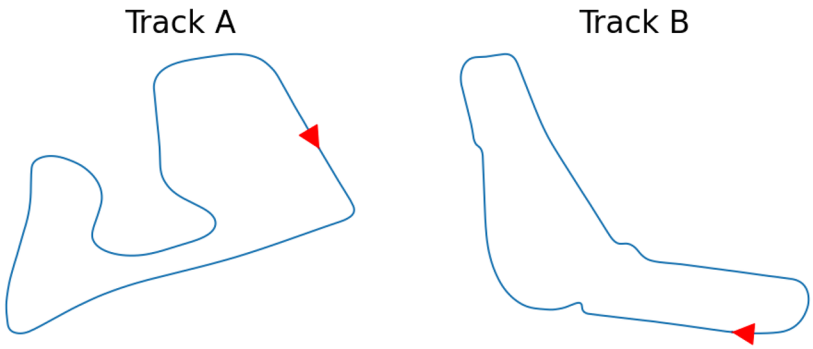


Figure 2. Two tracks chosen for training and evaluating the model in the Assetto Corsa simulator. Red triangles indicate the start positions and directions.

To encourage fuel efficientcy, we extended the reward function with a penalty for fuel consumption. The resulting reward function is defined as:

$$r = v \cdot (1 - a \cdot d) - b \cdot f$$

where f is the change in fuel since the last timestep, and b controls the strength of the penalty. Notably, fuel consumption data was excluded from the agent's observation space to encourage the agent to learn through implicit feedback.

We experimented with four values of the b coefficient – corresponding to approximate reward reductions of 2%, 5%, 10%, and 20%, relative to the original reward function formulation. All training runs were performed on an RTX 3060 laptop GPU. On average, it took approximately 48 hours to complete 500 training episodes.

Experiments

We examined the effect of fuel penalties on RL agent performance during training.

Figure 3 (Top) shows that low to moderate penalties (2-5%) often improved lap times during early training compared to the baseline, particularly with the GT3 car on Track A. However, higher penalties (20%) led to suboptimal policies that heavily prioritized fuel savings. On the more complex Track B, penalties above 2% consistently prevented agents from completing valid laps.

Figure 3 (Bottom) shows the evolution of the fuel consumption rates per lap during training. Across all setups, penalized agents consistently reduced fuel usage over time. This effect was strong on Track A for both vehicles. On Track B, fuel savings were less noticeable and limited by the lower magnitude of the penalty (2%), as higher penalties failed to converge.

Table 1 summarizes the best lap times and corresponding fuel consumption across all setups. On Track A, penalties led to fuel savings up to 0.4L per lap without major performance drops. On Track B, even mild penalties degraded performance and convergence. "DNF" values indicate failure to complete valid laps.

Table 1. **Comparison of best lap times and corresponding fuel consumption rates per lap during training across different setups**

Fuel Penalty	Random Seed	Best Lap Time (s) ↓	Fuel/Lap for Best Lap ↓
Track A, Car: GT3 series			
0%	0	94.97	1.88
	1	93.13	1.80
5%	0	96.59	1.88
	1	**92.57**	1.59
	2	95.09	1.47
10%	1	94.35	1.44
	2	93.80	1.43
20%	1	115.93	**1.01**
Track A, Car: F3 series			
0%	0	**83.62**	0.97
5%	0	83.86	**0.90**

Fuel Penalty	Random Seed	Best Lap Time (s) ↓	Fuel/Lap for Best Lap ↓
Track B, Car: GT3 series			
0%	0	**112.99**	2.45
2%	0	113.85	**2.39**
5%	0	DNF	DNF
10%	0	DNF	DNF

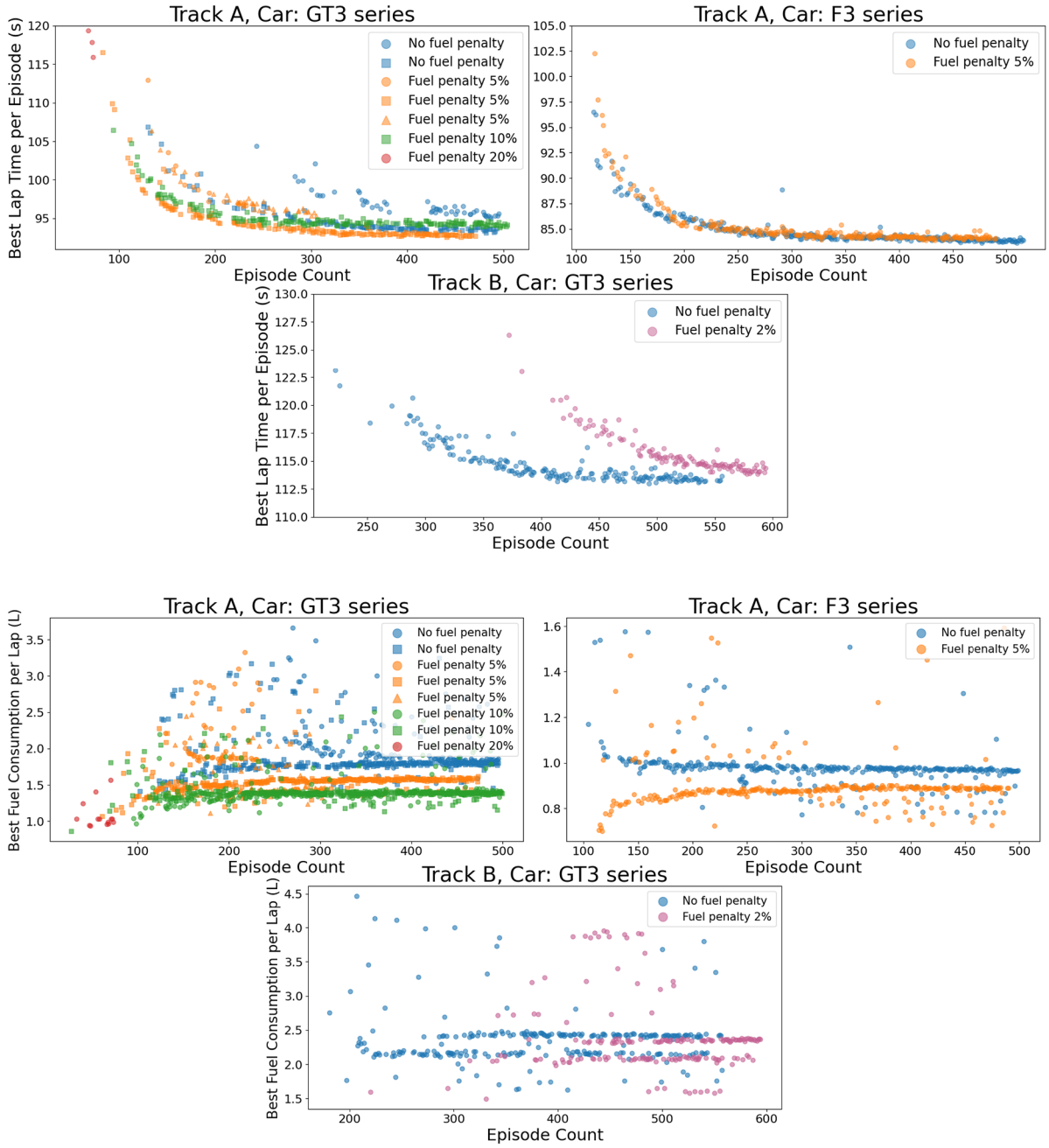


Figure 3. Top: Evolution of the best lap times per episode during training. Bottom: Best fuel consumption rates per lap. Different shapes represent different random seeds.

We compared the trained models against the baseline on Track A and Track B. On Track A, we tested models with 0% and 5% penalties. On Track B, models with 0% and 2% penalties were tested. Models were selected to have similar lap times for fair fuel efficiency comparison.

As per Table 2, the 5% penalty model on Track A used 0.19L less fuel per lap (10.7% savings) and was faster by 0.18 seconds (0.2%). On Track B, the 2% penalty model saved 0.064L (2.6%) but was slower by 0.84 seconds (0.7%). Training durations were similar between models.

Table 2. **Performance comparison of agents trained under different fuel penalties**

Fuel Penalty	Best Lap Time (s) ↓	Mean Lap Time (s) ↓	FC for Best Lap (L) ↓	Mean FC Per Lap (L) ↓	Training Episodes
Track A, Car: GT3 series					
0%	92.964	92.975	1.842	1.841	768
5%	**92.784**	**92.796**	**1.646**	**1.644**	474
Track B, Car: GT3 series					
0%	**113.198**	**113.206**	2.431	2.431	560
2%	114.04	114.203	**2.367**	**2.366**	595

Figure 4 shows spatial differences in fuel usage and lap time. Penalized models consumed less fuel, especially before major turns. On Track A, the penalty also led to faster lap times. On Track B, fuel savings came at the cost of slower lap times.

Figure 5 shows a comparison of driving behavior between the two models. On Track A, penalized agents reduced acceleration, braking, steering amplitude, and engine RPM – all factors responsible for the increased fuel consumption, directly or indirectly. On Track B, adaptations were weaker due to the lower penalty level: acceleration and RPM decreased, braking remained similar, and steering angle amplitude increased.

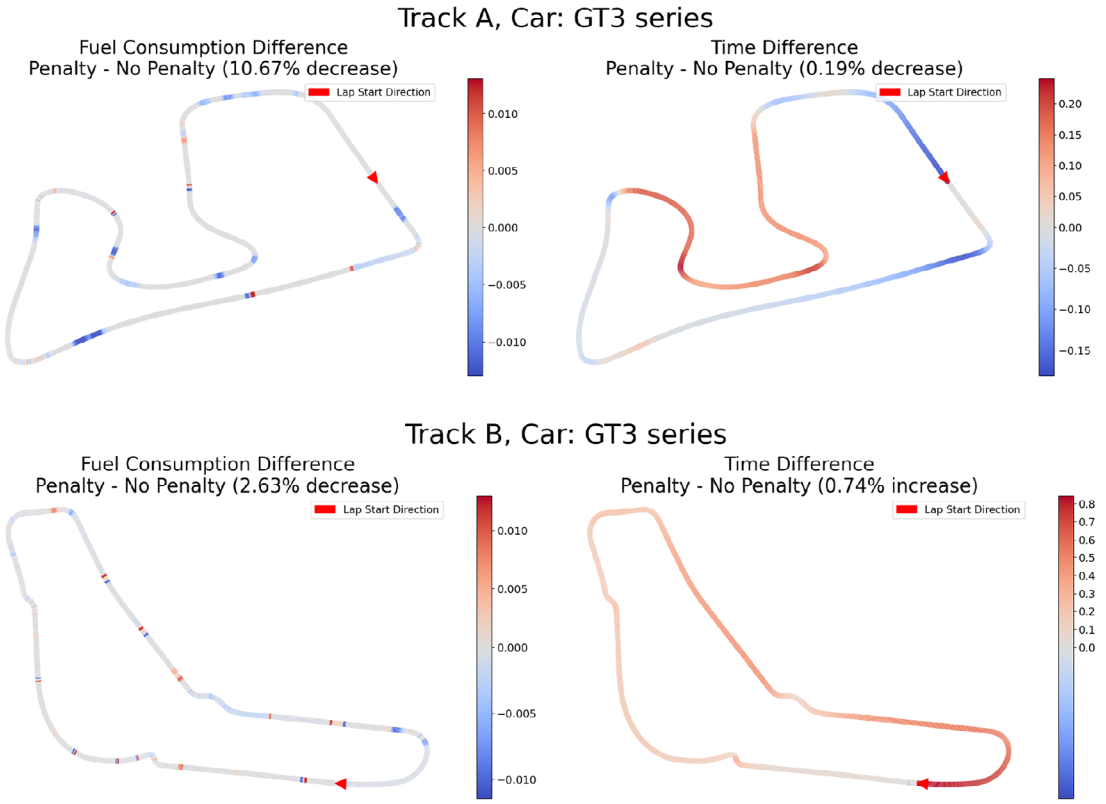


Figure 4. Spatial differences in fuel consumption and lap times between the two models on Tracks A and B. Percentage changes are computed as the difference in mean values between penalized and baseline laps. Blue regions indicate a decrease in the measured metric compared to the baseline, while red regions indicate an increase.

Results

Our experiments show that adding a fuel consumption penalty to the reward function leads to more fuel-efficient policies without significantly reducing lap time. On easier tracks, mild penalties (≤5%) often improved both fuel efficiency and speed, with faster early convergence during training. However, penalties above 10% led agents to prioritize fuel savings at the cost of increased lap time.

On the more challenging Track B, even a 2% penalty impaired performance, and higher penalties prevented agents from completing valid laps, confirming that penalty effectiveness depends on both track difficulty and the magnitude of the penalty.

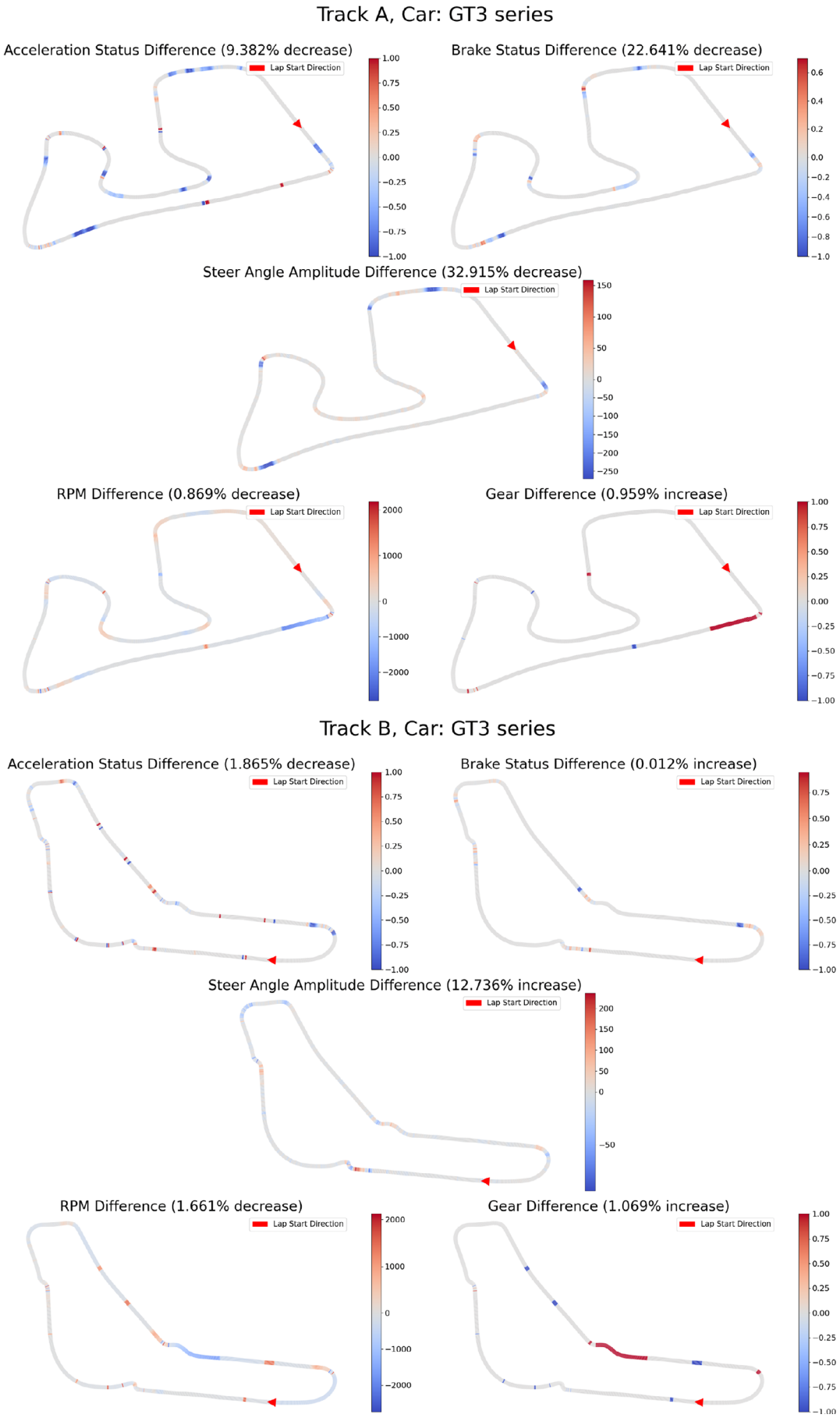


Figure 5. Spatial differences in driving behavior between the baseline and penalized agents

The agent learned key driving strategies to reduce fuel usage, mimicking real-world energy-saving strategies used in motorsport, such as maintaining momentum and staying in higher gears. Interestingly, it avoided early braking, likely suggesting that the agent prioritized maintaining speed over extra fuel savings.

Overall, the SAC algorithm effectively adapted to multi-objective rewards and learned fuel-efficient driving strategies under realistic constraints.

Discussions

Our work highlights the ability of RL agents to adopt fuel-efficient behaviors given appropriate reward shaping. However, several limitations remain that could be explored in future work.

First, experiments were limited to two vehicles and two tracks, raising concerns about generalizability. This is particularly relevant since strategies learned on the simpler Track A did not entirely transfer to Track B.

Second, the agent lacked manual gear-shifting control and could only influence gears indirectly via speed. This restricted fuel-saving techniques like short-shifting.

Third, the Assetto Corsa simulator runs only in real time, which considerably slows down training (~48 hours per 500 episodes). This limited our ability to test alternative reward designs or repeat training with different random seeds to ensure stability.

Notably, only one reward function design was explored. Future work could explore alternative formulations, like penalizing fuel-related factors directly or adding a short history of past fuel usage to the state space.

Finally, fuel usage was excluded from the observation space of the agent to simulate partial observability. While it did not prevent the agent from learning efficient strategies, future work could compare outcomes with and without partial observability.

It would also be valuable to test these methods with other RL algorithms beyond SAC, or with classical control frameworks such as MPC, LQR, or PID.

Conclusions

Our work explores the ability of RL agents in autonomous racing environments to adapt to multi-objective tasks that optimize both lap times and energy efficiency. We incorporated a fuel usage penalty into the reward function and demonstrated that low to moderate penalties lead to considerable fuel savings with minimal lap-time performance loss. The agents adapted by modifying their driving behaviors — reducing acceleration, managing engine RPM through gear changes, and increasing steering smoothness. However, these effects did not transfer to more complex tracks, where even small penalties impaired learning, highlighting the need for environment-specific penalty calibration.

Overall, our results suggest that RL can be effectively used to balance performance and energy efficiency. Future work could focus on generalizing these strategies across additional tracks and vehicles, and explore alternative reward function designs or observational conditions.

Acknowledgments

The Assetto Corsa simulation environment was used solely for the purposes of this research. We do not intend to use it in any public-facing or commercial context, nor in any activity involving public distribution or display.

The simulator is publicly available on the Steam platform. However, to use Assetto Corsa for research purposes, it is required to obtain the appropriate permissions from the Assetto Corsa Support Team.

Список літератури

1. Betz J. Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing [Electronic resource] / J. Betz IEEE // Open Journal of Intelligent Transportation Systems. — 2022. — Vol. 3. — Pp. 458—488. — Mode of access: https://doi.org/10.1109/ojits.2022.3181510.
2. Brockman G. OpenAI Gym [Electronic resource] / G. Brockman et al. // ArXiv — 2016. — Mode of access: https://doi.org/10.48550/arXiv.1606.01540.
3. Elallid B. A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving [Electronic resource] / B. Elallid et al. // Journal of King Saud University — Computer and Information Sciences. — 2022. — Vol. 34, no. 9. — Pp. 7366–7390. — Mode of access: https://doi.org/10.1016/j.jksuci.2022.03.013.
4. Harrnoja T. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor [Electronic resource] / T. Haarnoja et al. // ArXiv — 2018. — Mode of access: https://doi.org/10.48550/arXiv.1801.01290.

5. Hu X. How Simulation Helps Autonomous Driving: A survey of Sim2real, Digital Twins, and Parallel Intelligence [Electronic resource] / X. Hu et al. // IEEE Transactions on Intelligent Vehicles. — 2024. — Vol. 9, no. 1. — Pp. 593–612. — Mode of access: https://doi.org/10.1109/TIV.2023.3312777.
6. Kim H. Autonomous Vehicle Fuel Economy Optimization with Deep Reinforcement Learning [Electronic resource] / H. Kim et al. // Electronics. — 2020. — Vol. 9, no. 11. — 3. 1911. — Mode of access: https://doi.org/10.3390/electronics9111911.
7. Remonda A. A Simulation Benchmark for Autonomous Racingwith Large-Scale Human Data [Electronic resource] / A. Remonda et al. // ArXiv. — 2024. — Mode of access: https://doi.org/10.48550/arXiv.2407.1668.
8. Revell J. Sim2real: Issues in transferring autonomous driving model from simulation to real world [Electronic resource] / J. Revell, D. Welch, J. Hereford // SoutheastCon 2022. — 2022. — Pp. 296–301. — Mode of access: https://doi.org/10.1109/SoutheastCon48659.2022.9764110.
9. Sutton R. Reinforcement Learning: An Introduction / R. S. Sutton, A. G. Barto // MIT Press, London, 1998. — Mode of access: https://books.google.ca/books?id=CAFR6IBF4xYC.
10. Song Y. Autonomous Overtaking in Gran Turismo Sport Using Curriculum Reinforcement Learning [Electronic resource] / Y. Song et al. // IEEE International Conference on Robotics and Automation (ICRA). — 2021. — Pp. 9403–9409. — Mode of access: https://doi.org/ 10.1109/ICRA48506.2021.9561049.
11. Tang X. Highway Decision-Making and Motion Planning for Autonomous Driving via Soft Actor-Critic [Electronic resource] / X. Tang et al. // IEEE Transactions on Vehicular Technology. — 2022. — Vol. 71, no. 5. — Pp. 4706–4717. – Mode of access: https://doi.org/10.1109/TVT.2022.3151651.

References

Betz, J., Zheng, H., Liniger, A., Rosolia, U., Karle, P., Behl, M., Krovi, V., & Mangharam, R. (2022). Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing. *IEEE Open Journal of Intelligent Transportation Systems*, *3*, 458–488. https://doi.org/10.1109/OJITS.2022.3181510.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym* [Preprint]. arXiv. https://doi.org/10.48550/ARXIV.1606.01540.

Elallid, B. B., Benamar, N., Hafid, A. S., Rachidi, T., & Mrani, N. (2022). A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving. *Journal of King Saud University - Computer and Information Sciences*, *34* (9), 7366–7390. https://doi.org/10.1016/j.jksuci.2022.03.013.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor* (Version 2) [Preprint]. arXiv. https://doi.org/10.48550/ARXIV.1801.01290.

Hu, X., Li, S., Huang, T., Tang, B., Huai, R., & Chen, L. (2024). How Simulation Helps Autonomous Driving: A Survey of Sim2real, Digital Twins, and Parallel Intelligence. *IEEE Transactions on Intelligent Vehicles*, *9* (1), 593–612. https://doi.org/10.1109/TIV.2023.3312777.

Kim, H., Pyeon, H., Park, J. S., Hwang, J. Y., & Lim, S. (2020). Autonomous Vehicle Fuel Economy Optimization with Deep Reinforcement Learning. *Electronics*, *9* (11), 1911. https://doi.org/10.3390/electronics9111911.

Remonda, A., Hansen, N., Raji, A., Musiu, N., Bertogna, M., Veas, E., & Wang, X. (2024). *A Simulation Benchmark for Autonomous Racing with Large-Scale Human Data* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2407.16680.

Revell, J., Welch, D., & Hereford, J. (2022). Sim2real: Issues in transferring autonomous driving model from simulation to real world. *SoutheastCon 2022*, 296–301. https://doi.org/10.1109/SoutheastCon48659.2022.9764110.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press. https://books.google.ca/books?id=CAFR6IBF4xYC.

Song, Y., Lin, H., Kaufmann, E., Durr, P., & Scaramuzza, D. (2021). Autonomous Overtaking in Gran Turismo Sport Using Curriculum Reinforcement Learning. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 9403–9409. https://doi.org/10.1109/ICRA48506.2021.9561049.

Tang, X., Huang, B., Liu, T., & Lin, X. (2022). Highway Decision-Making and Motion Planning for Autonomous Driving via Soft Actor-Critic. *IEEE Transactions on Vehicular Technology*, *71* (5), 4706–4717. https://doi.org/10.1109/TVT.2022.3151651.

Беймук В. О., Кузьменко Д. О.

ЗБЕРЕЖЕННЯ ЕНЕРГІЇ ДЛЯ АВТОНОМНИХ АГЕНТІВ ІЗ ВИКОРИСТАННЯМ НАВЧАННЯ З ПІДКРІПЛЕННЯМ

Метою роботи є дослідження можливостей алгоритмів навчання з підкріпленням для формування стратегій автономного водіння з урахуванням компромісу між енергоефективністю та швидкістю.

Робота реалізована з використанням алгоритму Soft Actor-Critic у середовищі Assetto Corsa шляхом додавання штрафу за витрату пального у функцію винагороди. Досліджено вплив різних рівнів штрафу на витрати пального та швидкість руху. Також проаналізовано ключові поведінкові зміни, зокрема прискорення, оберти двигуна, передачі та амплітуди кермового кута.

Ключові слова: автономне водіння, навчання з підкріпленням, компроміс швидкість-ефективність, енергоефективність, симуляція перегонів.

Матеріал надійшов 24.06.2025

УДК 004.89:004.912
DOI: 10.18523/2617-3808.2025.8.76-83

Ванін Д. О.

ВИКОРИСТАННЯ ОДНО- ТА БАГАТОМОВНИХ МОДЕЛЕЙ НА БАЗІ BERT ДЛЯ ВИРІШЕННЯ ЗАДАЧ АВТОМАТИЧНОГО ОБРОБЛЕННЯ ТЕКСТІВ

Об'єктом дослідження цієї статті є одно- та багатомовні моделі на основі BERT. Предметом дослідження було порівняння продуктивності таких моделей на завданнях ОПМ із наголосом на їх застосуванні для української мови. Методологічну основу порівняльного аналізу становило використання стандартних підходів до навчання та оцінки моделей. У дослідженні використовувались доступні джерела інформації.

Загалом результати дослідження свідчать про те, що як одномовні, так і багатомовні моделі на основі BERT можуть бути ефективними для вирішення завдань ОПМ залежно від конкретної мови, завдання та доступних ресурсів. Хоча одномовні моделі часто перевершують багатомовні у завданнях своєї конкретної мови, багатомовні моделі можуть мати перевагу, коли ресурси для навчання одномовних моделей обмежені. Проведене порівняння роботи одно- та багатомовних моделей для різних мов додатково підкреслило важливість проведення окремого порівняння їх застосування для української мови.

Проведений аналіз сприятиме створенню комплексного україномовного бенчмарку, що покращить якість моделей і стимулюватиме нові дослідження у галузі ОПМ для української мови, розроблення нових, більш ефективних моделей.

Ключові слова: оброблення природної мови, великі мовні моделі, одно- та багатомовні моделі, BERT.

Вступ

Останні дослідження у сфері глибокого навчання, зокрема створення та використання моделей на основі архітектури трансформера [1], як-от BERT [2], значно розширили можливості для вирішення задач оброблення природньої мови (ОПМ).

Сучасні моделі зазвичай тренуються у два етапи: базове навчання на великих нерозмічених мовних корпусах і донавчання на менших наборах даних, специфічних для завдання. У результаті базового тренування моделі навчаються «розуміти» мову та передбачати слово, яке найкраще відповідає контексту. Якість результату цього навчання напряму залежить від якості й, найважливіше, розміру навчальних даних (сучасні моделі тренуються на терабайтах нерозмічених даних). Для популярних мов, наприклад англійської чи китайської, великі мовні корпуси відшукати достатньо легко, тимчасом як для мов із низькими ресурсами комплектування набору тренувальних достатнього розміру є відчутною проблемою.

Досить перспективною альтернативою для мов з обмеженими ресурсами стали багатомовні (multilingual) моделі. Ці моделі навчаються на злитих корпусах багатьох мов і показують високі результати на багатьох із них. Для мов із обмеженою кількістю ресурсів для навчання багатомовні моделі одразу показували передові на той час результати. Причина цього — крос-лінгвістичний трансфер, або перевикористання знань про одну мову для розуміння іншої. Схожий трансфер помітний у людях, які після вивчення французької мови можуть набагато швидше вивчати англійську — через схожі слова, абетки, будову речення, ідіоми та запозичення між мовами. Отже, багатомовні моделі можуть перевикористовувати певні «знання» про мови з великими ресурсами на мовах з обмеженими ресурсами.

У науковій літературі багато вивчають можливості одно- і багатомовних моделей. Деякі дослідження демонструють, що одномовні моделі перевершують багатомовні на багатьох завданнях однієї мови. Наприклад, дослідження [17] показало, що одномовна фінська модель BERT перевершила багатомовну модель BERT на різних завданнях фінської мови. Інші дослідження показують, що багатомовні моделі

можуть одночасно бути ефективними для великоресурсних мов і показувати кращі результати, ніж одномовні для тих мов, де ресурси обмежені (наприклад у випадку португальської мови).

Об'єктом дослідження цієї статті є одно- та багатомовні моделі на основі BERT. Предметом дослідження було порівняння продуктивності таких моделей на завданнях ОПМ із наголосом на їх застосуванні для української мови. Методологічну основу порівняльного аналізу становило використання стандартних підходів до навчання та оцінки моделей. У дослідженні використовувались доступні джерела інформації.

Дослідження має як наукове, так і практичне значення. З наукового погляду воно допомагає краще розуміти компроміс між використанням одно- та багатомовних моделей і потенційні особливості кожного з підходів. З практичного боку — результати дослідження можна використовувати для зваженого обрання найбільш оптимальної моделі для конкретного завдання. Зібрані результати порівняння та перелік моделей можна також узяти як основу для створення комплексного бенчмарку оцінки моделей для української мови.

1. Основні поняття та моделі ОПМ

Оброблення природної мови (ОПМ) передбачає безліч завдань, спрямованих на те, щоб інтелектуальні програмні системи могли розуміти, інтерпретувати та генерувати людську мову. Серед основних завдань, що вирішуються у сфері ОПМ, виділяють такі:

- класифікація тексту (Text Classification), що охоплює аналіз настроїв (Sentiment Analysis), класифікацію тем (Topic Classification), виявлення спаму (Spam Detection), визначення наміру (Intent Detection);
- маркування послідовностей (Sequence Labeling) з наголосом на розпізнавання іменованих сутностей (Named Entity Recognition, NER), розмічування частин мови (Part-of-Speech Tagging), групування (Chunking, Shallow Parsing), визначення параметрів (Slot Filling);
- генерація мови (Language Generation) з підзадачами генерації тексту (Text Generation), машинного перекладу (Machine Translation), стислого викладу (Summarization), перефразування (Paraphrasing), генерації діалогів (Dialogue Generation);
- порівняння тексту (Text Comparison), що охоплює підзадачі схожості тексту (Text Similarity), визначення перефразування (Paraphrase Identification), оцінки семантичної схожості тексту (Semantic Textual Similarity);
- інформаційний пошук (Information Retrieval);
- аналіз тексту (Text Analysis);
- розуміння мови (Language Understanding).

Моделі оброблення природної мови — це складні алгоритми, які створені для розуміння, інтерпретації та генерації людської мови. Їх тренування та налаштування відбуваються у кілька етапів, кожен з яких покращує їхню здатність виконувати конкретні завдання в обробленні природної мови.

Першим етапом у тренуванні є збирання даних та попереднє оброблення. Основою для будь-якої моделі ОПМ є великий (терабайти даних) і різноманітний (різні жанри, джерела, формати) набір текстових даних. Такі дані можна отримати з книжок, статей, вебсайтів, соціальних мереж та інших джерел, що відповідають цільовій задачі. Потім необроблений текст очищують від шуму, виправляють помилки та приводять у формат, придатний для використання моделлю.

Наступним етапом є вилучення характеристик. Моделі ОПМ не можуть безпосередньо працювати з необробленим текстом. Вони покладаються на числові представлення слів або їхніх частин (наприклад, символи чи фрагменти слів). Для цього використовуються такі методи, як вкладання слів (Word2Vec, GloVe) або більш складні моделі на основі архітектури трансформерів (BERT, GPT), що перетворюють слова на вектори у багатомірному просторі. Ці вектори захоплюють семантичні зв'язки між словами, які схожі за сенсом і будуть мати високий косинус подібності між векторами-репрезентаціями.

Вибір архітектури моделі залежить від конкретного завдання ОПМ. Наприклад, для нескладних завдань на зразок «послідовність — послідовність» (машинний переклад) можуть бути достатніми рекурентні нейронні мережі (RNN). Завдання класифікації спаму може бути вирішене і наївним баєсовим класифікатором. Поширена зараз трансформерна архітектура стала дуже популярною завдяки універсальності й здатності навчатися виявляти зв'язки між елементами тексту, розташованими на різних відстанях. Вони вирішують проблему згасання важливості контексту й регулярного перерахунку контекстних ваг у рекурентних архітектурах.

Модель тренується за допомогою алгоритму навчання з учителем, наприклад, стохастичного градієнтного спуску. Під час цього ваги функції, яку становить модель, поступово змінюються, щоб досягнути глобального чи локального мінімуму. Часто завданням на етапі базового тренування є просте передбачення слова у контексті. Наприклад, модель навчається передбачувати «замасковане» слово у реченні. Для такого завдання не потрібні розмічені дані, і для навчання потенційно можна скористатися будь-яким якісним текстом на цільовій мові.

Процес тонкого налаштування

Попри те, що моделі, такі як BERT, уже мають загальне розуміння мови (вміння передбачати слова у контексті), — цього недостатньо для більш специфічних завдань. Базова модель може лише передбачати масковані слова, а відповідно для більш складного завдання, як-от відповідь на питання чи переказ тексту, модель має бути донавчена на розмічених даних. Ці дані зазвичай мають значно менший обсяг і більш сфокусовані, ніж оригінальні тренувальні набори. Для тренування моделі, яка відповідає на питання, таким набором даних будуть пари «питання — відповідь».

Замість того, щоб навчати модель «з нуля», процес тонкого налаштування починається з використання вже наявної попередньо навченої моделі, яку адаптують під конкретне завдання. Це дає змогу «перевикористати» знання, отримані з великого неструктурованого набору даних під час попереднього навчання. Додавання нових даних, які близькі до кінцевого завдання, допомагає налаштувати модель для вирішення конкретних задач.

Залежно від завдання може виникнути потреба в незначних змінах архітектури моделі. Наприклад, для аналізу настроїв можна додати до моделі класифікаційний шар нейронів або розморозити деякі шари базової моделі. Такі потреби зазвичай визначають емпірично, хоча у спільноті дослідників часто вже є відпрацьований набір змін, які потрібні для вирішення конкретних проблем.

Тонке налаштування передбачає оптимізацію гіперпараметрів, як-от швидкість навчання, розмір пакета даних і кількість епох навчання. Це дозволяє досягти найкращої продуктивності на специфічних для завдання даних і також визначається емпірично.

Одномовний BERT

Одномовний BERT (Bidirectional Encoder Representations from Transformers — двонаправлене кодування представлень із трансформерів) — це мовна модель, яка використовує трансформерну архітектуру. Вона складається з шарів самоуваги (self-attention) і нейронних мереж прямого поширення. Модель попередньо навчається на великих одномовних корпусах через передбачення замаскованого токена у нерозміченому тексті. У цьому методі частина вхідних токенів випадково маскується, а модель навчається прогнозувати їх на основі контексту [2].

Такий підхід дозволяє BERT запам'ятовувати інформацію про мову з неструктурованих даних, як-от значення окремих слів, порядок слів у реченні та граматичні категорії. Завдяки цьому модель можна легко налаштувати для різних завдань із мінімальними модифікаціями, специфічними для завдання.

Після виходу BERT показав одні з найкращих результатів в 11 завданнях оброблення природної мови [2] і став основою для численних майбутніх моделей, таких як RoBERTa [21], DistilBERT [7] та інші.

Багатомовний BERT

Розроблені багатомовні варіанти BERT, наприклад mBERT і XLM-RoBERTa [24], розширюють можливості моделі для роботи з кількома мовами. Ці моделі навчаються на об'єднаних корпусах до 100 мов, використовуючи спільну інформацію між різними мовами для покращення роботи на окремих мовах, а особливо для мов із низьким рівнем ресурсів, таких як українська [24].

Основна ідея полягає в тому, що літери та фрагменти слів часто збігаються у схожих мовах, що спрощує тренування токенізатора, а також багато мов мають подібну семантичну структуру, логіку побудови слів чи навіть спільні слова. Використовуючи багатомовні корпуси, вдається значно збільшити розмір тренувальних даних, а модель може скористатися зі схожості різних мов.

Проте ефективність багатомовних моделей BERT може варіюватися залежно від конкретної мови та завдання. Вони демонструють високу продуктивність на мовах із великою кількістю ресурсів (наприклад, англійська, китайська, іспанська), однак їхні результати на мовах із низькими ресурсами

може бути обмежено. Базова причина цього — недостатня репрезентованість розуміння мови у вагах моделі. Чим більшу частку становлять дані певної мови в тренувальному наборі, тим кращими будуть результати моделі для цієї мови.

Хоча моделі BERT досягли значних успіхів на популярних мовах, усе ще існують виклики для мов із недостатніми ресурсами. Обмежена доступність якісних і достатніх за обсягом наборів навчальних даних для цих мов може перешкоджати продуктивності як одномовних, так і багатомовних моделей. Питання того, які з двох моделей є більш ефективними для конкретної мови, можна визначити лише емпірично.

2. Багато- та одномовність у мовних моделях

Вибір між багатомовними та одномовними моделями для задач оброблення природної мови є складним, оскільки кожен підхід має свої переваги та недоліки.

Багатомовні моделі, навчені на даних із кількох мов, продемонстрували великі можливості. Вони можуть досягати передових результатів у широкому діапазоні мов, зберігаючи при цьому високу продуктивність у мовах із великими ресурсами [14]. Такий «крос-лінгвістичний трансфер» дозволяє моделям використовувати знання, отримані в одній мові, для інших мов, що особливо корисно для мов з обмеженими ресурсами [24].

Автори [16] показали, що багатомовні моделі можуть досягати наближеної або навіть вищої продуктивності на декількох мовах порівняно з одномовними моделями. Цікаво, що такі результати досягаються, незважаючи на навчання на значно меншій кількості даних у цих конкретних мовах.

Водночас, одномовні моделі часто можуть перегнати свої багатомовні аналоги у деяких завданнях, зокрема у виявленні мови ворожнечі та генеруванні речень [19; 6]. Ця вища продуктивність може бути пов’язана зі здатністю одномовної моделі глибше розуміти нюанси і тонкощі однієї мови, а не розподіляти свої ресурси між кількома мовами [22]. Важливим також є тип даних, на яких тренувалася модель. Багатомовні моделі часто навчають на даних із Вікіпедії або онлайн-видавництв. Дослідники, які тренують одномовні моделі, можуть приділити більше часу на збирання більш нішевих наборів даних. Баланс між одномовними та багатомовними моделями досить цікавий. Наприклад, у деяких дослідженнях демонструють, що банальна заміна багатомовного токенізатора на одномовний може значно покращити продуктивність майже на кожних завданні та мові [13].

Ключовим фактором, що впливає на продуктивність, є рівень представлення мови в моделі. Мови, які добре репрезентовані у словниковому запасі (базовому тренувальному наборі) багатомовної моделі, часто демонструють незначне зниження точності порівняно з їхніми одномовними аналогами [13]. Проте одномовні моделі зазвичай показують кращі результати на мовах, які займають невеликий відсоток спільного корпусу багатомовної моделі. Збільшення розміру багатомовної моделі може допомогти вирішити цю проблему, дозволяючи моделі виділяти більше потужності на кожну мову [22]. Прикладом є модель Poro 34B із 34 мільярдами параметрів, навчена на фінській, англійській і мовах програмування, яка значно перевершує наявні одномовні моделі для менш репрезентованих мов, таких як фінська [20]. Ще одним важливим фактором є вплив багатомовності на упередженість. Дослідження показують, що багатомовне донавчання іноді може посилювати упередження порівняно з одномовним донавчанням, хоча ефект не завжди є пов’язаним [5]. Це підкреслює необхідність ретельної оцінки та зменшення упереджень у багатомовних моделях.

Зрештою, вибір між одномовними та багатомовними моделями залежить від конкретного завдання, доступних ресурсів і цільових мов. У деяких випадках, як із португальською мовою, обидва типи моделей можуть досягати порівнянних результатів, що робить вибір менш критичним [8]. Однак в інших випадках необхідно ретельно зважити компроміс між крос-лінгвістичним трансфером і спеціалізацією на конкретній мові. Порівняння результатів одномовних та багатомовних моделей для української мови може вказати, у яких напрямах дослідники повинні приділити більше уваги для розроблення якісних україномовних наборів даних і моделей.

3. ОПМ української мови та виклики

Українська мова, що належить до східнослов’янської мовної сім’ї індоєвропейських мов [10], має багату морфологічну структуру, характерну для синтетичних мов [4]. Із понад 40 мільйонами носіїв, вона є однією з найбільш поширених слов’янських мов [4]. Проте українська мова стикається зі

значними викликами в галузі оброблення природної мови, перш за все через нестачу даних для тренування. Через це її класифікують як мову з низьким рівнем ресурсів [4]. У 2020 році українську мову визначили як "rising star" (рис. 1.) — мову, що розвивається з культурною спільнотою, але страждає від нестачі розмічених наборів даних [12]. Українська мова належить до кластера 3 за розподілом мовних ресурсів (кількість розмічених і нерозмічених даних) [23].

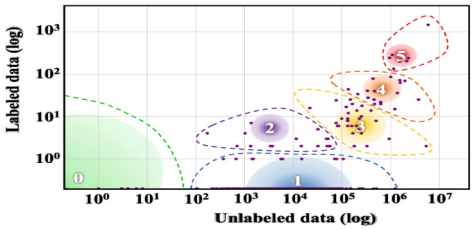


Рис. 1. Графік кластеризації мов [23]

Основною перешкодою для ОПМ української мови є нестача загальнодоступних, орієнтованих на конкретні завдання корпусів [4]. Це історично заважало розвитку надійних одномовних моделей для української мови та змушувало дослідників покладатися на крос-мовне трансферне навчання з інших мов [11]. На відміну від мов із високим рівнем ресурсів, таких як англійська, яка має величезні набори даних та обчислювальні ресурси, українська мова не мала комплексних ресурсів, необхідних для просування досліджень та розробок у галузі ОПМ [11].

Хоча українська мова має певні лінгвістичні зв'язки з іншими східнослов'янськими мовами [10], пряме трансферне навчання з цих мов не завжди є оптимальним через відмінності у лексиці, граматиці та культурному контексті. Тому розроблення спеціалізованих моделей і ресурсів ОПМ, адаптованих до унікальних особливостей української мови, є важливим для подолання цих викликів та розкриття повного потенціалу українських ОПМ застосунків.

Навчання моделей на корпусі з декількох подібних мов теоретично може показати кращі результати, ніж навчання на одномовному корпусі, у випадку нестачі одномовних ресурсів. У наступному нашому дослідженні ми плануємо перевірити гіпотезу, що модель, натренована на корпусі слов'янських мов, може показати кращі результати, ніж одномовна модель.

4. Аналіз досліджень порівняння одно- та багатомовних моделей

Декілька досліджень розглядали продуктивність одномовних та багатомовних моделей на основі BERT у різних завданнях оброблення природної мови. Результати цих досліджень були неоднозначними: деякі віддають перевагу одномовним моделям, тоді як інші демонструють ефективність багатомовних моделей.

Наприклад, дослідження, зосереджені на конкретних мовах і мовних сім'ях, виявили переваги одномовних моделей. Virtanen та ін. (2019) [17] показали, що одномовна фінська модель BERT (FinBERT) перевершує багатомовну модель BERT (mBERT) у різних фінських завданнях ОПМ. Деякі дослідження виявили, що мовно-специфічні моделі BERT перевершують mBERT у розпізнаванні іменованих сутностей (NER) у слов'янських мовах. Velankar та ін. (2022) [25] підтвердили цю тенденцію, показавши, що одномовні моделі Marathi BERT перевершують багатомовні моделі у різних завданнях ОПМ для маратської мови (індоарійська мова, 94 мільйони носіїв).

Ефективність одномовних моделей також спостерігалася при переході у межах мовних родин. Torge та ін. (2023) [18] показали, що одномовні та натреновані на декількох мовах з однієї мовної родини моделі для західнослов'янських мов перевершують великі багатомовні моделі у низхідних (downstream) завданнях.

Однак інші дослідження також показали ефективність багатомовних моделей. Feijo та Moreira (2020) [8] виявили, що багатомовна модель BERT незначно перевершує одномовні португальські моделі у різних завданнях NLP. Sido та ін. (2021) [6] також виявили, що багатомовна слов'янська модель BERT добре працює у кількох чеських завданнях ОПМ, хоча в деяких з них її перевершила одномовна чеська модель BERT.

Деякі дослідження вивчали ефекти доповнення даних і трансферного навчання. Yang та ін. (2023) [15] показали, що тримовна модель GPT-3, налаштована на англійських даних, може успішно виконувати інструкції угорською та китайською мовами, що свідчить про потенціал міжмовного транс-

ферного навчання. Vīksna та Skadina (2021) [26] виявили, що незважаючи на доповнення даних, багатомовна модель BERT не покращила показники в розпізнаванні іменованих сутностей у шести слов'янських мовах.

Висновки

Загалом результати дослідження свідчать про те, що як одномовні, так і багатомовні моделі на основі BERT можуть бути ефективними для вирішення завдань ОПМ залежно від конкретної мови, завдання та доступних ресурсів. Хоча одномовні моделі часто перевершують багатомовні у завданнях своєї конкретної мови, багатомовні моделі можуть мати перевагу, коли ресурси для навчання одномовних моделей обмежені. Проведене порівняння роботи одно- та багатомовних моделей для різних мов додатково підкреслило важливість проведення окремого порівняння їх застосування для української мови.

Ця робота має на меті зробити внесок у поточну дискусію, порівнюючи продуктивність одномовних і багатомовних моделей на основі BERT для різних завдань ОПМ. Конференція UNLP 2024, що відбулася 25 травня 2024 р., стала важливою подією для розвитку ОПМ в Україні. На конференції представили нові моделі, натреновані «з нуля» чи дотреновані на українських текстах [11; 9]. Крім того, конференція відзначилася розширенням бенчмарків для оцінки моделей [12; 3]. У рамках цієї роботи ми здійснили спробу інтегрувати деякі з цих результатів.

Проведений аналіз сприятиме створенню комплексного україномовного бенчмарку, що має покращити якість моделей і стимулюватиме нові дослідження у галузі ОПМ для української мови, розроблення нових, більш ефективних моделей.

У рамках подальшого дослідження вважаємо перспективним порівняти результати одно- та багатомовних моделей типу BERT на завданнях оброблення української мови.

Список літератури

1. Attention is all you need / Ashish Vaswani [et al.] // Proceedings of the 31st international conference on neural information processing systems. — Red Hook, NY, USA, 2017. — Pp. 6000–6010.
2. BERT: pre-training of deep bidirectional transformers for language understanding [Electronic resource] / Jacob Devlin [et al.] // Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers) / ed. by J. Burstein, C. Doran, T. Solorio. — Minneapolis, Minnesota, 2019. — Pp. 4171–4186. — Mode of access: https://aclanthology.org/N19-1423 (date of access: 23.05.2024). — Title from screen.
3. Chaplynskyi D. Introducing NER-UK 2.0: A Rich Corpus of Named Entities for Ukrainian [Electronic resource] / Dmytro Chaplynskyi, Mariana Romanyshyn // Proceedings of the Third Ukrainian Natural Language Processing Workshop (UNLP) @ LREC-COLING 2024, Torino / ed. by M. Romanyshyn et al. — [S. l.], 2024. — Pp. 23–29. — Mode of access : https://aclanthology.org/2024.unlp-1.4 (date of access: 31.05.2024). — Title from screen.
4. Chaplynskyi D. Introducing ubertext 2.0: a corpus of modern Ukrainian at scale [Electronic resource] / Dmytro Chaplynskyi // Proceedings of the second Ukrainian natural language processing workshop (UNLP), Dubrovnik / ed. by M. Romanyshyn. — [S. l.], 2023. — Pp. 1–10. — Mode of access: https://aclanthology.org/2023.unlp-1.1 (date of access: 28.05.2024). — Title from screen.
5. Comparing biases and the impact of multilingual training across multiple languages [Electronic resource] / Sharon Levy [et al.] // Association for computational linguistics. — 2023. — Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. — Pp. 10260–10280. — Mode of access: https://aclanthology.org/2023.emnlp-main.634 (date of access: 23.05.2024). — Title from screen.
6. Czert — czech bert-like model for language representation [Electronic resource] / Jakub Sido [et al.]. — [S. l. : s. n.], 2021. — Mode of access: https://doi.org/10.48550/arXiv.2103.13031 (date of access: 28.05.2024). — Title from screen.
7. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter [Electronic resource] / Victor Sanh [et al.] // ArXiv. — 2019. — Abs/1910.01108. — Mode of access: https://api.semanticscholar.org/CorpusID:203626972. — Title from screen.
8. Feijo D. de V. Mono vs multilingual transformer-based models: a comparison across several language tasks [Electronic resource] / Diego de Vargas Feijo, Viviane Pereira Moreira. — [S. l. : s. n.], 2020. — Mode of access: https://doi.org/10.48550/arXiv.2007.09757 (date of access: 28.05.2024). — Title from screen.
9. From bytes to borsch: fine-tuning gemma and mistral for the Ukrainian language representation [Electronic resource] / Artur Kiulian [et al.]. — [S. l. : s. n.], 2024. — Mode of access: https://doi.org/10.48550/arXiv.2404.09138 (date of access: 28.05.2024). — Title from screen.
10. Gomez F. P. A Low-Resource Approach to the Grammatical Error Correction of Ukrainian [Electronic resource] / Frank Palma Gomez, Alla Rozovskaya, Dan Roth // Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP), Dubrovnik / ed. by M. Romanyshyn. — [S. l.]. — Pp. 114–120. — Mode of access: https://aclanthology.org/2023.unlp-1.14 (date of access: 28.05.2024). — Title from screen.
11. Haltiuk M. LiBERTa: Advancing Ukrainian Language Modeling through Pre-training from Scratch / Mykola Haltiuk, Aleksander Smywiński-Pohl // Unlp 2024, Turin. — [S. l.], 2024.
12. Hamotskyi S. Eval-UA-tion 1.0: benchmark for evaluating Ukrainian (large) language models [Electronic resource] / Serhii Hamotskyi, Anna-Izabella Levbarg, Christian Hänig // Unlp 2024, Turin. — [S. l.], 2024. — Mode of access: https://hal.science/hal-04534651 (date of access: 28.05.2024). — Title from screen.
13. How good is your tokenizer? On the monolingual performance of multilingual language models [Electronic resource] / Phillip Rust [et al.] // Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference

on natural language processing (Volume 1: long papers) / ed. by C. Zong [et al.]. — [S. l.], 2021. — Pp. 3118–3135. — Mode of access: https://aclanthology.org/2021.acl-long.243 (date of access: 28.05.2024). — Title from screen.
14. Larger-Scale transformers for multilingual masked language modeling / Naman Goyal [et al.]. — [S. l. : s. n.], 2021.
15. Mono- and multilingual GPT-3 models for hungarian [Electronic resource] / Zijian Yang [et al.]. — [S. l.], 2023. — Pp. 94–104. — Mode of access: https://doi.org/10.1007/978-3-031-40498-6_9 (date of access: 28.05.2024). — Title from screen.
16. Multilingual instruction tuning with just a pinch of multilinguality [Electronic resource] / Uri Shaham [et al.]. — [S. l.], 2024. — Mode of access: https://doi.org/10.48550/arXiv.2401.01854 (date of access: 28.05.2024). — Title from screen.
17. Multilingual is not enough: BERT for Finnish [Electronic resource] / Antti Virtanen [et al.]. — [S. l. : s. n.], 2019. — Mode of access: https://doi.org/10.48550/arXiv.1912.07076 (date of access: 28.05.2024). — Title from screen.
18. Named entity recognition for low-resource languages — profiting from language families [Electronic resource] / Sunna Torge [et al.] // Proceedings of the 9th workshop on slavic natural language processing 2023 (slavicnlp 2023), Dubrovnik / ed. by J. Piskorski [et al.]. — [S. l.]. — Pp. 1–10. — Mode of access: https://aclanthology.org/2023.bsnlp-1.1 (date of access: 28.05.2024). — Title from screen.
19. Pires T. How multilingual is multilingual BERT [Electronic resource] / Telmo Pires, Eva Schlinger, Dan Garrette. — 2019. — Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — Pp. 4996–5001. — Mode of access: https://aclanthology.org/P19-1493 (date of access: 23.05.2024). — Title from screen.
20. Poro 34B and the blessing of multilinguality [Electronic resource] / Risto Luukkonen [et al.]. — [S. l. : s. n.], 2024. — Mode of access: https://doi.org/10.48550/arXiv.2404.01856 (date of access: 28.05.2024). — Title from screen.
21. RoBERTa: A robustly optimized BERT pretraining approach [Electronic resource] / Yinhan Liu [et al.] // CoRR. — 2019. — Abs/1907.11692. — Mode of access: http://arxiv.org/abs/1907.11692. — Title from screen.
22. Ruder S. The state of multilingual AI [Electronic resource] / Sebastian Ruder // ruder.io. — Mode of access: https://www.ruder.io/state-of-multilingual-ai/ (date of access: 23.05.2024). — Title from screen.
23. The State and Fate of Linguistic Diversity and Inclusion in the NLP World [Electronic resource] / Pratik Joshi [et al.] // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics / ed. by D. Jurafsky [et al.]. — [S. l.], 2020. — Pp. 6282–6293. — Mode of access: https://aclanthology.org/2020.acl-main.560 (date of access: 31.05.2024). — Title from screen.
24. Unsupervised cross-lingual representation learning at scale / Alexis Conneau [et al.]. — [S. l. : s. n.], 2020.
25. Velankar A. Mono vs multilingual BERT for hate speech detection and text classification: a case study in marathi / Abhishek Velankar, Hrushikesh Patil, Raviraj Joshi // Artificial neural networks in pattern recognition / ed. by N. El Gayar [et al.]. — Cham, 2023. — Pp. 121–128.
26. Vīksna R. Multilingual slavic named entity recognition [Electronic resource] / Rinalds Vīksna, Inguna Skadina // Proceedings of the 8th workshop on balto-slavic natural language processing, Kiyv. — [S. l.]. — Pp. 93–97. — Mode of access: https://aclanthology.org/2021.bsnlp-1.11 (date of access: 28.05.2024). — Title from screen.

References

Chaplynskyi, D. (2023). Introducing ubertext 2.0: a corpus of modern Ukrainian at scale. In M. Romanyshyn (Ed.), *Proceedings of the second ukrainian natural language processing workshop (UNLP)* (pp. 1–10). Association for Computational Linguistics. https://aclanthology.org/2023.unlp-1.1.

Chaplynskyi, D., & Romanyshyn, M. (2024). Introducing NER-UK 2.0: A Rich Corpus of Named Entities for Ukrainian. In M. Romanyshyn, N. Romanyshyn, A. Hlybovets & O. Ignatenko (Eds.), *Proceedings of the Third Ukrainian Natural Language Processing Workshop (UNLP) @ LREC-COLING 2024* (pp. 23–29). ELRA and ICCL. https://aclanthology.org/2024.unlp-1.4.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). *Unsupervised cross-lingual representation learning at scale*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran & T. Solorio (Eds.), *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics. https://aclanthology.org/N19-1423.

Feijo, D. de V., & Moreira, V. P. (2020). *Mono vs multilingual transformer-based models: a comparison across several language tasks*. https://doi.org/10.48550/arXiv.2007.09757.

Gomez, F. P., Rozovskaya, A., & Roth, D. (N. d.). A Low-Resource Approach to the Grammatical Error Correction of Ukrainian. In M. Romanyshyn (Ed.), *Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)* (pp. 114–120). Association for Computational Linguistics. https://aclanthology.org/2023.unlp-1.14.

Goyal, N., Du, J., Ott, M., Anantharaman, G., & Conneau, A. (2021). *Larger-Scale transformers for multilingual masked language modeling*.

Haltiuk, M., & Smywiński-Pohl, A. (2024). LiBERTa: Advancing Ukrainian Language Modeling through Pre-training from Scratch. In *Unlp 2024*.

Hamotskyi, S., Levbarg, A.-I., & Hänig, C. (2024). Eval-UA-tion 1.0: benchmark for evaluating Ukrainian (large) language models. In *Unlp 2024*. https://hal.science/hal-04534651.

Joshi, P., Santy, S., Budhiraja, A., Bali, K., & Choudhury, M. (2020). The State and Fate of Linguistic Diversity and Inclusion in the NLP World. In D. Jurafsky, J. Chai, N. Schluter & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 6282–6293). Association for Computational Linguistics. https://aclanthology.org/2020.acl-main.560.

Kiulian, A., Polishko, A., Khandoga, M., Chubych, O., Connor, J., Ravishankar, R., & Shirawalmath, A. (2024). *From bytes to borsch: fine-tuning gemma and mistral for the Ukrainian language representation*. https://doi.org/10.48550/arXiv.2404.09138.

Levy, S., John, N., Liu, L., Vyas, Y., Ma, J., Yoshinari, F., Ballesteros, M., Castelli, V., & Roth, D. (2023). Comparing biases and the impact of multilingual training across multiple languages. In *Association for computational linguistics*, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 10260–10280. https://aclanthology.org/2023.emnlp-main.634.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, *abs/1907.11692*. http://arxiv.org/abs/1907.11692.

Luukkonen, R., Burdge, J., Zosa, E., Talman, A., Komulainen, V., Hatanpää, V., Sarlin, P., & Pyysalo, S. (2024). *Poro 34B and the blessing of multilinguality*. https://doi.org/10.48550/arXiv.2404.01856.

Pires, T., Schlinger, E., & Garrette, D. (2019). How multilingual is multilingual BERT? In A. Korhonen, D. Traum & L. Màrquez (Eds.), *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 4996–5001). Association for Computational Linguistics. https://aclanthology.org/P19-1493.

Ruder, S. (2022, 14 листопада). The state of multilingual AI. *ruder.io*. https://www.ruder.io/state-of-multilingual-ai/.

Rust, P., Pfeiffer, J., Vulić, I., Ruder, S., & Gurevych, I. (2021). How good is your tokenizer? On the monolingual performance of multilingual language models. In C. Zong, F. Xia, R. Navigli & W. Li (Eds.), *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (Volume 1: long papers)* (pp. 3118–3135). Association for Computational Linguistics. https://aclanthology.org/2021.acl-long.243.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, *abs/1910.01108*. https://api.semanticscholar.org/CorpusID:203626972.

Shaham, U., Herzig, J., Aharoni, R., Szpektor, I., Tsarfaty, R., & Eyal, M. (2024). *Multilingual instruction tuning with just a pinch of multilinguality*. https://doi.org/10.48550/arXiv.2401.01854.

Sido, J., Pražák, O., Přibáň, P., Pašek, J., Seják, M., & Konopík, M. (2021). *Czert — czech bert-like model for language representation*. https://doi.org/10.48550/arXiv.2103.13031/

Torge, S., Politov, A., Lehmann, C., Saffar, B., & Tao, Z. (N. d.). Named entity recognition for low-resource languages - profiting from language families. In J. Piskorski, M. Marcińczuk, P. Nakov, M. Ogrodniczuk, S. Pollak, P. Přibáň, P. Rybak, J. Steinberger & R. Yangarber (Eds.), *Proceedings of the 9th workshop on slavic natural language processing 2023 (slavicnlp 2023)* (pp. 1–10). Association for Computational Linguistics. https://aclanthology.org/2023.bsnlp-1.1.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 6000–6010). Curran Associates Inc.

Velankar, A., Patil, H., & Joshi, R. (2023). Mono vs multilingual BERT for hate speech detection and text classification: a case study in marathi. In N. El Gayar, E. Trentin, M. Ravanelli & H. Abbas (Eds.), *Artificial neural networks in pattern recognition* (pp. 121–128). Springer International Publishing.

Vīksna, R., & Skadina, I. (N. d.). Multilingual slavic named entity recognition. In *Proceedings of the 8th workshop on balto-slavic natural language processing* (pp. 93–97). Association for Computational Linguistics. https://aclanthology.org/2021.bsnlp-1.11.

Virtanen, A., Kanerva, J., Ilo, R., Luoma, J., Luotolahti, J., Salakoski, T., Ginter, F., & Pyysalo, S. (2019). *Multilingual is not enough: BERT for Finnish*. https://doi.org/10.48550/arXiv.1912.07076.

Yang, Z., Laki, L., Váradi, T., & Prószéky, G. (2023). *Mono- and multilingual GPT-3 models for hungarian* (pp. 94–104). https://doi.org/10.1007/978-3-031-40498-6_9.

D. Vanin

THE APPLICATION OF MONOLINGUAL AND MULTILINGUAL BERT-BASED MODELS FOR TEXT AUTOMATION TASKS

This article investigates monolingual and multilingual BERT-based Transformer models. Its primary aim is to compare the behaviour of these models on a set of natural-language-processing (NLP) problems, with particular attention to their application to Ukrainian. The analysis is grounded in standard practice: large-scale masked-language pre-training, supervised fine-tuning, and evaluation on public, freely available corpora.

Five representative NLP tasks are examined—document classification, sentiment analysis, named-entity recognition, part-of-speech tagging, and sentence-level semantic similarity—because they cover core linguistic phenomena and underpin most applied pipelines. All checkpoints are trained and tested in identical experimental settings, an approach that is especially important for Ukrainian, which remains a low-resource language.

The results show that both model families are capable of solving the selected tasks, yet each excels under different conditions. Monolingual checkpoints deliver higher accuracy on problems that hinge on fine morpho-syntactic detail, such as handling case endings or varied word order. Multilingual checkpoints, in turn, offer a cost-effective solution when Ukrainian training data are scarce: knowledge transferred from related Slavic languages helps maintain acceptable quality while lowering annotation effort.

By clarifying when each strategy is preferable, the article provides practitioners with a concise decision framework and argues for the creation of a unified open benchmark to track progress. Such infrastructure will raise overall model quality, stimulate new Ukrainian-language research, and accelerate the development of more effective, resource-aware NLP technologies.

Keywords: natural language processing, large language models, monolingual and multilingual models, BERT.

Матеріал надійшов 23.06.2025

УДК 004.82
DOI 10.18523/2617-3808.2025.8.84-87

Андрощук М. В.

ВПЛИВ МЕТОДІВ ДОБУВАННЯ ЗНАНЬ НА ЕФЕКТИВНІСТЬ RAG-СИСТЕМ НА ОСНОВІ ГРАФІВ

Стаття досліджує, як методи добування знань впливають на ефективність RAG-систем, що використовують графи знань. Вона показує, що якість графа знань, сформованого різними методами добування знань, є ключовою для подолання обмежень великих мовних моделей (LLM), таких як «галюцинації». Робота аналізує архітектури LightRAG і GraphRAG та підкреслює, що вибір оптимальної KE-стратегії залежить від конкретних завдань і предметної області.

Ключові слова: RAG, графи знань, добування знань, ВММ.

Вступ

Останні роки демонструють прорив у галузі штучного інтелекту, зокрема завдяки стрімкому розвитку великих мовних моделей (LLMs). Моделі, такі як GPT-4.1, Claude Opus 4, LLaMA 4 та інші, генерують текст, який важко відрізнити від написаного людиною, перекладають тексти різними мовами, відповідають на запитання та виконують широкий спектр завдань, пов'язаних з обробленням природної мови. Їхня архітектура, що базується переважно на трансформерах, і навчання на величезних масивах текстових даних дозволили досягти безпрецедентної продуктивності. Однак, попри значні успіхи, ВММ мають низку фундаментальних обмежень, які стримують їхнє надійне застосування у критично важливих сферах, де важливі точність і пояснення результатів.

Одним із найсуттєвіших недоліків є схильність ВММ до так званих галюцинацій — генерування правдоподібної, але фактично неправильної або безглуздої інформації. Це явище пов'язане з тим, що ВММ, по суті, є статистичними моделями, які навчаються передбачати наступне слово в послідовності, не маючи справжнього розуміння світу чи доступу до актуальної фактичної бази даних. Другим важливим обмеженням є проблема «застарілих знань». Знання ВММ фіксуються на момент завершення їхнього тренування, і вони не можуть динамічно оновлюватися новою інформацією, що з'являється у світі. Це робить їх менш надійними для завдань, що потребують актуальних даних. Крім того, часто бракує прозорості у процесі прийняття рішень ВММ, що ускладнює верифікацію та довіру до їхніх відповідей [4].

Для розв'язку проблеми актуальності й точності створено системи генерації з доповненням пошуком, особливо на основі графів знань (KG-RAG). Та попри це залишається недостатньо дослідженою фундаментальна залежність ефективності таких систем від конкретних методів і стратегій, застосованих для добування знань та побудови самого графа. Необхідне глибоке розуміння того, як різні підходи до побудови та збагачення графів знань впливають на якість відповідей і загальну продуктивність таких систем.

Генерація з доповненням пошуком

Для подолання зазначених обмежень ВММ було запропоновано концепцію генерації з доповненням пошуком (Retrieval-Augmented Generation, RAG) [2]. RAG — це архітектурний підхід, який поєднує потужності генеративних моделей ВММ із зовнішньою базою знань. Замість того, щоб покладатися виключно на параметричні знання, засвоєні під час тренування, RAG-система спочатку здійснює пошук релевантної інформації із зовнішнього джерела у відповідь на запит користувача. Потім ця добута інформація передається ВММ разом із початковим запитом як контекст, на основі якого модель генерує остаточну відповідь.

Такий підхід має кілька ключових переваг: зниження ймовірності галюцинацій, забезпечення фактичної точності, вирішення проблеми застарілих знань шляхом легкого оновлення зовнішньої бази, а також підвищення прозорості через можливість надання посилань на джерела.

Графи знань

Графи знань є потужним інструментом для представлення інформації у структурованому вигляді, що становить великий інтерес для RAG-систем. Графи складаються з сутностей (вузлів) і відношень (ребер), що їх пов'язують, утворюючи семантичну мережу. Сутності можуть представляти реальні об'єкти, поняття, події, а відношення описують зв'язки між ними (наприклад, «Альберт Ейнштейн — є автором — теорії відносності»). Така структура дає змогу не лише зберігати факти, а і явно моделювати складні взаємозв'язки.

Потенціал графів знань для покращення RAG-систем є значним. На відміну від простого пошуку за ключовими словами у текстових корпусах, графи знань дають можливість здійснювати більш точний і контекстуально обґрунтований ретривінг. Завдяки чітко визначеним відношенням, можна виконувати складні запити, здійснювати багатоетапні логічні висновки та виявляти неявні зв'язки. Інтеграція графів знань у RAG-системи може забезпечити ВММ більш багатим і структурованим контекстом.

Методи добування знань для графів знань

Побудова високоякісного графу знань визначається застосованими методами добування знань з різнорідних джерел. Основні завдання добування знань такі: розпізнавання іменованих сутностей (NER), виявлення відношень (RE), зв'язування сутностей (EL) і виявлення подій (Event Extraction).

Основні методи добування знань:

– Методи, основані на правилах і лінгвістичних патернах (Rule-based KE): використання вручну створених або напівавтоматично індукованих шаблонів. Переваги: висока точність у вузьких доменах, інтерпретованість. Недоліки: низька повнота, погана масштабованість, трудомісткість. Для RAG: створюють високоточні, але розріджені графи знань.
– Методи на основі класичного машинного навчання (Classical ML-based KE): використання алгоритмів типу CRF, SVM. Переваги: краща узагальнювальна здатність, навчання на анотованих даних. Недоліки: залежність від інженерії ознак, потреба в анотованих даних. Для RAG: якість графу залежить від даних та ознак, можливий шум.
– Методи на основі глибокого навчання (Deep Learning-based KE): Ззастосування нейронних мереж (RNN, Transformers, BERT). Переваги: state-of-the-art результати, автоматичне вивчення ознак. Недоліки: потреба у великих даних, обчислювальні витрати, менша інтерпретованість. Для RAG: потенціал для повних і багатих графів знань, але помилки моделей можуть проникати в граф.
– Методи зв'язування сутностей (EL) і розширення графів (KG Completion): EL зіставляє сутності з канонічними ідентифікаторами. KG Completion передбачає відсутні зв'язки. Для RAG: EL критичний для усунення неоднозначності; KG Completion може збагатити, але й внести неточні зв'язки [5].

Вибір і комбінація цих методів визначають щільність, точність, повноту, актуальність та узгодженість графу, що безпосередньо впливає на здатність RAG-системи ефективно знаходити та використовувати релевантну інформацію до запиту користувача.

Архітектури наявних RAG-систем на основі графів знань

Загальний принцип роботи KG-RAG систем передбачає інтерпретацію запиту, пошук фактів з графу знань, формування доповненого контексту та генерацію відповіді ВММ.

У відкритому доступі є дві основні RAG-системи, що використовують графи знань, — GraphRAG [1] і LightRAG [3]. Відмінності у підходах до інтеграції графу знань полягають у тому, що LightRAG використовує його як швидкий довідник, GraphRAG — як модель предметної області. Тож для LightRAG критичною є точність видобутку знань, оскільки шум сильно деградує якість пошуку. Для GraphRAG важлива повнота видобутих знань для формування якісних результатів; певний рівень шуму може бути менш критичним, але систематичні помилки, що спотворюють структуру, також шкідливі.

При отриманні користувацького запиту LightRAG розбиває його на ключові слова, абстрактні і конкретні, для кращої контекстної обізнаності. Після цього у векторному сховищі знаходить найбільш подібні результати. Далі для них іде пошук в рамках графу знань і текстових фрагментів, що разом складуть контекст для відповіді ВММ. У GraphRAG аналіз покладається на роботу з узагальненнями, що були створені на етапі індексації, і ВММ отримує узагальнення та запит користувача і дає об'єднану відповідь.

Теоретичний аналіз впливу методів добування знань на ефективність RAG-систем

Граф знань для систем, що використовують його, є ключовим для відповідей через використання його в момент оброблення запиту користувача. Це призводить до необхідності аналізу цих процесів. Різні методи видобутку знань генерують граф знань із різною точністю, повнотою, щільністю, наявністю помилок та рівнем гранулярності. Наприклад, rule-based можуть дати високу точність, але низьку повноту, тоді як DL-методи можуть забезпечити вищу повноту, але з ризиком внесення шуму.

Кожна властивість графу знань впливатиме на ефективність роботи RAG-систем. Високоточні графи знань зменшують ризик передання ВММ помилкових фактів, знижуючи галюцинації. Більш повні графи знань надають багатший контекст, дозволяючи відповідати на складні запити (особливо multi-hop). Якість зв'язування сутностей забезпечує коректну інтеграцію інформації та навігацію всередині графу, уникаючи розпорошення інформації. Гранулярність відношень між сутностями дозволяє ВММ краще розуміти семантику при відповіді. За структурної цілісності ми отримуємо зв'язаний граф, який полегшує етап пошуку релевантних запиту сутностей і відношень.

Відповідно до цих властивостей графу і з'являються потенційні проблеми: шум у вилучених даних (помилкові факти, неправильні відношення), пропущені сутності / відношення, проблеми масштабування та підтримки правил. Для запобігання таким проблемам можна регулювати щільність графа, якість вихідного тексту для видобутку знань, складність і тип користувацьких запитів.

Аналіз видобутку знань у наявних системах

Основою RAG-систем є опрацювання документів, що надає користувач для подальшого пошуку в них. Подальше оброблення документів і перетворення їх на граф знань є частиною індексації вхідних даних.

У GraphRAG і LightRAG основна робота з видобутку знань лягає на ВММ, яка працює з частиною документа. GraphRAG орієнтований на одноразову побудову графу знань, і операції індексації нових документів вимагають перебудови графу знань для включення нового документа. ВММ у Graph є набором запитів до ВММ, де вказується структура очікуваної відповіді та налаштування контексту ВММ як асистента. При цьому GraphRAG допускає зміну цих запитів для покращення результатів при налаштуванні користувацьких потреб [1].

LightRAG теж покладається на ВММ модель, але додатково оперує «контекстом», що позначає суміжні сутності і відношення у графі знань. Додатково індексування нового документа вимагає менше кроків через пошук відповідних сутностей і відношень у графі. Це дає змогу органічно доповнювати граф знань без потреби повної перебудови графу знань.

Обидві системи покладаються на ВММ, що призводить до залежності від якості ВММ. Автори систем використовували моделі gpt-4-turbo для GraphRAG і GPT-4o-mini для LightRAG. ВММ чутливі до даних, на яких вони навчалися, що призводить до різних результатів. Іншим параметром, який впливатиме на результат, буде розмір фрагментів, на які розбивається документ: він має бути досить великим для збереження сенсу тексту у фрагменті і досить малим для оброблення ВММ без галюцинування [1; 3].

Висновки

Проведений аналіз підтверджує критичну залежність ефективності RAG-систем від методів видобування знань. Якість, повнота і точність графу знань безпосередньо впливають на здатність RAG-системи надавати релевантні, точні та правдиві відповіді. Різні архітектури RAG-систем (наприклад, LightRAG і GraphRAG) висувають різні вимоги до характеристик графів знань і, відповідно, до пріоритетних аспектів видобутку знань (точність або повнота).

Список літератури

1. Edge D. From Local to Global: A Graph RAG Approach to Query-Focused Summarization [Electronic resource] / D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, J. Larson // arXiv.org. — Mode of access: https://doi.org/10.48550/arXiv.2404.16130 (date of access: 10.06.2025).
2. Gao Y. Retrieval-Augmented Generation for Large Language Models: A Survey. [Electronic resource] / Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, H. Wang // arXiv.org. — Mode of access: https://doi.org/10.48550/arXiv.2312.10997 (date of access: 01.06.2025).
3. Guo Z. LightRAG: Simple and Fast Retrieval-Augmented Generation. [Electronic resource] /Z. Guo, L. Xia, Y. Yu, T. Ao, C. Huang // arXiv.org. — Mode of access: https://doi.org/10.48550/arXiv.2410.05779 (date of access: 10.06.2025).

4. Huang L. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions [Electronic resource] / L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, T. Liu // arXiv.org. — Mode of access: https://doi.org/10.48550/arXiv.2311.05232 (date of access: 07.06.2025).
5. Liu P. A Survey on Open Information Extraction from Rule-based Model to Large Language Model [Electronic resource] / P. Liu, W. Gao, W. Dong, L. Ai, Z. Gong, S. Huang, Z. Li, E. Hoque, J. Hirschberg, Y. Zhang // arXiv.org. — Mode of access: https://doi.org/10.48550/arXiv.2208.08690 (date of access: 25.05.2025).

References

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., & Larson, J. (2024). From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *arXiv preprint arXiv:2404.16130* (date of access: 10.06.2025). https://doi.org/10.48550/arXiv.2404.16130.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997* (date of access: 01.06.2025). https://doi.org/10.48550/arXiv.2312.10997.

Guo, Z., Xia, L., Yu, Y., Ao, T., & Huang, C. (2024). LightRAG: Simple and Fast Retrieval-Augmented Generation. *arXiv preprint arXiv:2410.05779* (date of access: 10.06.2025). https://doi.org/10.48550/arXiv.2410.05779.

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2023). A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *arXiv preprint arXiv:2311.05232* (date of access: 07.06.2025). https://doi.org/10.48550/arXiv.2311.05232.

Liu, P., Gao, W., Dong, W., Ai, L., Gong, Z., Huang, S., Li, Z., Hoque, E., Hirschberg, J., & Zhang, Y. (2022). A Survey on Open Information Extraction from Rule-based Model to Large Language Model. *arXiv preprint arXiv:2208.08690* (date of access: 25.05.2025). https://doi.org/10.48550/arXiv.2208.08690.

M. Androshchuk

INFLUENCE OF KNOWLEDGE EXTRACTION METHODS ON THE EFFECTIVENESS OF GRAPH-BASED RAG SYSTEMS

This paper investigates the impact of knowledge extraction methods on the effectiveness of RAG (Retrieval-Augmented Generation) systems that utilize knowledge graphs. It highlights that the quality of the knowledge graph, which is formed using various knowledge extraction methods, is crucial for overcoming limitations of large language models (LLMs), such as "hallucinations". The paper analyzes the architectures of LightRAG and GraphRAG, emphasizing that the selection of an optimal knowledge extraction strategy depends on specific tasks and the subject area. LLMs have advanced significantly, but they have limitations, including generating factually incorrect information ("hallucinations") and possessing "outdated knowledge". RAG systems were proposed to address these issues by combining LLMs with external knowledge bases. This approach reduces hallucinations, ensures factual accuracy, solves the problem of outdated knowledge, and increases transparency. Knowledge graphs are powerful tools for structuring information, consisting of entities (nodes) and relations (edges). They enhance RAG systems by enabling more precise and contextually grounded retrieval compared to keyword-based searches. The quality of a knowledge graph depends on the knowledge extraction methods used, which include named entity recognition (NER), relation extraction (RE), entity linking (EL), and event extraction. Different methods, such as rule-based, classical machine learning, and deep learning approaches, have varying trade-offs in terms of accuracy, completeness, and scalability. Entity linking and knowledge graph completion are also crucial for accuracy and richness. LightRAG and GraphRAG are two main graph-based RAG systems. LightRAG uses the knowledge graph as a quick reference, requiring high precision in knowledge extraction to avoid noise degradation. GraphRAG uses the knowledge graph as a domain model, where completeness of extracted knowledge is more critical, though systematic errors are still harmful. Both systems rely on LLMs for knowledge extraction, which makes them dependent on the LLM's quality and the size of document fragments processed. The theoretical analysis confirms that the effectiveness of RAG systems is critically dependent on knowledge extraction methods. The quality, completeness, and accuracy of the knowledge graph directly influence the RAG system's ability to provide relevant, accurate, and truthful answers. Different RAG system architectures like LightRAG and GraphRAG have distinct requirements for knowledge graph characteristics, prioritizing either accuracy or completeness in knowledge extraction.

Keywords: RAG, knowledge graphs, knowledge extraction, LLM.

Матеріал надійшов 23.06.2025

УДК 004.932
DOI: 10.18523/2617-3808.2025.8.88-96

Чоловський С. О., Бучко О. А.

АУГМЕНТАЦІЯ ДАНИХ У КОМП'ЮТЕРНОМУ ЗОРІ ІЗ ВИКОРИСТАННЯМ ГЕНЕРАТИВНИХ МОДЕЛЕЙ

У статті представлено огляд сучасних підходів до використання генеративних моделей для аугментації даних у задачах комп'ютерного зору. Показано, що ці моделі здатні генерувати високоякісні зображення та різні типи розмітки, що забезпечує їхню ефективність у широкому спектрі прикладних задач. Важливою умовою є недопущення витоку даних під час застосування переднавчених моделей. Проаналізовано методи оцінювання якості синтетичних даних, зокрема використання метрик візуальної якості та відповідності обумовлення, часто із залученням допоміжних моделей. Окреслено перспективні напрями подальших досліджень, зокрема забезпечення якості генерації без використання допоміжних моделей та розроблення методів вибору зразків для аугментації для найбільш ефективного навчання.

Ключові слова: аугментація, комп'ютерний зір, генеративні моделі, генеративно-змагальні мережі, дифузійні мережі.

Вступ

Аугментацією, або доповненням даних, називають набір прийомів, спрямованих на штучне розширення навчальної вибірки шляхом створення модифікованих версій наявних даних. Модель — нейронну мережу, яка навчається для розв'язання певної задачі з використанням аугментованих даних, — далі називатимемо цільовою моделлю, а саму задачу цільовою задачею відповідно.

Для аугментування даних можуть використовуватись як одиночні операції, так і багатоетапні схеми перетворень. Для пошуку оптимальних значень параметрів аугментації використовують, зокрема, навчання з підкріпленням [2], еволюційні підходи [23]. Стратегії, у яких вибір аугментаційних перетворень залежить від властивостей навчальних даних, відомі як метааугментації (meta-learning augmentations) [24]. Дослідження в цьому напрямі ведуть, зокрема, й українські науковці [3]. Додаткова обчислювальна складність визначається складністю самих перетворень зображень, простором та стратегією пошуку параметрів.

Синтетичні дані застосовуються при розв'язанні різних задач машинного навчання, в тому числі комп'ютерного зору. Метою цього огляду є опис та аналіз проблематики застосування генеративних моделей для аугментації даних у задачах комп'ютерного зору.

Базові аугментації

Для аугментування зображень спершу використовували примітивні операції над зображеннями: афінні трансформації, згорткові трансформації, перетворення в просторі кольорів, лінійне змішування та інші. На рис. 1 наведено приклади таких перетворень. У літературі стратегії аугментації, в яких використовують лише примітивні трансформації або їхні композиції, зазвичай класифікують як класичні методи аугментації. Навіть примітивні перетворення дають можливість відобразити значну частину варіативності вхідних даних. Їх широко використовують на практиці, про що свідчить, наприклад, той факт, що їх інтегровано у популярні бібліотеки машинного навчання. Вважають, що класичні методи аугментації дозволяють отримати невеликий, але гарантований приріст показників цільової моделі.

Варто зазначити, що існує формалізація [12; 13], в рамках якої диференційовані трансформації розглядаються як ще один шар нейронної мережі і для підбору параметрів перетворень використовується градієнтний спуск.

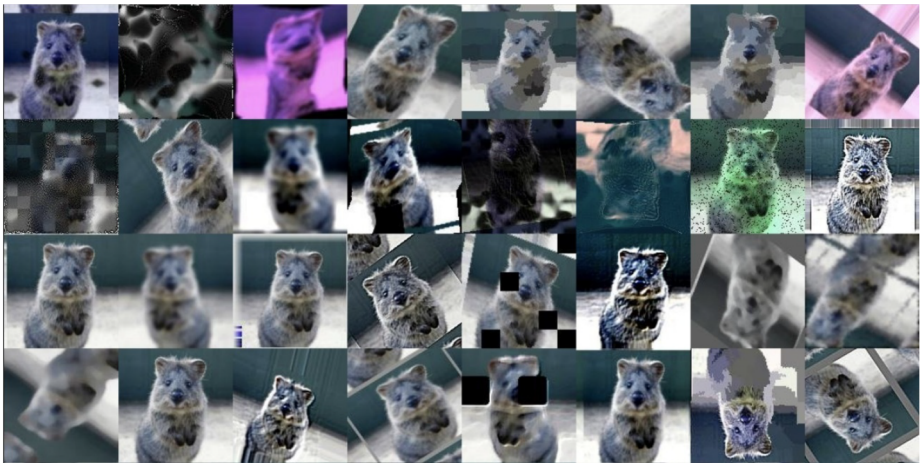


Рис. 1. Приклади базових аугментацій, застосованих до зображення

Застосування генеративних моделей для аугментації даних

Створення синтетичних даних є одним із важливих напрямів використання генеративних моделей, зокрема для аугментації навчальних вибірок. Переважна більшість таких моделей належить до одного з трьох основних класів: варіаційні автоенкодери (VAE), генеративно-змагальні мережі (GAN), дифузійні моделі (diffusion models).

1 *2* *3*

Рис. 2. Схематичне представлення можливостей генеративних аугментацій: вісь абсцис — інформація про положення на зображенні, вісь ординат — інформація про вміст зображення, чорним — елементи з вибірки, жовтим — генеральна сукупність, синім — простір генерації штучних зображень. *1* — у порівнянні з класичними аугментаціями (позначені червоним); *2* — моделювання генеральної сукупності за допомогою генеративної моделі; *3* — простір генерації перевищує простір даних

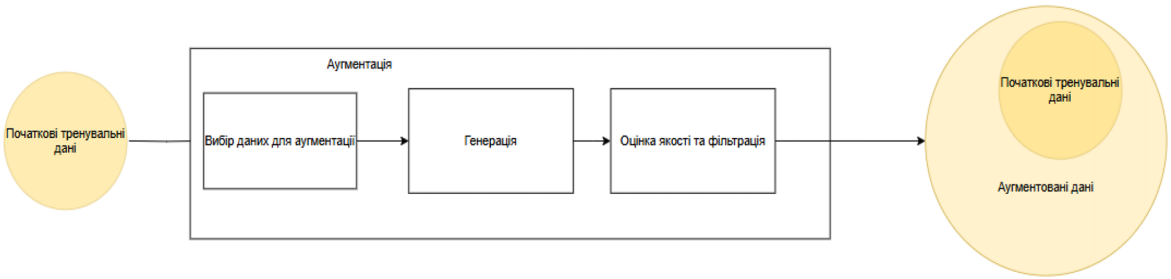


Рис. 3. Типова послідовність етапів аугментації із використанням генеративних моделей

Генеративні моделі дають можливість не лише отримувати більш різноманітні варіанти наявних зображень порівняно з базовими аугментаціями (рис. 2, *1*), а й ефективно моделювати розподіли навчальних даних (рис. 2, *2*). Використання генеративних моделей, попередньо навчених на великих наборах даних (мільярди зображень), дає змогу додати до цільового датасету інформацію про ширший клас можливих зображень (рис. 2, *3*). Однак це потребує накладання додаткових обмежень на згенеровані зображення, щоб уникнути витоку інформації чи спотворення відображення реальних даних. На рис. 3

наведено типову послідовність етапів аугментації із використанням генеративних моделей, яка передбачає такі кроки: формування штучної вибірки (вибір даних для аугментації), генерацію, оцінювання якості та фільтрацію штучно згенерованих даних. Як буде показано нижче, стратегії формування штучних даних можуть варіюватися — від рівномірної вибірки з простору ознак до аугментації конкретних екземплярів навчальних даних на основі оцінок їхньої інформативності.

Використання мереж типу кодувальник — декодувальник для аугментації

Автоенкодер є моделлю, яка кодує дані у вектор простору ознак і декодує їх з нього. Також цю модель називають латентним простором ознак. Ознаки латентного простору кодують як просторові, так і семантичні характеристики зображень.

Визначальною рисою варіаційних автоенкодерів (variational avtoencoder, VAE) є те, що енкодер навчається таким чином, щоб розподіл представлень у латентному просторі наближався до бажаного розподілу, зазвичай нормального. Це дає змогу генерувати нові приклади шляхом вибірки з цього розподілу.

У роботі [25] використано кодувально-декодувальну (encoder-decoder) згорткову нейронну мережу для перенесення стилів, із метою створення варіантів зображень при розв'язанні задач оцінки глибини (monocular depth estimation) і класифікації. Модель, що була попередньо навчена на наборі даних картин відомих художників, використали для створення варіантів фотографій об'єктів і дорожніх ситуацій. Стиль вибирався випадковим чином як вектор у просторі стилів. Проведений експеримент на наборі даних OFFICE, що складається з зображень 31 класу, кожне з яких належить до одного з трьох доменів, показав покращення узагальнюючих властивостей навчених моделей порівняно з класичними методами аугментації. Автори варіювали інтенсивність перенесення стилю та співвідношення штучних і природних зображень.

У роботах [4; 22] згенеровані варіаційним автоенкодером повністю синтетичні зображення апробовано на модельних наборах MNIST, Fashion-MNIST, Omniglot. У роботі [21] при розв'язанні задачі виявлення об'єктів, за допомогою варіаційного автоенкодера, створювали варіанти зображень об'єкта (людини) всередині розміченої рамки (bounding box), але залишали решту зображення незмінною.

Використання генеративно-змагальних мереж для аугментацій

У фреймворці генеративно-змагальних мереж (generative adversarial network, GAN) мережа складається з двох частин — генератора і дискримінатора, які суміщені в процесі навчання, але мають протилежні цільові функції. Генератор намагається створити дані, подібні до справжніх, тоді як дискримінатор прагне відрізнити згенеровані зразки від реальних.

Українські дослідники у своєму дослідженні [7] порівнювали класичні та генеративні аугментації на модельних датасетках MNIST і CIFAR-10. У роботі [19] при розв'язанні задачі класифікації зображень комах порівнюються базові аугментації та підхід із додаванням згенерованих за допомогою генеративно-змагальної мережі зображень. Для якісного аналізу використовується t-sne візуалізація представлень у латентному просторі.

В [11] запропоновано підхід для генерації зображень одночасно з масками сегментації. Автори спиралися на твердження, що для досить реалістичних генеративних моделей інформація про семантику окремих пікселів наявна в латентному просторі, а оскільки маски сегментації структурно простіші за відповідні зображення, навчити модель декодувати в маску сегментації можна з відносно невеликої кількості масок. Для генерації зображень використали мережу StyleGAN, із внутрішніх шарів якої будували карту ознак (feature-map) та отримували маски, класифікуючи кожен піксель за допомогою ансамблю багатошарових перцептронів (MLP). Отримані зображення додатково ранжувалися та фільтрувалися за значенням дивергенції Єнсена — Шеннона для оцінок кожного з пікселів — однієї з варіацій дивергенції Кульбака — Лейблера. Якість згенерованих масок оцінювалася шляхом підрахунку усередненого за категоріях індексу Жаккара (mIoU) з ручною анотацією згенерованих зображень. У роботі [1] цей підхід застосовано для аугментації медичних даних. Порівнювались методи аугментації, що використовують мережу StyleGAN, як у оригінальній роботі, та її наступника, StyleGAN-2.

У роботі [8] для отримання зображень рослин за їх масками сегментації використовували архітектуру SPADE, також відому за її реалізацією GAU-GAN. Отримані таким чином штучні зображення рослин вставляли на місце реальних на природні зображення та використовували для навчання моделі сегментації рослин. Аналогічний підхід застосували у пізнішій роботі з цього напряму [5].

Використання дифузійних мереж для аугментацій

Робота дифузійних моделей відбувається в два етапи: спершу до даних поступово додається шум (процес дифузії), унаслідок чого розподіл вхідних даних наближається до нормального, а потім відбувається відновлення даних, доданий шум усувається крок за кроком (знешумлення, denoising). При навчанні тренуються обидва оператори: дифузії та відновлення. Такий підхід дозволяє генерувати високоякісні зображення та інші типи даних.

Значний прогрес у генерації зображень досягнуто з появою генеративних мереж Stable Diffusion [18]. Автори перенесли дифузію у латентний простір автоенкодера, відділивши таким чином процес генерації від операцій переходу з простору пікселів до простору ознак і навпаки. Це дало можливість зменшити обчислювальну складність, покращити якість і забезпечити контрольованість генерації. Для контролю генерації використовуються мультимодальні обумовлення, найпоширеніші формати — це текстове обумовлення (промпт), зображення та псевдо-зображення (маски, карти глибин та подібні).

У роботі [26] використовували генеративну мережу Stable Diffusion в задачі опису зображень (image captioning). Проведено низку експериментів, для дослідження різних аспектів їхнього підходу, вплив на якість цільової моделі оцінювали на різних бенчмарках, зокрема для навчання на малих даних (few-shot learning). Розглянуто три стратегії побудови описів: часткові, де вказуються не всі об'єкти на зображенні, повні та синтетичні, отримані шляхом перефразування повних за допомогою мовної моделі чи з використанням моделі допоміжної моделі опису зображень. Для оцінки відповідності згенерованих зображень текстовим описам обчислювали показники CLIP-score та VIFIDEL, а для візуальної якості обрали показник MUSIQ замість стандартної відстані сприйняття за Фреше (FID). Оцінювали вплив фільтрації штучних зображень за цими показниками. За результатами проведених експериментів автори зазначили, що значення метрик для штучних даних, які близькі до реальних, не гарантують можливості повної заміни реальних даних штучними.

У [16] також використовується Stable Diffusion — генеративна мережа, де основну увагу приділено задачі класифікації при навчанні на невеликій кількості даних. Замість використання текстових обумовлень використовувалася техніка текстової інверсії (textual inversion) для побудови векторного представлення промпту. Окремо розглянуто умови для унеможливлення витоків даних при оцінюванні на стандартних бенчмарках.

При розв'язанні проблеми підрахунку кількості об'єктів запропоновано підхід, що використовує ControlNET для контрольованої генерації [14]. Щоби обійти необхідність забезпечення якості генерації, автори змінювали співвідношення між штучними та природними зображеннями при навчанні.

У роботі [6] представлено ще один підхід аугментації для задачі розпізнавання об'єктів на основі кількох зображень (few-shot detection), з використанням ControlNet. Нижче розглянемо цей підхід детально. До випадково вибраних зображень застосовується випадкове примітивне перетворення (зсув, обрізання або масштабування) та добувається апріорна інформація (prior extraction) для отримання обумовлювального зображення. На основі розмітки будується текстовий опис (промпт) і з побудованих обумовлень генерується штучне зображення. Якість генерації оцінюється за допомогою метрики CLIP-score, яка порівнює відповідність згенерованої ділянки текстовому опису та визначає її рейтинг серед інших згенерованих зображень для того самого класу об'єктів. Фінальний показник розраховується як середнє усіх згенерованих об'єктів, для подальшого навчання використовується лише певна частка найбільш відповідних зображень. Розглядалися різні стратегії отримання апріорної інформації: детекція контурів і сегментація, а також отримання описів типу «зображення {назви класів через кому}». Інші гіперпараметри: частка справжніх зображень, сила фільтрації.

У роботах [9; 10; 20] по-різному розвинуто ідею аналізу карт збудження шарів уваги (attention maps) для забезпечення одночасної генерації як штучних зображень, так і масок сегментації. Детальне порівняння наведено в табл. 1.

Таблиця 1. **Порівняння підходів одночасної генерації зображень і масок**

Робота	Основний фокус у роботі	Метод отримання масок	Формування текстових описів	Забезпечення якості генерації	Порівняння з іншими методами аугментації
DatasetDM [9]	Запропоновано метод генерації і його апробацію на few-shot learning задачах	Побудова маски сегментації за допомогою окремо навченого декодера	Перефразування інформації з розмітки за допомогою GPT-4	Окремо не розглядається	Виконується порівняння з класичними аугментаціями як частина ablation study
Dataset Diffusion [10]	Дослідження запропонованого методу на широкому наборі задач	Уточнення карти збудження за допомогою шарів уваги	Порівнюються стратегії побудови на основі розмітки та перефразування за допомогою мовних моделей	Якісний аналіз згенерованих даних. Розрахунок IoU зі вручну еталонними масками сегментації	Порівняння з іншим подібними методами аугментації та вичерпний ablation study
Mosaic Fusion [20]	Відтворення схеми аугментації, представленої у більш ранній роботі з використанням дифузійної моделі замість побудови колажів. Апробація запропонованого методу на широкому наборі задач	Бінаризація карт збудження за рівнем відсічення	На основі розмітки у форматі «a photo of {назва класу}»	Окремо не розглядається, виконується фільтрація масок шляхом аналізу компонент зв'язності. Отримані зображення реалістичні в деталях, але мозаїчні	Наявне порівняння з більш ранніми методами аугментації, порівнюються різні ваги однієї генеративної моделі

У роботі [17] використовується модель FreeStyleNet для генерації зображень за допомогою природних масок сегментації. Автори відмовляються від кількісного оцінювання якості згенерованих зображень загалом, натомість пропонують при навчанні ігнорувати лише «складні» пікселі на синтетичних зображеннях. Для обчислення складності окремих пікселів пропонується усереднювати функцію втрат, обчислену для допоміжної сегментаційної моделі. На основі цієї ж метрики, обчисленої для реальних зображень, пропонується генерувати з більшою ймовірністю варіанти зображень зі складнішими масками.

Обговорення

Попри доведену ефективність і гнучкість, використання генеративних моделей для аугментації має принципові обмеження. Розглянемо їх у порядку значущості: ризик витоку даних, необхідність забезпечення якості генерації та визначення меж застосовності конкретних методів. Найважливішим є забезпечення безпеки даних, оскільки витік даних ставить під сумнів як результати теоретичних досліджень, так і практичні застосування. Забезпечення якості генерації є менш критичним, оскільки навіть у випадку неідеальних результатів розглянуті методи можуть бути результативними на практиці у випадкових умовах (у форматі ad hoc).

Одним із важливих аспектів використання генеративних моделей для аугментації є забезпечення безпеки даних. Це пов'язано з тим, що інформація, яка міститься в оригінальних наборах даних, може потрапляти у згенеровані дані. Наприклад, розглянуті нами дифузійні генеративні моделі, навчені на варіантах датасету LAION, що складається з мільярдів зображень, зібраних із відкритих джерел. Існує ризик того, що частину цих зображень буде відтворено при генерації нових варіантів. Якщо згенероване зображення буде надмірно схожим на тестові дані з оцінювального набору даних, це може поставити під сумнів надійність результатів і призвести до витоку даних, що негативно позначиться на достовірності експериментів. Таким чином, є ризик компрометації експерименту. Схематично описаний витік даних зображено на рис. 4.

Окрім загального ризику витоку даних, важливо також забезпечити коректність використання генеративних моделей у рамках конкретної задачі. Деякі з розглянутих робіт застосовували аугментацію у контексті навчання на малих вибірках (few-shot learning). Якщо обмежитися лише вказанням текстового опису «зображення {назва класу}», то після генерації в навчальну вибірку потраплять не тільки варіанти об'єктів, які є у навчальній вибірці, а й абсолютно нові зображення об'єктів цього класу, що ставить під сумнів достовірність результатів експерименту.

Рис. 4. Схема витоку даних

Забезпечення якості генерації даних є ключовим аспектом при використанні генеративних моделей для аугментації зображень. Попри значно вищу якість отриманих зображень порівняно з попередниками, в усіх перелічених вище роботах, що використовують дифузійні моделі, автори так чи інакше вказують на обмеження якості, особливо при генерації складних сцен із великою кількістю об'єктів. Незважаючи на це, деякі дослідження обмежуються тільки опосередкованим оцінюванням якості, порівнюючи показники моделей, які навчалися на повністю штучних і повністю природних даних, що не дає точного уявлення про реальну ефективність застосування генеративних моделей для аугментації.

Поширеною мірою подібності наборів зображень є відстань сприйняття за Фреше (Frechet Inception Distance FID-score), вона вважається золотим стандартом у багатьох задачах генерації зображень і застосовується в розглянутих вище роботах, утім у [17; 26] вказується обмеження використання цієї метрики при аугментації. У розглянутих вище роботах також використовуються специфічні показники, характерні для різних задач. Наприклад, обчислення CLIP-score як показника відповідності між зображенням та його текстовим описом надає певну інтерпретованість результатів генерації. Цей підхід є зручним не тільки для задачі опису зображень, а й для широкого кола інших застосувань, оскільки дає можливість оцінити, наскільки добре згенероване зображення відповідає заданому текстовому опису, що є важливим для задач, де текстова інформація відіграє роль у контексті генерації.

Набувають популярності експерименти з підбором найкращих текстових описів (промптів), зокрема використання мовних моделей для перефразування. Такий підхід дає змогу покращити точність і релевантність згенерованих зображень. У задачах сегментації та детекції часто використовуються показники, як-от індекс Жаккара (intersection over union, IoU, mIoU), для оцінювання якості штучно згенерованих даних. Для обчислення цих метрик або здійснюється ручна розмітка штучного зображення, або використовується додаткова модель сегментації, що дає змогу порівняти згенеровані зображення з реальними.

Подальші дослідження в галузі аугментації даних із використанням генеративних моделей мають кілька перспективних напрямів. Одним з основних є застосування цих методів у сферах з обмеженим обсягом даних, таких як медичні зображення. В розглянутих роботах дослідження виконувалось на великих універсальних наборах даних (COCO, ADE, PASCAL VOC та ін.). З огляду на можливу відсутність якісних допоміжних моделей у спеціалізованих доменах, доцільно обмежити їх використання для оцінювання якості генерації. Як альтернативу оцінці за допомогою допоміжних моделей пропонується використовувати метрику на основі відстані сприйняття за Фреше для порівняння згенерованих і анотованих вручну (еталонних) масок сегментації.

Із подальшим розвитком генеративних моделей нинішні обмеження якості генерації, імовірно, буде подолано, а сучасні архітектури, зокрема дифузійні та генеративно-змагальні моделі, можуть бути замінені більш ефективними підходами. Водночас стратегії вибору, до яких саме екземплярів вибірки доцільно застосовувати аугментацію, значною мірою не залежать від конкретної архітектури генеративної моделі, а отже, можуть досліджуватися окремо. Перспективним напрямом є розвиток методів ранжування даних за ступенем невизначеності [11] або значенням функції втрат [17], а також їхнє поєднання з підходами, подібними до тих, що були запропоновані у [15], де авторами виконувався аналіз представлень «складних» і «простих» для класифікації екземплярів у просторі ознак.

Висновки

У статті розглянуто сучасні підходи до використання генеративних моделей для аугментації даних у задачах комп'ютерного зору. Такі моделі демонструють значно більшу варіативність доповнення даних порівняно з класичними методами та забезпечують вищу гнучкість при розв'язанні спеціалізованих задач. Напрям активно розвивається, поєднуючи методи з різних підгалузей комп'ютерного зору та машинного навчання.

Показано розвиток підходів до використання генеративних моделей від ранніх моделей на модельних наборах, до комплексних рішень, що дозволяють одночасне отримання зображень і відповідної розмітки завдяки механізмам обумовлення. Істотне розширення можливостей досягнуто з появою дифузійних моделей, які забезпечують високу якість та контрольованість синтезу даних.

Проаналізовано сучасні підходи до використання генеративних моделей для аугментації даних, виділено проблематику та окреслено перспективні напрями подальших досліджень. Зокрема, запропоновано метрику оцінювання якості генерації для сегментації, яку планується перевірити в наступних дослідженнях.

Список літератури

1. Application of DatasetGAN in medical imaging: preliminary studies [Electronic resource] / Zong Fan [et al.] // Medical Imaging 2022: Image Processing. — [S. l.], 2022. — Pp. 452–458. — Mode of access: https://doi.org/10.1117/12.2611191 (date of access: 24.06.2025). — Title from screen.
2. AutoAugment: Learning Augmentation Strategies From Data [Electronic resource] / Ekin D. Cubuk [et al.] // 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — [S. l.], 2019. — Pp. 113–123. — Mode of access: https://doi.org/10.1109/CVPR.2019.00020 (date of access: 24.06.2025). — Title from screen.
3. Baran I. Safe Augmentation: Learning Task-Specific Transformations from Data [Electronic resource] / Irynei Baran, Orest Kupyn, Arseny Kravchenko. — Mode of access: https://doi.org/10.48550/arXiv.1907.12896 (date of access: 24.06.2025). — Title from screen.
4. Chadebec C. Data Augmentation with Variational Autoencoders and Manifold Sampling [Electronic resource] / Clйment Chadebec, Stйphanie Allassonniиre // Deep Generative Models, and Data Augmentation, Labelling, and Imperfections: First Workshop, DGM4MICCAI 2021, and First Workshop, DALI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, October 1, 2021, Proceedings. — Berlin, Heidelberg, 2021. — Pp. 184–192. — Mode of access: https://doi.org/10.1007/978-3-030-88210-5_17 (date of access: 24.06.2025). — Title from screen.
5. Cho S.-B. Enhanced classification performance through GauGAN-based data augmentation for tomato leaf images [Electronic resource] / Seung-Beom Cho, Yu Cheng, Sanghun Sul // IET Image Processing. — 2024. — Vol. 18, no. 14. — Pp. 4887–4897. — Mode of access: https://doi.org/10.1049/ipr2.13069 (date of access: 24.06.2025). — Title from screen.
6. Data Augmentation for Object Detection via Controllable Diffusion Models [Electronic resource] / Haoyang Fang [et al.] // 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). — [S. l.], 2024. — Pp. 1246–1255. — Mode of access: https://doi.org/10.1109/WACV57701.2024.00129 (date of access: 24.06.2025). — Title from screen.
7. Data Augmentation Method Using Generative Adversarial Networks [Electronic resource] / Oleksandr Chaikovskyi [et al.] // Technical Sciences and Technologies. — 2021. — Pp. 83–91. — Mode of access: https://doi.org/10.25140/2411-5363-2021-2(24)-83-91. — Title from screen.
8. Data Augmentation Using GANs for Crop/Weed Segmentation in Precision Farming [Electronic resource] / Mulham Fawakherji [et al.] // 2020 IEEE Conference on Control Technology and Applications (CCTA). — [S. l.], 2020. — Pp. 279–284. — Mode of access: https://doi.org/10.1109/CCTA41146.2020.9206297 (date of access: 24.06.2025). — Title from screen.
9. Dataset Diffusion: Diffusion-based Synthetic Data Generation for Pixel-Level Semantic Segmentation [Electronic resource] / Quang Nguyen [et al.] // Advances in Neural Information Processing Systems. — 2023. — Vol. 36. — Pp. 76872–76892. — Mode of access: https://proceedings.neurips.cc/paper_files/paper/2023/hash/f2957e48240c1d90e62b303574871b47-Abstract-Conference.html (date of access: 24.06.2025). — Title from screen.
10. DatasetDM: synthesizing data with perception annotations using diffusion models / Weijia Wu [et al.] // Proceedings of the 37th International Conference on Neural Information Processing Systems. — Red Hook, [NY], [USA], 2023. — Pp. 54683–54695.
11. DatasetGAN: Efficient Labeled Data Factory with Minimal Human Effort [Electronic resource] / Yuxuan Zhang [et al.] // 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — [S. l.], 2021. — Pp. 10140–10150. — Mode of access: https://doi.org/10.1109/CVPR46437.2021.01001 (date of access: 24.06.2025). — Title from screen.
12. Differentiable Automatic Data Augmentation [Electronic resource] / Yonggang Li [et al.] // Computer Vision — ECCV 2020 / ed. by A. Vedaldi [et al.]. — Cham, 2020. — Pp. 580–595. — Mode of access: https://doi.org/10.1007/978-3-030-58542-6_35. — Title from screen.
13. Differentiable Data Augmentation with Kornia [Electronic resource] / Jian Shi [et al.]. — Mode of access: https://doi.org/10.48550/arXiv.2011.09832 (date of access: 02.02.2024). — Title from screen.
14. Diffusion-based data augmentation for object counting problems [Electronic resource] / Zhen Wang [et al.] // ICASSP 2025 — 2025 IEEE international conference on acoustics, speech and signal processing (ICASSP). — [S. l.], 2025. — Pp. 1–5. — Mode of access: https://doi.org/10.1109/ICASSP49660.2025.10888449 (date of access: 24.06.2025). — Title from screen.
15. Distilling Model Failures as Directions in Latent Space [Electronic resource] / Saachi Jain [et al.]. — Mode of access: https://doi.org/10.48550/arXiv.2206.14754 (date of access: 23.06.2025). — Title from screen.
16. Effective Data Augmentation With Diffusion Models [Electronic resource] / Brandon Trabucco [et al.]. — Mode of access: https://doi.org/10.48550/arXiv.2302.07944 (date of access: 24.06.2025). — Title from screen.
17. FreeMask: synthetic images with dense annotations make stronger segmentation models / Lihe Yang [et al.] // Proceedings of the 37th International Conference on Neural Information Processing Systems. — Red Hook, [NY], [USA], 2023. — Pp. 18659–18675.

18. High-Resolution Image Synthesis with Latent Diffusion Models [Electronic resource] / Robin Rombach [et al.] // 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — [S. l.], 2022. — Pp. 10674–10685. — Mode of access: https://doi.org/10.1109/CVPR52688.2022.01042 (date of access: 24.06.2025). — Title from screen.
19. Lu C.-Y. Generative Adversarial Network Based Image Augmentation for Insect Pest Classification Enhancement [Electronic resource] / Chen-Yi Lu, Dan Jeric Arcega Rustia, Ta-Te Lin // IFAC-PapersOnLine. — 2019. — Vol. 52, no. 30. — Pp. 1–5. — Mode of access: https://doi.org/10.1016/j.ifacol.2019.12.406 (date of access: 23.06.2025). — Title from screen.
20. MosaicFusion: Diffusion Models as Data Augmenters for Large Vocabulary Instance Segmentation [Electronic resource] / Jiahao Xie [et al.] // Int. J. Comput. Vision. — 2024. — Vol. 133, no. 4. — Pp. 1456–1475. — Mode of access: https://doi.org/10.1007/s11263-024-02223-3 (date of access: 24.06.2025). — Title from screen.
21. Nikolov I. A. Variational Autoencoders for Pedestrian Synthetic Data Augmentation of Existing Datasets: 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISAPP 2024 [Electronic resource] / Ivan Adriyanov Nikolov // Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. — 2024. — Vol. 2. — Pp. 829–836. — Mode of access: https://doi.org/10.5220/0012570700003660 (date of access: 23.06.2025). — Title from screen.
22. Norouzi S. Exemplar VAE: Linking Generative Models, Nearest Neighbor Retrieval, and Data Augmentation [Electronic resource] / Sajad Norouzi, David J. Fleet, Mohammad Norouzi. — Mode of access: https://doi.org/10.48550/arXiv.2004.04795 (date of access: 23.06.2025). — Title from screen.
23. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules [Electronic resource] / Daniel Ho [et al.]. — Mode of access: https://doi.org/10.48550/arXiv.1905.05393 (date of access: 16.05.2025). — Title from screen.
24. Shorten C. A survey on Image Data Augmentation for Deep Learning [Electronic resource] / Connor Shorten, Taghi M. Khoshgoftaar // Journal of Big Data. — 2019. — Vol. 6, no. 1. — Pp. 60. — Mode of access: https://doi.org/10.1186/s40537-019-0197-0 (date of access: 02.02.2024). — Title from screen.
25. Style Augmentation: Data Augmentation via Style Randomization [Electronic resource] / Philip T. Jackson [et al.]. — Mode of access: https://doi.org/10.48550/arXiv.1809.05375 (date of access: 23.06.2025). — Title from screen.
26. Xiao C. Multimodal Data Augmentation for Image Captioning using Diffusion Models [Electronic resource] / Changrong Xiao, Sean Xin Xu, Kunpeng Zhang // Proceedings of the 1st Workshop on Large Generative Models Meet Multimodal Applications. — [S. l.], 2023. — Pp. 23–33. — Mode of access: https://doi.org/10.1145/3607827.3616839 (date of access: 02.02.2024). — Title from screen.

References

Baran, I., Kupyn, O., & Kravchenko, A. (2019). *Safe Augmentation: Learning Task-Specific Transformations from Data* (arXiv:1907.12896). arXiv. https://doi.org/10.48550/arXiv.1907.12896.

Chadebec, C., & Allassonnière, S. (2021). Data Augmentation with Variational Autoencoders and Manifold Sampling. *Deep Generative Models, and Data Augmentation, Labelling, and Imperfections: First Workshop, DGM4MICCAI 2021, and First Workshop, DALI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, October 1, 2021, Proceedings*, 184–192. https://doi.org/10.1007/978-3-030-88210-5_17.

Chaikovskyi, O., Volokyta, A., Kyrianov, A., & Loutskii, H. (2021). Data Augmentation Method Using Generative Adversarial Networks. *Technical Sciences and Technologies*, 83–91. https://doi.org/10.25140/2411-5363-2021-2(24)-83-91.

Cho, S.-B., Cheng, Y., & Sul, S. (2024). Enhanced classification performance through GauGAN-based data augmentation for tomato leaf images. *IET Image Processing*, *18* (14), 4887–4897. https://doi.org/10.1049/ipr2.13069.

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., & Le, Q. V. (2019). AutoAugment: Learning Augmentation Strategies From Data. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 113–123. https://doi.org/10.1109/CVPR.2019.00020.

Fan, Z., Kelkar, V., Anastasio, M. A., & Li, H. (2022). Application of DatasetGAN in medical imaging: Preliminary studies. *Medical Imaging 2022: Image Processing*, *12032*, 452–458. https://doi.org/10.1117/12.2611191.

Fang, H., Han, B., Zhang, S., Zhou, S., Hu, C., & Ye, W.-M. (2024). Data Augmentation for Object Detection via Controllable Diffusion Models. *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 1246–1255. https://doi.org/10.1109/WACV57701.2024.00129.

Fawakherji, M., Potena, C., Prevedello, I., Pretto, A., Bloisi, D. D., & Nardi, D. (2020). Data Augmentation Using GANs for Crop/Weed Segmentation in Precision Farming. *2020 IEEE Conference on Control Technology and Applications (CCTA)*, 279–284. https://doi.org/10.1109/CCTA41146.2020.9206297.

Ho, D., Liang, E., Stoica, I., Abbeel, P., & Chen, X. (2019, May 14). *Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules* (Issue arXiv:1905.05393). arXiv. https://doi.org/10.48550/arXiv.1905.05393.

Jackson, P. T., Atapour-Abarghouei, A., Bonner, S., Breckon, T., & Obara, B. (2019, April 12). *Style Augmentation: Data Augmentation via Style Randomization* (Issue arXiv:1809.05375). arXiv. https://doi.org/10.48550/arXiv.1809.05375.

Jain, S., Lawrence, H., Moitra, A., & Madry, A. (2022, December 2). *Distilling Model Failures as Directions in Latent Space* (Issue arXiv:2206.14754). arXiv. https://doi.org/10.48550/arXiv.2206.14754.

Li, Y., Hu, G., Wang, Y., Hospedales, T., Robertson, N. M., & Yang, Y. (2020). Differentiable Automatic Data Augmentation. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer Vision — ECCV 2020* (pp. 580–595). Springer International Publishing. https://doi.org/10.1007/978-3-030-58542-6_35.

Lu, C.-Y., Arcega Rustia, D. J., & Lin, T.-T. (2019). Generative Adversarial Network Based Image Augmentation for Insect Pest Classification Enhancement. *IFAC-PapersOnLine*, *52* (30), 1–5. https://doi.org/10.1016/j.ifacol.2019.12.406.

Nguyen, Q., Vu, T., Tran, A., & Nguyen, K. (2023). Dataset Diffusion: Diffusion-based Synthetic Data Generation for Pixel-Level Semantic Segmentation. *Advances in Neural Information Processing Systems*, *36*, 76872–76892.

Nikolov, I. A. (2024). Variational Autoencoders for Pedestrian Synthetic Data Augmentation of Existing Datasets: 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISAPP 2024. *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, *2*, 829–836. https://doi.org/10.5220/0012570700003660.

Norouzi, S., Fleet, D. J., & Norouzi, M. (2020, November 24). *Exemplar VAE: Linking Generative Models, Nearest Neighbor Retrieval, and Data Augmentation* (Issue arXiv:2004.04795). arXiv. https://doi.org/10.48550/arXiv.2004.04795.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10674–10685. https://doi.org/10.1109/CVPR52688.2022.01042.

Shi, J., Riba, E., Mishkin, D., Moreno, F., & Nicolaou, A. (2020, November 19). *Differentiable Data Augmentation with Kornia* (Issue arXiv:2011.09832). arXiv. https://doi.org/10.48550/arXiv.2011.09832.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6* (1), 60. https://doi.org/10.1186/s40537-019-0197-0.

Trabucco, B., Doherty, K., Gurinas, M., & Salakhutdinov, R. (2025, June 10). *Effective Data Augmentation With Diffusion Models* (Issue arXiv:2302.07944). arXiv. https://doi.org/10.48550/arXiv.2302.07944.

Wang, Z., Li, Y., Wan, J., & Vasconcelos, N. (2025). Diffusion-based Data Augmentation for Object Counting Problems. *ICASSP 2025 — 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49660.2025.10888449.

Wu, W., Zhao, Y., Chen, H., Gu, Y., Zhao, R., He, Y., Zhou, H., Shou, M. Z., & Shen, C. (2023). DatasetDM: Synthesizing data with perception annotations using diffusion models. *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 54683–54695.

Xiao, C., Xu, S. X., & Zhang, K. (2023). Multimodal Data Augmentation for Image Captioning using Diffusion Models. *Proceedings of the 1st Workshop on Large Generative Models Meet Multimodal Applications*, 23–33. https://doi.org/10.1145/3607827.3616839.

Xie, J., Li, W., Li, X., Liu, Z., Ong, Y. S., & Loy, C. C. (2024). MosaicFusion: Diffusion Models as Data Augmenters for Large Vocabulary Instance Segmentation. *Int. J. Comput. Vision*, *133* (4), 1456–1475. https://doi.org/10.1007/s11263-024-02223-3.

Yang, L., Xu, X., Kang, B., Shi, Y., & Zhao, H. (2023). FreeMask: Synthetic images with dense annotations make stronger segmentation models. *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 18659–18675.

Zhang, Y., Ling, H., Gao, J., Yin, K., Lafleche, J.-F., Barriuso, A., Torralba, A., & Fidler, S. (2021). *DatasetGAN: Efficient Labeled Data Factory with Minimal Human Effort*. 10140–10150. https://doi.org/10.1109/CVPR46437.2021.01001.

S. Cholovskyi, O. Buchko

DATA AUGMENTATION IN COMPUTER VISION USING GENERATIVE MODELS

Modern generative models provide high quality of generation, with their usage considered alongside classic data augmentation techniques. The paper provides a review of existing approaches for data augmentation with generative models in computer vision tasks. Reviewed pipelines utilize enhanced conditioning mechanisms of modern generative models to produce both images and labels for various tasks including image captioning, classification, object detection and segmentation.

Data leak prevention is crucial when large pretrained generative models are used for augmentation. Models like Stable Diffusion were trained on billions of publicly available images, which might also be present in popular datasets. A potential methodological weakness in the application to few-shot learning tasks was identified: images generated based on textual prompts are sampled from all possible images of a certain class, but not only from a few given training examples. Controllable sampling should be introduced to prevent possible data leaks.

Despite the high overall quality of generated images, novel diffusional models are still error-prone in the generation of complex scenes. To mitigate this various quality assessment procedures for generated images are used. These methods include visual naturalness evaluation with Fréchet Inception Distance (FID), prompt correspondence control via CLIP-score, intersection-over-union (IoU) with ground truth or predictions of auxiliary segmentation models for segmentation masks.

Further utilization of generative augmentations in data-scarce domains such as medical imaging is needed. To achieve this, it is preferable to eliminate auxiliary predictors from generation quality assessment. It is proposed to compare generated and natural segmentation masks with the FID-score.

Another area for further research is data augmentation sampling strategies that are less dependent on specific generative pipelines and therefore can be considered separately. Targeted augmentation of hard to predict examples is more effective than uniform sampling. It is planned to improve sampling strategies from reviewed papers in future research.

Keywords: augmentation, computer vision, generative models, generative-adversarial networks, diffusional networks.

Матеріал надійшов 27.06.2025

УДК 81’322.4
DOI: 10.18523/2617-3808.2025.8.97-101

Махаммедов Ж. Ж., Кирієнко О. В., Ткаченко В. О.

ОЦІНКА ТРАНСФОРМЕРНИХ МОДЕЛЕЙ mT5 ДЛЯ УКРАЇНСЬКО-АНГЛІЙСЬКОГО ПЕРЕКЛАДУ

Цю статтю присвячено кількісному вивченню впливу розміру архітектури Transformer на точність українсько-англійського машинного перекладу з використанням моделі mT5. Досліджено ефективність роботи моделей mT5 різних розмірів (small, base, large) щодо часу навчання, часу генерації перекладів і якості перекладу, оціненої метриками BLEU та chrF++. Результати показують, що більші моделі mT5 демонструють вищу якість перекладу, але потребують значно більше обчислювальних ресурсів. Результати дослідження підтверджують доцільність застосування моделей mT5 для українсько-англійського перекладу, навіть на типових обчислювальних системах.

Ключові слова: трансформер, оброблення природної мови, машинний переклад, нейронний машинний переклад, mT5, HPLT, BLEU, chrF++, NLLB-200.

Вступ

Машинний переклад є однією з ключових галузей оброблення природної мови. Історично ця сфера зазнала значної еволюції, здійснивши перехід від ранніх систем, основаних на правилах, до статистичного машинного перекладу [5], який домінував на початку XXI століття. Проте парадигматичний зсув відбувся з появою нейронного машинного перекладу, який продемонстрував здатність генерувати значно більш якісні, контекстуально обґрунтовані та точні переклади. На відміну від статистичного машинного перекладу, що оперував на рівні фраз, перші моделі нейронного машинного перекладу на базі рекурентних нейронних мереж дали можливість обробляти речення як єдине ціле, що стало значним кроком уперед [1].

Ключовим моментом, що визначив сучасний стан нейронного машинного перекладу, стало впровадження архітектури Transformer у 2017 р. [8]. Ця архітектура відмовилася від рекурентних і згорткових шарів на користь механізмів уваги, зокрема самоуваги (self-attention). Такий підхід дозволив моделі зважувати важливість різних слів у вхідній послідовності та одночасно обробляти всі її елементи. Можливість паралельного оброблення даних не лише кардинально прискорила процес навчання та перекладу, а й значно підвищила ефективність у захопленні довготривалих залежностей у тексті. Завдяки своїй ефективності та масштабованості, архітектура Transformer стала фундаментальним будівельним блоком для переважної більшості сучасних великих мовних моделей і де-факто стандартом для найсучасніших систем машинного перекладу.

Останнім часом популярність багатомовних моделей значно зросла, цю сферу також активно досліджують і в Україні [4], однією з таких моделей є mT5 (Massively Multilingual Text-to-Text Transfer Transformer) [9]. Дослідження їхньої ефективності для менш поширених або складних мов, зокрема української, залишаються актуальними. Хоча автори mT5 у своїй статті охоплюють загальні відомості про роботу моделі та демонструють загальні спроможності архітектури, вони не надали детального аналізу або тестування, сфокусованих на задачі перекладу. Це створює прогалину в розумінні того, як різні розміри моделей mT5 проявляють себе у таких задачах, тому ця робота прагне дослідити цю проблему й оцінити можливості моделі саме для українсько-англійського перекладу.

Передумови дослідження

Модель mT5 є багатомовною версією моделі T5 (Text-to-Text Transfer Transformer), головною особливістю якої є уніфікація завдань з оброблення природної мови у формат «text-to-text», що дозволяє використовувати одну архітектуру кодера-декодера для вирішення широкого спектра завдань: від машинного перекладу та узагальнення до відповідей на запитання та класифікації тексту. Модель

mT5 була попередньо навчена (pre-trained) на багатомовному корпусі даних mC4, що дає їй можливість ефективно працювати зі 101 мовою. Це робить її надзвичайно цінним інструментом для завдань, що потребують роботи з текстами або генерації тексту різними мовами.

Мовний корпус mC4 (Massively Multilingual Common Crawl Cleaned Corpus) охоплює 101 мову та має обсяг у 27 терабайтів. Цей датасет містить 2733 мільярди токенів англійською і 41 мільярд токенів українською, що становить 5,67 % і 1,51 % відповідно. Він є багатомовною версією корпусу C4 (Colossal Clean Crawled Corpus), який використовувався для навчання оригінальної моделі T5. Обидва корпуси базуються на даних, зібраних проєктом Common Crawl, що систематично сканує веб.

Моделі mT5 існують у 5 розмірах: mT5-small, mT5-base, mT5-large, mT5-xl, mT5-xxl. Розмір моделі Transformer визначається кількістю її параметрів: кількістю блоків у кодері та декодері, розмірністю векторів ембедінгу, розмірністю матриці проєкції ключів / значень, кількістю голів уваги, розмірністю проміжного вектора всередині блоку трансформера. Ці параметри є критично важливим фактором, що впливає на продуктивність моделі та її обчислювальні вимоги. Моделі Transformer можуть варіюватися від сотень мільйонів до сотень мільярдів параметрів, що безпосередньо корелює з їхньою здатністю до захоплення складних мовних закономірностей і контекстів. Зазвичай, що більша модель і що більше даних вона обробляє під час навчання, то кращої якості перекладу вона може досягти. Однак збільшення розміру тягне за собою значне зростання обчислювальних ресурсів та пам'яті, необхідних для навчання та розгортання. Це створює важливий компроміс між бажаною точністю перекладу та практичною можливістю використання моделі на доступному обладнанні.

Методологія дослідження

Для оцінки було використано три версії mT5 — small (300 млн параметрів, 1,2 ГБ), base (580 млн параметрів, 2,4 ГБ), large (1,3 млрд параметрів, 4,9 ГБ). Ці версії було обрано, оскільки більшість споживчих графічних прискорювачів мають об'єм відеопам'яті від 8 до 12 ГБ, а отже більші моделі, як-от mT5-xl (3,7 млрд, 15 ГБ), не вмістилися б у такий об'єм.

Для кількісної оцінки якості машинного перекладу використовуються автоматичні метрики, які порівнюють згенерований моделями переклад з еталонним перекладом з корпусу. У цьому дослідженні для оцінки були обрані дві широко використовувані метрики: BLEU та chrF++, вони дають змогу швидко та об'єктивно порівнювати різні моделі.

BLEU (Bilingual Evaluation Understudy) [6] — це одна з найдавніших і найпоширеніших метрик для оцінки машинного перекладу, представлена у далекому 2002 році. Вона вимірює схожість між машинним і еталонним перекладами шляхом розрахунку модифікованої точності для n-грам. Кінцеве значення BLEU є геометричним середнім модифікованих точностей для n-грам різної довжини, помноженим на штраф за стислість. Значення BLEU лежить у діапазоні від 0 до 100, де вищі показники вказують на кращу якість.

chrF++ (character n-gram F-score) [7] — метрика, розроблена 2017 року як розширення оригінальної chrF. Вона працює на рівні символьних n-грам і додатково враховує словесні уніграми та біграми, що дозволяє враховувати як дрібні лінгвістичні деталі, так і ширший контекст. Як F-міра, chrF++ балансує між точністю та повнотою.

Для тонкого налаштування (fine-tuning) моделей було використано англійсько-українську частину корпусу HPLT v2 (High-Performance Language Technologies version 2) [2]. Цей корпус є розширеним та очищеним багатомовним набором даних, спеціально розробленим для задач оброблення природної мови. Його вибір зумовлений високою якістю і релевантністю для завдань машинного перекладу. Корпус було відфільтровано за показниками score-aligner не менше ніж 0,4 і score-bicleaner не менше ніж 0,9 для збільшення якості тренувальних даних. Після цього з отриманих пар було взято 10 % випадковим чином. Усього для навчання моделей було використано 1 068 199 пар речень.

Для оцінки якості перекладів був використаний тестовий набір даних із 10 000 українсько-англійських речень, відібраних з корпусу HPLT v2, які не використовувалися під час тонкого налаштування. Вхідні українські речення отримували префікс «translate Ukrainian to English: ». Оцінку BLEU проводили з параметром «tokenize='flores200'». Для chrF++ були вказані параметри «word_order=2» та «char_order=6», що відповідає стандартним налаштуванням.

Для розуміння результатів роботи mT5 були включені дві дистильовані моделі з сімейства NLLB-200 (No Language Left Behind) [3]: NLLB-200-distilled-600M та NLLB-200-distilled-1.3B. Обидві моделі NLLB-200 базуються на архітектурі Transformer і були навчені за допомогою техніки дистиляції,

коли менша модель вчиться на виходах більшої моделі, вони навчалися за допомогою моделі NLLB-MOE-54B (54 млрд параметрів, 250 ГБ), що дозволяє їм перекладати безпосередньо між будь-якими з 200 підтримуваних мов без використання проміжної мови.

Апаратне та програмне забезпечення

Процес тонкого налаштування моделей mT5, який потребує значних обчислювальних ресурсів і об'ємів відеопам'яті, виконувався на графічному прискорювачі Nvidia RTX 5090. Відеокарту було обрано через високу продуктивність (104.8 TFLOPS) і великий обсяг відеопам'яті (32 ГБ), вона є найпотужнішим споживчим графічним прискорювачем на момент проведення дослідження. Етап генерації перекладів для оцінки моделей виконувався на графічному прискорювачі Nvidia RTX 3070, який добре відображає умови типових сучасних обчислювальних систем. Цей графічний прискорювач має меншу продуктивність (20.31 TFLOPS) і відеопам'ять (8 ГБ) порівняно з RTX 5090, але його можливостей було достатньо для швидкого отримання перекладів на тестовому наборі даних.

Програмне середовище було побудовано на основі мови Python і провідних бібліотек для глибинного навчання:

- PyTorch — основний фреймворк для глибинного навчання, забезпечує гнучкість у роботі з нейронними мережами та оптимізовану роботу з GPU;
- Hugging Face Transformers — для взаємодії з попередньо навченими моделями Transformer, такими як mT5, і спрощення процесу тонкого налаштування;
- Hugging Face Datasets — для ефективної роботи з наборами даних;
- sacrebleu — інструмент для автоматичної оцінки машинного перекладу.

Результати

Дані, отримані після попереднього оброблення даних і тонкого налаштування моделей mT5, наведено в табл. 1 і 2.

Таблиця 1. **Час тренування моделей на графічному прискорювачі Nvidia RTX 5090**

Модель	Кількість параметрів	Час тренування (години)
mT5-small	300 млн	6:06
mT5-base	580 млн	15:20
mT5-large	1.2 млрд	48:39

Таблиця 2. **Оцінка перекладу моделей на графічному прискорювачі Nvidia RTX 3070**

Модель	Кількість параметрів	BLEU	chrF++	Час генерації прикладів (години)
mT5-small	300 млн	41,53	62,33	0:14
mT5-base	580 млн	49,97	68,17	0:21
mT5-large	1.2 млрд	54,50	71,13	1:06
NLLB-200-distilled-600M	600 млн	42,89	63,44	0:27
NLLB-200-distilled-1.3B	1.3 млрд	46,61	66,90	1:41

Аналіз табл. 2 засвідчує, що більші моделі mT5 демонструють вищу якість перекладу за обома метриками. Найкращі результати серед моделей mT5 показала mT5-large, що підтверджує гіпотезу про те, що збільшення розміру моделі позитивно корелює з якістю перекладу.

Порівняно з моделями сімейства NLLB-200 тонко налаштовані моделі mT5 демонструють кращі результати. NLLB-200-distilled-600M показала, що вона працює трохи краще, ніж mT5-small. Однак mT5-base перевершує NLLB-200-distilled-600M за обома метриками, незважаючи на схожий розмір. Найцікавіше, що NLLB-200-distilled-1.3B досягла другого за якістю результату, хоча і була найбільшою моделлю; mT5-large значно перевершує показники NLLB-200-distilled-1.3B, що свідчить про ефективність тонкого налаштування mT5 на спеціалізованому українсько-англійському корпусі даних і демонструє перевагу моделей вузького спрямування.

Отримані результати чітко ілюструють компроміс між якістю перекладу та обчислювальними ресурсами. Хоча mT5-large забезпечує найвищу якість перекладу, вона потребує суттєво більше часу для навчання та генерації порівняно з меншими моделями. Водночас mT5-base демонструє значне

покращення якості порівняно з mT5-small за помірного збільшення часу навчання та генерації, що робить її привабливим вибором для сценаріїв, де важливий баланс між якістю і доступністю ресурсів. Навіть на типових обчислювальних системах, що представлені, mT5-small, mT5-base і навіть mT5-large показують прийнятний час генерації перекладів.

Для глибшого розуміння якісних відмінностей у перекладах, згенерованих моделями різних розмірів, було проведено ручний аналіз 50 речень. У табл. 3 наведено розбір одного типового речення.

Таблиця 3. **Приклади перекладу**

Модель	Оригінал речення	Еталонний переклад	Переклад моделі
mT5-small	Швидка бура лисиця перестрибує через лінивого собаку.	A quick brown fox jumps over a lazy dog.	Quick blade of the fox spreads through a linear dog.
mT5-base	Швидка бура лисиця перестрибує через лінивого собаку.	A quick brown fox jumps over a lazy dog.	Quick brown fox jumps through a lazy dog.
mT5-large	Швидка бура лисиця перестрибує через лінивого собаку.	A quick brown fox jumps over a lazy dog.	The fast brown fox jumps over a lazy dog.

Почнемо з mT5-small. Його переклад значною мірою некоректний. Модель не змогла правильно передати значення багатьох слів («бура» — «blade», «перестрибує» — «spreads», «лінивого» — «linear») і загалом спотворила зміст речення. Спостерігається втрата сенсу та граматичні помилки. Модель mT5-base показує значно кращий переклад, ніж mT5-small. Більшість ключових фраз перекладено правильно: «quick brown fox», «lazy dog». Проте є контекстуальна помилка у прийменнику «через» — «through» замість «over», що спотворює сенс речення. Модель mT5-large створила переклад речення, дуже схожий на еталонний, навіть контекстуально правильно переклала слово «через». Варто зазначити, що mT5-large іноді дає більш якісний переклад, ніж еталон з HPLT, mT5-base зазвичай тримається на рівні еталону, а mT5-small часто показує гірший результат, ніж еталон, хоча і ненабагато.

Ці результати підкреслюють доцільність використання моделей mT5 для українсько-англійського перекладу після тонкого налаштування.

Висновок

Це дослідження було присвячене кількісному вивченню впливу розміру моделі mT5 на точність українсько-англійського перекладу. Ми проаналізували продуктивність трьох версій моделі mT5 — small, base та large — з огляду на час навчання, час генерації перекладів та якість перекладу, використовуючи метрики BLEU та chrF++. Для повнішого контексту результати моделей mT5 було порівняно з показниками дистильованих версій NLLB-200.

Наше тестування підтвердило значний вплив розміру моделі Transformer на точність. Важливо зазначити, що mT5-large показала кращий результат та працювала швидше, ніж навіть більша модель широкої направленості NLLB-200-distilled-1.3B після тонкого налаштування на корпусі у мільйон мовних пар. Це демонструє користь моделей вузького направлення, які спеціалізовані на конкретній мовній парі та завданні й тому можуть перевершувати більш універсальні рішення.

Результати цього дослідження показують доцільність використання моделей mT5 для українсько-англійського перекладу навіть на типових обчислювальних системах.

Список літератури

1. Bahdanau D. Neural Machine Translation by Jointly Learning to Align and Translate [Electronic resource] / D. Bahdanau, K. Cho, Y. Bengio // arXiv. — 2014. — Mode of access: https://arxiv.org/pdf/1409.0473. — Title from screen.
2. Burchell L. An Expanded Massive Multilingual Dataset for High-Performance Language Technologies (HPLT) [Electronic resource] / L. Burchell, O. Gilbert, et al. // arXiv. — 2025. — Mode of access: https://arxiv.org/pdf/2503.10267. — Title from screen.
3. Costa-jussa M. No Language Left Behind: Scaling Human-Centered Machine Translation [Electronic resource] / M. Costa-jussa, J. Cross, et al. // arXiv. — 2022. — Mode of access: https://arxiv.org/pdf/2207.04672. — Title from screen.
4. Glybovets M. Natural Language Processing Using Large Language Models and Machine Learning Methods [Електронний ресурс] / M. Glybovets, D. Zadokhin, et al. // Наукові записки НаУКМА. Комп'ютерні науки. — 2024. — Т. 7. — С. 102–111. — Режим доступу: https://doi.org/10.18523/2617-3808.2024.7.102-111. — Назва з екрана.
5. Och F. J. Statistical Machine Translation. European Association for Machine Translation [Electronic resource] / F. J. Och, H. Ney. — Ljubljana, Slovenia, 2000. 5th EAMT Workshop: Harvesting Existing Resources. — Mode of access: https://aclanthology.org/2000.eamt-1.5.pdf. — Title from screen.
6. Papineni K. Bleu: a Method for Automatic Evaluation of Machine Translation [Electronic resource] / K. Papineni, S. Roukos, et al. — Association for Computational Linguistics. Philadelphia, Pennsylvania, USA, 2002. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. — Pp. 311–318. — Mode of access: https://aclanthology.org/P02-1040.pdf. — Title from screen.

7. Popović M. chrF++: words helping character n-grams [Electronic resource] / M. Popović. — Association for Computational Linguistics. Copenhagen, Denmark, 2017. Proceedings of the Second Conference on Machine Translation. — Pp. 612–618. — Mode of access: https://aclanthology.org/W17-4770.pdf. — Title from screen.
8. Vaswani A. Attention Is All You Need [Electronic resource] / A. Vaswani, N. Shazeer, et al. // arXiv. — 2017. — Mode of access: https://arxiv.org/pdf/1706.03762. — Title from screen.
9. Xue L. mT5: A massively multilingual pre-trained text-to-text transformer [Electronic resource] / L. Xue, N. Constant, et al. // arXiv. — 2020 — Mode of access: https://arxiv.org/pdf/2010.11934. — Title from screen.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate*. arXiv. https://arxiv.org/abs/1409.0473.

Burchell, L., Gilbert, O., Arefyev, N., Aulamo, M., Banon, M., Chen, P., Fedorova, M., Guillou, L., Haddow, B., Haije, J., Helel, J., Hentiksson, E., Klimaszewski, M., Komulainen, V., Kutuzov, A., Kytoniemi, J., Laippala, V., Maehlum, P., Malik, B., ... Zaragoza-Bernabeu, J. (2025). *An expanded massive multilingual dataset for high-performance language technologies (HPLT)*. arXiv. https://arxiv.org/abs/2503.10267.

Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., Sun, A., Wang, S., Wenzek, G., Youngblood, A., Akula, B., Barrault, L., Mejia, G., Hansanti, P., Hoffman, J., ... Wang, J. (2022). *No language left behind: Scaling human-centered machine translation*. arXiv. https://arxiv.or/g/abs/2207.04672.

Glybovets, M., Zadokhin, D., Dekhtiar, B., & Pyechkurova, O. (2024). Natural language processing using large language models and machine learning methods. *NaUKMA Research Papers. Computer Science*, *7*, 102–111. https://doi.org/10.18523/2617-3808.2024.7.102-111.

Och, F. J., & Ney, H. (2000). Statistical machine translation. In *Proceedings of the 5th EAMT Workshop*: *Harvesting Existing Resources*. European Association for Machine Translation. https://aclanthology.org/2000.eamt-1.5.pdf.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311–318). Association for Computational Linguistics. https://aclanthology.org/P02-1040.pdf.

Popović, M. (2017). chrF++: Words helping character n-grams. *In Proceedings of the Second Conference on Machine Translation* (pp. 612–618). Association for Computational Linguistics. https://aclanthology.org/W17-4770.pdf.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. arXiv. https://arxiv.org/abs/1706.03762.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2020). *mT5: A massively multilingual pre-trained text-to-text transformer.* arXiv. https://arxiv.org/pdf/2010.11934.

Z. Makhammedov, O. Kyriienko, V. Tkachenko

EVALUATING MT5 TRANSFORMER MODELS FOR UKRAINIAN-ENGLISH TRANSLATION

This study quantitatively investigates the impact of Transformer architecture size on the accuracy of Ukrainian-to-English machine translation using the multilingual mT5 model. The research evaluates three distinct mT5 versions (small, base, and large) by fine-tuning them on a subset of the HPLT v2 corpus. The technical implementation relied on a standard Python-based deep learning stack, utilizing PyTorch and Hugging Face (Transformers, Datasets) libraries for model management and training. Fine-tuning was executed on a high-performance GPU to handle the significant computational load, while inference speed was benchmarked on a typical consumer-grade GPU to reflect real-world deployment scenarios. Translation quality was assessed using the standard BLEU and chrF++ metrics.

The results confirm a direct correlation between model size and translation quality, with larger models consistently achieving higher scores on both evaluation metrics. This improved accuracy, however, comes at the cost of significantly increased computational demand for both training and inference. Notably, when benchmarked against other large-scale, general-purpose translation models such as NLLB-200 (distilled-600M and distilled-1.3B), the fine-tuned mT5 variants demonstrated superior performance for Ukrainian-English translation, underscoring the benefits of task-specific adaptation. This study confirms the feasibility of using all three mT5 models for this task on typical computing systems, presenting users with a clear trade-off between desired translation quality and available resources.

Keywords: transformer, natural language processing, machine translation, neural machine translation, mT5, HPLT, BLEU, chrF++, NLLB-200.

Матеріал надійшов 27.06.2025

УДК 004.42
DOI: 10.18523/2617-3808.2025.8.102-107

Дубовик А. В., Волинець Є. А.

АВТОМАТИЧНА КЛАСИФІКАЦІЯ ТЕКСТІВ

У цьому дослідженні здійснено аналіз сучасних підходів до класифікації текстової інформації. Особливу увагу приділено автоматичній класифікації текстів, що передбачає їхній розподіл за визначеними категоріями без використання ручного аналізу. Розглянуто й порівняно ефективність різних методів класифікації з акцентом на гібридні системи, які здатні поєднувати переваги окремих підходів і забезпечувати підвищену точність та продуктивність моделей. Також обґрунтовано вибір інструментальних засобів для подальшої програмної реалізації системи автоматизованої класифікації текстів за категоріями. Для навчання моделей запропоновано використовувати збірку AG News Classification Dataset з платформи kaggle.com. Доцільним вважається обмеження класифікаційного процесу комбінацією трьох моделей — Naive Bayes, Support Vector Machine (SVM) та Recurrent Neural Networks (RNN), які вирізняються невисокими вимогами до обчислювальних ресурсів і часу на тренування.

Ключові слова: класифікація текстів, машинне навчання, оброблення української мови, Naive Bayes, SVM, RNN, попереднє оброблення тексту.

Вступ

Оброблення текстової інформації має суттєве значення в різних сферах знань сучасного світу, таких як бізнес, наука, соціальні медіа та інші. З розвитком інформаційних технологій і зростанням обсягу доступної інформації стає надзвичайно важливим ефективне управління цими даними та їхній аналіз. Однією з ключових задач у зазначеній сфері є автоматична класифікація текстів, яка полягає в їхньому розподілі за певними категоріями без використання при цьому ручного аналізу (під «категорією» може матися на увазі тема, почуття, мова тексту тощо). Це завдання може бути важливим для багатьох установ під час створення архівів і організації великих обсягів документів. Воно також стає дедалі актуальнішим з розвитком Інтернету, де величезні обсяги різноманітної інформації потребують швидкого та точного аналізу. Зазначимо при цьому, що інформаційні системи, які використовують технології машинного навчання, такі як нейронні мережі та методи оброблення природної мови (NLP), демонструють високу ефективність у розв'язанні цієї задачі завдяки своїй здатності адаптуватися до нових даних і високій швидкості оброблення [1; 4; 8; 20].

Основні складнощі, пов'язані з автоматичною класифікацією текстів, це різноманітність форматів, структур текстів і семантична складність мови, і для їхнього розв'язання потрібні ефективні алгоритми та методи оброблення природної мови.

Основні поняття

Класифікація є процесом розподілу об'єктів або даних на певні категорії або класи відповідно до їхніх характеристик або властивостей. У контексті оброблення текстів класифікація полягає у призначенні кожному текстовому документу або фрагменту відповідної категорії або класу, що допомагає організувати, проаналізувати та зрозуміти великі обсяги інформації [4]. До XX століття класифікацію текстів здійснювали виключно вручну. У XIX столітті, з поширенням бібліотек та друкарства, було створено системи класифікації книг, такі як десяткова класифікація Дьюї та класифікація Бібліотеки Конгресу.

Потреба у класифікації текстів стала ще більш актуальною з появою комп'ютерів, які дали змогу автоматизувати процес обробки текстів і розробити алгоритми для автоматичної класифікації текстових даних. Перші комп'ютерні системи для класифікації текстів з'явилися ще в другій половині XX століття, але вони були дуже примітивними порівняно з сучасними системами. Технології штуч-

ного інтелекту, зокрема методи машинного навчання, дозволили значно покращити ефективність та точність класифікації текстів.

Автоматична класифікація тексту є одним із фундаментальних завдань оброблення природної мови, і її дослідження матиме важливі наслідки у багатьох сферах життя. Наприклад, у наукових дослідженнях відсутність ефективних засобів для аналізу великих обсягів наукових публікацій може ускладнити синтез знань і виявлення тенденцій розвитку науки. У сфері бізнесу автоматична класифікація текстів може допомогти під час розподілу замовлень або при аналізі ринку, що сприятиме прийняттю обґрунтованих управлінських рішень. Також у сфері соціальних мереж і медіа класифікація текстових повідомлень може допомогти під час аналізу громадської думки та виявлення суспільних тенденцій.

Усі системи автоматичної класифікації текстів можна умовно розділити на такі три групи: системи на основі правил, системи на основі машинного навчання, гібридні системи.

Одним із найпростіших та найбільш зрозумілих методів класифікації текстів є класифікація за допомогою систем на основі правил. Ці системи базуються на використанні набору правил, які визначають спосіб прийняття рішень щодо класифікації текстових даних [19]. Прикладом може бути проста система фільтрації спаму електронної пошти, яка відносить повідомлення, насичені словами «акція», «знижка», «продаж», «виграш», «лотерея», до спаму. Хоча цей підхід може спершу здатися доволі обмеженим порівняно з більш складними методами машинного навчання, він все ще знаходить своє застосування у деяких випадках. Перевагою систем на основі правил є їхня прозорість: ви легко можете зрозуміти, які правила були застосовані до конкретного тексту під час класифікації. Однак недоліком цих систем є їхня обмежена гнучкість, а також відносна складність побудови та підтримки, особливо якщо застосовувати зазначені системи для обробки великих обсягів текстових даних.

У системах класифікації текстів на основі машинного навчання використовуються різноманітні алгоритми та моделі для автоматичного визначення категорій або класів, до яких належать тексти [19]. Ці системи зазвичай навчаються на зразках текстів з уже відомими мітками категорій, які дозволяють зрозуміти певні закономірності у вхідних даних та правильно класифікувати нові зразки. Для класифікації текстів найбільш широко застосовують такі рішення, як Naive Bayes, SVM (Support Vector Machine), RNN (Recurrent Neural Network), BERT (Bidirectional Encoder Representations from Transformers), LLM (Large Language Models).

Naive Bayes — це проста техніка для побудови класифікаторів, що ґрунтується на теоремі Баєса та припущенні про незалежність ознак [16]. Наприклад, якщо фрукт зелений або червоний, круглої форми й має радіус близько п’яти сантиметрів, то його можна вважати яблуком — саме до цього висновку приводить використання наївного баєсового класифікатора, який при визначенні ймовірності того, що цей фрукт є яблуком, розглядає кожну ознаку як незалежну від інших. Перевагами методу Naive Bayes є його ефективність для великих обсягів даних, простота реалізації та швидкість. Недоліком є те, що припущення про незалежність ознак не завжди може бути справедливим, і в цих випадках метод показуватиме нижчу ефективність порівняно з іншими методами.

Для текстової класифікації найчастіше використовується мультиноміальний баєсів класифікатор, в якому розподіл є параметризованим векторами $\theta_y = (\theta_{y1}, \dots, \theta_{yn})$ для кожного класу y, де n — це кількість ознак (у цьому випадку розмір словника) та θ_{yi} — це вірогідність $P(x_i|y)$ ознаки i з’явитись в екземплярі класу y. Параметри θ_{yi} оцінюються підрахунком відносної частоти:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \ ,$$

де $N_{yi} = \sum_{x \in T} x_i$ — це кількість разів, коли ознака i трапляється в класі y в тренувальному наборі даних T, і $N_y = \sum_{i=1}^{n} N_{yi}$ — це загальна кількість всіх ознак в класі y. Попереднє згладжування $\alpha \geq 0$ враховує функції, яких немає у зразках навчання, і запобігає нульовій ймовірності в подальших обчисленнях. Встановлення $\alpha = 1$ називають згладжуванням Лапласа, $\alpha < 1$ — згладжуванням Лідстоуна [19].

SVM — алгоритм, який шукає в просторі ознак гіперплощину, що найкращим чином розподіляє екземпляри по різних категоріях [16]. Перевагою такого методу є ефективність при роботі з розрідженими даними, а недоліком є те, що за великої кількості зразків для тренування або великої кількості ознак алгоритм може стати обчислювально-вимогливим і також може демонструвати гірші, ніж інші алгоритми, результати при обробленні великих корпусів текстів.

Загалом SVM працює таким чином. Ми маємо набір n точок із тренувального набору даних (x_1, y_1), ... (x_n, y_n), де y_i (1 або –1) вказує на належність точки x_i до певного класу. Кожен x_i — це p-розмірний вектор, і SVM шукає гіперплощину, яка з максимальним запасом розділяє групу точок x_i, для яких $y_1 = 1$, від тих точок, для яких $y_1 = -1$.

RNN — рекурентна нейронна мережа, що може ефективно працювати з послідовностями даних, такими як рядки тексту, і враховувати контекстуальні зв'язки між словами [4]. Перевагою мереж такого типу є здатність до врахування контексту, що може бути корисним для визначення залежностей між словами та фразами в реченні. Проте оброблення довгих текстових послідовностей для RNN може більш обчислювально-затратною операцією, ніж для Naive Bayes та SVM. Також рекурентні нейронні мережі мають проблему затухання градієнта і можуть погано зв'язувати інформацію при великій відстані між словами. Як вихід із такої ситуації в 1997 р. було запропоновано модель LSTM, якій протягом останніх років удалось набагато покращити можливості RNN. Наприклад, повідомляється, що у 2015 р. розпізнавання мовлення від Google значно покращилось завдяки використанню LSTM [15].

BERT — це одна з найпродуктивніших моделей у сфері оброблення природної мови, що базується на трансформерах [5]. Модель зазвичай попередньо навчається на великому корпусі текстів, а потім підлаштовується для виконання конкретних завдань. Основною перевагою BERT є здатність розуміти контекстне представлення слів і його масштабованість. Недоліками моделі можна вважати потребу в значних обчислювальних ресурсах для тренування та підлаштування, а також складність її налаштування порівняно з більш традиційними методами.

LLM — моделі, які базуються на нейронних мережах із великою кількістю параметрів та навчаються на великих обсягах текстових даних із метою розуміти та генерувати контент [12]. Перевагою таких моделей є їхня гнучкість, адже уже натреновані LLM здатні визначити дуже велику кількість категорій тексту. Проте значним недоліком моделей такого типу є використання потужної обчислювальної машини для роботи та необхідність надзвичайно великого обсягу даних для тренування у випадку, якщо ми створюємо власну модель, а не використовуємо наявну. Деякі відомі LLM — це моделі серії GPT від OpenAI (наприклад, GPT-3.5 і GPT-4, що використовуються в ChatGPT і Microsoft Copilot), PaLM і Gemini від Google (останній сьогодні використовується в однойменному чат-боті), xAI Grok, сімейство моделей LLaMA від Meta. Але саме браузерний ChatGPT 2022 року, орієнтований на споживачів, захопив уяву широких верств населення та викликав ажіотаж у ЗМІ [3].

Гібридні системи класифікації текстів можуть поєднувати переваги різних підходів і методів для створення більш точних та ефективних моделей [2]. Ці системи можуть використовувати комбінації всіх інших методів для досягнення кращих результатів. Вони становлять потужний інструмент, який може бути налаштований та оптимізований для конкретних потреб. Такі моделі досягають високої точності та ефективності у різних сценаріях та сферах застосування. Отже, гібридні системи класифікації текстів є ефективним інструментом, який можна використовувати для різноманітних завдань у багатьох галузях, від аналізу текстових даних до автоматичної обробки природної мови. Проте один з основних недоліків гібридних систем полягає в їхній складності. Поєднання різних методів і підходів може призвести до значного ускладнення самої системи — як її розробки, так і підтримки та масштабування.

Отже, потрібно розробити систему, яка зможе доволі швидко та точно класифікувати тексти. У користувача має бути можливість зручним способом натренувати систему на власних даних та налаштувати її параметри для отримання оптимальних результатів. Система буде попередньо навчена на певному наборі даних (наприклад, AG News Classification Dataset), що дозволить одразу розпізнавати деякі базові категорії текстів.

Аргументація вибору основних інструментів розробки

Для того щоб ефективно опрацювати вхідні дані та реалізувати алгоритм класифікації, необхідно насамперед визначити мову програмування та основні бібліотеки, які зможуть надати необхідний функціонал.

На нашу думку, тут найбільше підходить мова програмування Python, оскільки саме для цієї мови вже сформована екосистема бібліотек машинного навчання, оброблення даних, матричної

математики тощо. Синтаксис Python є лаконічним і зрозумілим, що в деяких випадках дозволяє реалізовувати більш складні алгоритми швидше та з меншими зусиллями. Великий вибір різноманітних бібліотек, простота та зручність використання, а також всі інші згадані фактори роблять Python стандартним вибором для проєктів, що стосуються машинного навчання та нейронних мереж.

Основними бібліотеками з наявним функціоналом ми обираємо такі. TensorFlow — безкоштовна бібліотека для машинного навчання з відкритим кодом [18]. Вона відома своєю масштабованістю та високою продуктивністю, що дозволяє легко створювати та навчати складні нейронні мережі. Ще одна бібліотека для машинного навчання — scikit-learn, вона надає простий та зрозумілий інтерфейс для реалізації класичних алгоритмів [16]. Ця бібліотека містить великий набір інструментів для класифікації та вирішення інших завдань. Бібліотеку nltk (Natural Language Toolkit) використовують для опрацювання природної мови [10]. Вона надає широкий спектр інструментів для реалізації різноманітних завдань, пов'язаних з обробленням природної мови. Також корисними будуть бібліотеки NumPy [11] — для зручного доступу до великих і багатовимірних масивів, а також функцій високого рівня для роботи з такими масивами; Pandas [13] — для використання швидкої, гнучкої та інтуїтивно зрозумілої табличної структури даних DataFrame; Matplotlib [9] та seaborn [17] — для візуалізації даних за допомогою графіків різних типів.

Цей список засобів не є вичерпним, адже для деяких задач можна використати також інші бібліотеки з аналогічним функціоналом.

Висновки

У цьому дослідженні проаналізовано задачу класифікації текстів за категоріями з наголосом на оброблення текстів українською мовою. Обґрунтовано інструментарій розроблення майбутньої програмної реалізації автоматичної системи класифікації текстів за категоріями.

Для навчання моделей можна застосувати AG News Classification Dataset з сайту kaggle.com [6]. Всього цей датасет містить 120 тисяч коротких текстів, зібраних з різних новинних джерел. Щоб робота з даними була більш ефективною, потрібно розробити модуль, який виконував би попереднє обробленння текстів.

Для вирішення нашої задачі ми вважаємо недоцільним підхід із використанням систем на основі правил, адже ми не зможемо адаптувати старі правила для нових категорій у випадку, якщо користувач натренує модель на власних текстах. Тому для класифікації ми пропонуємо обмежитися використанням комбінації трьох моделей: Naive Bayes, SVM та RNN. Вони не потребують потужних обчислювальних ресурсів та великої кількості часу для тренування.

Важливим при цьому має бути можливість використовувати будь-яку комбінацію моделей для тренування на користувацьких текстах. Так само і для класифікації повинна бути можливість використовувати будь-яку комбінацію натренованих моделей.

Список літератури

1. Глибовець А. М. Програмна система перевірки на плагіат українських текстів / А. М. Глибовець, М. О. Бікчентаєв // Наукові записки НаУКМА. Комп'ютерні науки. — 2022. — Т. 5. — https://doi.org/10.18523/2617-3808.2022.5.16-25.
2. A Hybrid Text Classification Approach for Analysis of Student Essays [Electronic resource] / C. P. Rosé, A. Roque, D. Bhembe, K. Vanlehn. — Mode of access : https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7f7e133f636308b5f67600ad321335f716a7a14a.
3. ChatGPT a year on: 3 ways the AI chatbot has completely changed the world in 12 months [Electronic resource]. — Mode of access: https://www.euronews.com/next/2023/11/30/chatgpt-a-year-on-3-ways-the-aihttps://www.euronews.com/next/2023/11/30/chatgpt-a-year-on-3-ways-the-ai-chatbot-has-completely-changed-the-world-in-12-monthschatbot-has-completely-changed-the-world-in-12-months.
4. Gasparetto A. A Survey on Text Classification Algorithms: From Text to Predictions [Electronic resource] / A. Gasparetto, M. Marcuzzo, A. Zangari, A. Albarelli. — Mode of access : https://www.mdpi.com/2078-2489/13/2/83.
5. Google voice search: faster and more accurate [Electronic resource]. — Mode of access: https://research.google/blog/google-voice-search-fasterhttps://research.google/blog/google-voice-search-faster-and-more-accurateand-more-accurate.
6. Kaggle [Electronic resource]. — Mode of access: https://www.kaggle.com.
7. Large language model (LLM) [Electronic resource]. — Mode of access: https://www.growthloop.com/university/article/llm.
8. Machine Learning Glossary [Electronic resource]. — Mode of access: https://developers.google.com/machine-learning/glossary.
9. Matplotlib [Electronic resource]. — Mode of access: https://matplotlib.org.
10. Natural Language Toolkit [Electronic resource]. — Mode of access: https://www.nltk.org.
11. NumPy [Electronic resource]. — Mode of access: https://numpy.org.

12. Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing [Electronic resource]. — Mode of access: https://research.google/blog/open-sourcing-bert-state-of-the-art-pre-traininghttps://research.google/blog/open-sourcing-bert-state-of-the-art-pre-training-for-natural-language-processingfor-natural-language-processing.
13. Pandas [Electronic resource]. — Mode of access: https://pandas.pydata.org.
14. Scikit-learn: Naive Bayes [Electronic resource]. — Mode of access: https://scikit-learn.org/stable/modules/naive_bayes.html.
15. Support vector machine [Electronic resource]. — Mode of access: https://en.wikipedia.org/wiki/Support_vector_machine.
16. Scikit-learn [Electronic resource]. — Mode of access: https://scikit-learn.org/stable.
17. Seaborn [Electronic resource]. — Mode of access: https://seaborn.pydata.org.
18. TensorFlow [Electronic resource]. — Mode of access: https://www.tensorflow.org.
19. Understanding Text Classification in Python [Electronic resource]. — Mode of access: https://www.datacamp.com/tutorial/text-classificationhttps://www.datacamp.com/tutorial/text-classification-pythonpython.
20. Zie Eya Ekolle. A Generative Learning Model for Heterogeneous Text Classification Based on Collaborative Partial Classifications [Electronic resource] / Zie Eya Ekolle, Ryuji Kohno. — Mode of access: https://www.mdpi.com/2076-3417/13/14/8211.

References

ChatGPT a year on: 3 ways the AI chatbot has completely changed the world in 12 months. (2023). https://www.euronews.com/next/2023/11/30/chatgpt-a-year-on-3-ways-the-aihttps://www.euronews.com/next/2023/11/30/chatgpt-a-year-on-3-ways-the-ai-chatbot-has-completely-changed-the-world-in-12-monthschatbot-has-completely-changed-the-world-in-12-months.

Ekolle, Zie Eya, & Kohno, Ryuji. (2023). *A Generative Learning Model for Heterogeneous Text Classification Based on Collaborative Partial Classifications*. https://www.mdpi.com/2076-3417/13/14/8211.

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). *A Survey on Text Classification Algorithms: From Text to Predictions*. https://www.mdpi.com/2078-2489/13/2/83.

Google voice search: faster and more accurate. (2015). https://research.google/blog/google-voice-search-faster-and-more-accurate.

Hlybovets, A., & Bikchentaev, M. (2022). Prohramna systema perevirky na plahiat ukrainskykh tekstiv. *NaUKMA Research Papers. Computer Science*, *5*, 16–25. https://doi.org/10.18523/2617-3808.2022.5.16-25 [in Ukrainian].

Kaggle. (n. d.). https://www.kaggle.com.

Large language model (LLM). (2024). https://www.growthloop.com/university/article/llm.

Machine Learning Glossary. (n. d.). https://developers.google.com/machine-learning/glossary.

Matplotlib Documentation. (n. d.). https://matplotlib.org.

Natural Language Toolkit Documentation. (n. d.). https://www.nltk.org.

NumPy Documentation. (n. d.). https://numpy.org.

Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing. (2018). https://research.google/blog/open-sourcing-bert-state-of-the-art-pre-traininghttps://research.google/blog/open-sourcing-bert-state-of-the-art-pre-training-for-natural-language-processingfor-natural-language-processing.

Pandas Documentation. (n. d.). https://pandas.pydata.org.

Rosé, C. P., Roque, A., Bhembe, D., & Vanlehn, K. (2003). *A Hybrid Text Classification Approach for Analysis of Student Essay*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7f7e133f636308b5f67600ad321335f716a7a14a.

Scikit-learn Documentation. (n. d.). https://scikit-learn.org/stable.

Scikit-learn: Naive Baye. (n. d.). https://scikit-learn.org/stable/modules/naive_bayes.html.

Seaborn Documentation. (n. d.). https://seaborn.pydata.org.

Support vector machine. (n. d.). https://en.wikipedia.org/wiki/Support_vector_machine.

TensorFlow Documentation. (n. d.). https://www.tensorflow.org.

Understanding Text Classification in Python. (2022). https://www.datacamp.com/tutorial/text-classificationhttps://www.datacamp.com/tutorial/text-classification-pythonpython.

A. Dubovyk, Y. Volynets

AUTOMATIC TEXT CLASSIFICATION

This study explores modern methodologies in the field of automatic text classification, a critical task in natural language processing (NLP) that enables the categorization of unstructured textual data into predefined groups without manual intervention. The rapid growth of digital text across domains such as business, media, science, and social networks has created a pressing need for scalable and accurate classification systems. The research provides an analytical overview of three primary approaches: rule-based systems, machine learning methods, and hybrid models. Particular attention is paid to evaluating the strengths and limitations of several popular machine learning algorithms, including Naive Bayes, Support Vector Machines (SVM), and Recurrent Neural Networks (RNN). While advanced techniques such as BERT and Large Language Models (LLMs) demonstrate high performance, they are not considered optimal for lightweight, user-trainable applications due to their high computational costs.

To support practical implementation, the study proposes a system architecture based on the Python programming language and a suite of supporting libraries (e.g., TensorFlow, scikit-learn, NLTK, NumPy,

Pandas, Matplotlib, and Seaborn). The AG News Classification Dataset is recommended as the initial training corpus, providing a robust foundation for multi-class categorization tasks.

The final system design emphasizes modularity and user configurability. It allows end users to preprocess their own text data, train classification models on domain-specific content, and utilize combinations of models to improve performance. The research recommends a model ensemble consisting of Naive Bayes, SVM, and RNN due to their balance between effectiveness and computational efficiency.

This study not only highlights the technical viability of automated text classification systems but also presents a practical, extensible framework suitable for real-world applications, especially for underrepresented languages such as Ukrainian. The resulting system aims to bridge the gap between academic research and deployable technology, offering a customizable platform for tasks ranging from document organization and content filtering to sentiment analysis and market research.

Keywords: text classification, machine learning, Ukrainian language processing, Naive Bayes, SVM, RNN, text preprocessing.

Матеріал надійшов 14.06.2025

УДК 004.932
DOI: 10.18523/2617-3808.2025.8.108-112

Бучко О. А., Плахотна Д. О.

ПОРІВНЯННЯ АРХІТЕКТУР НЕЙРОННИХ МЕРЕЖ ДЛЯ СЕГМЕНТАЦІЇ ПУХЛИН МОЗКУ

У роботі досліджено ефективність сучасних архітектур глибокого навчання для сегментації медичних зображень у задачі виявлення пухлин мозку. Проведено порівняльний аналіз моделей FCN, DeepLabv3+, PSPNet та Attention U-Net. Окрему увагу приділено впливу попереднього самоконтрольованого навчання на якість сегментації в умовах обмеженої кількості розмічених даних. Для оцінювання результатів використано метрику подібності.

Ключові слова: сегментація зображень, глибоке навчання, нейронні мережі, Attention U-Net, BraTS2020, самоконтрольоване навчання.

Вступ

Сегментація зображень є однією з фундаментальних задач комп'ютерного зору, що полягає у поділі зображення на семантично однорідні області. Залежно від типу поставленої задачі, сегментація може мати різні форми: від знаходження фону та об'єкта до точного піксельного виділення багатьох класів. Особливу актуальність ця задача набуває в галузі медицини, де автоматизоване оброблення візуальних даних дає можливість покращити точність діагностики та прискорити процес ухвалення клінічних рішень.

Традиційні алгоритми сегментації, основані на простих евристиках, таких як порогове оброблення, виявлення контурів або кластеризація, мають низьку стійкість до змін у структурі даних, неоднорідності освітлення, шумів або варіативності об'єктів. Їхня ефективність різко падає в умовах, де спостерігається складна морфологія, як, наприклад, у випадках сегментації пухлин головного мозку за МРТ-знімками.

З огляду на це зростає потреба у використанні методів, здатних адаптуватися до різноманітних сценаріїв, ураховувати як локальні, так і глобальні закономірності, а також навчатися з великих обсягів розмічених даних. Саме такі властивості реалізуються в сучасних підходах, що базуються на глибоких згорткових нейронних мережах (CNN), які сьогодні становлять основу комп'ютерного зору.

У цій статті представлено результати експериментального порівняння кількох сучасних сегментаційних архітектур глибокого навчання, а також досліджено вплив попереднього самоконтрольованого навчання енкодера на якість сегментації. За мету було поставлено визначення архітектурно-тренувальних факторів, які найбільше впливають на точність, узагальнюваність та обчислювальну ефективність моделі сегментації в умовах обмежених ресурсів та доступу до анотованих даних.

Різновиди сучасних підходів до сегментації зображень

Сучасні підходи до сегментації зображень базуються насамперед на глибоких згорткових нейронних мережах, що забезпечують автоматичне вилучення багаторівневих ознак і точне піксельрівне розмежування об'єктів [6]. Еволюцію методів доцільно розглядати крізь призму архітектурних парадигм, які розв'язують типові проблеми: низьку роздільну здатність глибоких представлень, втрату просторового контексту та брак анотованих даних у спеціалізованих доменах, зокрема в медицині [7].

Повністю згорткові мережі (англ. fully convolutional networks, FCN) стали першими моделями, що усунули повнозв'язні шари класичних CNN, здійснивши пряме згорткове перетворення зображення у карту класів [10]. Використання транспонованих згорток (англ. upsampling) дозволило відновлювати початкову роздільну здатність, однак через відсутність симетричного декодера FCN схильні до розмивання меж об'єктів, критичних у медичній діагностиці.

Найширше застосування отримала енкодер-декодерна архітектура, зокрема модель U-Net, яка поєднує контракційну гілку з розширювальною та реалізує прямі пропуски (англ. skip connections), що компенсують втрату низькорівневих деталей [1]. Завдяки здатності ефективно навчатися на малих вибірках U-Net стала «золотим стандартом» для медичних томографічних даних [4], зокрема для набору BraTS2020, використаного у нашому дослідженні [2]. Її модифікації, як-от Feature Pyramid Network (FPN) та SegNet, удосконалюють багаторівневе представлення ознак або підвищують ефективність декодування шляхом збереження індексів підсумплювання [5].

Проблема обмеженого сприймального поля без втрати роздільної здатності вирішується за допомогою розширених згорток (англ. dilated convolutions) та модуля Atrous Spatial Pyramid Pooling (ASPP), реалізованого в архітектурі DeepLabv3+ [3]. Остання поєднує ASPP із енкодер-декодером і використовує глибокі сепарабельні згортки, що забезпечує збалансованість між локальною точністю та глобальним контекстом.

Альтернативний спосіб урахування різнорівневого контексту реалізує Pyramid Scene Parsing Network (PSPNet), що агрегує глобальні статистики через пірамідальний пулінг [12]. У високоточних клінічних сценаріях PSPNet дає змогу врахувати складну топологію пухлин, хоча поступається DeepLabv3+ за деталізацією дрібних структур.

Високоточні задачі, які вимагають субпіксельної локалізації, мотивували розроблення High-Resolution Network (HRNet), яка підтримує паралельні гілки з різною дискретизацією та поступовим об'єднанням ознак [11]. Цей підхід дає змогу досягти чіткого збереження геометричних меж, що є критичним при визначенні, наприклад, інвазивних країв пухлин.

Для підсилення дискримінативності було запропоновано механізми уваги (англ. attention mechanisms). Attention U-Net, що використовується в нашій роботі, вводить модулі просторової уваги, які пригнічують нерелевантні регіони й підсилюють семантично значущі [9]. Подальший розвиток призвів до створення гібридних архітектур на основі трансформерів (як-от TransUNet), де self-attention-блоки використовуються для моделювання довготривалих залежностей.

Недостатня кількість розмічених медичних зображень сприяла поширенню самоконтрольованого навчання (англ. self-supervised learning, SSL). Одним із базових підходів є передбачення обертання зображення, тоді як сучасні методи (MoCo, DINO, BYOL) використовують контрастивні втрати для навчання інваріантних до перетворень представлень [7]. У межах нашого дослідження було застосовано самоконтрольоване попереднє навчання енкодера Attention U-Net на основі передбачення кута обертання зображення, що дозволило покращити якість сегментації при обмеженій кількості анотованих даних.

Експериментальна оцінка

Для проведення дослідження було використано відкритий медичний датасет BraTS2020 (Brain Tumor Segmentation Challenge), що містить МРТ-знімки мозку з багатоканальними 3D-об'ємами та відповідними анотованими масками. У дослідженні було задіяно 369 пацієнтів, розділених на тренувальну (70 %), валідаційну (15 %) і тестову (15 %) підвибірки на рівні пацієнтів. Для кожного зразка доступні чотири МРТ-послідовності: T1, T2, T1ce та FLAIR, що дають змогу охопити різні морфологічні характеристики пухлинної ділянки.

На етапі підготовки даних було здійснено нормалізацію значень піксельної інтенсивності кожного каналу в діапазоні [0, 1]. Сегментаційні маски, представлені в датасеті BraTS2020 у формі трьох бінарних каналів, де кожен канал відповідає окремому типу пухлинної тканини, було перетворено в одноканальний формат. У цьому форматі кожному пікселю надавалося одне з чотирьох значень: 0 — фон, 1 — некротична ділянка, 2 — набряк, 3 — активна пухлина. Для формування навчального набору було використано двовимірні аксіальні зрізи з тривимірних об'ємів.

Для сегментації було обрано чотири сучасні архітектури згорткових нейронних мереж: FCN, DeepLabv3+, PSPNet та Attention U-Net, реалізовані за допомогою бібліотеки Segmentation Models PyTorch. Обрані архітектури представляють різні підходи до оброблення просторового контексту: ASPP у DeepLabv3+, пірамідальний пулінг у PSPNet та просторові модулі уваги в Attention U-Net.

Мережі було навчено на GPU з використанням оптимізатора Adam, функції втрат у вигляді суми Dice Loss та зваженої CrossEntropy Loss, а також з агресивним застосуванням аугментацій: випадкового повороту, горизонтального відображення, зміни яскравості та випадкового масштабування.

Приклад вхідного зрізу та відповідної маски наведено на рис. 1.

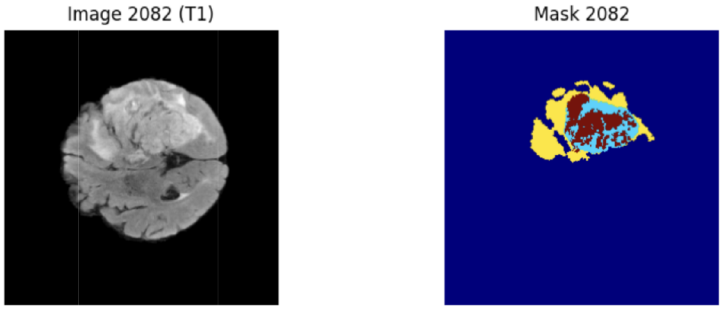


Рис. 1. Приклад МРТ-зрізу (T1) та відповідної сегментаційної маски з датасету BraTS2020

Кількісну оцінку якості сегментації було здійснено за допомогою коефіцієнта подібності кубиків (англ. Dice Similarity Coefficient) [8], визначеного за формулою:

$$Dice = \frac{2TP}{2TP+FP+FN}, \quad (1)$$

де *TP* — кількість пікселів, правильно класифікованих як позитивні, *FP* — кількість хибнопозитивних пікселів, *FN* — кількість хибнонегативних пікселів. Ця метрика є чутливою до якості локалізації об'єкта та широко використовується у медичних задачах.

Для оцінювання якості сегментації було враховано чотири класи: фон, некроз, перитуморальний набряк і активна пухлина (табл. 1). Для кожної моделі було обчислено середні значення коефіцієнта Dice за кожним класом на тестовій вибірці.

Таблиця 1. **Середні значення Dice-коефіцієнта для кожного класу (BraTS2020, тестова вибірка)**

Модель	Фон (0)	Некроз (1)	Набряк (2)	Активна пухлина (3)	Середнє значення
FCN	0.975	0.744	0.802	0.763	0.821
Attention U-Net	0.982	0.784	0.843	0.813	0.856
DeepLabv3+	0.978	0.767	0.828	0.796	0.842
PSPNet	0.974	0.719	0.794	0.755	0.811

Як видно з таблиці, Attention U-Net досягла найвищих показників Dice-коефіцієнта як для кожного окремого класу, так і в середньому. Найбільший приріст спостерігався у класах активної пухлини та набряку, що є критичними для клінічної оцінки.

Особливої уваги заслуговує високий показник для класу фону (0) у всіх моделей (>0.97), що свідчить про низький рівень хибнопозитивних спрацювань поза межами пухлинної області. Водночас відносно нижчі значення для некротичної тканини (1) підтверджують, що саме ця ділянка є найскладнішою для сегментації через розмитість меж та внутрішню неоднорідність.

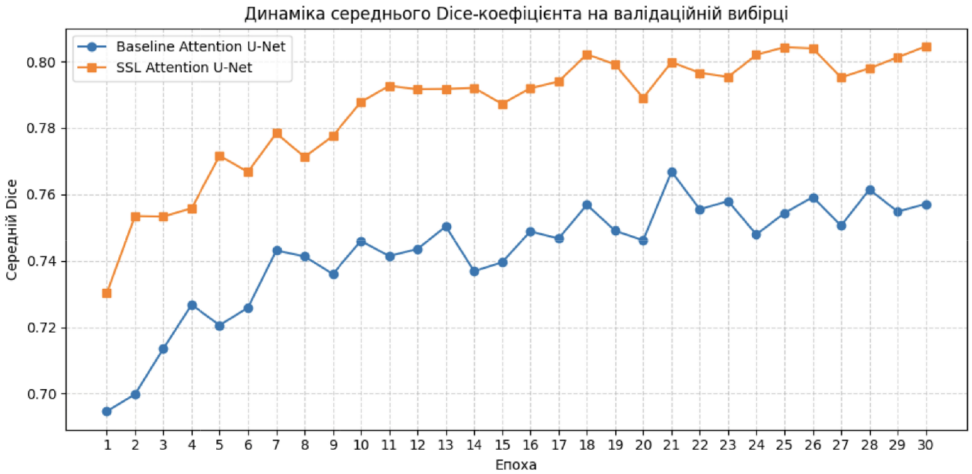


Рис. 2. Динаміка середнього Dice-коефіцієнта на валідаційній вибірці для Attention U-Net з і без самоконтрольованого попереднього навчання

Із метою підвищення узагальнюваності моделі Attention U-Net в умовах обмеженої кількості анотованих прикладів було застосовано попереднє самоконтрольоване навчання енкодера на основі завдання передбачення кута обертання (0°, 90°, 180°, 270°). Це дозволило моделі попередньо навчитися розпізнавати базові структурні особливості зображень без використання анотованих міток.

Після подальшого навчання на задачі сегментації середній Dice-коефіцієнт зріс з 0.856 до 0.879. Покращення виявлено насамперед у складних класах. Динаміку зміни Dice-коефіцієнта на валідаційній вибірці протягом 30 епох наведено на рис. 2, що демонструє стабільну перевагу SSL-ініціалізованої моделі над базовою.

Висновки

Із проведеного дослідження можна зробити висновок, що застосування сучасних архітектур глибокого навчання дає змогу досягти високої точності сегментації медичних зображень навіть в умовах обмежених анотованих даних. Найкращі результати було отримано для моделі Attention U-Net, зокрема у варіанті з попереднім самоконтрольованим навчанням, що підтверджує ефективність таких підходів у задачі сегментації зображень.

Переваги таких моделей відображено у кількісних метриках Dice, які перевищують показники класичних архітектур, а також у візуальній якості сегментацій. Разом із тим слід враховувати обчислювальну складність та вимоги до апаратного забезпечення, що може обмежувати практичне використання деяких моделей у клінічних умовах.

Використання нейромережевих методів сегментації, зокрема з урахуванням контексту та попереднього навчання, є перспективним напрямом для подальшого розвитку автоматизованих систем медичної діагностики, але потребує подальшої оптимізації та перевірки на більш різноманітних даних.

Список літератури

1. Badrinarayanan V. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation [Electronic resource] / Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2017. — Vol. 39, no. 12. — Pp. 2481–2495. — https://doi.org/10.1109/tpami.2016.2644615.
2. BraTS2020 Dataset [Electronic resource]. — Mode of access: https://www.med.upenn.edu/cbica/brats2020/.
3. Chen L.-C. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs [Electronic resource] / Liang-Chieh Chen [et al.] // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2018. — Vol. 40, no. 4. — Pp. 834–848. — https://doi.org/10.1109/tpami.2017.2699184.
4. Çiçek Ö. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation [Electronic resource] / Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, Olaf Ronneberger // Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016. — Cham, 2016. — Pp. 424–432. — https://doi.org/10.1007/978-3-319-46723-8_49.
5. Drozdzal M. The Importance of Skip Connections in Biomedical Image Segmentation [Electronic resource] / Michal Drozdzal [et al.] // Deep Learning and Data Labeling for Medical Applications. — Cham, 2016. — Pp. 179–187. — https://doi.org/10.1007/978-3-319-46976-8_19.
6. Haralick R. M. Image Segmentation Techniques [Electronic resource] / Robert M. Haralick, Linda G. Shapiro // Computer Vision, Graphics, and Image Processing. — 1984. — Vol. 27, no. 3. — P. 389. — https://doi.org/10.1016/0734-189x(84)90043-4.
7. Image Segmentation Techniques: A Comprehensive Review [Electronic resource] // International Research Journal of Modernization in Engineering Technology and Science. — 2024. — https://doi.org/10.56726/irjmets49604.
8. McGurk R. TH-C-WAB-08: Modeling of the Dice Coefficient for PET Segmentation Studies [Electronic resource] / R. McGurk [et al.] // Medical Physics. — 2013. — Vol. 40, no. 6, part 32. — P. 538. — https://doi.org/10.1118/1.4815765.
9. Ronneberger O. U-Net: Convolutional Networks for Biomedical Image Segmentation [Electronic resource] / Olaf Ronneberger, Philipp Fischer, Thomas Brox // Lecture Notes in Computer Science. — Cham, 2015. — Pp. 234–241. — https://doi.org/10.1007/978-3-319-24574-4_28.
10. Shelhamer E. Fully Convolutional Networks for Semantic Segmentation [Electronic resource] / Evan Shelhamer, Jonathan Long, Trevor Darrell // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2017. — Vol. 39, no. 4. — Pp. 640–651. — https://doi.org/10.1109/tpami.2016.2572683.
11. Sun K. Deep High-Resolution Representation Learning for Human Pose Estimation [Electronic resource] / Ke Sun [et al.] // 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, CA, USA, 15–20 June 2019. — [S. l.], 2019. — https://doi.org/10.1109/cvpr.2019.00584.
12. Zhao H. Pyramid Scene Parsing Network [Electronic resource] / Hengshuang Zhao [et al.] // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 21–26 July 2017. — [S. l.], 2017. — https://doi.org/10.1109/cvpr.2017.660.

References

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39* (12), 2481–2495. https://doi.org/10.1109/tpami.2016.2644615.

BraTS2020 Dataset. (n.d.). *The Cancer Imaging Archive (TCIA)*. https://www.med.upenn.edu/cbica/brats2020/.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 40* (4), 834–848. https://doi.org/10.1109/tpami.2017.2699184.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016* (pp. 424–432). Springer. https://doi.org/10.1007/978-3-319-46723-8_49.

Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., & Pal, C. (2016). The importance of skip connections in biomedical image segmentation. In *Deep Learning and Data Labeling for Medical Applications* (pp. 179–187). Springer. https://doi.org/10.1007/978-3-319-46976-8_19.

Haralick, R. M., & Shapiro, L. G. (1984). Image segmentation techniques. *Computer Vision, Graphics, and Image Processing, 27* (3), 389–405. https://doi.org/10.1016/0734-189x(84)90043-4.

Image segmentation techniques: A comprehensive review. (2024). *International Research Journal of Modernization in Engineering Technology and Science.* https://doi.org/10.56726/irjmets49604.

McGurk, R., Klein, S., Yushkevich, P., & Rohlfing, T. (2013). TH-C-WAB-08: Modeling of the Dice coefficient for PET segmentation studies. *Medical Physics, 40* (6, part 32), 538. https://doi.org/10.1118/1.4815765.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science* (pp. 234–241). Springer. https://doi.org/10.1007/978-3-319-24574-4_28.

Shelhamer, E., Long, J., & Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39* (4), 640–651. https://doi.org/10.1109/tpami.2016.2572683.

Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, June 15–20, 2019. IEEE. https://doi.org/10.1109/cvpr.2019.00584.

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, July 21–26, 2017. IEEE. https://doi.org/10.1109/cvpr.2017.660.

O. Buchko, D. Plakhotna

COMPARISON OF NEURAL NETWORK ARCHITECTURES FOR BRAIN TUMOR SEGMENTATION

Image segmentation plays a crucial role in medical diagnostics, where precise identification of tumor boundaries is essential for treatment planning and prognosis. However, accurate segmentation remains a challenge due to the complexity of anatomical structures and the limited availability of annotated data. Traditional methods are not robust to variability in medical images, which often results in inconsistent and inaccurate outcomes.

This paper investigates the effectiveness of modern neural network-based architectures for brain tumor segmentation using MRI data. The primary goal of the study is to evaluate and compare the accuracy of several convolutional segmentation models under identical training conditions, and to examine whether self-supervised pretraining can improve segmentation quality in cases with limited labeled samples.

The research is based on the BraTS2020 dataset, which contains multi-modal MRI scans with manual annotations of glioma subregions. Five segmentation models were trained: FCN, FPN, PSPNet, DeepLabv3+, and Attention U-Net. All models were evaluated using the Dice Similarity Coefficient. The best result was achieved by Attention U-Net with a mean Dice score of 0.842. A self-supervised learning (SSL) strategy was further applied to pretrain the encoder of this model using a rotation prediction task, which increased the Dice score to 0.869.

The findings confirm that neural network-based methods provide higher segmentation quality compared to classical approaches. More importantly, the integration of SSL enables performance improvements without requiring additional labeled data. This is particularly valuable in the medical field, where collecting expert annotations is expensive and time-consuming.

This article demonstrates that high-quality segmentation of brain tumors is possible even under limited supervision, provided that suitable architectures and training strategies are selected. The presented approach can be adapted for other medical image segmentation tasks and may support the development of practical clinical decision support systems.

Keywords: image segmentation, neural networks, Attention U-Net, self-supervised learning, BraTS2020.

Матеріал надійшов 14.06.2025

УДК 004.04, 004.65
DOI: 10.18523/2617-3808.2025.8.113-117

Зважій Д. В.

ОСОБЛИВОСТІ ІНДЕКСАЦІЇ У PostgreSQL

У статті наведено перелік основних типових індексів, реалізованих у PostgreSQL. Розглянуто можливості їх розширення та вдосконалення з урахуванням бізнес-завдань. Проаналізовано підхід Access Methods API. Описано методи життєвого циклу індексу в PostgreSQL. Також запропоновано інтерфейс для реалізації нового індексу в PostgreSQL на основі суфіксного дерева.

Ключові слова: бази даних, пошук рядків, PostgreSQL, суфіксні дерева, SP-GiST, GiST, хеш-таблиці, Б-дерева, Access Methods API.

Вступ

Індекси є одним із ключових підходів для оптимізації продуктивності в системах управління базами даних (СУБД). Завдяки використанню індексів можна значно пришвидшити пошук, сортування та доступ до даних, зменшуючи кількість операцій читання диска. Під час роботи з великими обсягами даних або використання складних запитів правильне використання індексів має критичне значення для забезпечення ефективної роботи програмних систем.

PostgreSQL, як одна з найбільш поширених СУБД із відкритим вихідним кодом, пропонує широкий набір стандартних типів індексів, які дають можливість вирішувати широкий спектр завдань. Проте специфічні випадки, такі як геопросторовий пошук, оброблення векторних моделей даних або оптимізація пошуку підрядків, можуть потребувати створення власних, спеціалізованих типів індексів.

Одна з визначних рис PostgreSQL полягає в можливості розширення його функціональності. Серед іншого, розробники можуть створювати власні типи індексів, інтегруючи їх у ядро системи через механізм Access Methods API [12]. Це дозволяє гнучко адаптувати базу даних під особливі вимоги застосунку без необхідності змінювати самі дані або логіку їхнього збереження.

У цій статті розглянуто:
– які стандартні індекси підтримує PostgreSQL;
– які можливості PostgreSQL надає для розширення наявних та реалізації власних індексів;
– які методи необхідні для створення власного індексу;
– запропонований набір інтерфейсних методів для індексу на базі суфіксного дерева.

Метою є надати практичне уявлення про те, як індексація в PostgreSQL може бути розширена відповідно до потреб конкретних проєктів, а саме для створення індексу на основі суфіксного дерева.

Вбудовані типи індексів у *PostgreSQL*

PostgreSQL підтримує кілька типів індексів, кожен із яких оптимізований під конкретні види запитів або структури даних. Вибір правильного типу індексу є вирішальним для досягнення високої продуктивності бази даних. Нижче наведено короткий огляд основних типів індексів, які доступні для негайного використання.

Б-дерево (B-tree) [14] є типовим індексом для PostgreSQL і використовується при створенні індексу без явного вказання типу індексу. Для PostgreSQL реалізовано збалансоване Б-дерево, що не має фіксованої кількості елементів у кожному вузлі [3]. Кількість елементів визначається динамічно залежно від того, скільки елементів можна записати у фіксований розмір сторінки PostgreSQL. За замовчуванням розмір сторінки у PostgreSQL дорівнює 8 КВ (8192 байти) [9]. Дизайн цієї структури даних дозво-

ляє підтримувати дерево збалансованим, що забезпечує ефективність операцій =, <, >, BETWEEN, ORDER BY. Цей індекс часто застосовують для індексації числових, рядкових полів, міток часу.

Хеш-таблиця є ще одним типом індексів, який підтримує PostgreSQL. Цей вид індексу підтримує лише оператор рівності. Результатом функції хешування у PostgreSQL є 32-бітове ціле число. Хеш індекс зберігає хеші у відсортованому вигляді, що дає змогу використовувати бінарний пошук для доступу до потрібного елементу [8]. У разі колізій елементи з однаковими хешами зберігаються разом із додатковими даними для їхньої подальшої ідентифікації. Цей тип індексу найкраще підходить для оптимізації операції точного збігу для великих обсягів даних.

GIN (Generalized Inverted Index) — узагальнений інвертований індекс — структура даних, що добре зарекомендувала себе в інформаційному пошуку. Цей індекс дає змогу ефективно працювати з композитними значеннями, які складаються з багатьох елементів. Зазвичай це текстові документи або масиви. Інвертований індекс дозволяє ефективно відповідати на запитання, чи містить документ той чи інший елемент. Узагальненість цієї структури полягає в тому, що її реалізація дозволяє працювати із різними типами даних, орієнтуючись на типові стратегії [6]. Також реалізація цього індексу передбачає розширення власними методами доступу, залежно від бізнес-завдань.

BRIN (Block Range Index) — індекс, що спроєктований для роботи з таблицями великого розміру, значення колонок яких прямо корелюють із їхнім розташуванням у таблиці [7], наприклад відсортовані колонки з датами. Такий індекс має невеликий розмір і не потребує значних зусиль для підтримки. Цей індекс належить до таких, що можуть повертати результати, які не відповідають умові запиту. У такому разі додаткова відповідальність покладається на рівень виконавця запиту: він мусить повторно перевіряти кортежі. Така концепція може бути виправданою, якщо індекс дає змогу уникнути повного перебору величезних таблиць.

GiST (Generalized Search Tree) — узагальнене дерево пошуку — це збалансована деревовидна структура, що слугує взірцем для створення власних індексних стратегій [4]. Вихідний код PostgreSQL також містить кілька готових реалізацій на основі цього індексу. Запропонована реалізація GiST дозволяє імплементувати Б-дерево, R-дерево та багато інших залежно від бізнес-завдання і заданого типу даних. Ядро цього індексу перебирає на себе управління внутрішньою комунікацією з базою даних. Сюди також відносяться вирішення задач багатопоточності, керування блокуваннями, логування та інші. Це дає можливість розробнику зосередитися на розробленні методів доступу, пов'язаних із семантикою запропонованого типу даних.

SP-GiST (Space-partitioned generalized search tree) [5; 10] — узагальнене дерево пошуку із просторовим розбиттям. Ідея цього індексу дуже схожа на GiST, проте основний акцент зроблено на підтримці незбалансованих дерев. До таких структур належать дерева квадрантів [15], k-вимірні дерева [11], префіксні дерева та їхні варіації і багато інших. Одна з важливих характеристик, що об'єднує ці структури даних, це те, що такі структури даних можуть мати різну глибину заглиблення для різних вузлів. SP-GiST також дозволяє створювати нові розширення залежно від завдань та обраного типу даних. Головна ідея цього підходу полягає в тому, що ядро індексу бере на себе взаємодію із PostgreSQL, підтримує виділення пам'яті, реалізовує реагування на блокування. Розробник власного розширення має можливість реалізувати певні функції зворотного виклику, які будуть викликані під час життєвого циклу індексу. Таким чином можна впливати на процес побудови дерева та його обходу для пошуку потрібних елементів у структурі.

Одним з обмежень у дизайні та реалізації SP-GIST індексу є те, що він надає доступ лише до певної частини дерева при вставці елементів. Тому розробник не має можливості додавати елементи, які пов'язані з індексованим кортежем, до інших частини дерева. Така можливість є дуже важливою при побудові суфіксного дерева [16]. Підхід до побудови суфіксного дерева значною мірою залежить від обраного алгоритму та початкових умов, проте жоден із можливих алгоритмів не дозволяє отримати необхідний результат, користуючись рушієм SP-GiST. Така реалізація потребуватиме великої кількості додаткового коду, який багато в чому дублюватиме наявний SP-GiST рушій.

У наступному розділі цієї статті буде запропоновано кроки для реалізації індексу на основі суфіксного дерева в межах можливостей, які доступні у PostgreSQL.

Інтерфейс взаємодії індексу на базі суфіксного дерева з PostgreSQL

Як уже було зазначено у попередньому розділі, архітектура PostgreSQL передбачає можливість для створення власної реалізації для індексування. Рушій бази даних не має знання про алгоритм

індексації чи використані структури даних. Натомість взаємодія відбувається через методи інтерфейсу доступу до індексу [13]. Такий підхід забезпечує незалежність між кодом, що реалізує логіку організації та зберігання сирих даних, і алгоритмом створення та підтримки індексу. На практиці це дає змогу розробнику зосередитися на деталях реалізації нового методу індексації, уникаючи певних особливостей роботи рушія з даними.

Під час ініціалізації індексу PostgreSQL очікує, що функція —обробник індексу у відповідь на аргумент типу internal повертатиме структуру типу IndexAmRoutine, що містить всю необхідну інформацію для роботи з індексом [1]. Найважливіша інформація про поля структури IndexAmRoutine міститься у табл. 1. Крім того, для індексу потрібно задати набір операторів, які він здатний підтримувати. Ця інформація є важливою для планувальника запитів під час формування стратегії виконання запиту.

Таблиця 1. **Основні поля структури IndexAmRoutine у PostgreSQL**

Поле	Короткий опис
ambuild	указник на метод для створення індексу
ambuildempty	указник на метод для побудови порожнього індексу
aminsert	указник на метод для вставки одного кортежу
ambeginscan	указник на метод ініціалізації сканування індексу
amgettuple	метод повертає один підхожий кортеж
amgetbitmap	указник на метод для повернення всіх кортежів, які відповідають критерію пошуку
amrescan	указник на метод для повторного сканування індексу
amendscan	указник на метод для завершення сканування
ambulkdelete / amvacuumcleanup	указник на метод для операції видалення елементів з індексу

У такому разі для підтримки індексу на базі суфіксного дерева обов'язково потрібно забезпечити реалізацію інтерфейсу функцій підтримки доступу. Під час аналізу вихідного коду PostgreSQL було помічено, що найчастіше для префікса методів інтерфейсу використовується ключове слово, яке згодом дає змогу ідентифікувати підхід, що застосовується. Відповідно, реалізації методів індексу на основі суфіксного дерева матимуть префікс "stree".

Варто почати з функції Datum streehandler(PG_FUNCTION_ARGS). Як уже зазначено вище, цей метод є необхідним для взаємодії з основним кодом PostgreSQL. Метод streehandler повертає структуру типу IndexAmRoutine, яка містить усю необхідну інформацію для життєвого циклу індексу.

Однією з обов'язкових частин цієї структури є метод IndexBuildResult * streebuild(Relation heapRelation, Relation indexRelation, IndexInfo *indexInfo). Цей метод відповідає за побудову всіх структур, необхідних для роботи індексу. Йому на вхід приходить два аргументи типу Relation. Аргумент з ім'ям heapRelation — це представлення таблиці, для якої створюється індекс, аргумент indexRelation — дає доступ до об'єкта, доступного для запису, де будуть зберігатися дані індексу на базі суфіксного дерева. Аргумент indexInfo містить необхідну інформацію про сам індекс. Його структура значною мірою перегукується зі структурою вищезгаданого об'єкту IndexAmRoutine. Результатом виконання функції є структура, яка містить статистику щодо кількості вже просканованих кортежів у таблиці, а також кількості кортежів, доданих до індексу.

Також необхідно визначити метод void streebuildempty(Relation indexRelation). Цей метод призначений для ініціалізації порожнього індексу на диску без сканування та вставки кортежів. Він використовується як частина процесу ініціалізації бази даних — ще до того, як будь-які дані були додані. Також цей метод може застосовуватися під час клонування бази даних.

Одним із наступних методів інтерфейсу є bool streeinsert(Relation indexRelation, Datum *values, bool *isnull, ItemPointer ht_ctid, Relation heapRel, IndexUniqueCheck checkUnique, bool indexUnchanged, IndexInfo *indexInfo). Цей метод відповідає за вставку кортежу в індекс. Усередині нього відбувається робота з пам'яттю PostgreSQL, обробка виняткових ситуацій, пов'язаних із блокуваннями, а також виклик основної логіки підтримки суфіксного дерева. На вхід метод приймає вже знайомий об'єкт для запису indexRelation. Аргумент values містить вказівник на рядок із даними. Аргумент isnull — це масив, розмірність якого дорівнює кількості елементів у values; він вказує, чи містить відповідна колонка значення NULL у рядку. Аргумент ht_ctid містить вказівник на ідентифікатор кортежу. На практиці такий підхід дає змогу PostgreSQL підтримувати концепцію MVCC (Multi-Version Concurrency Control), відповідно до якої PostgreSQL не модифікує наявні значення, а створює нові

записи — це дає можливість уникати блокування бази даних і гарантує актуальність інформації. Результатом виконання функції є булеве значення, яке є важливим у разі певних значень аргументу checkUnique. Позитивний результат означає, що вхідне значення завідомо унікальне, а негативний — що значення, передане на вхід, можливо, не є унікальним і потребує перевірки на унікальність. Зазвичай, якщо аргумент checkUnique не має значення, документація PostgreSQL рекомендує повертати негативний результат [2].

Ще одним важливим методом, який обов'язково потрібно реалізувати, є void streecostestimate (PlannerInfo *root, IndexPath *path, double loop_count, Cost *indexStartupCost, Cost *indexTotalCost, Selectivity *indexSelectivity, double *indexCorrelation, double *indexPages). Цей метод відіграє важливу роль для планувальника під час генерації стратегій виконання запитів. Завдання цього методу — надати оцінки, які відображають вартість виконання умови WHERE при використанні індексу. Оцінка враховує різні чинники: наскільки витратно розпочати використання індексу; які будуть загальні витрати; наскільки добре порядок, що задається індексом, співвідноситься з очікуваним порядком результату; а також скільки сторінок індексу, ймовірно, буде прочитано.

Що стосується сканування індексу, то виникає потреба реалізувати підтримку таких методів інтерфейсу:

– IndexScanDesc streebeginscan (Relation indexRelation, int nkeys, int norderbys);
– void streerescan (IndexScanDesc scan, ScanKey keys, int nkeys, ScanKey orderbys, int norderbys);
– bool streegettuple (IndexScanDesc scan, ScanDirection direction);
– int64 streegetbitmap (IndexScanDesc scan, TIDBitmap *tbm);
– void streeendscan (IndexScanDesc scan).

Усі ці методи беруть участь у процесі сканування індексу та пошуку результатів. На першому етапі викликається streebeginscan, який ініціалізує процес сканування та виділяє необхідні ресурси для підтримки його стану. Далі результат streebeginscan — об'єкт, що описує стан сканування, — передається в метод streerescan. У методі streerescan запускається або перезапускається процес сканування, а також задаються нові умови пошуку, наприклад: WHERE name LIKE '%query'. Після цього уже виконується streegettuple або streegetbitmap, де streegettuple повертає true, якщо кортеж був знайдений, та false, якщо ні. Метод streegetbitmap повертає число усіх результатів, що задовольняють умову, і також наповнює ідентифікаторами структуру типу TIDBitmap. Метод streeendscan завершує процес сканування та вивільняє усі використані для сканування ресурси.

Висновок

У статті було описано основні можливості індексації в PostgreSQL. Розглянуто готові індекси, які надає PostgreSQL, можливості їх розширення та ймовірні сценарії використання.

Основну увагу зосереджено на можливостях і інструментах для створення власних індексів, зокрема на Index Access Method Interface [13]. Було проаналізовано ключові методи інтерфейсу, які необхідно реалізувати для створення власного індексу. Також подано аналіз і запропоновано визначення основних методів інтерфейсу доступу для реалізації індексу на основі суфіксного дерева.

Список літератури

1. 62.1. Basic API Structure for Indexes. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/index-api.html.
2. 62.2. Index Access Method Functions. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/index-functions.html.
3. 64.1. B-Tree Indexes. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/btree.html.
4. 64.2. GiST Indexes. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/gist.html.
5. 64.3. SP-GiST Indexes. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/spgist.html.
6. 64.4. GIN Indexes. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/gin.html.
7. 64.5. BRIN Indexes. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/brin.html.
8. 64.6. Hash Indexes. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/hash-index.html.
9. 65.2. TOAST. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/storage-toast.html.
10. Aref W. G. SP-GiST: An Extensible Database Index for Supporting Space Partitioning Trees / W. G. Aref, I. F. Ilyas. — 2001. —https://doi.org/10.1023/A:1012809914301.
11. Bentley J. L. Multidimensional binary search trees used for associative searching / J. L. Bentley // Commun. ACM. — 1975. — Vol. 18 (9). — Pp. 509–517. — https://doi.org/10.1145/361002.361007.

12. Chapter 61. Table Access Method Interface Definition. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/tableam.html.
13. Chapter 62. Index Access Method Interface Definition. PostgreSQL Documentation [Electronic resource]. — Mode of access: https://www.postgresql.org/docs/17/indexam.html.
14. Comer D. Ubiquitous B-Tree / D. Comer // ACM Comput. Surv. — 1979. — Vol. 11 (2). — Pp. 121–137. — https://doi.org/10.1145/356770.356776.
15. Finkel R. A. Quad trees a data structure for retrieval on composite keys / R. A. Finkel, J. L. Bentley // Acta Informatica. — 1974. — Vol. 4 (1). — Pp. 1–9. — https://doi.org/10.1007/BF00288933.
16. Weiner P. Linear pattern matching algorithms / P. Weiner // 14th Annual Symposium on Switching and Automata Theory (Swat 1973). — 1973. — Pp. 1–11. — https://doi.org/10.1109/SWAT.1973.13.

References

62.1. Basic API Structure for Indexes. PostgreSQL Documentation. https://www.postgresql.org/docs/17/index-api.html.
62.2. Index Access Method Functions. PostgreSQL Documentation. https://www.postgresql.org/docs/17/index-functions.html.
64.1. B-Tree Indexes. PostgreSQL Documentation. https://www.postgresql.org/docs/17/btree.html.
64.2. GiST Indexes. PostgreSQL Documentation. https://www.postgresql.org/docs/17/gist.html.
64.3. SP-GiST Indexes. PostgreSQL Documentation. https://www.postgresql.org/docs/17/spgist.html.
64.4. GIN Indexes. PostgreSQL Documentation. https://www.postgresql.org/docs/17/gin.html.
64.5. BRIN Indexes. PostgreSQL Documentation. https://www.postgresql.org/docs/17/brin.html.
64.6. Hash Indexes. PostgreSQL Documentation. https://www.postgresql.org/docs/17/hash-index.html.
65.2. TOAST. PostgreSQL Documentation. https://www.postgresql.org/docs/17/storage-toast.html.
Aref, W. G., & Ilyas, I. F. (2001). *SP-GiST: An Extensible Database Index for Supporting Space Partitioning Trees*. https://doi.org/10.1023/A:1012809914301.
Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, *18* (9), 509–517. https://doi.org/10.1145/361002.361007.
Chapter 61. Table Access Method Interface Definition. (2025, February 20). PostgreSQL Documentation. https://www.postgresql.org/docs/17/tableam.html.
Chapter 62. Index Access Method Interface Definition. PostgreSQL Documentation. https://www.postgresql.org/docs/17/indexam.html.
Comer, D. (1979). Ubiquitous B-Tree. *ACM Comput. Surv.*, *11* (2), 121–137. https://doi.org/10.1145/356770.356776.
Finkel, R. A., & Bentley, J. L. (1974). Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, *4* (1), 1–9. https://doi.org/10.1007/BF00288933.
Weiner, P. (1973). Linear pattern matching algorithms. *14th Annual Symposium on Switching and Automata Theory (Swat 1973)*, 1–11. https://doi.org/10.1109/SWAT.1973.13.

D. Zvazhii

INDEXING FEATURES IN PostgreSQL

This article provides an overview of the main standard index types implemented in PostgreSQL, emphasizing their internal structures, use cases, and efficiency characteristics. The study discusses the possibilities for extending and adapting these index types to meet specific business needs. Special attention is given to PostgreSQL's extensible architecture and the Access Methods API, which enables the creation of custom indexing solutions. The article analyzes the core functions that define the lifecycle of an index in PostgreSQL, including index creation, scan operations, maintenance, and cost estimation procedures. These functions not only structure the interaction between the planner and the index access methods but also open up opportunities for experimental or domain-specific extensions. As a case study of such extensibility, the article proposes an interface for a suffix tree index that utilizes the Access Methods API. The proposed suffix tree index serves as a demonstration of the flexibility provided by PostgreSQL's Access Methods API. The article details how the API's modular architecture allows for the integration of custom data structures, such as suffix trees, by implementing a well-defined set of callback functions governing index creation, scanning, insertion, and cost estimation. This case illustrates how the Access Methods API can be leveraged to expand PostgreSQL's indexing capabilities beyond traditional use cases, making it a powerful framework for experimental development and research-driven optimization of complex query patterns.

Keywords: databases, string search, PostgreSQL, suffix tree, SP-GiST, GiST, hash index, B-tree, Access Methods API.

Матеріал надійшов 30.05.2025

УДК 004.056
DOI: 10.18523/2617-3808.2025.8.118-125

Михайленко О. І., Гороховський К. С., Гороховський С. С.

МЕТОД ШИФРОВАНОЇ КОМУНІКАЦІЇ У СТРАТЕГІЧНИХ ВЗАЄМОДІЯХ

У роботі досліджено можливості застосування наскрізного шифрування в середовищах з обмеженою довірою, зокрема в контексті стратегічних ігор та симуляцій. Незважаючи на широке впровадження таких технологій у сфері цифрової комунікації, їхній потенціал у специфічних сценаріях залишався недостатньо вивченим. Запропоновано новий підхід до захищеної взаємодії між учасниками шляхом адаптації криптографічного протоколу Double Ratchet. Розроблено application-level протокол, оптимізований для ігрових сценаріїв, і вперше реалізовано його компіляцію у WebAssembly, досліджено можливості використання протоколів наскрізного шифрування у браузерному середовищі, а також надання безпеки при зберіганні пар криптографічних ключів поза безпечними середовищами апаратного забезпечення, наведено приклад використання розробленого протоколу для забезпечення приватності у стратегічній комунікації. Проведено оцінку ефективності та безпеки рішення в умовах симульованого середовища з недовірою до сервера. Актуальність дослідження зумовлено зростанням потреби у захищеному зв'язку в умовах кібератак, зокрема в період воєнних дій.

Ключові слова: наскрізне шифрування, Double Ratchet, WebAssembly, стратегічні ігри, WebSocket, Rust, захищена комунікація.

Вступ

Існує багато задач, які потребують безпеки у комунікаційному каналі між двома сторонами. Традиційними методами для створення секретності у комунікації є методи симетричного та асиметричного шифрування, однак використання таких систем має певні недоліки, коли йдеться про системи з мінімальною довірою. До них зокрема можна віднести месенджери та шифровані дзвінки через Інтернет. За останнє десятиліття стрімко набрали популярність протоколи наскрізного шифрування. Через те, що Інтернет перетворювався і продовжує перетворюватися на усе більш централізовану систему, де контроль над даними консолідується у 7–8 компаніях (Meta, Apple, Amazon, Google та ін.), які за будь-якою вимогою держав змушені передавати приватні дані користувачів, виникли підходи, що застосовують комбінацію симетричного та асиметричного шифрування задля забезпечення комунікації двох або більше учасників без розкриття нешифрованих даних сервера, що реалізує передавання цих даних. Таким чином, з'явилися як приватні месенджери на кшталт Signal (раніше Whisper), так і функціональність приватних діалогів у інших популярних месенджерах, як-от WhatsApp та Telegram. Такі месенджери набули широкого використання на найвищих рівнях управління державою, що підтверджує нещодавній скандал із Signal та плануванням військових операцій США [1]. Певні особливості наскрізного шифрування роблять використання такої функціональності обмеженим. Це, в основному, означає втрату даних діалогу у випадку втрати самого девайсу, на якому відбувається комунікація. Іншою важливою проблемою наскрізного шифрування є сувора необхідність використання HSM [12] для безпечного зберігання криптографічних ключів на пристрої. Утім, у реаліях російського вторгнення в Україну потреба у наскрізному шифруванні значно зросла, адже не усі держави можуть забезпечити своїм збройним силам безпечний зв'язок на великих відстанях, який достатньо захищений від зовнішнього впливу та кібератак. Кібератаки становлять серйозний виклик комунікації, адже проникнення у систему, що зберігає ключі шифрування до багатьох пристроїв, автоматично означає ризик дешифрування цих даних і витоку секретної інформації. Також, у випадку роботи груп із захисту прав людей, деякі держави зацікавлені у небезпеці та фізичному знищенні цих груп. Наскрізне шифрування дає можливість створити безпечний канал комунікації у тому випадку, коли серверу неможливо повністю довіряти. Попри вже

широкий досвід використання наскрізного шифрування у системах для комунікації, певні ділянки залишаються недостатньо дослідженими. До них можна віднести стратегічні комунікації на кшталт симуляції військових дій із двома або більше учасниками, класичні ігри та інші. У цій статті розглядається можливість розширення застосування протоколів наскрізного шифрування у середовищах з відсутнім безпечним середовищем, створення application-level протоколу шифрованої комунікації на базі алгоритму Double Ratchet [8] та приклад його використання у класичній покроковій грі.

1. Протокол шифрування

Пропонований протокол шифрування є імплементацією протоколу Double Ratchet без використання шифрування хедерів. Для імплементації були вибрані такі криптографічні алгоритми:

1. Для реалізації функцій ***ENCRYPT*** і ***DECRYPT*** — алгоритм шифрування AES-256-GCM-SIV [5]. Вибір цього алгоритму обґрунтовується додатковим захистом системи від зламів у випадку перевикористання послідовності байтів nonce.
2. Для реалізації функцій ***GENERATE_DH*** і ***DH*** — алгоритм Diffie-Hellman, що використовує криву Curve25519 [2], або ж скорочено X25519. Це відповідає рекомендаціям авторів специфікації протоколу Double Ratchet.
3. Для реалізації ***KDF_RK*** — алгоритм HKDF [7] з хеш-функцією SHA-256. У рекомендації авторів вказано, що автор імплементації має самостійно вибрати послідовність байтів для інформаційного параметру. Ми вибрали набір байтів, який кодує слово “Checkers”.
4. Для реалізації ***KDF_CK*** — алгоритм HMAC [6] з хеш-функцією SHA-256. Це відповідає рекомендаціям авторів специфікації протоколу.

Імплементація протоколу виконана мовою Rust [13]. Мова Rust вибрана для імплементації тому, що вона забезпечує: безпеку пам’яті мови, широкий вибір криптографічних бібліотек, можливість компіляції у бінарний формат WebAssembly [14].

Протокол шифрування, окрім реалізації функцій, визначених у Double Ratchet, також реалізує API:

1. Функція ***w_init_ratchet_sender*** дозволяє генерувати інтерфейс мовою Javascript та ініціалізує криптографічні параметри відправника першого повідомлення у протоколі.
2. Функція ***w_init_ratchet_receiver*** дозволяє генерувати інтерфейс мовою Javascript та ініціалізує криптографічні параметри отримувача першого повідомлення у протоколі.
3. Структура ***Header*** відповідає структурі ***Header*** у Double Ratchet та містить публічний ключ відправника, створеного алгоритмом X25519; кількість надісланих повідомлень у минулому ланцюгу надсилання, та кількість надісланих повідомлень у поточному ланцюгу надсилання.
4. Структура ***UserClientData*** зберігає у собі стан усіх криптографічних параметрів, що потрібні для шифрування та дешифрування повідомлень, а також ініціалізації сторін. Протокол Double Ratchet визначає дві операції зміни ключів: symmetric key ratchet та asymmetric key ratchet. Symmetric key ratchet відбувається при шифруванні та дешифруванні кожного нового повідомлення в одному ланцюгу надсилання / отримання. Таким ланцюгом називають упорядковану послідовність повідомлень, які використовували стале значення результату алгоритму Diffie-Hellman операції двох користувачів. Для шифрування або дешифрування кожного повідомлення у ланцюгу надсилання або ланцюгу отримання відповідно протокол мусить добути новий ключ із батьківського ключа ланцюга. Батьківський ключ ланцюга оновлюється з кожним новим шифрованим або дешифрованим повідомленням. Щоби забезпечити функціональність, структура зберігає початковий батьківський ключ для видобування першого ключа ланцюга надсилання / отримання, поточні ключі ланцюгів надсилання / отримання, кількість надісланих повідомлень, кількість отриманих повідомлень, довжину попереднього ланцюга надсилання, а також пропущені повідомлення. Особливістю імплементації протоколу є використання X25519 публічного ключа отримувача у ролі ***associated data*** [3]. Оскільки публічний ключ отримувача може змінюватися при операції asymmetric key ratchet, протокол зберігає дані про пропущені повідомлення у хеш-таблиці та в такому форматі: [*публічний_ключ_відправника, номер_повідомлення*] -> [*ключ_дешифрування, публічний_ключ_отримувача*]. Ця сукупність даних дозволяє проводити операцію symmetric key ratchet. Втім, протокол Double Ratchet також визначає операцію asymmetric key ratchet. Для забезпечення цієї операції, у структурі ***UserClientData*** зберігаються параметри ***dh_s*** (приватний X25519 ключ відправника) і ***dh_r*** (публічний X25519 ключ отримувача).

5. Структура ***HeaderCiphertext*** представляє трійку значень Header, байтове представлення зашифрованого тексту і новий стан ***UserClientData*** після шифрування. Повертається методом ***w_ratchet_encrypt***.
6. Структура ***Plaintext*** представляє пару значень, що містить байтове представлення незашифрованого тексту і новий стан ***UserClientData*** після дешифрування. Повертається методом ***w_ratchet_decrypt***.
7. Функції ***w_ratchet_encrypt*** та ***w_ratchet_decrypt***. Ці функції є головними при шифруванні та дешифруванні повідомлень. Утім, окрім отримання звичних вхідних параметрів (незашифрований текст для ***encrypt***, хедер та зашифрований текст для ***decrypt***), функції також отримують копію структури ***UserClientData***, закодованої як значення з Javascript. Визначивши API, що доступне Javascript, потрібно зробити крок від програми мовою Rust до модуля Javascript.

WebAssembly нативно підтримується у більшості великих браузерів. Це означає, що браузери можуть слугувати хостами для контейнеризованого виконання бінарного коду WebAssembly, виділяти пам'ять WebAssembly модулям і викликати функції цих модулів. Оскільки WebAssembly модуль у Javascript є обгорткою над виконуваним файлом, з точки зору користувача бібліотеки, потрібно знати вказівники до функцій, типи аргументів цих функцій тощо. Щоби полегшити практичне використання бібліотеки, частою практикою є створення зовнішніх Javascript функцій та класів, що звертаються до відомих на момент компіляції секцій у бінарному файлі. Для мови програмування Rust зазначену інфраструктуру надають такі компоненти:

1) нативна компіляція коду у формат *wasm32;*
2) бібліотека ***wasm-bindgen*** [10], що надає інформацію про те, для яких компонентів генерувати Javascript відповідники;
3) застосунок командного рядка ***wasm-pack*** [11], котрий генерує Javascript модуль для WebAssembly коду.

Завдяки цим компонентам можливо виконати перехід від наявності Rust бібліотеки, яку можуть використовувати програми, написані мовою Rust, C, або C++, до WebAssembly модуля, який може виконуватися в браузерному середовищі. Альтернативними підходами, які використала *libsignal*, є створення клієнтських бібліотек мовами програмування нативних платформ — Swift для iOS, Java для Android.

У нашому випадку, ***wasm-pack*** виконує критичну роль, генеруючи Javascript модуль, що містить чотири файли: ***double_ratchet.d.ts***, ***double_ratchet.js***, ***double_ratchet_bg.wasm.d.ts***, ***double_ratchet_bg.wasm***. Погляньмо на них детальніше. ***double_ratchet.d.ts*** містить типізовані визначення функцій як розробленої бібліотеки, так і внутрішніх функцій ***wasm-bindgen***.

```
1 /* tslint:disable */
2 /* eslint-disable */
3 export function w_new_heade:(sender_public_key: Uint8Array, previous_sending_chain_length: bigint, num_sent: bigint): any;
4 export function w_init_ratchet_sender(dh_r: Uint8Array, shared_secret: Uint8Array): any;
5 export function w_init_ratchet_receiver(dh_s: Uint8Array, shared_secret: Uint8Array): any;
6 export function w_ratchet_encrypt(data: any, plaintext: Uint8Array): HeaderCiphertext;
7 export function w_ratchet_decrypt(data: any, header: Header, ciphertext: Uint8Array): Plaintext;
```

Рис. 1. Сигнатури функцій, що генеруються для мови Typescript

Файл ***double_ratchet.js*** містить імплементацію наведених у рис. 1 функцій, а також внутрішніх функцій ***wasm-bindgen***.

Файл ***double_ratchet_bg.wasm.d.ts*** містить додаткові генеровані Typescript сигнатури для коректної роботи ***wasm-pack***.

Файл ***double_ratchet_bg.wasm*** є виконуваним файлом, що генерується після компіляції Rust коду у цільову архітектурну трійку wasm32-unknown-unknown.

Протокол шифрування в загальному випадку складається з трьох кроків: встановлення значення спільного секрету, ініціалізації сторін, шифрування та дешифрування повідомлень.

Однією з небагатьох «логічних» вразливостей цього алгоритму є саме перший крок — створення спільного секрету. Автори Double Ratchet пропонують [8] встановлювати спільний секрет за допомогою протоколів узгодження ключів. Обравши алгоритм X25519, клієнти створюють пари ключів, де перший клієнт надсилає свій публічний ключ другому клієнту, після чого другий клієнт виконує операцію скалярного множення публічного ключа (точка на еліптичній кривій Curve25519) на секретний ключ (скаляр із поля, визначеного Curve25519) та надсилає свій публічний ключ першому

клієнту для виконання такої самої операції множення. Таким чином, користувачі прийдуть до одних і тих самих значень спільного секрету. Важливою проблемою такого підходу є атака Man-In-The-Middle з боку активного нападника, який може надати клієнтам власні значення публічного ключа і створити шифрований канал комунікації з будь-яким із клієнтів. Традиційно, для подолання таких проблем використовується протокол TLS, втім це все одно вимагатиме довіри до сервера, що є посередником у цій комунікації. Для розв'язання цієї проблеми ведуться дослідження в напрямі застосування zero-knowledge proofs у протоколі TLS [4]. У наступному розділі ми наведемо приклад генерації спільного секрету.

Другий крок — ініціалізація сторін — відбувається на боці відправника першого повідомлення та отримувача першого повідомлення незалежно.

Третій крок — шифрування та дешифрування повідомлень. Отримавши доступ до структури ***UserClientData***, можливо шифрувати будь-який plaintext для отримувача.

Практична значущість розробленого протоколу особливо актуальна в контексті сучасних безпекових викликів, зокрема в ситуаціях, де не можна довіряти мережевій інфраструктурі або існує ризик кібератак, що компрометують важливі операції. Створення альтернативи ***libsignal*** із можливістю інтеграції у середовища WebAssembly значно розширює сферу можливих застосувань наскрізного шифрування, зокрема у веборієнтованих застосунках, ігрових платформах та системах командування і контролю військами, що виконуються у браузері або інших середовищах [15].

2. Сервер комунікації

У цьому розділі розглядається створення сервера для комунікації учасників певного стратегічного завдання. Створений протокол може використовуватися у задачах, що абстрагують текстову комунікацію від клієнта і представляють певні дані у відомому і реплікованому форматі. Прикладом є симуляції поля бою або покрокові ігри з відомими публічними параметрами. До останніх можна віднести шахи і шашки: учасники починають комунікацію з наперед відомого та відкритого стану, тобто шахової дошки з розставленими на ній фігурами. Специфічний вибір таких задач дає змогу вилучити сервер із ланцюга прийняття ігрових рішень, адже немає недетермінованих процесів, що відповідають за зміну стану гри. У нашому випадку функції сервера, зокрема створення каналу комунікації між двома наперед відомими користувачами, були розширені можливістю створювати відкриті сесії, до яких можуть доєднатися будь-які користувачі.

Щоб підтримувати безперервну комунікацію між користувачами, потрібно обрати протокол, що дозволяє як отримувати повідомлення користувачів, так і передавати повідомлення користувачам на сторону клієнта. Для такої задачі був обраний протокол Websocket. Наступним поняттям сервера є сесія, яка складається з двох гравців, кожен з яких підключений до сервера окремим Websocket підключенням і має внутрішній ідентифікатор підключення, що зберігається у пам'яті сервера. Для кожного користувача виділяється канал необмеженого розміру, що дозволяє надсилати користувачу відповіді та отримувати від користувача запити. Важливо зазначити, що користувачів та сесії сервер розглядає як *ефемерні*. Ефемерність сесії означає, що той самий фізичний користувач може закрити підключення до сервера, у випадку чого сервер видалить цього користувача з хеш-таблиці користувачів і з сесії. За повторного підключення користувача до сервера користувач розглядатиметься як абсолютно новий.

Сервер імплементує певну логічну структуру повідомлень. Для коректної роботи, користувач має мати змогу створити відкриту сесію (анонсувати свою присутність іншим користувачам), отримавши ідентифікатор сесії; створити запит на долучення до наявної сесії; підтвердити запит на долучення до наявної сесії; надіслати повідомлення.

Загальною структурою повідомлення, що надсилають користувачі, є структура ***ClientMessage***, що містить поля ***sid***, ***cmd*** та ***sig***. Поле ***sid*** може опційно містити ідентифікатор сесії, представлений додатним двобайтним цілим числом. Поле ***cmd*** містить серіалізовану команду, визначену у переліку варіантів ***Cmd***. Поле ***sig*** обов'язково містить цифровий підпис певного набору байтів, що використовується для верифікації автентичності відправленого повідомлення. Верифікація цифрового підпису є головним механізмом автентифікації ефемерних користувачів у імплементації протоколу. Алгоритм EdDSA (Ed25519) був обраний для створення та верифікації цифрових підписів. Вибір алгоритму EdDSA обумовлений його високою продуктивністю, малим розміром ключів та підписів, що особливо важливо у вебсередовищах, де ресурси обмежені. До переваг EdDSA над альтернативними

алгоритмами (ECDSA на кривій secp256k1, Schnorr-Ristretto на кривій Ed25519) також можна додати широку доступність бібліотек для клієнтських середовищ.

Структура ***AnnounceArgs*** містить аргументи для створення сесії користувачем — ім'я цього користувача та його публічний ключ алгоритму EdDSA для верифікації цифрових підписів. Структура ***JoinSessionArgs*** містить параметри для створення запиту на доєднання користувача до вже наявної сесії — ім'я користувача, що доєднується, публічний ключ алгоритму X25519, та публічний ключ алгоритму EdDSA для верифікації цифрових підписів. Структура ***ConfirmJoinSessionArgs*** визначає параметри для прийняття вказаного у параметрі ***guest_username*** користувача та публічний ключ алгоритму X25519 користувача, що створив сесію. На цьому етапі користувачі мають можливість створити спільний секрет та ініціалізуватися як відправник і отримувач. Після доєднання другого користувача до сесії сервера відомі всі необхідні публічні ключі алгоритму EdDSA, проте у структурі ***SendSesssionMessage*** обов'язковою є наявність параметра ***public***, що містить публічний ключ EdDSA відправника повідомлення для визначення ідентичності надсилача. Крім цього, вона містить шифрований текст повідомлення у кодуванні Base64 і дані структури ***Header*** з протоколу шифрування. Для опрацювання усіх повідомлень використовується формат серіалізації JSON. Як бачимо з набору повідомлень, що може отримувати сервер, сервер не підтримує реєстрації користувачів зі збереженням їхніх даних. Перевагою цього підходу є архітектурна простота рішення, недоліком — можливість підробки повідомлень завдяки підслуховуванню повідомлень користувачів та перевикористання цифрових підписів. Ця проблема розв'язується дизайном сервера: сесії є ефемерними, тобто для кожної нової сесії користувача вимагається створення нової пари EdDSA ключів і використання підписів для автентифікації. На практиці це означає, що підписи однією парою ключів публічних даних відбуваються лише один раз, бо для однієї сесії виділяється можливість лише один раз підписати та надіслати повідомлення ***Announce***, ***JoinSession***, ***ConfirmJoinSession***. У випадку ***SendSessionMessage*** використання підпису статичних даних зробило б перевикористання підпису статичних даних вразливим, адже підпис не залежав би від даних, що надсилаються. Щоби цьому запобігти, для сесії користувача сервер зберігає число, що відоме клієнту та серверу одночасно, і це число додається до даних, на яких виконується цифровий підпис.

У аргумент ***session_id*** може передаватися будь-яке позитивне додатне число, яке може бути постійним, що менш безпечно, і змінним, що унеможливлює підроблення або перевикористання підпису за дотримання умови ефемерності.

Для внутрішнього управління комунікаційними сесіями сервер зберігає дві структури даних — хеш-таблицю з сесіями, де ключем є ідентифікатор сесії, значенням — стан сесії, і хеш-таблицю, що дозволяє визначити сесію за ім'ям користувача. Стан сесії — це набір даних, наданих обома членами цієї сесії. У цій імплементації стан сесії містить стан хоста (відправника першого повідомлення) і стан гостя (отримувача першого повідомлення).

Щоб користувачі могли не тільки відправляти повідомлення у сесію, а й отримувати «відповіді», тобто факт успішно опрацьованого повідомлення від учасника сесії, сервер має надсилати трансформовані відповіді одному з учасників. Загалом, відповідь має таку саму структуру, як і запит, окрім наявності публічного ключа EdDSA в полях структури. Залежно від отриманого повідомлення, сервер визначає отримувача відповіді. Наприклад, анонсувавши сесію, сервер не надішле відповідь іншим користувачам, окрім автора повідомлення, позаяк до сесії ніхто більше не належить. У випадку запиту ***GetSessions*** відповідь також надається автору повідомлення. Відповідь для ***JoinSession*** працює дещо інакше: автор повідомлення чекає на підтвердження від хоста, тому відповідь потрібно передати хосту. Схожим чином працює відповідь до ***ConfirmJoinSession***, де автор сповіщає гостя про підтвердження сесії. ***SendSessionMessage*** за своїм визначенням завжди надсилає повідомлення протилежній стороні.

При обробленні повідомлень можуть виникнути помилки. Наприклад, надсилати повідомлення у сесії, де немає другого користувача, є помилковою дією. Так само усі цифрові підписи мають успішно верифікуватися. Врахувавши різні випадки, де можуть виникнути помилки, можна використати конструкцію enum мови Rust та бібліотеку, що дозволяє перетворювати окремі помилки в описовий текст.

Таким чином, сервер комунікації містить перелік повідомлень, відповідей та помилок для безпечного використання у рамках шифрованої комунікації. Реалізований серверний компонент має важливу практичну значущість, зокрема у задачах, що потребують швидкої, безпечної та приватної кому-

нікації у вебдодатках, стратегічних симуляціях, покрокових іграх та інших сценаріях з високими вимогами до конфіденційності та низької затримки передавання інформації.

3. Використання методу у покроковій грі

Як було зазначено у вступі, наскрізне шифрування може використовуватися у різних сферах, де існує передавання інформації між двома сторонами. Однією з таких сфер є покрокові ігри для двох людей. Вибір цієї сфери обґрунтовується схожістю впровадження наскрізного шифрування у попередньо незашифровану комунікацію, адже стратегічні взаємодії між двома сторонами, такі як симуляції військових дій, є вичерпним набором станів та повідомлень. Для ілюстрації такого впровадження були обрані шашки. Шашки — це добре досліджена у сфері комп'ютерних наук класична гра, котра ідеально підпадає під використання шифрованої комунікації між гравцями.

Проблема імплементації полягає у тому, що дані про будь-яку мультиплеєрну гру зазвичай зберігаються на сервері, що не відповідає критеріям наскрізного шифрування. Щоби метод відтворення стану працював коректно, потрібно перенести задачу зберігання даних та оброблення ходів на сторону клієнта. Для подолання цієї проблеми використовується WebAssembly: уся гра відбувається на сторонах двох клієнтів, де кожен надсилає стан дошки після закінчення власного ходу. Клієнти можуть проводити валідацію зміни стану, тобто перевіряти на правильність послідовність ходів, зроблених у межах однієї зміни стану. Для простоти імплементації цей крок пропускається.

Як базова імплементація був обраний проєкт [9] авторства Toms Zvirbulis під назвою chemkers. Проєкт містить імплементацію шашок мовою Rust, що компілюється у WebAssembly механізмами, описаними у розділі 1. Також у проєкті наявна імплементація клієнтської частини застосунку — вебдодаток мовою Typescript з використанням бібліотеки ***preact***.

У базовій імплементації був механізм генерування ходів за допомогою підходів minimax та alpha-beta pruning, проте у цьому випадку такі підходи були зайвими. Замість цього гравці очікують закінчення ходу від противника і застосовують наданий противником стан до дошки на клієнті. Для підтримки такої роботи застосунку був доданий модуль ***network-context*** за патерном Redux. У модулі, відповідно до патерна, є actions, reducer, dispatch типи.

Підключення до сервера гри відбувається завдяки встановленню WebSocket з'єднання.

Шифрування та дешифрування повідомлень можливе завдяки використанню модуля Javascript, створеного в розділі 1.

Остання велика деталь, яка уможливлює зашифрований мультиплеєр для шашок, — сесії. Для того, щоби не дозволити початок гри до формування сесії, було розроблено окремий TSX компонент, котрий реалізує створення та підключення до сесій.

Висновки

У цій роботі було досліджено можливість розширення застосування протоколів наскрізного шифрування на основі алгоритму Double Ratchet у застосунках із низькою довірою до сервера, зокрема у покрокових іграх та стратегічних взаємодіях. Головним досягненням цієї роботи є власна імплементація протоколу Double Ratchet мовою Rust із можливістю компіляції у формат WebAssembly, реалізований сервер на базі WebSocket, а також вебклієнт гри у шашки для демонстрації роботи розробленого протоколу. Область наскрізного шифрування потребує дослідження додаткових застосувань поза вже звичними, як-от шифрована комунікація у текстових месенджерах. Унікальність роботи полягає у першості імплементації Double Ratchet, що компілюється у WebAssembly. Розроблена імплементація протоколу може безпечно використовуватися у будь-яких застосунках, що мають на меті впровадження наскрізного шифрування та задовольняються критерієм ефемерності сесій (у тих випадках, де секрети зберігаються поза безпечним середовищем). Вибір Rust гарантував безпеку пам'яті та ефективність роботи криптографічних алгоритмів. WebAssembly дозволив швидко та безпечно інтегрувати ці алгоритми у вебклієнти. WebSocket забезпечив низьку затримку передавання даних між клієнтами, критично важливу для інтерактивних застосунків. Реалізований сервер підтримує ефемерні сесії, які гарантують мінімальні ризики компрометації інформації, і використовує цифрові підписи (EdDSA) для автентифікації користувачів. Логічна маршрутизація запитів забезпечує ефективне передавання повідомлень у захищених сценаріях. Вибір класичної гри у шашки як прикладу дозволив ефективно продемонструвати переваги наскрізного шифрування та

можливості реалізованого протоколу. Всі криптографічні операції, включно із генерацією ключів, шифруванням та дешифруванням повідомлень, успішно виконуються на клієнтських пристроях. Важливим є покращення механізмів обробки помилок та оптимізація роботи WebAssembly. Цікавою ділянкою подальших досліджень є створення механізмів zero-knowledge proofs для запобігання атакам типу Man-In-The-Middle під час створення спільного секрету, оптимізація інтеграції з криптографічними модулями апаратної безпеки (HSM) та дослідження можливостей масштабування рішення. Описане дослідження важливе в умовах частих кібератак на інфраструктуру різного рівня, коли вимоги до конфіденційності та безпеки інформації значно зросли. Запропонований підхід може використовуватися для розв'язання реальних задач захисту інформації, де критично важливою є довіра до каналу передавання даних. Таким чином, у процесі роботи створено комплексне рішення, що охоплює криптографічний протокол, серверну частину і вебклієнт, яке демонструє життєздатність застосування наскрізного шифрування у браузерних середовищах та мультиплеєрних іграх. Роботу може бути використано як основу для подальших досліджень та розробок у сфері безпеки комунікаційних систем та приватності у мультиплеєрних іграх.

Список літератури

1. BBC [Electronic resource]. — Signalgate. — 2025. — Mode of access: https://www.bbc.com/ukrainian/articles/c4g9n525lp7o (date of access: 26.03.2025).
2. Bernstein D. Curve25519: New Diffie-Hellman Speed Records / D. Bernstein // M. Yung, Y. Dodis, A. Kiayias, T. Malkin, eds. Public Key Cryptography - PKC 2006. PKC 2006. Lecture Notes in Computer Science. — Springer, Berlin, Heidelberg, 2026. — Vol. 958. — Pp. 207–228. — doi.org/10.1007/11745853_14.
3. Black J. Authenticated encryption / J. Black // H. C. A. van Tilborg, ed. Encyclopedia of Cryptography and Security. — Springer, Boston, MA, 2005. — https://doi.org/10.1007/0-387-23483-7_15.
4. Grubbs P. Zero-Knowledge Middleboxes [Electronic resource] / P. Grubbs, A. Arun, Y. Zhang, J. Bonneau, M. Walfish // 31st USENIX Security Symposium (USENIX Security 22). — 2022. — Mode of access: https://www.usenix.org/conference/usenixsecurity22/presentation/grubbs (date of access: 15.06.2025).
5. Gueron S. AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption [Electronic resource] / S. Gueron, A. Langley, Y. Lindell. — 2019. — Mode of access: https://www.rfc-editor.org/info/rfc8452.
6. Krawczyk H. HMAC: Keyed-Hashing for Message Authentication [Electronic resource] / H. Krawczyk, M. Bellare, R. Canetti. — 1997. — Mode of access: https://www.rfc-editor.org/info/rfc2104 (date of access: 15.06.2025).
7. Krawczyk H. HMAC-based Extract-and-Expand Key Derivation Function (HKDF) [Electronic resource] / H. Krawczyk, P. Eronen. — 2010. — Mode of access: https://www.rfc-editor.org/info/rfc5869 (date of access: 15.05.2025).
8. Perrin T. The Double Ratchet Algorithm [Electronic resource] / T. Perrin, M. Marlinspike. — 2016. — Mode of access: https://signal.org/docs/specifications/doubleratchet, https://doi.org/10.1007/978-3-031-33386-6_16 (date of access: 15.06.2025).
9. Redux. A JS library for predictable and maintainable global state management [Electronic resource] // Redux. — 2025. — Mode of access: https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers (date of access: 15.06.2025).
10. Rustwasm. wasm-bindgen [Electronic resource] // Github. — Mode of access: https://github.com/rustwasm/wasm-bindgen (date of access: 15.06.2025).
11. Rustwasm. wasm-pack [Electronic resource] // Github. — Mode of access: https://github.com/rustwasm/wasm-pack (date of access: 15.06.2025)
12. Sommerhalder M. Hardware Security Module [Electronic resource] / M. Sommerhalder // V. Mulder, A. Mermoud, V. Lenders, B. Tellenbach, eds. Trends in Data Protection and Encryption Technologies. Springer, Cham, 2023. — Mode of access: URL: https://www.rfc-editor.org/info/rfc8452 (date of access: 24.06.2025).
13. The Rust Programming Language [Electronic resource]. — Mode of access: https://www.rust-lang.org/ (date of access: 15.06.2025).
14. WebAssembly Community Group [Electronic resource] // WebAssembly Specification. — 2025. — Mode of access: https://webassembly.github.io/spec/core/ (date of access: 15.06.2025).
15. Zvirbulis T. Chemkers [Electronic resource] / T. Zvirbulis // Github. — 2022. — Mode of access: https://github.com/tomszir/chemkers (date of access: 15.06.2025).

References

BBC. (2025). Signalgate. https://www.bbc.com/ukrainian/articles/c4g9n525lp7o.

Bernstein, D. J. (2006). Curve25519: New Diffie-Hellman speed records. In M. Yung, Y. Dodis, A. Kiayias, T. Malkin (Eds.), *Public Key Cryptography - PKC 2006*. PKC 2006. Lecture Notes in Computer Science, Vol. 3958. Springer, Berlin, Heidelberg.

Black, J. (2005). Authenticated encryption. In H. C. A. van Tilborg (Ed.), *Encyclopedia of cryptography and Security*. Springer, Boston, MA. https://doi.org/10.1007/0-387-23483-7_15.

Grubbs, P., Arun, A., Zhang, Y., Bonneau, J., & Walfish, M. (2022). Zero-knowledge middleboxes. In *31st USENIX Security Symposium (USENIX Security 22)*. https://www.usenix.org/conference/usenixsecurity22/presentation/grubbs.

Gueron, S., Langley, A., & Lindell, Y. (2019). AES-GCM-SIV: Nonce misuse-resistant authenticated encryption.

Krawczyk, H., & Eronen, P. (2010). HMAC-based extract-and-expand key derivation function (HKDF). https://www.rfc-editor.org/info/rfc5869.

Krawczyk, H., Bellare, M., & Canetti, R. (1997). HMAC: Keyed-hashing for message authentication. https://www.rfc-editor.org/info/rfc2104.

Perrin, T., & Marlinspike, M. (2016). The Double Ratchet Algorithm. https://signal.org/docs/specifications/.

Redux. (2025). A JS library for predictable and maintainable global state management. https://redux.js.org.

Rust Project Developers. (n. d.). The Rust programming language. https://www.rust-lang.org/.
Rustwasm. (n. d.). wasm-bindgen. GitHub. https://github.com/rustwasm/wasm-bindgen.
Rustwasm. (n. d.). wasm-pack. GitHub. https://github.com/rustwasm/wasm-pack.
Sommerhalder, M. (2023). Hardware security module. In V. Mulder, A. Mermoud, V. Lenders, & B. Tellenbach (Eds.), *Trends in Data Protection and Encryption Technologies*. Springer, Cham. https://www.rfc-editor.org/info/rfc8452.
WebAssembly Community Group. (2025). WebAssembly specification. https://webassembly.github.io/spec/.
Zvirbulis, T. (2022). Chemkers. GitHub. https://github.com/tomszir/chemkers.

O. Mykhailenko, K. Gorokhovsky, S. Gorokhovsky

METHOD OF ENCRYPTED COMMUNICATION IN STRATEGIC INTERACTIONS

The paper explores the possibility of expanding the use of end-to-end encryption protocols based on the Double Ratchet algorithm in applications with low trust in the server, particularly in turn-based games and strategic interactions. The relevance of the research is due to the growing need for secure communication in cyberattacks, especially during military operations. The field of end-to-end encryption requires the study of additional applications beyond the usual ones, such as encrypted communication in text messengers. The developed implementation of the protocol can be safely used in any applications that aim to implement end-to-end encryption and satisfy the criterion of session ephemerality (in cases where secrets are stored outside a secure environment). The implemented server supports ephemeral sessions, which guarantee minimal risks of information compromise, and uses digital signatures (EdDSA) for user authentication. Logical routing of requests ensures efficient message transmission in secure scenarios. The choice of the classic game of checkers as an example allowed the authors to effectively demonstrate the advantages of end-to-end encryption and the capabilities of the implemented protocol. All cryptographic operations, including key generation, encryption and decryption of messages, are successfully performed on client devices. It is important to improve error handling mechanisms and optimize the operation of WebAssembly. An interesting area of further research is the creation of zero-knowledge proof mechanisms to prevent Man-In-The-Middle attacks during the creation of a shared secret, optimizing integration with cryptographic hardware security modules (HSM), and exploring the scalability of the solution. The proposed approach can be used to solve real-world information security problems where trust in the data transmission channel is critically important. Thus, the work has created a comprehensive solution that includes a cryptographic protocol, a backend, and a web client, which demonstrates the viability of end-to-end encryption in browser environments and multiplayer games. The work can be used as a basis for further research and development in the field of security of communication systems and privacy in multiplayer games.

Keywords: end-to-end encryption, Double Ratchet, WebAssembly, strategy games, WebSocket, Rust, secure communication.

Матеріал надійшов 25.06.2025

УДК 519.8
DOI: 10.18523/2617-3808.2025.8.126-131

Тригуб О. С., Олецький О. В., Франчук О. В.

ОЦІНЮВАННЯ УПРАВЛІНСЬКИХ РІШЕНЬ НА ОСНОВІ МЕТОДУ АНАЛІЗУ ІЄРАРХІЙ ТА МОДЕЛІ «СТАН — ІМОВІРНІСТЬ ДІЇ»

Пропонується підхід до оцінювання і моніторингу управлінських рішень на основі аналізу невідповідностей між рішеннями, які були фактично прийняті уповноваженими органами, і тими, які могли б бути прийняті при загальному голосуванні колективом агентів, якби таке голосування відбулося. В основі підходу лежить поєднання методу аналізу ієрархій і моделі «стан — імовірність дії» з урахуванням того, що на рішення впливає низка факторів із різним ступенем важливості, а також того, що при оцінці слід враховувати не тільки негайний виграш або програш, а й виграш або програш у перспективі. Наведено ілюстративний приклад.

Ключові слова: прийняття рішень, оцінювання рішень, метод аналізу ієрархій, модель «стан — імовірність дії».

1. Вступ

Типовий підхід до оцінювання управлінських рішень полягає в аналізі поточної ситуації та можливих змін ситуації унаслідок ухвалення та реалізації тих чи інших рішень. Математично задачу може бути сформульовано таким чином: приймати рішення так, щоб здійснювався перехід до ситуацій (станів) із кращими значеннями тих чи інших критеріїв, тобто від поточного стану $x^{(1)}$ зі значенням критеріїв ($f_1(x^{(1)})$, ..., $f_K(x^{(1)})$) перейти до стану $x^{(2)}$ зі значеннями критеріїв ($f_1(x^{(2)})$, ..., $f_K(x^{(2)})$) так, щоб $f_k\left(x^{(2)}\right) \geq f_k\left(x^{(1)}\right) \forall i = \overline{1,K}$ (тут K — кількість критеріїв), і хоча б для одного з критеріїв вказана нерівність була строгою. Після ухвалення та реалізації рішень здійснюється оцінка (моніторинг) нової ситуації. У випадку державного управління такий моніторинг може здійснюватися як силами самих управлінських органів, так і з боку суспільства.

Може бути поставлено задачу вибору найбільш ефективної управлінської дії. Для оцінки ефективності тих чи інших дій видається перспективним застосування методів навчання з підкріпленням (reinforcement learning) [14; 17] — від простих моделей типу багаторукого бандита до складного аналізу на основі марковських процесів прийняття рішень.

Але тут виникають певні проблеми, основними з яких є такі:

1. Часто не видається можливим покращити один критерій так, щоб не погіршилося значення якогось іншого критерію (таку ситуацію можна охарактеризувати як парето-оптимальну за критеріями). Типовим підходом до вирішення цієї проблеми є комбінування критеріїв на основі тих чи інших методик.
2. Часто буває так, що покращення відбувається після певної послідовності управлінських дій, а до того — тільки погіршення (так звана проблема локального оптимуму). Тому має бути забезпечено можливість того, щоб із певною ймовірністю приймалися рішення, гірші за поточне. Проблема локального оптимуму певною мірою вирішується в рамках марковських процесів прийняття рішень, оскільки наслідки прийнятого рішення оцінюються не тільки з погляду негайного виграшу, а й з точки зору того, наскільки близьким є новий стан до станів, які оцінюються як «хороші».
3. Наслідки прийняття рішень можуть бути недетермінованими: наприклад, у більшості випадків вони можуть бути хорошими, але з певною ймовірністю — поганими або навіть катастрофічними. Підходи до вирішення цієї проблеми розглядають на основі побудови і аналізу матриць ризиків, а також на основі марковських процесів прийняття рішень.
4. Часто буває так, що значення критерію не можна виміряти безпосередньо, а пов’язані з ним вимірювані індикатори не є достатньо надійними. У цьому випадку прийнято застосовувати експертні

оцінки; добре зарекомендували себе методики на основі попарних порівнянь та метод аналізу ієрархій (MAI) [5; 11; 15 та ін.].

Розглянемо приклад, який ілюструє підходи до вирішення деяких із перелічених проблем на основі методу аналізу ієрархій.

2. Ілюстративний приклад

Уряд деякої країни інвестує в будівництво заводу, який має почати працювати через рік. Унаслідок цього:

– помітно погіршується екологія;
– доходи населення та соціальні витрати спочатку зменшуються, але в перспективі очікується суттєвий економічний виграш, коли розпочнеться виробництво та продаж нової продукції (будемо вважати, що населення відчує цей ефект через два роки).

Сформуємо систему критеріїв. До цього підійдемо диференційовано з урахуванням дисконтування, тобто міркувань про те, що виграш або програш у близькій перспективі має більше значення, ніж у більш віддаленій, — подібно до того, як це прийнято при застосуванні методів навчання з підкріпленням. Тому розглянемо такі критерії:

– K1 — економічний виграш у цьому році;
– K2 — економічний виграш через рік;
– K3 — економічний виграш через два роки;
– K4 — економічний виграш через три роки;
– K5 — погіршення екології в цьому році;
– K6 — погіршення екології через рік;
– K7 — погіршення екології через два роки;
– K8 — погіршення екології через три роки.

Вважатимемо також, що рішення приймається, коли розрахована міра схвалення цього рішення перевищує деякий поріг B. Тут приймемо $B = 0.5$.

Застосуємо MAI. Побудуємо матриці попарних порівнянь за кожним із критеріїв (тут можна розглядати різні шкали попарних порівнянь [6], але для простоти візьмемо стандартну шкалу Сааті). Будемо порівнювати поточну ситуацію (перший рядок матриці попарних порівнянь) із тією, яка утвориться після прийняття відповідного рішення (другий рядок).

Для критеріїв *K1* та *K2* мають місце економічні втрати. Матриця попарних порівнянь має вигляд:

$$M^1 = M^2 = \begin{pmatrix} 1 & 2 \\ \frac{1}{2} & 1 \end{pmatrix}.$$

У наступні роки вже маємо економічний виграш, тому

$$M^3 = M^4 = \begin{pmatrix} 1 & \frac{1}{4} \\ 4 & 1 \end{pmatrix}.$$

Що стосується екологічних втрат, вони починаються з другого року, тому

$$M^5 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix},$$

$$M^6 = M^7 = M^8 = \begin{pmatrix} 1 & 3 \\ \frac{1}{3} & 1 \end{pmatrix}.$$

Вагові коефіцієнти критеріїв будемо отримувати в два етапи. Спочатку побудуємо матрицю попарних порівнянь між економічним і екологічним критеріями на першому році. Таким чином, перший рядок матриці буде відповідати критерію K1, другий — критерію K5. Вважаємо, що економічний критерій має певну перевагу, маємо наступну матрицю попарних порівнянь:

$$\begin{pmatrix} 1 & 4 \\ \frac{1}{4} & 1 \end{pmatrix}.$$

Її перонів вектор (нормалізований головний власний вектор) приблизно дорівнює

(0.8 0.2).

На основі цього вектора побудуємо розширений вектор важливостей усіх критеріїв з урахуванням коефіцієнта дисконтування за таким принципом: важливість критерію через k років дорівнює базовій важливості, помноженій на γ^k, де $\gamma(0 < \gamma < 1)$ — коефіцієнт дисконтування; після цього здійснюється нормалізація так, щоб сума елементів дорівнювала 1.

Візьмемо коефіцієнт дисконтування $\gamma = 0.95$ (змістовно це означає, що рішення значною мірою приймається з урахуванням майбутніх виграшів, тобто перспективи). Тоді отримуємо вектор:

$$w' = (0.8,\ 0.8 \cdot \gamma,\ 0.8 \cdot \gamma^2,\ 0.8 \cdot \gamma^0,\ 0.2,\ 0.2 \cdot \gamma,\ 0.2 \cdot \gamma^2,\ 0.2 \cdot \gamma^3)$$

і після нормалізації

$$w = (0.2156 \quad 0.2049 \quad 0.1946 \quad 0.1849 \quad 0.0539 \quad 0.0512 \quad 0.0487 \quad 0.0462).$$

Матриця, кожний i-й рядок якої дорівнює нормалізованому пероновому вектору матриці M^i (тобто змістовно — оцінкам альтернатив за i-м критерієм), набуває вигляду

$$Q = \begin{pmatrix} 0.6667 & 0.3333 \\ 0.6667 & 0.3333 \\ 0.1250 & 0.8750 \\ 0.1250 & 0.8750 \\ 0.5000 & 0.5000 \\ 0.7500 & 0.7500 \\ 0.7500 & 0.7500 \\ 0.7500 & 0.7500 \end{pmatrix}.$$

Комбінований вектор оцінки альтернатив набуває вигляду

$$p = wQ = (0.4643 \quad 0.5357).$$

Це означає, що приймають рішення будувати завод.

Очевидно, на рішення впливає низка факторів, зокрема:

– оцінки альтернатив за кожним критерієм;
– оцінки важливості самих критеріїв;
– поріг B;
– коефіцієнт дисконтування γ.

Для прикладу візьмемо коефіцієнт дисконтування $\gamma = 0.75$ (змістовно це означає, що вплив майбутніх виграшів зменшується порівняно з попереднім випадком). Тепер вектор оцінок альтернатив буде дорівнювати

$$(0.5090 \quad 0.4910)$$

і рішення про будівництво заводу має бути відхилено.

3. Оцінка ухваленого рішення

Однак важливе значення має не тільки саме рішення, а й оцінка цього рішення соціумом. В основу може бути покладений аналіз можливої невідповідності між тим, яке рішення приймається відповідним органом прийняття рішень, і тим, яке рішення було б прийнято на загальному референдумі.

Для оцінки того, яким могло б бути колективне рішення, застосуємо однорівневу модель «стан — імовірність дії», запропоновану в [13]. Нехай стани моделі відповідають можливим рівням коефіцієнта дисконтування; для визначеності візьмемо множину можливих рівнів

$$\{1,\ 0.95,\ 0.9,\ 0.8,\ 0.75,\ 0.6,\ 0.5,\ 0.4,\ 0.3,\ 0.2,\ 0.1,\ 0.05,\ 0\}.$$

Побудуємо матрицю «стан — імовірність дії» на основі методики, описаної в розділі 2 (рядки цієї матриці відповідають імовірностям прийняття рішень за умови перебування у відповідному стані). Для вказаної системи станів ця матриця набуває вигляду

$$H = \begin{pmatrix} 0.4542 & 0.5458 \\ 0.4643 & 0.5357 \\ 0.4749 & 0.5251 \\ 0.4973 & 0.5027 \\ 0.5090 & 0.4910 \\ 0.5456 & 0.4544 \\ 0.5700 & 0.4300 \\ 0.5928 & 0.4072 \\ 0.6123 & 0.3877 \\ 0.6266 & 0.3734 \\ 0.6340 & 0.3660 \\ 0.6348 & 0.3652 \\ 0.6333 & 0.3667 \end{pmatrix}.$$

Тепер вирішального значення набуває вектор ймовірностей $\overline{p}$ того, що агент перебуває в тому чи іншому стані. Якщо $\overline{p}$ та H відомі, вектор імовірностей прийняття того чи іншого рішення в рамках моделі «стан — імовірність дії» розраховують за формулою

$$p = \overline{p} \cdot H.$$

Нехай коефіцієнт дисконтування $\gamma = 0.75$. Як ми бачили раніше, в рамках моделі за таких умов рішення про будівництво має бути відхилене.

Але суспільне схвалення чи несхвалення цього рішення залежить від вектора $\overline{p}$. Нехай він дорівнює

$$\overline{p} = (0.0, 0.0, 0.0, 0.04, 0.06, 0.1, 0.1, 0.2, 0.2, 0.2, 0.06, 0.03, 0.01)$$

(змістовно такий вектор означає, що більшість виборців не є достатньо далекоглядними та живуть сьогоднішнім днем).

Тоді

$$p = \overline{p} \cdot H = (0.5917 \quad 0.4083).$$

Змістовно це означає, що на референдумі, якби він відбувся, більшість громадян підтримала б рішення, прийняте органом управління. Суспільство скоріш задоволене, ніж незадоволене.

А якщо

$$\overline{p} = (0.2, 0.3, 0.25, 0.15, 0.05, 0.03, 0.02, 0, 0, 0, 0, 0, 0)$$

(змістовно це означає, що громадяни є більш далекоглядними і думають про майбутнє), то

$$p = \overline{p} \cdot H = (0.4767 \quad 0.5233).$$

Тепер суспільство скоріше не схвалює прийняте рішення (референдум, якби він відбувся, ухвалив би рішення про будівництво заводу), і можна казати про очікування деякого зростання суспільного невдоволення щодо того, що рішення про будівництво не було ухвалене.

4. Висновки та обговорення

У статті запропоновано підходи до оцінювання управлінських рішень на основі поєднання методу аналізу ієрархій і моделі «стан — імовірність дії». При цьому враховується те, що оцінювання має здійснюватися з огляду на різні критерії (можливо, суперечливі), і що слід враховувати не тільки негайні виграші, а й перспективу.

Наведено приклад, в якому ключовим параметром є коефіцієнт дисконтування майбутніх виграшів (чим ближче коефіцієнт дисконтування до 1, тим більш далекоглядними є особи, які приймають рішення). Показано на моделі, яким чином різне ставлення до майбутніх виграшів (далекоглядне чи недалекоглядне) спричиняє невідповідності між тими рішеннями, які ухвалив відповідний орган і рішеннями, які могли б бути прийняті на референдумі, якби він відбувся, і як наслідок — позитивну або негативну оцінку ухвалених рішень з боку суспільства.

У цьому контексті видається доцільним застосування багаторівневих схем моделі «стан — імовірність дії» у поєднанні з методом аналізу ієрархій, які в загальних рисах описані в [3; 8; 10], що

дозволить більш повноцінно враховувати вплив тих чи інших факторів на рішення, що ухвалюються. При побудові матриць попарних порівнянь видається перспективним також застосування транзитивних шкал [2; 6], оскільки стандартна шкала Сааті не завжди дає достатньо гарні результати.

Варто звернути увагу на те, що оцінка суспільного задоволення або незадоволення суттєво залежить від міри невідповідності між рішеннями, які фактично приймаються управлінськими органами й тими, які могли б бути прийняті на референдумі, якби він відбувся. При моделюванні важливе значення має також рівень інформованості суспільства про фактори, які мають вплив на прийняття рішення, а також на моделі поширення інформації та її вплив до ставлення до тих чи інших рішень і до факторів, які зумовлюють ухвалення цих рішень [4; 7; 9; 12; 16; 18–20 та ін.].

Список літератури

1. Глибовець М. М. Штучний інтелект : підручник / М. М. Глибовець, О. В. Олецький. — Київ : Вид. дім «Києво-Могилянська академія», 2002.
2. Олецький О. В. Про застосування методу аналізу ієрархій для автоматизованого оцінювання студентських робіт / О. В. Олецький , О. С. Тригуб // Наукові записки НаУКМА. Комп'ютерні науки. — 2020. — Т. 3. — С. 127–131. — https://doi.org/10.18523/2617-3808.2020.3.127-131.
3. Олецький О. В. Про підхід до формування дворівневої моделі «стан — ймовірність дії» на основі попарних порівнянь та методу аналізу ієрархій / О. В. Олецький, І. О. Франчук, В. В. Гуминський // Наукові записки НаУКМА. Комп'ютерні науки. — 2023. — Т. 6. — С. 4–10. — https://doi.org/10.18523/2617-3808.2023.6.4-10.
4. Aref A. An integrated trust establishment model for the internet of agents / A. Aref, T. Tran // Knowledge and Information Systems. — 2020. — Vol. 62. — Pp. 79–105.
5. Brunelli M. Introduction to the analytic hierarchy process [Electronic resource] / M. Brunelli. — Springer, Cham, 2015. — Mode of access: https://doi.org/10.1007/978-3-319-12502-2.
6. Choo E. A common framework for deriving preference values from pairwise comparison matrices / E. Choo, W. Wedley // Computers and Operations Research. —2004. — Vol. 31, no. 6. — Pp. 893–908.
7. Dang L. Simulating the spatial diffusion of memes on social media networks / L. Dang, Z. Chen, J. Lee, M.-H. Tsou, X. Ye // International Journal of Geographical Information Science. — 2016. — Vol. 33. — Pp. 1545–1568.
8. Dosyn D. An approach to modeling elections in bipartisan democracies on the base of the "state-probability of action" model / D. Dosyn , O. Oletsky // CEUR Workshop Proceedings. — 2024. — Pp. 74–85.
9. Fullam K. K. A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies / K. K. Fullam, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, M. Voss // Proc. 4th Int. Joint Conf. Auto. Agents Multiagent Syst. — 2005. — Pp. 512–518.
10. Ivokhin E. Restructuring of the model "state–probability of choice" based on products of stochastic rectangular matrices [Electronic resource] / E. Ivokhin, O. Oletsky // Cybern. Syst. Anal. — 2022. — Vol. 58-2. — Pp. 242–250. — Mode of access: https://doi.org/10.1007/s10559-022-00456-z.
11. Koczkodaj W. Important facts and observations about pairwise comparisons [Electronic resource] / W. Koczkodaj, L. Mikhailov, G. Redlarski, M. Soltys, J. Szybowski, G. Tamazian, E. Wajch, Kevin Kam Fung Yuen // Fundamenta Informaticae. — 2016. — Vol. 144. — Pp. 1–17. — Mode of access: https://doi.org/10.3233/fi-2016-1336.
12. Mallipeddi R. A framework for analyzing influencer marketing in social networks: selection and scheduling of influencers [Electronic resource] / R. Mallipeddi, S. Kumar, C. Sriskandarajah, Y. Zhu // SSRN Electronic Journal. — 2018. — Mode of access: https://doi.org/10.2139/ssrn.3255198.
13. Oletsky O. Formalizing the procedure for the formation of a dynamic equilibrium of alternatives in a multi-agent environment in decision-making by majority of votes [Electronic resource] / O. Oletsky, E. Ivohin // Cybern Syst Anal. — 2021. — Vol. 57-1. — Pp. 47–56. — Mode of access: https://doi.org/10.1007/s10559-021-00328-y.
14. Russell S. Artificial intelligence: A modern approach / S. Russell, P. Norwig. — 4th Edition. — Pearson Education, Inc., 2021.
15. Saaty T. L. The analytic hierarchy process / T. L. Saaty. — McGraw-Hill, New York, 1980.
16. Sobkowicz P. Opinion mining in social media: modeling, simulating, and forecasting political opinions in the web / P. Sobkowicz, M. Kaschesky, G. Bouchard // Government Information Quarterly. — 2012. — Vol. 29. — Pp. 470–479.
17. Sutton R. S. Reinforcement learning: an introduction / R. S. Sutton, A. G. Barto. —Second Edition. — MIT Press, London, 2018.
18. Vosoughi S. The spread of true and false news online / S. Vosoughi, D. Roy, S. Aral // Science. —2018. — Vol. 359. — Pp. 1146–1151.
19. Wasserman S. Social network analysis: methods and applications / S. Wasserman, K. Faust. — Cambridge University Press, 1994.
20. Yu H. A survey of multi-agent trust management systems / H. Yu, Z. Shen, C. Leung, C. Miao, V. Lesser // IEEE Access. — 2013. — Vol. 1. — Pp. 35–50.

References

Aref, A., & Tran, T. (2020). An integrated trust establishment model for the internet of agents. *Knowledge and Information Systems*, *62*, 79–105.

Brunelli, M. (2015). *Introduction to the analytic hierarchy process*. Springer International Publishing. https://doi.org/10.1007/978-3-319-12502-2.

Choo, E., & Wedley, W. (2004). A common framework for deriving preference values from pairwise comparison matrices. *Computers and Operations Research*, *31* (6), 893–908.

Dang, L., Chen, Z., Lee, J., Tsou, M.-H., & Ye, X. (2019). Simulating the spatial diffusion of memes on social media networks. *International Journal of Geographical Information Science*, *33* (8), 1545–1568. https://doi.org/10.1080/13658816.2019.1591414.

Dosyn, D., & Oletsky, O. (2024). An approach to modeling elections in bipartisan democracies on the base of the "state-probability of action" model. In *CEUR workshop proceedings* (pp. 74–85).

Fullam, K. K., Voss, M., Klos, T. B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K. S., Rosenschein, J. S., & Vercouter, L. (2005). A specification of the agent reputation and trust (ART) testbed. In *The fourth international joint conference*. ACM Press. https://doi.org/10.1145/1082473.1082551.

Han Yu, Zhiqi Shen, Leung, C., Chunyan Miao & Lesser, V. R. (2013). A survey of multi-agent trust management systems. *IEEE Access*, *1*, 35–50. https://doi.org/10.1109/access.2013.2259892.

Hlybovets, M. M., & Oletskyi, O. V. (2002). *Shtuchnyi intelekt*. Vydavnychyi dim "KM Akademiia" [in Ukrainian].

Ivokhin, E. V., & Oletsky, O. V. (2022). Restructuring of the model "state–probability of choice" based on products of stochastic rectangular matrices. *Cybernetics and Systems Analysis*. https://doi.org/10.1007/s10559-022-00456-z.

Katherine, F. (Ed.). (1994). *Social network analysis: Methods and applications*. Cambridge University Press.

Koczkodaj, W. W., Mikhailov, L., Redlarski, G., Soltys, M., Szybowski, J., Tamazian, G., Wajch, E., & Yuen, K. K. F. (2016). Important Facts and Observations about Pairwise Comparisons (the special issue edition). *Fundamenta Informaticae*, *144* (3–4), 291–307. https://doi.org/10.3233/fi-2016-1336.

Mallipeddi, R., Kumar, S., Sriskandarajah, C., & Zhu, Y. (2018). A framework for analyzing influencer marketing in social networks: *Selection and scheduling of influencers. SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3255198.

Oletsky, O. V., & Tryhub, O. S. (2020). Pro zastosuvannia metodu analizu iierarkhii dlia avtomatyzovanoho otsiniuvannia studentskykh robit. *NaUKMA Research Papers. Computer Science*, *3*, 127–131. https://doi.org/10.18523/2617-3808.2020.3.127-131 [in Ukrainian].

Oletsky, O. V., Franchuk, I. O., & Humynskyi, V. V. (2023). Pro pidkhid do formuvannia dvorivnevoi modeli "stan – ymovirnist dii" na osnovi poparnykh porivnian ta metodu analizu iierarkhii. *NaUKMA Research Papers. Computer Science*, *6*, 4–10. https://doi.org/10.18523/2617-3808.2023.6.4-10 [in Ukrainian].

Oletsky, O. V., & Ivohin, E. V. (2021). Formalizing the procedure for the formation of a dynamic equilibrium of alternatives in a multi-agent environment in decision-making by majority of votes. *Cybernetics and Systems Analysis*, *57* (1), 47–56. https://doi.org/10.1007/s10559-021-00328-y.

Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach*. Pearson Education, Limited.

Sobkowicz, P., Kaschesky, M., & Bouchard, G. (2012). Opinion mining in social media: Modeling, simulating, and forecasting political opinions in the web. *Government Information Quarterly*, *29* (4), 470–479. https://doi.org/10.1016/j.giq.2012.06.005.

Sutton, R. S., Barto, A. G., & Bach, F. (2018). *Reinforcement Learning: An Introduction*. MIT Press.

The analytic hierarchy process: Planning, priority setting, resource allocation. (1980). McGraw-Hill International Book Co.

Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science*, *359* (6380), 1146–1151. https://doi.org/10.1126/science.aap9559.

O. Tryhub, O. Oletsky, O. Franchuk

ESTIMATING DECISIONS BASED ON THE ANALYTIC HIERARCHY PROCESS AND MODELING THE STATE-PROBABILITY OF ACTION

An approach to estimating and monitoring decisions in situations where there are many criteria influencing possible results, and these criteria typically contradict each other, is suggested. Typically, a Pareto-optimal situation, when it is impossible to improve any criterion without worsening another criteria, takes place. Another problem regarded in the paper is the problem of a local optimum, when it is impossible to improve an objective function without temporarily worsening it, that is, improving may be expected after several steps.

The approach is based on combining the Analytic Hierarchy Process and pairwise comparisons, from the one side, and modeling the "state-probability of action", from the other side. Such a combining leads to the construction of a two-level model "state-probability of choice", in which the required matrices are obtained by using pairwise comparisons and the Analytic Hierarchy Process. The main idea is to consider discrepancies between decisions actually made by authorities and those which could be made by the whole bulk of agents by a majority of votes, if such a voting took place. The suggested approach takes into account two considerations: firstly, decisions are influenced by several factors having different measures of importance, and secondly, not only instant wins and losses but those in prospect matter for evaluating these decisions.

The suggested approach is illustrated by the example provided in the paper. Within this example, the question is whether a new factory should be built or not. Two contradictory factors influencing decisions have been considered: economic wins and environmental losses expected if the factory is actually built. The main idea of the example is to consider different values of time discount, which meaningfully stand for how important instant wins and losses are in comparison with those in the future. It was showcased in the paper that discrepancies in estimating discount values can lead to large discrepancies in agents' attitudes toward estimates of decisions.

The standard scale of preference suggested earlier was used in this example, though other types of scales, especially transitive ones, definitely could be considered.

Keywords: decision making, assessing decisions, Analytic Hierarchy Process, modeling the "state-probability of action".

Матеріал надійшов 25.06.2025

УДК 004.4’2,004.75
DOI: 10.18523/2617-3808.2025.8.132-137

Моголівський В. О.

ОПТИМІЗАЦІЯ ВИБОРУ ТА РОЗМІЩЕННЯ ДАТЧИКІВ ЦИФРОВОГО ДВІЙНИКА ЛАБОРАТОРІЇ 3D-ДРУКУ НА ОСНОВІ ГЕНЕТИЧНИХ АЛГОРИТМІВ

Розглянуто підходи до створення цифрових двійників в університетському середовищі. Досліджено рекомендований набір датчиків, необхідний для створення цілісного цифрового двійника університетської лабораторії. Розглянуто підхід до вибору та розміщення датчиків для цифрового двійника лабораторії 3D-друку на основі генетичного алгоритму. Сформульовано математичну постановку задачі. Запропоновано формат хромосоми, зважену фітнес-функцію та представлення тривимірного простору лабораторії.

Ключові слова: цифровий двійник, генетичний алгоритм, лабораторія 3D-друку, оптимізація розміщення датчиків, оптимізаційна задача.

Вступ

Концепція Digital Twin набула широкого поширення в багатьох галузях: дослідження космосу, промислове виробництво, архітектурний дизайн, розумні міста, міське планування, моделювання клімату та енергоефективність [4]. Цифрові двійники дають змогу отримувати цінну інформацію про фізичні системи та сприяють впровадженню інновацій.

Цифровий двійник здатен допомогти університетам керувати складною інфраструктурою та динамічною освітньою діяльністю. Впровадження цифрових двійників надає уніфіковану структуру для розуміння й оптимізації взаємопов’язаних систем шляхом постійного збору даних, моніторингу в реальному часі та моделюванню з метою прогнозування. Цифровий двійник надає можливість передбачати проблеми та будувати стратегії для їх недопущення, замість пасивного реагування на несподівані проблеми.

Перспективним видається впровадження концепції цифрового двійника у середовищі університету [7]. На основі створення віртуальних копій приміщень навчальних корпусів та їхніх систем опалення, вентиляції, кондиціонування і критично важливого обладнання навчальні заклади можуть запроваджувати протоколи прогнозованого обслуговування, які мінімізують збої в роботі та оптимізують розподіл ресурсів. Зокрема така концепція може бути корисною для університетських лабораторій, таких як лабораторії електроніки чи 3D-друку [3; 6; 10].

У цьому дослідженні обговорюються деякі підходи до вирішення проблеми вибору та розміщення датчиків у лабораторії 3D-друку, необхідних для функціонування цифрового двійника, на основі генетичного алгоритму [1]. Сформульовані деякі рекомендації щодо побудови цифрового двійника для університетської лабораторії 3D-друку.

Огляд літератури

Концепцію цифрового двійника вперше запропонував Майкл Грівс у 2002 р. [9]. Спершу вона стосувалася тільки промислового виробництва, та згодом активно поширилася на різноманітні матеріальні та нематеріальні об’єкти, як-от промислові процеси або складні фізичні системи [11]. Цифрові двійники — це точні цифрові копії об’єктів реального світу з багатьма рівнями деталізації [8]. Їх активно впроваджують у різні галузі промисловості, наприклад у молочне виробництво [15]. Цифрові двійники також активно починають застосовувати для інших галузей, безпосередньо не пов’язаних із промисловим виробництвом, наприклад для моделювання бізнес-процесів в установах та організаціях [12]; перспективним видається їх застосування у більш загальному контексті концептуального моделювання.

Концепція цифрового двійника демонструє високу перспективність для університетського середовища [7]. Сото й колеги дослідили створення цифрового двійника офісного приміщення відкритого плану в університетському містечку [7]. Створений ними цифровий двійник збирав та аналізував такі показники, як температура, вологість і зайнятість приміщення [7]. Отже, цифровий двійник сприяє підтримці ефективного та сталого будівництва на території університету [7].

Характеристика лабораторних датчиків

Побудова цифрового двійника для університетських лабораторій є досить складним завданням з урахуванням того, що самі датчики можуть бути різних типів із широким спектром характеристик. У статті ми розглядаємо лабораторію 3D-друку. Проте описаний підхід може бути застосований і до інших типів університетських лабораторій, таких як лабораторії електроніки, робототехніки тощо.

Датчики, характерні для цифрового двійника університетської лабораторії, можна розділити на п’ять категорій: датчики середовища, стану обладнання, людської діяльності, безпеки та інвентаризації. Датчики навколишнього середовища вимірюють температуру, вологість, присутність частинок VOC і шумове забруднення. Всі перелічені фактори, очевидно, є важливими для комфортного робочого середовища [5]. Крім того, температура і вологість мають вирішальне значення не лише для лаборантів, а й для належного функціонування такого обладнання, як 3D-принтери чи паяльні станції [13]. Датчики стану обладнання відстежують рівень вібрації, споживання енергії та розподіл тепла. Вони гарантують, що обладнання перебуває в хорошому стані й використовується належним чином. Датчики активності людини відстежують зайнятість лабораторії: використання простору та обладнання. Вони дають змогу адміністрації визначати моделі використання лабораторії та планувати розширення або зміни робочого графіка. Крім того, вони допомагають оцінити фізичний вплив організму людини на температуру і вологість, які є ключовими для деяких процесів. Датчики безпеки відстежують небезпеку пожежі, як-от дим, надмірне тепло або полум’я. Датчики запасів — це датчики ваги та тиску для витратних матеріалів.

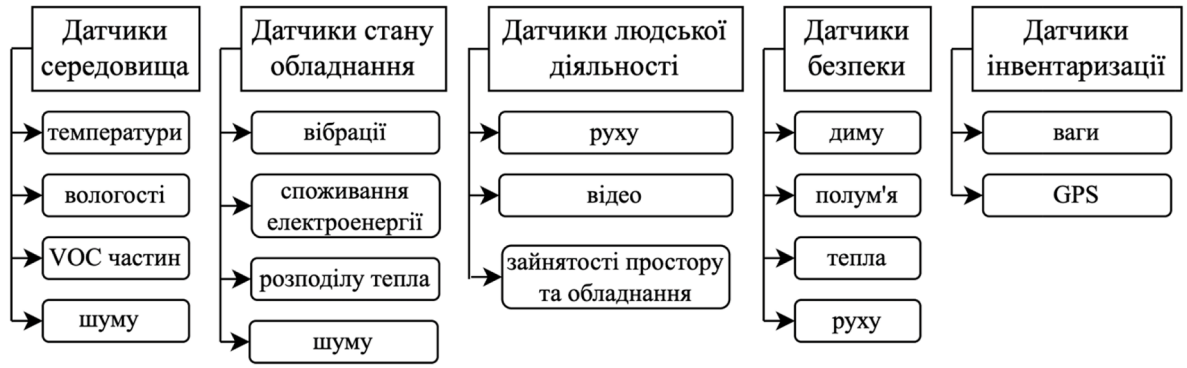


Рис. 1. Класифікація датчиків, необхідних для створення цифрового двійника університетської лабораторії

Формалізація задачі

Для оптимізації вибору та розміщення датчиків лабораторії 3D-друку пропонується використовувати генетичні алгоритми. Середовище лабораторії будемо моделювати як набір точок у тривимірному просторі. Простір, який займає певний об’єкт, задається двома точками:

- $P_{\min}$ — лівий дальній нижній край об’єкта;
- $P_{\max}$ — правий ближній верхній край об’єкта.

Простір лабораторії визначається шістьма наборами пар точок: S_{room}, $S_{obstacles}$, $S_{fdmPrinters}$, $S_{slaPrinters}$, $S_{washStations}$, $S_{cadWorkstations}$. Для експериментів було використане середовище, задане наборами точок (рис. 2).

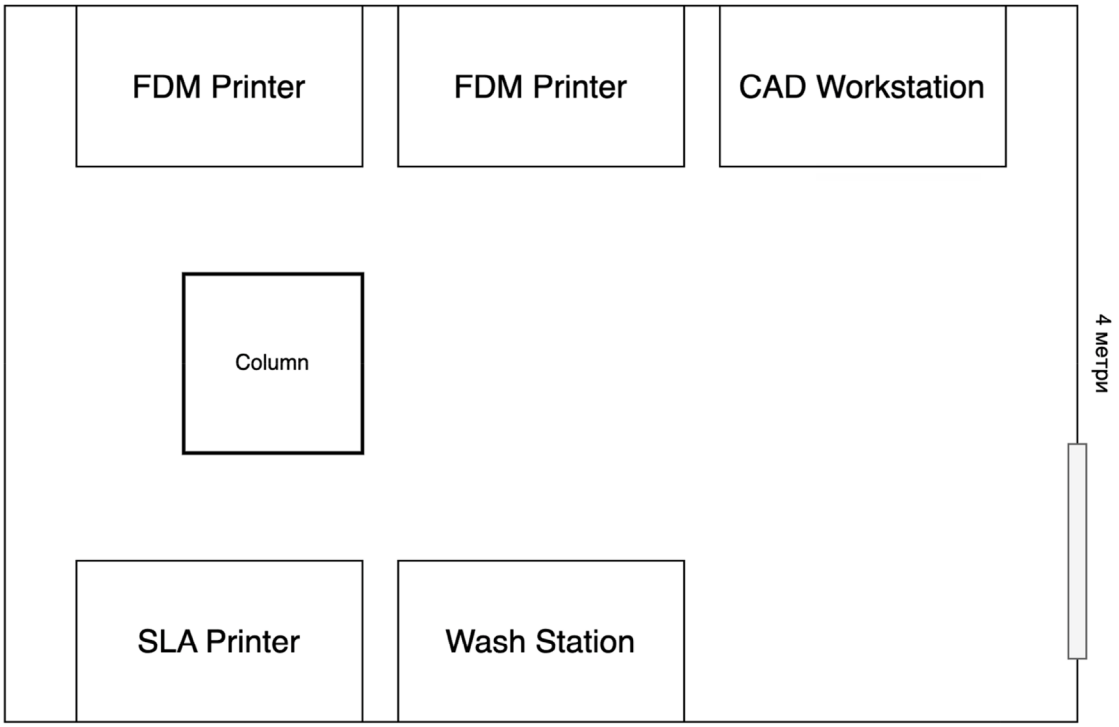


Рис. 2. Представлення лабораторії 3D-друку (вигляд згори)

Рішення задачі були представлені як хромосоми з таким виглядом:

$$(0001_{type_s}\ 010\ldots101_{pos_x}\ 010\ldots101_{pos_y}\ 010\ldots101_{pos_z})_{s_1}\ldots(\ldots)_{s_n}, \tag{1}$$

де 0001_{type_s} — це 4 біти, які ідентифікують певний вид датчика, наступні 48 бітів зберігають розміщення датчика у тривимірному просторі (по 16 бітів на кожну з осей x, y, z). Одержуємо фрагмент хромосоми s_1, що представляє датчик певного виду та його розміщення у просторі, таких фрагментів може бути n. Цей формат хромосоми може представляти до 15 датчиків, розміщених у тривимірному просторі розміром 65 535 одиниць по кожній з осей. За необхідності довжину окремих частин хромосоми можна збільшити, тоді її довжина буде розраховуватися за формулою:

$$L_c = (l_{type_s} + l_x + l_y + l_z) \cdot n, \tag{2}$$

де l_{type_s} — це кількість бітів для представлення типу датчика, а l_x, l_y, l_z — кількості бітів для представлення позиції датчика по осям x, y, z.

Початкова популяція з 1000 особин генерується випадковим чином.

Фітнес-функція може набувати значень від 0 до 1 та визначається як

$$F_k = w_1 f_1 + w_2 f_2 + w_3 f_3, \tag{3}$$

де f_1, f_2, f_3 — дочірні фітнес-функції, що оцінюють хромосому за різними параметрами, w_1, w_2, w_3 — ваги кожної з дочірніх фітнес-функцій у результуючій F_k, встановлені як 0.4, 0.3, 0.3 відповідно. Дочірні фітнес-функції можуть набувати значень від 0 до 1 та оцінюють такі параметри:

- f_1 — оцінює можливість монтажу датчика у приміщенні. Дозволяється розміщення на стінах і стелі. Не дозволяється поза приміщенням, на підлозі чи всередині перешкоди. Віддає перевагу розміщенням датчиків на стінах, також враховує висоту розміщення: що вище розміщений датчик, то вище значення f_1.
- f_2 — оцінює покриття датчиком необхідних об'єктів (3D принтерів тощо). Функція враховує мінімальну та максимальну дальність дії датчика, щоб визначити, чи покриває він об'єкт. Також враховує наявність перешкод (колон тощо) на шляху.
- f_3 — оцінює вартість обраних датчиків.

Таким чином виникає оптимізаційна задача, яка полягає у мінімізації вартості використаних датчиків:

$$\sum_x \sum_y \sum_z u_{ixyz} \cdot c_i \to \min(4),$$

за обмежень на функції f_1, f_2, f_3, де u_{ixyz} —1, якщо в місце (x, y, z) встановлено пристрій i-го типу, інакше 0; c_i — ціна пристрою i-го типу.

Розглянемо розміщення тепловійних датчиків чотирьох видів із діапазоном ефективної дії 150, 200, 300, 500 одиниць і вартістю 10, 25, 40, 100 одиниць відповідно.

Застосувавши запропонований підхід, отримаємо таке розміщення датчиків у приміщенні лабораторії:

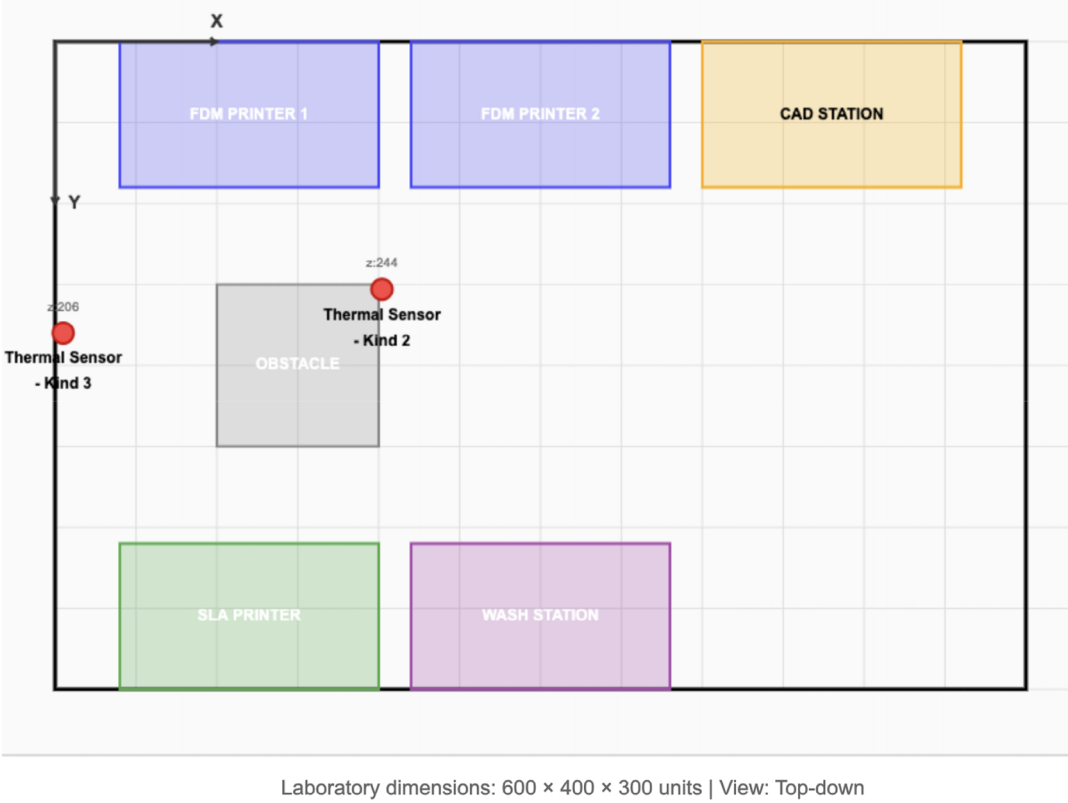


Рис. 3. Розміщення тепловійних датчиків у лабораторії 3D-друку (вигляд згори)

Висновки та обговорення

Концепція цифрового двійника є перспективною для університетського середовища, зокрема лабораторій електроніки чи 3D-друку. Для її застосування описано математичну модель цифрового двійника університетської лабораторії 3D-друку. Сформульовано математичну задачу вибору параметрів моделі, а саме типів датчиків і їх розміщення. Для вирішення задачі застосовано генетичний алгоритм зі зваженою фітнес-функцією, яка оцінює можливість встановлення датчика в певній позиції, охоплення ним об'єктів інтересу та вартість датчика. Охарактеризовано п'ять категорій датчиків, необхідних для цілісного цифрового двійника лабораторії: датчики середовища, стану обладнання, людської діяльності, безпеки та інвентаризації. Ретельний підбір датчиків важливий як для забезпечення комфортного робочого середовища для лаборантів, так і для належного функціонування такого обладнання лабораторії, як 3D-принтери чи паяльні станції.

Перспективним видається створення системи керування цифровим двійником на основі мікросервісної архітектури з використанням машин станів для координації операційних процесів, таких як аналіз даних, отриманих з датчиків, прийняття рішень щодо зміни режимів роботи датчиків і самих пристроїв. Основні підходи до подібного моніторингу та координації в загальних рисах сформульовано в [2; 14], водночас для подальшого розвитку цих підходів необхідне максимальне врахування специфіки предметної галузі.

Список літератури

1. Глибовець М. М. Еволюційні алгоритми / Микола Миколайович Глибовець, Наталія Михайлівна Гулаєва. — Київ : НаУКМА, 2013. — 828 с.
2. Олецький О. В. Координація мікросервісів із використанням машин станів [Електронний ресурс] / Олексій Віталійович Олецький, Віталій Олегович Моголівський // Наукові записки НаУКМА. Комп'ютерні науки. — 2025. — Т. 7. — С. 4–10. — Режим доступу: https://doi.org/10.18523/2617-3808.2024.7.4-10 (дата звернення: 18.06.2025). — Назва з екрана.
3. 3D printing and implementation of digital twins: current trends and limitations [Electronic resource] / Antreas Kantaros [et al.] // Applied system innovation. — 2021. — Vol. 5, no. 1. — P. 7. — Mode of access: https://doi.org/10.3390/asi5010007 (date of access: 12.06.2025). — Title from screen.
4. Attaran M. Digital Twin: benefits, use cases, challenges, and opportunities [Electronic resource] / Mohsen Attaran, Bilge Gokhan Celik // Decision analytics journal. — 2023. — Vol. 6. — P. 100165. — Mode of access: https://doi.org/10.1016/j.dajour.2023.100165 (date of access: 15.06.2025). — Title from screen.
5. Albelda-Estellés Ness M. C. Indoor relative humidity: relevance for health, comfort, and choice of ventilation system [Electronic resource] / Maria Coral Albelda-Estellés Ness // 3rd Valencia International Biennial of Research in Architecture, VIBRArch, 9–11 November 2022. — València, 2022. — Mode of access: https://doi.org/10.4995/vibrarch2022.2022.15237 (date of access: 18.06.2025). — Title from screen.
6. Building digital twins of 3D printing machines [Electronic resource] / T. DebRoy [et al.] // Scripta materialia. — 2017. — Vol. 135. — Pp. 119–124. — Mode of access: https://doi.org/10.1016/j.scriptamat.2016.12.005 (date of access: 12.06.2025). — Title from screen.
7. Developing a Digital Twin on a university campus to support efficient and sustainable buildings [Electronic resource] / Borja Garcia de Soto [et al.] // Creative construction e-conference 2023, Keszthely, Hungary, 20–23 June 2023. — Online, 2023. — Mode of access: https://doi.org/10.3311/ccc2023-007 (date of access: 15.06.2025). — Title from screen.
8. Digital Twin: architectures, networks, and applications. — [S. l.] : Springer, 2024.
9. Digital Twin evolution: a 30-year journey that changed industry | simio [Electronic resource] // Simio. — Mode of access: https://www.simio.com/digital-twin-evolution-a-30-year-journey-that-changed-industry/ (date of access: 15.06.2025). — Title from screen.
10. Digital Twins for defect detection in FDM 3D printing process [Electronic resource] / Chao Xu [et al.] // Machines. — 2025. — Vol. 13, no. 6. — P. 448. — Mode of access: https://doi.org/10.3390/machines13060448 (date of access: 12.06.2025). — Title from screen.
11. Digital Twins: recent advances and future directions in engineering fields [Electronic resource] / Kamran Iranshahi [et al.] // Intelligent systems with applications. — 2025. — P. 200516. — Mode of access: https://doi.org/10.1016/j.iswa.2025.200516 (date of access: 12.06.2025). — Title from screen.
12. Grieves M. W. Digital Twins: past, present, and future [Electronic resource] / Michael W. Grieves // The digital twin. — Cham, 2023. — Pp. 97–121. — Mode of access: https://doi.org/10.1007/978-3-031-21343-4_4 (date of access: 15.06.2025). — Title from screen.
13. Hamid R. A. Influence of Humidity on the Tensile Strength of 3D Printed PLA Filament [Electronic resource] / Rahimah Abdul Hamid, Fatima Hanem Hamezah, Jeefferie Abd Razak // Lecture Notes in Mechanical Engineering. — Singapore, 2022. — Pp. 497–502. — Mode of access: https://doi.org/10.1007/978-981-16-8954-3_47 (date of access: 18.06.2025). — Title from screen.
14. Oletsky O. On supervising and coordinating microservices within web applications on the basis of state machines / Oleksiy Oletsky, Vitalii Moholivskyi // CEUR workshop proceedings. — 2025. — Vol. 3909.
15. Risk analysis and cybersecurity enhancement of Digital Twins in dairy production [Electronic resource] / Tetiana Savchenko [et al.] // Technology audit and production reserves. — 2025. — Vol. 2, no. 2 (82). — Pp. 37–49. — Mode of access: https://doi.org/10.15587/2706-5448.2025.325422 (date of access: 15.06.2025). — Title from screen.

References

Albelda-Estellés Ness, M. C. (2022). Indoor relative humidity: Relevance for health, comfort, and choice of ventilation system. In *3rd valencia international biennial of research in architecture, vibrarch*. Editorial Universitat Politècnica de València. https://doi.org/10.4995/vibrarch2022.2022.15237.

Attaran, M., & Celik, B. G. (2023). Digital Twin: Benefits, use cases, challenges, and opportunities. *Decision Analytics Journal*, *6*, 100165. https://doi.org/10.1016/j.dajour.2023.100165.

DebRoy, T., Zhang, W., Turner, J., & Babu, S. S. (2017). Building digital twins of 3D printing machines. *Scripta Materialia*, *135*, 119–124. https://doi.org/10.1016/j.scriptamat.2016.12.005.

Digital Twin evolution: A 30-year journey that changed industry | simio. (n. d.). Simio. https://www.simio.com/digital-twin-evolution-a-30-year-journey-that-changed-industry/.

Digital Twin: Architectures, networks, and applications. (2024). Springer.

Garcia de Soto, B., Sanchis, Z., Ezzeddine, A., Mengiste, E., Padilla, M., & Karmacharya, S. (2023). Developing a Digital Twin on a university campus to support efficient and sustainable buildings. In *Creative construction e-conference 2023*. Budapest University of Technology and Economics. https://doi.org/10.3311/ccc2023-007.

Grieves, M. W. (2023). Digital Twins: Past, present, and future. In *The digital twin* (pp. 97–121). Springer International Publishing. https://doi.org/10.1007/978-3-031-21343-4_4.

Hamid, R. A., Hamezah, F. H., & Abd Razak, J. (2022). Influence of humidity on the tensile strength of 3D printed PLA filament. U *Lecture notes in mechanical engineering* (pp. 497–502). Springer Singapore. https://doi.org/10.1007/978-981-16-8954-3_47.

Hlybovets, M. M., & Hulaieva, N. M. (2013). *Evoliutsiini alhorytmy*. NaUKMA [in Ukrainian].

Iranshahi, K., Brun, J., Arnold, T., Sergi, T., & Müller, U. C. (2025). Digital Twins: Recent advances and future directions in engineering fields. *Intelligent Systems With Applications*, 200516. https://doi.org/10.1016/j.iswa.2025.200516.

Kantaros, A., Piromalis, D., Tsaramirsis, G., Papageorgas, P., & Tamimi, H. (2021). 3D printing and implementation of digital twins: Current trends and limitations. *Applied System Innovation*, *5* (1), 7. https://doi.org/10.3390/asi5010007.

Oletsky, O., & Moholivskyi, V. (2025). On supervising and coordinating microservices within web applications on the basis of state machines. *CEUR Workshop Proceedings*, *3909*.

Oletskyi, O. V., & Moholivskyi, V. O. (2025). Koordynatsiia mikroservisiv iz vykorystanniam mashyn staniv. *NaUKMA Research Papers. Computer Science*, *7*, 4–10. https://doi.org/10.18523/2617-3808.2024.7.4-10 [in Ukrainian].

Savchenko, T., Lutska, N., Vlasenko, L., Sashnova, M., Zahorulko, A., Minenko, S., Ibaiev, E., & Tytarenko, N. (2025). Risk analysis and cybersecurity enhancement of Digital Twins in dairy production. *Technology Audit and Production Reserves*, *2* (2(82)), 37–49. https://doi.org/10.15587/2706-5448.2025.325422.

Xu, C., Lu, S., Zhang, Y., Zhang, L., Song, Z., Liu, H., Liu, Q., & Ren, L. (2025). Digital Twins for defect detection in FDM 3D printing process. *Machines*, *13* (6), 448. https://doi.org/10.3390/machines13060448.

V. Moholivskyi

OPTIMIZATION OF 3D PRINTING LABORATORY DIGITAL TWIN SENSORS SELECTION AND PLACEMENT BASED ON GENETIC ALGORITHMS

The concept of the Digital Twin shows great prospects for the university environment. It can assist universities in managing complex infrastructure and dynamic educational activities. The Digital Twin provides a unified framework for understanding and optimizing interconnected systems through continuous data collection, real-time monitoring, and predictive modeling. It empowers instituitions to predict issues and create strategies to prevent them, instead of passively reacting to unexpected difficulties. In particular, the concept of the Digital Twin could be helpful for university laboratories such as electronics or 3D printing laboratories.

Approaches for creating digital twins in a university environment are studied. The set of sensors necessary to create a comprehensive digital twin of a university laboratory is discussed. Five categories of sensors are described: environmental, equipment condition, human activity, safety, and inventory. Careful selection of sensors is crucial both for ensuring a comfortable working environment for lab technicians and proper functioning of equipment such as 3D printers or soldering stations.

An approach to selecting and placing sensors for a digital twin of a 3D printing laboratory based on a genetic algorithm is considered. A mathematical model of the problem is formulated. The chromosome format, the weighted fitness function, and the representation of the three-dimensional space of the laboratory are proposed. The weighted fitness function evaluates the possibility to install a sensor in a particular location, the coverage of objects of interest, and the cost of a sensor. It seems promising to create a digital twin control system based on microservice architecture using state machines to coordinate operational processes such as analyzing data received from sensors, making decisions on changing the operating modes of sensors and the devices themselves.

Keywords: digital twin, genetic algorithm, 3D printing laboratory, sensor placement optimization, optimization problem.

Матеріал надійшов 18.06.2025

УДК 004.432.2
DOI: 10.18523/2617-3808.2025.8.138-148

Бублик В. В., Фітель Д. Р.

ОБ'ЄКТНО-ОРІЄНТОВАНА ПАРАДИГМА: PRO I CONTRA

У статті наведено критику парадигми об'єктно-орієнтованого програмування (ООП) та її найбільш поширених реалізацій. Досліджено історію виникнення та подальшої еволюції ООП, її сильні й слабкі сторони, а також спільні й відмінні риси між ООП та іншими парадигмами. Проаналізовано приклади вдалого співіснування парадигм ООП і узагальненого програмування на прикладі шаблонів у мові C++.

Ключові слова: парадигма програмування, об'єктно-орієнтована парадигма, узагальнене програмування.

Вступ

Стаття є розширеною версією доповіді авторів [2] на Міжнародній конференції TAAPSD 24, присвяченої особливостям застосування об'єктно-орієнтованої парадигми. Щоб уникнути суперечок про виникнення концепції об'єктно-орієнтованого програмування, одразу зазначимо, що в епоху становлення алгоритмічних мов програмування, скажімо, Фортрану, створеного в 1954–1957 рр. [9], В. С. Королюк і К. Л. Ющенко створили перший варіант мови адресного програмування, фіналізований у 1963 р. [3]. Адресне програмування виявилося предтечою важливих концепцій структурованих даних — структур і указників. На жаль, через відомі причини адресне програмування залишилося непоміченим на Заході, хоча його структури і указники досягли практично нинішньої форми, відомої з мови програмування C [3], створеної Деннісом Рітчі в 1972–1973 рр. під значним впливом із боку Алголу і Фортрану. За оцінкою Дональда Кнута, «the way C handles pointers, for example, was a brilliant innovation; it solved a lot of problems that we had before in data structuring and made the programs look good afterwards» [11].

Щоб дістатися об'єктно-орієнтованої парадигми, залишилося лише додати до структур і указників класи і об'єкти, успадкування і віртуальність. Усе це, одначе, було ще в 1967 р. у першій мові об'єктно-орієнтованого програмування, яка одержала назву SIMULA 67. З історією створення мови можна ознайомитися за статтею її авторів Крістена Нюгора і Уле-Югана Дала [13]. За свідченням авторів, головного впливу мова SIMULA 67 зазнала з боку алгоритмічної мови ALGOL 60.

Нарешті, за свідченням Б'ярне Страуструпа, коло Фортран-Алгол-C-SIMULA зімкнулося поєднанням у 1983 р. мов C і SIMULA 67 в новій мові програмування C++ [14]. За понад 40 років свого існування і інтенсивних застосувань мова C++ розвивалася в напрямі зростання рівня вбудованих до неї абстракцій, набуваючи багатьох нових рис. При цьому C++ залишалася мультипарадигменною, не наполягаючи на штучному впровадженні об'єктів усюди і скрізь, порівняно з деякими іншими мовами програмування.

C++:

```
int main()
{
    std::cout << «Hello, World!“ << std::endl;
}
```

```
Java:
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println(«Hello, World!»);
  }
}
```

У контексті мови C++ несподіваним видається висловлювання, приписане Бобом Кроуфордом (у важкодоступній роботі [6]) світовому програмісту номер один Едсгеру Дейкстрі: «Object-oriented programming is an extremely bad idea which could only have originated in California». Це звучить тим більш дивно, тому що одна з найцитованіших праць про структурне програмування [7], написана Дейкстрою у співавторстві з Уле-Юганом Далом і Тоні Хоаром, містить об'єктно-орієнтований розділ. Справді, впровадження об'єктів, успадкування та поліморфізму може зробити програми складнішими для розуміння та підтримки. А критичне ставлення Дейкстри можна зрозуміти, зважаючи на його позицію про те, що (добре розроблена) програма має близьку структурну аналогію з будь-якою (добре розробленою) математичною теорією [8]. Тому, на думку Дейкстри, робота програміста суттєво не відрізняється від роботи творчого математика.

В останньому твердженні прихована головна причина розбіжності підходу Дейкстри з сучасними тенденціями в інженерії програмного забезпечення: превалювання інженерного підходу над математичним. Зрештою, об'єктно-орієнтована парадигма — це не більше, ніж інструмент, призначений для розв'язування широкого кола задач. Але цей інструмент проявив себе у багатьох всесвітньо відомих проєктах. Особливо у випадках, коли проєкти мають великий обсяг, а структури даних виявляються надто складними. Інженерний підхід до програмування певним чином використовує рух знизу догори. Спочатку вивчається проблема, визначаються і виділяються її найважливіші частини. Вони відокремлюються від усього іншого, несуттєвого з погляду розробника, і утворюють абстрактне подання проблеми. Саме у виділенні і побудові релевантних абстракцій прихована потужність об'єктно-орієнтованої парадигми.

Узагальнені абстракції: варіативні шаблони

Нового поштовху парадигмі надала робота Андрея Александреску [4], у якій поєднано об'єктно-орієнтовану парадигму з парадигмою узагальненого програмування. На час виходу книжки узагальнене програмування вже з 1970-х років застосовували в деяких мовах програмування, а на межі 1980–1990-х воно потрапило до C++ у вигляді шаблонів функцій і класів. Починаючи з C++11 мова перетворилася на найпотужніший інструмент метапрограмування. Виявилося, що разом об'єктна орієнтованість і узагальненість досягають результатів, недосяжних ними поодинці.

Але все одно тотальна «об'єктизація» не завжди виправдана, як і механізми успадкувань, які вважають головним досягненням об'єктно-орієнтованого підходу. На думку Петера Готшлінга, «the most important benefit of classes in C++ for us is not the inheritance mechanisms but the ability to establish new abstractions and to provide alternative realizations for them» [10]. Тезу підтверджує наведений у книжці приклад шаблону для задачі пошуку екстремуму методом градієнтного спуску, максимально наближений до математичного алгоритму.

```
template <typename Value, typename T1, typename T2,
    typename FF, typename GG>
auto gradient_descent(Value& u, T1 s, T2 eps, FF f, GG g)
{
    auto val = f(u), delta = val;
    do {
          u -= s * g(u);
          auto new_val = f(u);
          delta = abs(new_val - val);
          val = new_val;
    } while (delta > eps);
    return u;
}
```

Цей код, написаний для функцій двох змінних, допускає прості варіанти узагальнень до *n*-місних функцій. Визначимо структуру R, відповідну *n*-вимірному простору,

```
struct R
{
    static const int n=10;
    double x[n]{};
};
```

на якому задаємо оператори множення і віднімання, задіяні в алгоритмі,

```
const R inline operator*(double s, const R& u)
{
    R res;
    for (int i = 0; i < R::n; ++i)
          res.x[i] = s * u.x[i];
    return res;
}

R& operator-=(R& u, const R& v)
{
    for (int i = 0; i < R::n; ++i)
          u.x[i] -= v.x[i];
    return u;
}
```

Цільову параболічну функцію і її градієнти задаємо лямбда виразами прямо в операторі виклику

```
gradient_descent(u, h, eps,
        [](const R& u)
        {
            double res = 0;
            for (int i = 0; i <R:: n; ++i)
                res += u.x[i] * u.x[i];
            return res;
        },
        [](const R& u)->const R
        {
            R res;
            for (int i = 0; i < R::n; ++i)
                res.x[i] = 2 * u.x[i];
            return res;
        });
```

Цікаво, що застосування варіативних шаблонів дозволить уникнути явної залежності від розмірності простору. Спочатку задано варіативну структуру простору довільної розмірності. Тут застосована особлива техніка часткової спеціалізації шаблону структури R, не визначеного, а лише попередньо декларованого

```
template <typename ...T> struct R;

template <typename T, typename ...P>
struct R<T, P...>
{
    T x;
    R<P...> y;
};

template<>
struct R<> {};
```

Завдяки такому визначенню в одному і тому самому фрагменті коду можуть співіснувати різні розмірності, наприклад,

```
R<double, double, double> u = { 1, 2, 3 };
R<double, double, double, double> v = { 7, 8, 9, 10 };
```

Наведемо варіативні рекурсивні визначення для операторів множення і віднімання, застосовані у функції gradient_descent. Зверніть увагу, параметр розмірності простору n став повністю зайвим.

```
template <typename ...T>
inline const R<T...> operator*(double s, const R<T...>& u);

template <typename T, typename ...P>
inline const R<T, P...> operator*(double s, const R<T, P...>& u)
{
    return { s * u.x, s * u.y};
}

template <>
inline const R<> operator*(double s, const R<>& u)
{
    return u;
}

template <typename ...T>
R<T...>& operator-=(R<T...>& u, const R<T...>& v);

template <typename T, typename ...P>
inline R<T, P...>& operator-=(R<T, P...>& u, const R<T, P...>& v)
{
    u.x -= v.x;
    u.y -= v.y;
    return u;
}

template <>
inline R<>& operator-=(R<>& u, const R<>& v)
{
    return u;
}
```

І, нарешті, цільова функція і її градієнти

```
template <typename ...T>
inline double f(const R<T...>& u);

template <typename T, typename ...P>
inline double f(const R<T, P...>& u)
{
    return u.x*u.x+f(u.y);
}

template <>
inline double f(const R<>& u)
{
    return 0;
}

template <typename ...T>
inline R<T...> g(const R<T...>& u);

template <typename T, typename ...P>
inline R<T, P...> g(const R<T, P...>& u)
{
    return { 2 * u.x, g (u.y)};
}

template <>
inline R<> g(const R<>& u)
{
    return {};
}
```

Виклик, як і раніше, може містити лямбда вирази у вигляді обгорток для варіативних функцій.

```
gradient_descent(u, h, eps,
        [](auto u) { return f(u); },
        [](auto u) { return g(u); });
```

Обидва приклади повністю відповідають процедурному стилю програмування, наявному в C++. Хоча застосування структур і операторів над ними (по суті об'єктів) виводить нас за рамки чистого C, не кажучи вже про варіативність шаблонів. Зауважимо, що, на думку загальновизнаного експерта в об'єктно-орієнтованому програмуванні на C++ Скотта Меєрса, перетворення функцій над класами на їхні методи безпідставно порушує їхню безпечність [12]. Ясно, що деякі функції класів мусять розміщуватися всередині класу. Це конструктори, деструктори, присвоєння і деякі інші. Вичерпний алгоритм розміщення функцій всередині і поза класами наводять Герб Саттер і Андрей Александреску [15].

Статичний і динамічний поліморфізм

Однією з причин розміщення функцій усередині класів може бути поліморфізм. Поліморфізм вимагає віртуальних методів, і це важлива суперечлива тема в об'єктно-орієнтовній частині C++. Незважаючи на тенденцію віртуалізації функцій у сучасних мовах програмування, C++ успішно поєднує використання невіртуальних і віртуальних функцій.

Традиційний спосіб організації ієрархій полягає у створенні абстрактного класу або інтерфейсу з чисто віртуальними функціями, реалізованими в похідних класах. Простим прикладом може бути інтерфейс функторів

```
class Functor
{
public:
    virtual double operator()(double) const =0;
    virtual ~Functor(){};
};
```

У похідному класі повинна бути реалізація віртуального оператора, наприклад, так

```
class Elliptic: public Functor
{
private:
    double _a, _b;
public:
    Elliptic (double a=1, double b=2):_a(a),_b(b){};
    ~Elliptic(){};

    double operator()(double x) const
    {
          return 1.0
                /sqrt(_a*_a*cos(x)*cos(x)
                +(_b*_b)*(sin(x)*sin(x)));
    }
};
```

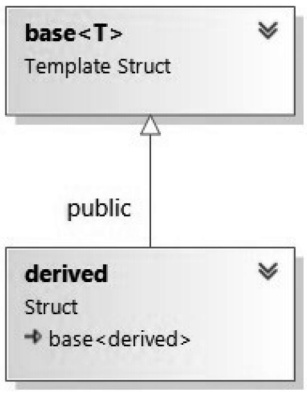


Рис. 1. Дивно рекурсивний шаблон

Але існує спосіб уникнення використання віртуальних функцій. Як приклад, розглянемо так званий «дивно рекурсивний шаблон» побудови неполіморфної ієрархії класів функторів. Рис. 1 пояснює будову шаблону, у якому базовий клас ієрархії параметризується похідним класом. Така архітектура дає змогу уникнути віртуальності в похідному класі.

Варіант застосування для статичної ієрархії функторів.

```
template<typename AnyFunctor>
struct Functor
{
    AnyFunctor& _functor;
    Functor(AnyFunctor& functor) :_functor(functor) {}
    const double operator()(double x) const
    {
          return _functor(x);
    }
};

class Elliptic : public Functor<Elliptic>
{
private:
    double _a, _b;
public:
    Elliptic(double a = 1, double b = 2):_a(a),_b(b),
              Functor<Elliptic>(*this){}
    ~Elliptic() {}

    double operator()(double x) const
    {
          return 1.0
                / sqrt(_a * _a * cos(x) * cos(x)
                + (_b * _b) * (sin(x) * sin(x)));
    }
};
```

Використання неполіморфного функтора дозволяє його відкритий (inline) виклик.

```
template<class AnyFunctor>
inline double process(const Functor<AnyFunctor>& f, double x)
{
    return f(x);
}
Приклад виклику:
Elliptic el(1, 2);
cout << process(el, 3) << endl;
```

Такий гібридний підхід дозволяє за певних умов досягти ефекту, який зазвичай потребує динамічного поліморфізму, без втрати швидкодії, до якої призводить виклик віртуальних функцій. Усе те саме, але без виклику віртуальної функції.

Іншим прикладом може бути використання статичного поліморфізму для вирішення проблеми подвійної диспетчеризації [1].

Сучасна критика ООП

Тож чи справді Дейкстра настільки критично висловився про ООП? І якщо так, то в чому могла бути суть критики?

Такий погляд, імовірно, пов'язаний із тим фактом, що Дейкстра працював над можливістю формального доведення коректності процедурних програм за допомогою логіки Флойда — Хоора. ООП, на відміну від процедурного, не має за собою формального математичного апарату, і тому важче формально аргументувати коректність об'єктно-орієнтованих програм. Але навряд чи це єдина або вичерпна причина.

Спробуємо уявити, якою б могла бути більш детальна критика сучасних реалізацій ООП.

Проблеми починаються з самого визначення парадигми. На жаль, поширені «доступні» визначення надто загальні, і спільні ознаки зводяться до понять класу і об'єктів. Мабуть, найбільш відоме визначення — формула «ООП = Інкапсуляція + успадкування + поліморфізм» — зовсім не відповідає суті парадигми, а лише перелічує окремі атрибути, які часто притаманні її реалізаціям. Найбільше уваги, як правило, приділяють успадкуванню, яке на практиці виявляється найбільш проблематичною рисою.

Спершу дамо визначення парадигмі ООП.

- ООП ґрунтується на об›єктах, пов'язаних ієрархіями вкладення, а не на алгоритмах.
- Кожен об'єкт суть екземпляр певного класу.
- Класи утворюють ієрархію успадкувань.

Елементи об'єктної моделі (за Г. Бучем [5])

- Абстрагування
- Інкапсуляція
- Модульність
- Ієрархічність
- Типізація
- Паралелізм
- Збережність

Ключова властивість ООП — ***абстрагування***, тобто можливість визначення нових абстракцій. Інші елементи певною мірою слугують для забезпечення абстрагування.

Інкапсуляція забезпечує вищий рівень абстракції цілої конструкції порівняно з рівнем її складових, виділяючи її істотні характеристики.

Пошук адекватних абстракцій — головне завдання об'єктно-орієнтованого програмування.

This is why the most important benefit of classes in C++ for us is not the inheritance mechanisms but the ability to establish new abstractions and to provide alternative realizations for them (Peter Gottschling [10]).

Стереотипи

ООП страждає від кількох поширених стереотипів, які значно ускладнюють його розуміння новачками.

«Все є об'єктом».

Імовірно, така фраза — це результат того, що деякі «чисті» ООП-мови змушують визначати всі функції в межах класів і автоматично загортають примітивні типи даних в об'єкти.

«Об'єкти взаємодіють між собою».

Під взаємодією мають на увазі те, що метод одного об'єкта може викликати метод іншого. Але ця взаємодія не визначає керування ходом виконання програми — об'єкти не є автономними підпрограмами, кожен з власним стеком виконання програми — це ознака потоків, а не об'єктів.

Надлишковість

Порівняно з процедурними, ООП мовам часто притаманна надлишковість, яка найпростіше демонструється прикладом Hello World на Java:

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}
```

У цій «чистій» ООП програмі немає жодного об'єкта, як і інкапсуляції, успадкування чи поліморфізму. Мінімальна ООП програма не є об'єктно-орієнтованою, бо ООП — це факультативний набір абстракцій, який можна поєднати з іншими, більш базовими парадигмами.

Успадкування і абстрактні класи

Успадкуванню класів приділяють більше уваги, ніж будь-якому іншому інструменту в ООП, і ним зловживають найбільше. Це потужний інструмент, який, як правило, важко використати правильно

і важко прочитати в готовому коді. Тимчасом як ООП парадигма має на меті зробити проєктування програм простішим, надаючи можливість зосередитись на вищому рівні абстракції, успадкування класів на практиці робить програми складнішими. Тому виникають підстави для критики.

- Успадкування — механізм визначення ієрархій класів, а не повторного використання. Успадкування часто використовують, щоб надати новому класу певний функціонал наявного. Але на практиці завжди краще виокремити потрібний функціонал в абстракцію і агрегувати її.
- Успадкування ускладнює читання і налагодження програм замість того, щоб спрощувати.
- Абстрактні класи завжди краще замінити інтерфейсами.
- Перевизначення методів (override) майже завжди порушує принцип підстановки Лісков (Liskov Substitution Principle). Під час перевизначення дуже легко порушити інваріанти базового класу.
- Успадкування може призвести до конфлікту сигнатур.
- Множинне успадкування — ще більш складний, і, як наслідок, ризикований інструмент.

Квінтесенція антипатернів успадкування — це абстрактний клас, який містить спільну реалізацію, але окремі функції залишає для визначення похідним класам.

```
public abstract class MyAbstractClass
 {
     public int DoWork(int value)
     {
         int preProcessed = this.PreProcess(value);
         int processed = this.Process(preProcessed);
         int postProcessed = this.PostProcess(processed);
         return postProcessed;
     }
     protected abstract int Process(int value);
     private int PreProcess(int value) { … }
     private int PostProcess(int value) { … }
 }
```

Однак така архітектура швидко наражається на ряд проблем:

- Неможливо суттєво змінити хід виконання у похідних класах, можна лише визначити окремий функціонал, передбачений базовим класом.
- Нащадкам легко порушити контракт, особливо якщо клас має змінні атрибути.
- Під час налагодження потрібно перемикатися між різними рівнями абстракції.
- Важко тестувати логіку в базовому класі.
- Важко тестувати логіку в похідних класах.
- Ніякого reuse на практиці.
- Ще гірше, якщо абстрактний метод залежить від атрибутів, визначених у базовому класі.

На практиці завжди краще змінити таке «успадкування» агрегацією чітко визначених абстракцій:

```
public class MyAbstractClass
{
    private readonly IProcessor preProcessor;
    private readonly IProcessor processor;
    private readonly IProcessor postProcessor;
    public MyAbstractClass(IProcessor preProcessor,
           IProcessor processor, IProcessor postProcessor)
    {
       this.preProcessor = preProcessor;
       this.processor = processor;
       this.postProcessor = postProcessor;
    }
    public int DoWork(int value)
    {
        int preProcessed = this.preProcessor.Process(value);
        int processed = this.processor.Process(preProcessed);
        int postProcessed = this.postProcessor.Process(processed);
        return postProcessed;
    }
}
```

Інтерфейси та незмінність

Повертаючись до конкретних інструментів, наявних в ООП мовах, безумовно, найпотужнішим і найчистішим інструментом для створення абстракцій є інтерфейс. Інтерфейс — *мінімальний* контракт між частинами програми без зайвих умов чи обмежень. Інтерфейс робить можливим поліморфізм і заохочує повторне використання.

Поліморфізм випливає напряму з можливості визначати абстракції через інтерфейси. Схематика поліморфізму описує ідеальну ієрархію класів. Як приклад розглянемо ієрархію узагальнених стеків. Кожен клас другого рівня слугує реалізацією абстракції стеку за рахунок застосування чисто віртуальних функцій, наявних в інтерфейсі.

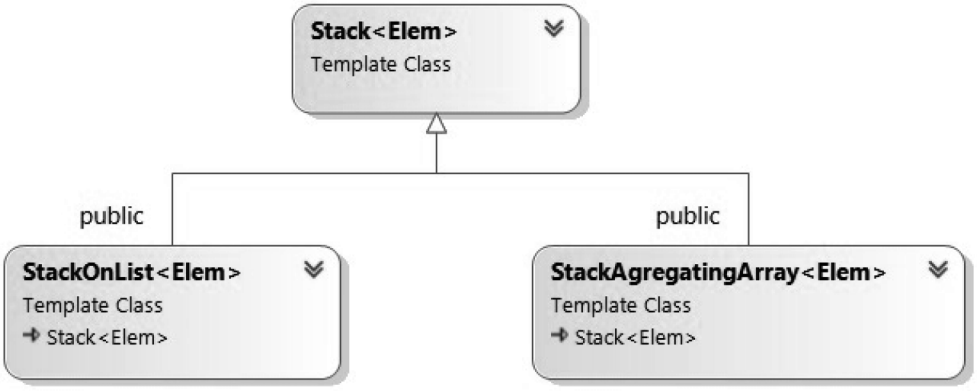


Рис. 2. Ієрархія успадкувань стеку

Така ієрархія допускає її доповнення і розширення, не втручаючись до внутрішньої будови її елементів. Наприклад, її можна розширити до конструкції стеку з підгляданням, доповнивши успадкування іншими видами ієрархії, а саме агрегацією.

Важливо розуміти, що між похідними класами не повинно бути успадкування. Успадкуванню підлягає інтерфейс. Інші класи можуть використовувати реалізацію один одного знову ж із застосуванням агрегації.

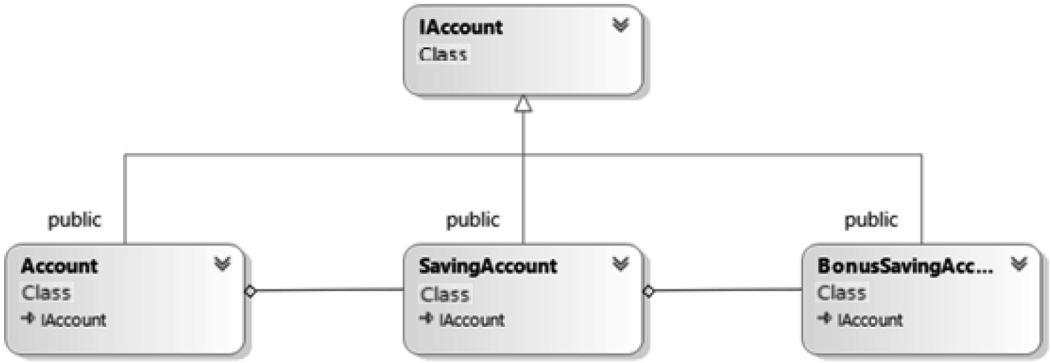


Рис. 3. Спільна ієрархія успадкування інтерфейсу і агрегації реалізацій

Існує багато інших корисних властивостей, акумульованих в об'єктно-орієнтованій парадигмі. Одна з найважливіших — це незмінність. Це, мабуть, найкорисніша властивість незалежно від парадигми. Вона значно спрощує читання і налагодження програм, робить можливими програми з багатьма потоками виконання, зближує об'єктно-орієнтовану і функціональну парадигми, особливо на рівні метапрограмування, детальний розгляд можливостей якого виходить за рамки цієї статті. **Все, що може бути незмінним, мусить бути незмінним!**

Нарешті, важливий аспект незмінності — інваріанти, наявність яких переводить прості зібрання даних у класи. Конструктор створює об'єкт у коректному стані, він обов'язково перевіряє коректність значень параметрів. Кожен метод переводить об'єкт з коректного стану в коректний стан. Особливо це стосується пересувної семантики. Після переміщення об'єкт перебуває в невизначено-

му, але дієспроможному стані. Кожен відкритий метод перевіряє коректність значень параметрів. Об'єкт перебуває в коректному стані після аварійної ситуації в одному з методів. Об'єкт перебуває в коректному стані, якщо він використовується багатьма потоками одночасно. Не повинно бути неявних умов на порядок виклику відкритих методів. Якщо компілятор дозволяє викликати метод, то жодних додаткових обмежень не повинно бути.

Висновки

Об'єктно-орієнтована парадигма після майже шістдесяти років досі залишається популярною, незважаючи на те, що певні її недоліки були добре відомі та задокументовані ще у 1980-х роках [6]. Наведена вище критика стосується як концепції ООП, так і конкретних інструментів, які стали стандартними в більшості об'єктно-орієнтовних мов. Тим не менш, певні ідеї, зароджені в ООП, довели свою ефективність і здатність бути застосованими за нових умов.

Поза сумнівом, найважливіша риса ООП — це можливість створення абстракцій за допомогою механізмів мови, адже вона дозволяє масштабувати розробку програм, водночас роблячи їх простішими для розуміння. Значна частина критики ООП походить якраз від того, що деякі мови роблять такий механізм обов'язковим для всіх програм, що не завжди є виправданим. Але збалансоване створення абстракцій із дотриманням належних принципів організації архітектури програмного забезпечення слугує явним свідченням дієвості об'єктно-орієнтованої парадигми.

Найбільш вдалий механізм створення абстракцій, який походить від об'єктно-орієнтованої парадигми — інтерфейс, адже він дозволяє виокремити необхідний контракт між частинами програми, при цьому не ускладнюючи їх, на відміну від успадкування. Успадкування класів зазвичай ускладнює програми, і в більшості випадків може бути замінене агрегацією класів.

Механізми абстрагування, які надає ООП, часто підсилюються іншими парадигмами програмування. У статті наведено приклади вдалого поєднання ООП з узагальненим програмуванням у мові C++, яке дозволило досягти результатів, які раніше були недосяжними цими парадигмами поодинці.

Список літератури

1. Бублик В. В. До питання створення статичного патерну проєктування для подвійної диспетчеризації модельних сигнатур / В. В. Бублик // Наукові записки НаУКМА. Комп'ютерні науки. — 2021. — Том 4. — С. 64–71.
2. Бублик В. В. Деякі особливості застосування об'єктно-орієнтованої парадигми [Електронний ресурс] / В. В. Бублик, Д. Р. Фітель // Теоретичні та прикладні аспекти побудови програмних систем: праці 15 міжнародної науково-практичної конференції, Київ, 23–24 грудня 2024. — Режим доступу: https://taapsd.ukma.edu.ua/.
3. Ющенко Е. Л. Адресное программирование / Е. Л. Ющенко. — Киев : Техн. лит., 1963. — 286 с.
4. Alexandrescu A. Modern C++ Design: Generic Programming and Design Patterns Applied / A. Alexandrescu. — Addison Wesley, 2001. — 352 p.
5. Booch G. Object-Oriented Analysis and Design with Applications / G. Booch. — 3rd edition. Addison-Wesley Professional, 2007. — 720 p.
6. Crawford B. Object-Oriented Programming: The Good, the Bad, and the Ugly / B. Crawford // TUG Lines. — 1989. — Vol. 32. — Aug.–Sep. — Pp. 7–11.
7. Dahl O.-J. Structured Programming / O.-J. Dahl, E. W. Dijkstra, C. A. R. Hoare. — Academic Press, 1972. — 220 p.
8. Dijkstra E. W. Hoe wiskundig programmeren is [Electronic resource] / E. W. Dijkstra // EWD 261. — Mode of access: https://www.cs.utexas.edu/~EWD/transcriptions/EWD02xx/EWD261.html.
9. Fortran. Programmer's Reference Manual. The Fortran Automatic Coding System for the IBM 704 EDPM [Electronic resource] // IBM Corp. — 1956. — 51 p. — Mode of access: http://bitsavers.informatik.uni-stuttgart.de/pdf/ibm/704/704_FortranProgRefMan_Oct56.pdf.
10. Gottschling P. Discovering Modern C++ / P. Gottschling. — Addison Wesley, 2015. — 472 p.
11. Knuth D. Computer Literacy Bookshops Interview / D. Knuth. — December 7th, 1993.
12. Meyers S. How Non-Member Functions Improve Encapsulation [Electronic resource] / S. Meyers. — Mode of access: https://www.drdobbs.com/cpp/how-non-member-functions-improve-encapsu/184401197.
13. Nygaard K. The Development of the SIMULA Languages / K. Nygaard, O.-J. Dahl // ACM SIGPLAN Notices. — 1978. — Vol. 13, no. 8. — Pp. 245–272.
14. Stroustrup B. The Design and Evolution of C / B. Stroustrup. — 1st edition. —Addison-Wesley Pub Co, 1994. — 506 p.
15. Sutter H. C++ Coding Standards: 101 Rules, Guidelines, and Best Practices / H. Sutter, A. Alexandrescu. — Addison-Wesley, 2004. — 240 p.

References

Alexandrescu, A. (2001). *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley Professional.

Booch, G. (2007). *Object-Oriented Analysis and Design with Applications*. Addison-Wesley Professional.

Boublik, V. (2021). Do pytannia stvorennia statychnoho paternu proektuvannia dlia podviinoi dyspetcheryzatsii modelnykh syhnatur. *NaUKMA Research Papers. Computer Science*, *4*, 64–71 [in Ukrainian].

Boublik, V., & Fitel, D. (2024). Deiaki osoblyvosti zastosuvannia obiektno-oriientovanoi paradyhmy. *Teoretychni ta prykladni aspekty pobudovy prohramnykh system: pratsi 15 mizhnarodnoi naukovo-praktychnoi konferentsii*. Kyiv, December 23–24, 2024. https://taapsd.ukma.edu.ua/ [in Ukrainian].

Crawford, B. (1989). Object-Oriented Programming: The Good, the Bad, and the Ugly. *TUG Lines*, *32*, 7–11.

Dahl, O.-J., Dijkstra, E. W., & Hoare, C. A. R. (1972). *Structured Programming*. Academic Press, 1972.

Dijkstra, E. W. Hoe wiskundig programmeren is. *EWD 261*. https://www.cs.utexas.edu/~EWD/transcriptions/EWD02xx/EWD261.html.

Fortran. Programmer's Reference Manual. The Fortran Automatic Coding System for the IBM 704 EDPM. IBM Corp. 1956. http://bitsavers.informatik.uni-stuttgart.de/pdf/ibm/704/704_FortranProgRefMan_Oct56.pdf.

Gottschling, P. (2015). *Discovering Modern C++*. Addison Wesley, 2015.

Knuth, D. (1993) *Computer Literacy Bookshops Interview*. December 7.

Meyers, S. (2000). *How Non-Member Functions Improve Encapsulation*. https://www.drdobbs.com/cpp/how-non-member-functions-improve-encapsu/184401197.

Nygaard, K., & Dahl, O.-J. (1978). The Development of the SIMULA Languages. *ACM SIGPLAN Notices*, *13* (8), 245–272.

Stroustrup, B. (1994). *The Design and Evolution of C*. Addison-Wesley Pub Co.

Sutter, H., & Alexandrescu, A. (2004). *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices*. Addison-Wesley.

Yushchenko, E. L. (1963). *Adresnoe programmyrovanye*. K. Tech. Lit. [in Russian].

V. Boublik, D. Fitel

OBJECT-ORIENTED PARADIGM: PROS AND CONS

This paper critically examines the object-oriented programming (OOP) paradigm, tracing its theoretical foundations and evaluating its most widely adopted modern implementations. Particular attention is given to the criticisms that have emerged over time, notably those stemming from Edsger Dijkstra's remark that "object-oriented programming is an exceptionally bad idea…." We aim to interpret and contextualize Dijkstra's dissatisfaction with contemporary OOP, offering a practical perspective on its implications.

We begin by exploring the origins of the object-oriented paradigm, highlighting its evolution from procedural programming and the fundamental conceptual shift it introduces—namely, the incorporation of programmable abstractions. Given OOP's widespread adoption across numerous programming languages, we distill its core principles and examine their shared characteristics. Additionally, we provide a refined definition of the object-oriented paradigm based on these key principles, emphasizing its role as a powerful and flexible mechanism for creating abstractions, rather than a simplistic combination of more specific programming language features.

Our analysis also outlines the advantages and limitations inherent in common OOP principles, including the SOLID principles. We present specific recommendations, along with examples of both successful and flawed implementations of various OOP concepts. Importantly, we stress that the object-oriented paradigm functions best as an optional tool for abstraction rather than an imposed constraint.

Finally, we examine the paradigm's most effective attributes, particularly its ability to coexist with and enhance other programming methodologies. To illustrate this adaptability, we present a case study on template metaprogramming in C++, demonstrating how OOP can facilitate the creation of sophisticated custom abstractions with precise control over their behavior and degrees of reusability that are not achievable by a single programming paradigm.

Keywords: programming paradigm, object-oriented paradigm, generic programming, template metaprogramming.

Матеріал надійшов 28.05.2025

УДК 519.682.1
DOI: 10.18523/2617-3808.2025.8.149-157

Проценко В. С.

ДЕНОТАЦІЙНА СЕМАНТИКА ОДНОВИМІРНИХ МАСИВІВ

Розглянуто імперативну мову програмування, об'єктами якої є цілі змінні — скалярні та одновимірні масиви. Оператори мови — присвоєння, введення, виведення, умовний, циклу і блок. Призначення боку — введення локальних цілих змінних. Одновимірний масив a[k], де k>0 — ціле додатне число (розмірність масиву). Робота з масивами здійснюється поелементно. Доступ до окремого елементу масиву здійснює операція індексування a[e], e — цілий вираз. Значення e — число від 0 до k-1. Наведено повну формальну специфікацію мови. На основі специфікації побудовано інтерпретатор мови програмування.

Ключові слова: мова програмування, програма, оператор, вираз, змінні, одновимірні масиви, синтаксис, денотат, семантична функція, Haskell, синтаксичний аналіз, інтерпретатор.

Мова з одновимірними масивами

Розглянуто імперативну мову програмування, що має цілі скалярні дані і одновимірні масиви. Ця мова є розширенням мови зміни станів [1] одновимірними масивами. Кожна змінна a в цій мові — або ціла скалярна змінна a, або цілий одновимірний масив a[k], де k>0 — ціле додатне число (розмірність масиву). Такий масив має k елементів, які розміщуються послідовно і нумеруються як a[0], a[1], ..., a[k-1]. Робота з масивами здійснюється тільки поелементно.

Кожна програма мови може вводити цілі значення, обробляти їх і виводити цілі значення. Програма — це окремий оператор, як правило, блок із описом локальних цілих змінних (скалярних або масивів) та списком операторів — тілом блоку. Головну обробку даних виконують оператори присвоювання v := e, введення **read** v, виведення **write** e, умовний **if** (e) s, циклу **while** (e) s, де v — змінна, e — вираз, s — оператор. Усі вирази e в операторах обчислюють скалярне ціле значення. Якщо a[k] — цілий одновимірний масив, то доступ до окремого елементу масиву здійснює операція індексування — запис виду a[e]. Значення iv цілого виразу e — повинно бути ціле число від 0 до k-1. В операторах умовному і циклу значення iv>0 цілого виразу e еквівалентно логічному значенню True.

Далі наведено приклад програми в цій мові. Програма вводить масив a з 10 елементів, в окремому блоці (з локальними змінними is і c) впорядковує його, використовуючи «бульбашкове» сортування, і виводить впорядкований масив.

```
{ int i, a[10];
  i := 0; while (10 - i) {read a[i]; i:= i+1 };
  {  int is, c; is := 1;
     while (is) { is := 0: i := 0;
          while (9 – i) {
        if (a[i] – a[i+1]) {
           is := 1; c := a[i+1]; a[i+1] := a[i]; a[i] := c };
        i := i+1} }
 };
 i := 0; while (10-i) {write a[i]; i := i+1}
}
```

Опис мови програмування охоплює синтаксис і семантику. Синтаксис мови програмування — це опис усіх конструкцій, що утворюють елементи мови. Семантика мови вказує «значення» синтаксичних конструкцій. Для опису семантики використовується денотаційна семантика. Спочатку фіксуються денотати (як правило, математичні об'єкти) найпростіших синтаксичних об'єктів. Потім із кожною складеною синтаксичною конструкцією пов'язується семантична функція, яка за денотатами компонентів конструкції обчислює її значення — денотат. Оскільки програма є конкретною

синтаксичною конструкцією, то її денотат можна визначити, застосувавши відповідну семантичну функцію. Зауважимо, що сама програма при обчисленні її денотата не виконується.

Конкретний і абстрактний синтаксис

Конкретний синтаксис, що описується за допомогою БНФ [2], можна розділити на три частини: лексика, вирази і оператори.

У лексиці описуються все лексичні конструкції, які використовує мова.

symbol = *';'* | *'{'* | *'}'* | *'('* | *')'* | *'['* | *']'* | *'z'* ;
addOp = '+' | '_' ;
mulOp = '*' | '/' | '%' ;
identifier = *letter* , { (*digit* | *letter*) } ;
крім 'int' 'if' 'while' 'read' 'write'
decimal = *digit* , { *digit* } ;
letter = 'A' | … | 'Z' | 'a' | … | 'z' ;
digit = *'0'* | *'1'* | *'2'* | *'3'* | *'4'* | *'5'* | *'6'* | *'7'* | *'8'* | *'9'* ;
identif = *letter* , { (*digit* | *letter*) } ;
reserved = 'int' | 'if' | 'while' | 'read' | 'write' ;

Обчислення значень виконують вирази.

var = *identifier* , [*'['* , *expr* , *']'*] ;
expr = *term* , {*addOp* , *term*} ;
term = *factor* , {*mulOp* , *factor*} ;
factor = *decimal* | '(' , *expr* , ')' | *var* ;

Послідовність дій описують оператори

program = *stmt ;*
stmt = 'while' , '(' , *expr* , ')' , *stmt* ;
| 'if' , '(' , *expr* , ')' , *stmt* ;
| 'read' , *identifier* ;
| 'write' , *expr* ;
| *var* , ':=' , *expr* ;
| '{' , [*defin*] , { *defPrS* } , *stmt* , {';' , *stmt* } , '}' ;
defin = 'int' , *defvar* , { ',' , *defvar* } , '; ' ;
defvar = *identifier* , [*'['* , *decimal* , *']'*] ;

callSt = '(' , *identifier* , { ',' , *identifier* } , ')' ;
definS = 'int' , *identifier* , { ',' , *identifier* } , '; ' ;
defPrS = 'proc' , *identifier* , *procedS* ;
procedS = '(' , [*identifier* , { ',' , *identifier* }] , ')' , *stmt* ;

Для опису семантики мови віддають перевагу абстрактному синтаксису. Зв'язок між об'єктами конкретного і абстрактного синтаксису виконує синтаксичний аналіз.

Абстрактний синтаксис визначається типами: бінарної операції Op, змінної Var, виразу Expr, оголошення змінної VarDef, оператору Stmt і програми Program.

```
data Op    = Plus | Minus | Times | Div | Mod
                deriving (Show, Eq)
type Var   = (String, Maybe Expr)
data Expr  = VarOp Var  | Const Integer   | BinOp Op Expr Expr
                deriving (Show, Eq)
type VarDef = (String, Maybe Int)
data Stmt  = Assign Var Expr  | Read Var |  Write Expr | If Expr Stmt
           |  While Expr Stmt  | Block [VarDef] [Stmt]
                deriving (Show, Eq)
type Program = Stmt
```

Синтаксичний аналіз

Для реалізації синтаксичного аналізу використана бібліотека Parsec. Синтаксичний аналіз у цій бібліотеці виконують спеціальні функції — аналізатори. Синтаксичний аналізатор p, що розпізнає значення типу a на початку рядка типу String, має тип Parser a. У випадку успіху залишок рядка, що аналізується, передається наступному аналізатору.

На першому кроці будуються аналізатори, що розпізнають лексичні одиниці мови. Мова з одновимірними масивами дозволяє вживати проміжки між довільними лексичними одиницями. Синтаксичний аналізатор lexem p за аналізатором p будує аналізатор, який розпізнає конструкцію p і всі проміжки за нею. identif — аналізатор, що розпізнає послідовність букв і десяткових цифр, котра починається з букви. Синтаксичний аналізатор identifier — розпізнає ідентифікатори, що не збігаються із зарезервованими словами, відповідно аналізатор reserved st — розпізнає зарезервоване слово st.

```
identif :: Parser String
identif = do {c <- letter; cs <- many alphaNum; return (c:cs)}
lexem :: Parser a -> Parser a
lexem p = do {a <- p; spaces; return a}
identifier :: Parser String
identifier = try (do {nm <- lexem identif;
                 if (any(nm==) [«int»,»read»,»write»,»if»,»while»])
                  then unexpected («reserved word « ++ show nm) else return nm} )
reserved :: String -> Parser ()
reserved st = try( lexem (do {_ <- string st; notFollowedBy alphaNum}))
symbol :: String ->  Parser ()
symbol st = lexem (do {_ <- string st; return ()})
oper  :: String -> Op -> Parser (Expr -> Expr -> Expr)
oper str bop = do {symbol str; return (BinOp bop)}
mulOp, addOp :: Parser (Expr -> Expr -> Expr)
mulOp = (oper «*» Times) <|> (oper «/» Div) <|> (oper «%» Mod)
addOp = (oper «+» Plus) <|> (oper «-» Minus)
```

На наступному кроці будуємо аналізатори виразів. Головні з них: decimal — розпізнає число, var — розпізнає змінну, можливо з індексом, factor, term, expr — розпізнають різні частини виразу.

```
parens, brackets :: Parser a -> Parser a
parens p = do {symbol «(«; e <- p; symbol «)»; return e}
brackets p = do {symbol «[«; e <- p; symbol «]»; return e}
decimal :: Parser Integer
decimal = do {ds <- many1 digit; return (read ds)}
dimension :: Parser Int
dimension = do {ds <- many1 digit; return (read ds)}
var  :: Parser Var
var  = do {v  <- identifier;
      me <- option Nothing (do {e<-brackets expr; return (Just e)}) ;
      return (v,me) }
factor, term, expr :: Parser Expr
factor  = parens expr  <|> do {v <- lexem decimal; return (Const v)}
     <|> do {v <- var; return (VarOp v)} <?> «factor»
term = chainl1 factor mulOp
expr = chainl1 term addOp
```

На заключному кроці будуються аналізатори операторів, оголошень даних і програми. Ведуча функція parseProgram s запускає основний аналізатор program, що розпізнає в рядку s програму, і аналізує результат його роботи. Якщо рядок s містить синтаксично правильну програму, то функція повертає її представлення в абстрактному синтаксисі — дане типу Program. У випадку синтаксичної помилки процес обчислення переривається і генерується інформація про помилку.

```
braces :: Parser a -> Parser a
braces p = do {_ <- symbol «{«; e <- p; _ <- symbol «}»; return e}

semiSep1, commaSep1 :: Parser a -> Parser [a]
semiSep1 p = sepBy p (symbol «;»)
commaSep1 p = sepBy p (symbol «,»)
stmt :: Parser Stmt
stmt = do {reserved «while»; e <- parens expr; s <- stmt; return (While e s)}
   <|> do {reserved «if»; e <- parens expr; s <- stmt; return (If e s)}
   <|> do {reserved «read»; v <- var; return (Read v)}
   <|> do {reserved «write»; e <- expr; return (Write e)}
   <|> do {v <- var; symbol «:=»; e <- expr; return (Assign v e)}
   <|> braces (do {dll <- option [] defin;
                   sl <- semiSep1 stmt; return (Block dll sl)})
```

```
   <?> «statement»
 defin :: Parser [VarDef]
 defin = do {reserved «int»; il <- commaSep1 varDef; symbol «;»; return il}
 varDef :: Parser VarDef
 varDef = do {v  <- identifier; mi <- option Nothing
                 (do {i<- brackets dimension; return (Just i)}) ;
                 return (v,mi) }
program :: Parser Stmt
program = do {spaces; r <- stmt; eof; return r}
parseProgram :: String -> Program
parseProgram s = case parse program «» s  of
          {Left er -> error (show er); Right p -> p}
```

Контекстні умови

На жаль, повний опис деяких складних конструкцій процедурної мови контекстно-вільним синтаксисом (конкретним чи абстрактним) дати не можна. В цьому випадку використовують контекстно залежні правила, що описують необхідні додаткові умови.

Контекстні умови є предикатами, які накладають на об'єкти, визначені правилами абстрактного синтаксису, додаткові контекстно залежні обмеження. В подальшому при заданні семантики вважають, що використовують лише об'єкти, які задовольняють ці обмеження. Більшість контекстних умов пов'язана з правильним уживанням даних у програмі.

Для зберігання інформації про дані програми (Sctp) використовують «статичне» середовище (EnvSt)

```
data Sctp    = Ar | Sc deriving (Show, Eq)
type EnvSt =  [(String, Sctp)]
```

Використаному в контекстних умовах середовищу env відповідає список усіх, відомих у цьому місці програми, імен змінних. Ідентифікатор id — ім'я скалярної змінної, якщо в списку є пара (id, Sctp), або ім'я змінної-масиву, якщо в списку є пара (id, Ar). Контекстні умови задаємо для об'єктів: Program (iswfProgram), Stmt (iswfStmt) і Expr (iswfExpr). Функція iswfExpr, що перевіряє контекстні умови виразу Expr, повертає значення Maybe Sctp.

Якщо iswfExpr e env == Nothing, то контекстні умови для виразу e в статичному середовищі env не виконуються. Якщо iswfExpr e env == Just st, то умови виконуються і значення виразу або скалярне (st==Sc) або масив (st==Ar). Контекстні умови використовують допоміжний предикат: distinct is який перевіряє, що список is не має дублікатів.

```
distinct :: Eq a => [a] -> Bool
distinct []       = True
distinct (v:vs) = notElem v vs  &&  distinct vs
iswfProgram :: Program -> Bool
iswfProgram pr = iswfStmt pr []
iswfStmt :: Stmt -> EnvSt -> Bool
iswfStmt (Block vdl sl) env =
    (distinct (map fst vdl)) &&
    (let env1 = [(v,(maybe Sc (\_->Ar)) d) | (v,d) <- vdl] ++ env
     in all (flip iswfStmt env1) sl)
iswfStmt (Assign (v, Just ei) e) env =
    (lookup v env == Just Ar) &&
   (iswfExpr ei env == Just Sc) && (iswfExpr e env == Just Sc)
iswfStmt (Assign (v,Nothing) e) env =
    (lookup v env == Just Sc) && (iswfExpr e env == Just Sc)
iswfStmt (Read (v, Just ei)) env  =
     (lookup v env == Just Ar) && (iswfExpr ei env == Just Sc)
iswfStmt (Read (v, Nothing)) env  = (lookup v env == Just Sc)
iswfStmt (Write e) env                = (iswfExpr e env == Just Sc)
iswfStmt (If e s) env   =
      (iswfExpr e env == Just Sc) && iswfStmt s env
iswfStmt (While e s) env   =
     (iswfExpr e env == Just Sc) && iswfStmt s env
iswfExpr :: Expr -> EnvSt -> Maybe Sctp
iswfExpr (VarOp (v,Just e)) env  =
```

```
        if (lookup v env == Just Ar) && (iswfExpr e env == Just Sc)
                    then Just Sc else Nothing
iswfExpr (VarOp (v, Nothing)) env = lookup v env
iswfExpr (Const _) _                  = Just Sc
iswfExpr (BinOp _ e1 e2) env     =
   if (iswfExpr e1 env == Just Sc) && (iswfExpr e2 env == Just Sc)
                    then Just Sc else Nothing
```

Денотати

Завданням денотаційної семантики мови програмування є пов'язати з конструкціями мови певні математичні об'єкти (списки, таблиці, функції) —денотати, які задають «значення» (семантику) конструкцій. Семантика мови — набір семантичних функцій, які відображають синтаксичні конструкції мови, зокрема програму, у відповідні денотати.

```
type Work = ([Integer], [Maybe Integer], [Integer])
```

Для опису семантики мови використовують стан — дане типу Work. Це кортеж із трьох елементів (inp, stg, out), що моделює три набори значень, із якими працює програма: список цілих [Integer] inp містить вхідні дані (файл введення); список значень [Maybe Integer] stg зберігає поточні значення всіх змінних програми (пам`ять); список цілих [Integer] out містить результуючі дані (файл виведення). Зауважимо, що значення скалярної змінної програми — це елемент списку stg, а значення масиву розмірності k — це k послідовно розташованих елементів списку stg. Якщо поточне значення деякої змінної є ціле v, то в списку stg її поточне значення відображають як Just v, а якщо змінній ще не було присвоєно ніякого значення, то поточне значення відображають як Nothing.

Для роботи зі станом семантичні функції використовують функції, які управляють розміром другої компоненти робочого стану allocate, getBase і free; працюють зі значенням змінних getValue і updValue; працюють з файлом виведення writeValue та файлом введення readInput і dropInput.

```
allocate :: Int -> Work -> Work
allocate k = \(inp,stg,out) ->
    let beg = [Nothing | _ <- [1..k]]    in (inp, beg++stg,out)
getBase :: Work -> Int
getBase = \(_,stg,_) -> length stg
free :: Int -> Work -> Work
free k = \(inp, stg,out) -> (inp, drop k stg,out)
getValue:: Int -> Work -> Integer
getValue k = \(_,stg,_) ->
    let p = length stg - k  in case stg!!p of
        { Just v  -> v;  Nothing -> error «valueNothing»}
updValue :: Int ->  Integer -> Work -> Work
updValue k v  = \(inp,stg,out) ->
    let {p = length stg – k;  (beg,end) = splitAt p stg;
         stg1 = beg ++ [Just v] ++ (tail end)}
    in  (inp,stg1,out)
writeValue :: Integer -> Work -> Work
writeValue v = \(inp, stg, out) -> (inp, stg, out ++ [v])
readInput :: Work -> Integer
readInput = \(inp,_,_) ->
      if null inp then error «readInput» else head inp
dropInput :: Work -> Work
dropInput = \(inp,stg,out) -> (tail inp,stg,out)
```

Управління динамічною пам'яттю в робочому стані (inp,stg,out) реалізує друга компонента stg. Функції allocate k (inp,stg,out) і free k (inp,stg,out) працюють зі списком stg як зі стеком: перша — на початку виконання блоку, який вводить локальні змінні, додає в список stg k елементів для розміщення значень локальних змінних блоку, а друга після закінчення виконання блоку вилучає їх. (Значення k визначається розмірностями локальних масивів і кількістю локальних скалярів. Наприклад для означення **int** i, c[10], m; маємо k=12.)

Функція getBase (inp,stg,out) повертає довжину списку stg. Якщо список stg, перед виконанням блоку, який вводить локальні змінні **int** i, c[10], m; має b (результат функції getBase) елементів stg = [a1, ..., ab], то після виконання функції allocate список stg буде мати (12+b) елементів stg = [vm, vc9, ..., vc1, vc0, vi, a1, ..., ab]. vci — позначає елемент списку stg, в якому буде зберігатися поточ-

не значення елементу c[i] локального масиву c в процесі виконання блоку. Коли необхідно при виконанні блоку отримати доступ до змінної vi з адресою k, досить взяти елемент списку stg з індексом p = length stg – k.

Функція getValue k (inp,stg,out) знаходить поточне значення змінної адресою k. Якщо це значення Just v, то повертає v, а якщо Nothing, то генерує помилку "valueNothing". Функція updValue k v (inp,stg,out) реалізує зміну поточного значення змінної з адресою k. Значення елементу списку stg з індексом p = length stg – k покладають Just v.

Функція writeValue v (inp,stg,out) змінює стан, додаючи в кінець списку out ціле значення v.

Функція readInput (inp,stg,out) повертає перше значення зі списку inp, якщо список порожній, то генерує помилку "readInput". Функція dropInput (inp,stg,out) змінює стан, вилучаючи перший елемент списку inp. Використовують відразу після функції readInput.

Семантичні функції

Наявність блоків приводить до того, що в конкретних програмах одне ім'я може посилатися на різні значення (опис скалярної змінної у внутрішньому і у зовнішньому блоках). Крім того, одне ім'я масиву пов'язане з декількома значення (елементи з різними індексами).

Для пов'язування імен змінних з їхніми денотатами (номерами елементів списку stg, де зберігається поточне значення змінної) використовують середовище — значення типу Env.

```
type Env = [(String,(Maybe Int, Int))]
```

Якщо **int** a — скалярна змінна, з якою в env зв'язано елемент (a,(Nothing,3)), то stg[3] містить поточне значення a. Якщо **int** b[4] — масив, з яким в env зв'язано елемент (b,(Just 4,7)), то stg[7] містить поточне значення елемента b[0], stg[6] містить значення b[1], stg[5] — значення b[2], stg[4] — значення b[3] відповідно.

У семантичних функціях використовують такі допоміжні функції.

```
applyBo :: Op -> Integer -> Integer -> Integer
applyBo Plus v1 v2    = v1 + v2
applyBo Minus v1 v2 = v1 - v2
applyBo Times v1 v2 = v1 * v2
applyBo Div v1 v2      = if v2 /= 0 then div v1 v2 else error «DivOnZero»
applyBo Mod v1 v2     = if v2 /= 0 then mod v1 v2 else error «ModOnZero»
alloc :: Int -> (Maybe Int) -> Int
alloc s mi = s+(maybe 1 id mi)
extEnv :: [VarDef] -> Int -> Env -> Env
extEnv vs b = \env ->
          let {(ns,mi) = unzip vs; mls  = zip mi (scanl alloc 0 mi);
                nenv1 = (zip ns [(m, (b+i+1)) | (m,i)<- mls]) ++ env}
           in nenv1
  getLocation :: String -> Env -> (Int,Int)
  getLocation s env = case fromJust $ lookup s env of
           {((Just n), k) -> (n,k);  (Nothing,  k) -> (1,k)}
  eLocation :: Var -> Env -> Work -> Int
  eLocation (s, Just ei) env = \w ->
           let {i  = fromIntegral (eExpr ei env w) ; (n,k) = getLocation s env}
           in if i>=0 && i<n then k+i else error "Index"
  eLocation (s, Nothing) env = \_w -> let (_,k) = getLocation s env in k
```

При обчисленні виразу використовують функцію applyBop op v1 v2, яка обчислює результат застосування бінарної операції op до цілих значень v1 і v2. У випадку обчислення операції Div або Mod, якщо другий операнд рівний нулю, то генерується помилка "DivOnZero" або "ModOnZero". Синтаксичний аналіз, обробляючи оголошення змінної, формує інформацію про її розмірність — об'єкт типу Maybe Int. Якщо змінна скаляр, то її розмірність — Nothing, а якщо масив із k елементів — то Just k. Функція alloc s mi за розміром s списку stg визначає його новий розмір після виділення пам'яті під значення змінної з розмірністю mi. (Друга компонента стану — список stg — реалізує динамічне управління пам'яттю: при вході в блок — на початку списку виділяється пам'ять, а при виході — звільнюється.) Функція extEnv vs ps b env за середовищем env, в якому потрібно виконати блок із локальними змінними vs і розміром b списку stg, який моделює пам'ять, будує нове розширене середовище.

Функція getLocation s env за іменем змінної s у середовищі env формує інформацію про частину списку stg, виділеного під зберігання значення змінної s, повертаючи пару (n,k): n — розмір участку (для скалярної змінної 1)і k — адреса його початку. Функція eLocation (s, mei) env за іменем змінної s і індексом mei вираховує адресу елемента пам'яті, де зберігається значення, в середовищі env. Якщо s — скалярна змінна, то mei == Nothing і адрес розміщення її значення getLocation s env. В іншому випадку mei == Just ei, за виразом ei вираховується значення індексу і. Якщо це значення задовольняє розмірність масиву, то вираховується адреса, в іншому випадку генерується помилка "Index".

Денотатами виразів Expr і операторів Stmt мови є відповідно функції зміни стану типу Work -> Integer і Work -> Work, а денотатом програми Program — функція типу [Integer] -> [Integer]. Ці денотати будують семантичні функції eExpr, iStmt і iProgram, відповідно. Функція extEnv vs b env викликається в функції iStm при формуванні денотату блоку (Block vs sts). Функція виділяє пам'ять під значення оголошених у блоці локальних змінних vs і за поточним розміром b списку stg формує денотати локальних змінних vs, розширюючи середовище env. Більшість семантичних функцій для доступу до денотатів змінних використовують середовище як додатковий параметр.

```
eExpr :: Expr -> Env -> Work -> Integer
eExpr (VarOp var) env      = \w -> getValue (eLocation var env w) w
eExpr (Const v) _              = \_ -> v
eExpr (BinOp op e1 e2) env = \w ->
                        applyBo op (eExpr e1 env w) (eExpr e2 env w)
extEnv :: [VarDef] -> Int -> Env -> Env
extEnv vs b = \env ->
        let {(ns,mi) = unzip vs; mls  = zip mi (scanl alloc 0 mi);
             nenv1 = (zip ns [(m, (b+i+1)) | (m,i)<- mls]) ++ env}
        in nenv1
iStmt :: Stmt -> Env -> Work -> Work
iStmt (Assign var e) env = \w ->
            updValue (eLocation var env w)(eExpr e env w) w
iStmt (If e s) env            = \w ->
             if eExpr e env w > 0 then iStmt s env w else w
iStmt wh@(While e s) env = \w ->
             if eExpr e env w > 0 then iStmt wh env (iStmt s env w) else w
iStmt (Block vs sts) env      = \w ->
             let {k  = foldl alloc 0 (map snd vs); w1 = allocate k w;
                  nenv = extEnv vs (getBase w) env
                  w2 = foldl (\wr s -> iStmt s nenv wr) w1 sts }
             in free k w2
iStmt (Read var) env        = \w ->
              let {v = readInput w;
                   w1 = updValue (eLocation var env w) v  w}
              in dropInput w1
   iStmt (Write e) env         = \w -> writeValue (eExpr e env w) w
   iProgram :: Program -> [Integer] -> [Integer]
   iProgram prog ix =
         let {w = (ix, [],[]); (_,_,ox) = iStmt prog [] w} in ox
```

Реалізація

Використавши мову Haskell [3], наведено денотаційний опис імперативної мови, що містить цілі скалярні змінні і цілі одновимірні масиви. Опис охоплює синтаксис і денотаційну семантики мови.

Синтаксис містить: конкретний синтаксис, який описується синтаксичними правилами в розширеній БНФ; абстрактний синтаксис, що містить типи бінарної операції Op, змінної Var, виразу Expr, оголошення змінної VarDef, оператору Stmt і програми Program; контекстні умови, які накладають на об`єкти, визначені правилами абстрактного синтаксису, додаткові контекстно залежні обмеження.

Денотаційна семантика охоплює: основу денотатів мови — тип Work, що задає стан програми мови з масивами, та базові функції, які працюють із ним; денотати синтаксичних конструкцій мови (Expr, Stmt, Program) — функції зміни стану Work -> Integer, Work -> Work і [Integer] -> [Integer]; семантичні функції, які будують денотати синтаксичних конструкцій: eExpr, iStmt, iProgram.

Зауважимо, що всі функції базові, денотати і семантичні — чисті функції.

Використавши чисту функцію parseProgram, яка реалізує синтаксичний аналіз мови, будуємо інтерпретатор interpret (тобто реалізацію) мови. Якщо програма (рядок s) синтаксично правильна, то інтерпретатор, застосувавши до результату синтаксичного аналізу pr семантичну функцію iProgram, будує денотат програми, який потім застосовує до вхідних даних ix — iProgram pr ix. interpret — чиста функція, але якщо на кроці синтаксичного аналізу, перевірки контекстних умов або при застосуванні денотату виникає помилка, то обчислення переривається, генеруючи відповідне повідомлення (функція error).

Для зручності роботи будуємо основну функцію interpretFile sf ix, яка на початку вводить програму мови з файлу sf, а потім використовує чисту функцію interpret для інтерпретації програми.

```
    interpret :: String -> [Integer] -> [Integer]
interpret st ix = let {pr = parseProgram st; wf = iswfProgram pr}
                  in if wf then iProgram pr ix  else error «Contex»
interpretFile :: String -> [Integer] -> IO()
interpretFile sf ix = do {st <- readFile sf; print (interpret st ix)}
```

Найпростіший спосіб реалізації, враховуючи невеликий об'єм наведених Haskell-функцій, зібрати всі елементи в одному файлі DenotArFull.hs і скористатися інтерпретатором Haskell ghci.

Наведемо структуру файлу DenotArFull.hs.

```
{-# OPTIONS_GHC -Wall #-}
module DenotArFull where
import Text.ParserCombinators.Parsec
-- Абстракний синтаксис і типи,
-- що використовуються: Work, Sctp, EnvSt, Env
......
-- Синтаксичні аналізатори і Конкретний синтаксис БНФ в коментарях
.....
-- Контекстні умови
.......
-- Семантичні функції
.......
-- Інтерпретація
......
```

Далі наведено приклад роботи з програмою DenotArFull.hs, яка інтерпретує програму сортування (файл bubbleSort.txt), в командному рядку (програма cmd.exe). Програму cmd потрібно виконувати в каталозі, що містить файли DenotArFull.hs і bubbleSort.txt.

```
> ghci DenotArFull.hs
.....
ghci> interpretFile «bubbleSort.txt» [45,2,4,78,12,45,78,13,67,20]
[2,4,12,13,20,45,45,67,78,78]
```

Список літератури

1. Проценко В. С. Опис імперативної мови програмування у Haskell / В. С. Проценко // Наукові записки НаУКМА. Комп'ютерні науки. — 2021. — Т. 4. — С. 72–77.
2. Information technology. syntatic metalanguage. extended BNF. 1996.
3. Kurt W. Get programming with Haskell / W. Kurt. — Manning Publications, 2018.

References

Information technology. syntatic metalanguage. extended BNF (ISO/IEC 14977). (1996).

Kurt,W. (2018). *Get programming with Haskell.* Manning Publications.

Protsenko, V. S. (2021) Opys imperatyvnoi movy prohramuvannia u Haskell. *NaUKMA Research Papers. Computer Science, 4*, 72–77.

V. Protsenko

DENOTATIONAL SEMANTICS OF ONE-DIMENSIONAL ARRAYS

An imperative programming language is considered that has integer scalar data and one-dimensional arrays. Each variable a in this language is either an integer scalar variable a or an integer one-dimensional array a[k], where k>0 is a positive integer. Such an array has k elements, which are sequentially arranged and numbered as a[0], a[1], ..., a[k-1]. A program is a separate statement, usually a block with a

description of local integer variables (scalar or arrays) and a list of statements – the body of the block. All expressions in the statements calculate a scalar integer value. Work with an array is carried out only element by element. If a[k] is an entire one-dimensional array, then access to an individual element of the array is performed by an indexing operation - a record of the form a[e]. The value of the integer expression e – must be an integer from 0 to k-1.

In a program, one name can refer to different values (a variable description in the inner and outer blocks). In addition, one array name is associated with multiple values (elements with different indices). To associate variable names with their denotations (memory addresses where the current value of the variable is stored), an environment is used – a value of type Env.

type Env = [(String,(Maybe Int, Int))]

If Int a is a scalar variable with the denotation (Nothing,3) associated with it in the environment, then 3 is the memory address containing the current value of a. If Int b[4] is an array with which the denotation (Just 4,7) is associated, then 7 is the memory address containing the current value of element b[0], 6 is the address where the value b[1] is found, 5 is the address of b[2], and 4 is the address of b[3], respectively.

The functional language Haskell is used to describe this imperative language, which includes syntax and denotational semantics. All functions included in the description are pure. It is shown how to use these functions to build a pure function – a language interpreter.

Keywords: programming language, program, statement, expression, variable, one-dimensional array, syntax, denotation, semantic function, Haskell, interpreter.

Матеріал надійшов 25.06.2025

УДК 004.41
DOI: 10.18523/2617-3808.2025.8.158-166

Постніков М. А., Гороховський С. С.

АВТОМАТИЗОВАНА СИСТЕМА ВИЯВЛЕННЯ АНОМАЛІЙ У БІЗНЕС-ДАНИХ

У статті описано проведений аналіз процесу виявлення аномалій у бізнес-даних, відомі програмні рішення, сформульовано вимоги до системи та описано розроблену автоматизовану програмну систему виявлення аномалій. Ця система складається з програмних модулів, має високу адаптивність, є легкою до модифікації і зручною у використанні.

Розроблена система повністю відповідає поставленим раніше вимогам: легкість у налаштуванні забезпечується інтерфейсом користувача і інтерактивним процесом, гнучкість і легкість кастомізації — обраними технологіями та архітектурними абстракціями, надійність — розділенням компонентів через чергу задач, функціональні вимоги — розробленими складовими модулями. Система виконує поставлену задачу автоматизованого виявлення аномалій у бізнес-даних і відповідає сучасним стандартам у галузі даних.

Ключові слова: виявлення аномалій, програмна система, вимоги до системи, архітектура, технології, вебдодаток, FastAPI, Celery, REST API, Pandas.

Вступ

«Дані — нафта XXI століття» — ці слова британського математика Кліва Гамбі у 2006 р. [1] (тут і далі переклад наш) ознаменували головну характеристику сучасного цифрового світу. Дані сьогодні є найціннішим і затребуваним ресурсом. Однак дані не мають прямої користі в тому вигляді, в якому вони є, — для застосування вони потребують оброблення і інтерпретації [1]. Великий обсяг даних потребує автоматизованого оброблення.

Таким чином, для забезпечення своєї діяльності компанії мусять базувати свою аналітику на даних, що стає дедалі важчим завданням через постійне зростання їхньої варіативності та об'єму. Головними проблемами тут є надійність даних (data reliability) та швидка ідентифікація зміни трендів. Проблема надійності даних є наріжною в індустрії, оскільки сучасні бізнеси оперують даними, що походять із різноманітних джерел (бухгалтерські системи, системи звітності, продажів, CRM системи, дані активності соціальних мереж, дані систем обліку співробітників тощо.). Такі дані часто непридатні до використання одразу, оскільки містять помилки та неточності. Їх знаходження є класичною проблемою аналізу даних. Також бізнес стикається з потребою швидко ідентифікувати незвичну поведінку даних, що є індикатором певних суттєвих змін, які вносять корективи до бізнес-процесів. Вирізняти такі випадки серед великих обсягів даних швидко — складна задача, що важко розв'язується за допомогою звичних практик аналізу даних.

Тому багато бізнесів звертаються до техніки автоматизації виявлення аномалій (outlier detection), що допомагає визначати одиниці даних, які не дотримуються патерну поведінки інших даних. Ці методи широко застосовують для вирішення проблем на кшталт оптимізації витрат, виявлення випадків шахрайства тощо.

1. Аналіз досліджень про виявлення аномалій

Станом на сьогодні можна сказати, що виявлення аномалій (outlier detection, також anomaly detection) — це усталена галузь науки про дані (data science), що досліджує підходи і методи вирішення проблеми визначення аномалій у даних. Однією з перших відомих наукових робіт на цю тему вважають статтю викладача Королівської школи Лондона Ф. Еджворс (Francis Edgeworth) «On discordant observations» [4], де автор визначає дискордантні спостереження як ті, що «...справляють враження відмінності від інших спостережень відносно до закону, за яким розподілені їхні частоти». З плином часу дослідженням цієї проблеми приділяли дедалі більше уваги.

Перші ґрунтовні напрацювання щодо визначення аномалій зробили дослідники в галузі статистики. Так, класичне визначення аномалії наводить Ф. Ґрабс (Frank Grubbs): «Аномальним спостереженням, або "аномалією", є те, що помітно виділяється серед інших членів вибірки, де воно зустрічається» [9, p. 1]. Автор виділяє два типи аномальних спостережень: ті, «...що є серйозним свідченням про варіативність даних випадкового характеру...» [9, p. 1], і ті, «...що є результатом серйозних відхилень від процедури [статистичного] експерименту або помилки в підрахунках чи записах числового значення» [9, p. 1]. Дослідник наводить опис розроблених ним критеріїв визначення аномалій у статистичних даних і зазначає, що «...майже всі [розроблені] критерії для аномалій ґрунтуються на передбаченні про нормальний (Ґаусовий) розподіл або популяцію, що лежить в основі [даних]» [9, p. 3]. Додатково автор підкреслює, що знаходження аномалій є насамперед індикатором потреби подальшого дослідження і виявлення причини їхньої появи, оскільки наявність аномалій не обов'язково означає, що такі спостереження потребують особливого поводження чи мають бути відкинуті [9, pp. 20–21].

Ґрунтовним дослідженням визначення аномалій у статистичних даних стала монографія Д. Гокінза (Douglas Hawkins) «Identification of Outliers» [10]. За Гокінзом, аномалія — це «...спостереження, що настільки сильно відхиляється від інших спостережень, що виникає підозра, що воно було згенероване іншим механізмом» (тут і далі переклад наш) [10, p. 1]. Автор виділяє два основні механізми походження аномалій: коли дані походять від розподілу з «важкими хвостами» (heavy-tailed distributions) і коли дані походять від двох розподілів, один з яких генерує нормальні дані, а інший — аномалії [10, pp. 1–2].

Найбільш значущою роботою останніх років у сфері стало дослідження науковців з Університету Міннесоти В. Чандола (Varun Chandola), А. Банерджі (Arindam Banerjee) і В. Кумара (Vipin Kumar), опубліковане в 2009 р. [3], де було проаналізовано понад сотню досліджень і підсумовано наявні звершення в цій галузі. Так, дослідники дають визначення аномаліям як «...патернам в даних, що не підкорюються поняттю про нормальну поведінку» [3, p. 2].

Через те, що аномалії є широким поняттям, техніки виявлення аномалій мають різноманітне практичне застосування: *виявлення шкідливої поведінки, виявлення дефектів, виявлення новизни (novelty detection).* Загалом, виявлення аномалій може бути застосоване у будь-якій галузі, де є потреба відстежувати зміни в даних. Особливо корисною вона буде в тих випадках, де на аналіз усіх потрібних даних бракує ресурсів.

Проведене нами дослідження показало, що серед відкритого програмного забезпечення на сьогодні немає системи, яка могла б задовольнити потребу бізнесу у швидкому виявленні аномалій у найбільш поширеній формі бізнес-даних, а саме — в часових рядах. Ця проблема і стала поштовхом для нашого дослідження. Потрібно було на основі досвіду наявних розробок сформулювати вимоги до програмної системи (ПС), провести проєктування архітектури і вибір технологій і розробити систему автоматизованого виявлення аномалій у бізнес-даних (часових рядах).

Виявлення аномалій може бути «на сторожі» певних важливих параметрів, сповіщаючи лише тоді, коли справді потрібна додаткова увага. Ключовим є те, що виявлення аномалій не є інтерпретацією даних: аномальні дані не завжди є чимось добрим або поганим, вони не завжди означають, що щось пішло не так, як планувалось. Вони дають знати про критичну зміну в поведінці певних показників. Як і з будь-яким моделюванням, аномалії не є стовідсотково точними: так, можливість хибнопозитивних (false-positive) чи хибнонегативних (false-negative) результатів залежить від якості вхідних даних, їхньої попередньої обробки, вибраної моделі і її параметрів, варіативності і обсягу вибірки.

Нині відомо багато різних технік виявлення аномалій. Вони різняться передусім за рахунок предметної сфери застосування, характеру вхідних даних, бажаної точності передбачень та об'єму зусиль на впровадження. Основними методами серед них є [3, p. 3] статистичні, гістограмні, класифікаційні, кластеризаційні, інтервальні. Наприклад, гістограмний метод полягає в такому: за допомогою гістограми будується частотний профіль даних (тільки аномальних або тільки нормальних), його використовують для порівняння з гістограмою тих даних, які ми хочемо аналізувати, і оцінка аномальності випливає з різниці частотних профілів даних [3, p. 37; 8, p. 4]. Ядерна оцінка щільності (kernel density) використовується дуже схожим чином; можна сказати, що цей метод є гістограмним, який використовує згладжування, за рахунок чого вирішується проблема впливу кількості стовпчиків гістограми на дані [3, p. 38; 8, p. 4].

3. Вимоги до програмної системи

Для бізнесу сьогодні ефективне використання даних розглядається не тільки як конкурентна перевага, а і як необхідність для виживання [2]. Автори дослідження говорять про те, що інтерпретація даних для отримання з них користі є основною задачею бізнесу, і це зробити зовсім непросто. Зокрема, одною з проблем тут є якість даних (поняття «якість даних» тут і далі використовується як для позначення власне якості даних, англ. data quality, так і цілісності даних, англ. data integrity, та інших термінів, що мають під собою здатність даних бути використаними для отримання правильних висновків). За даними авторів дослідження, 71 % бізнесу стикається з проблемами якості даних, і вони називають це «бар'єром для досягнення цілі (*отримання користі з*)» [2].

Велика частина цих даних належить до *часових рядів* (time series data). За нашим досвідом, більшість потреб бізнесу у даних задовольняють саме такі дані: вони представляють зміну певних показників із часом, наприклад, це може бути кількість продажів, дохід, кількість активних користувачів, маркетингові показники, такі як retention rate (частка користувачів, які продовжують користуватись продуктом після певного часу). Цих даних на сьогодні набагато більше, ніж можливостей у більшості бізнесів для їхнього аналізу. Дослідники з SRM Institute of Science and Technology та VIT Bhopal University виділяють такі характеристики часових рядів: тренд, сезонність, циклічність і випадкову компоненту (шум), і наголошують, що аналіз часових рядів дозволяє бізнесу покращити процес прийняття рішень за рахунок кращого розуміння трендів на ринку [11].

Отже, особливості бізнес-даних можна сформулювати так: *необхідність визначати незвичну інформацію серед великої кількості даних, представлених у вигляді часових рядів.* Тобто метою ПС буде *надання можливості легкого визначення аномалій у часових рядах*. Система має бути інструментом, за допомогою якого можна *швидко* і *легко* налаштувати відслідковування аномалій у таких даних.

Для аналізу були відібрані ті ПС, які декларують про вирішення задачі відстежування аномалій: *ChaosGenius* [6], *CueObserve* [7], *Elementary Data* [8].

Серед особливостей усіх цих систем можна виділити такі: підтримка багатьох баз і сховищ даних (Postgres, Google BigQuery, Amazon Redshift); підтримка інтеграцій із різними системами для відправки сповіщень (Slack, електронна пошта); автоматизоване виявлення аномалій, яке запускається за розкладом; наявність вебінтерфейсу для перегляду інформації і налаштувань; підтримка різних моделей визначення аномалій; наявність налаштувань для моделей визначення аномалій.

Серед недоліків цих систем називають такі. Жодна з основних систем більше не підтримується станом на час проведення дослідження. Налаштування систем є заплутаним і неочевидним, зокрема в процесі налаштування бракує інтерактивності і можливості подивитись, як налаштування впливають на результат. У системах мала кількість підтримуваних моделей визначення аномалій.

Однак головними недоліками цих систем є їхня непристосованість до потреб бізнесу. Ці інструменти не дають можливості швидко і інтерактивно налаштувати відстеження аномалій, їхні налаштування не інтуїтивні і не дозволяють швидко розібратись в системі або в тому, як найкраще налаштувати параметри визначення, враховуючи особливості тих чи інших даних. Бізнес потребує інструменту, який дозволяв би за короткий час налаштувати відстеження аномалій у часових рядах, які періодично оновлюються. Це дозволило б бізнесу без особливих витрат часу і грошей стежити за багатьма показниками.

Тому ми визначили основні вимоги до нашої ПС: легке і швидке визначення аномалій у часових рядах за допомогою інтерактивних налаштувань; легкість у налаштуванні з використанням вебінтерфейсу; гнучкість і легкість кастомізації системи; надійність. Оскільки бізнес буде покладатись на таку систему у своїй діяльності, система має мати високі показники стійкості до помилок і відмов і мати змогу обробляти великі масиви даних. Передусім це має бути забезпечено архітектурними рішеннями: наприклад, використання черг задач.

4. Програмна реалізація системи

Як мову програмування ми обрали мову Python. В індустрії оброблення даних ця мова є найпопулярнішою і стандартом де-факто галузі, що дозволить користувачам за потреби легко модифікувати систему.

Для реалізації серверної частини обрали бібліотеку FastAPI: це популярний асинхронний вебфреймворк, який почали розробляти в 2018 р. Фреймворк повністю асинхронний і базується на

typehints — оголошеннях про типи в Python [5]. Підтримка typehints реалізована через бібліотеку Pydantic, що є найпоширенішою бібліотекою для валідації даних у Python. Асинхронність FastAPI дозволила підвищити продуктивність роботи вебсервера за рахунок зменшення процесорного часу, що в синхронному режимі витрачається на затратні I/O операції.

FastAPI часто називають мікрофреймворком (micro framework) за аналогією з Flask через те, що він надає «з коробки» небагато можливостей, а весь додатковий функціонал потрібно реалізовувати окремо через під'єднання сторонніх бібліотек або власноруч написану логіку. Разом з фреймворком були використані такі бібліотеки: Psycopg3 як рушій для спілкування з БД Postgres, SQLAlchemy як ORM для СКБД Postgres, Jinja2 як рушій серверного рендерингу, Alembic для міграцій БД.

Для маніпуляції даними ми обрали бібліотеку Pandas, вона є найбільш популярною такою бібліотекою на Python і також є стандартом де-факто у галузі даних, що також дозволить легко робити зміни в системі. Дуже багато людей критикують Pandas за неінтуїтивність і нелогічність API, відсутність розподіленості і порівняно низьку швидкість роботи, але для роботи з помірним розміром даних цього цілком достатньо, тому вибір було зроблено на користь цієї бібліотеки, а не її конкурентів на кшталт Polars, які ще не набули такої популярності.

Правильним є відокремити роботу вебсервера від роботи з даними (запуску перевірок), тому ці компоненти мають якимось чином спілкуватись між собою. Для цього ми обрали Celery — чергу задач на Python, яка дозволяє легко передавати задачі на виконання іншим компонентам, зручно стежити за їхнім виконанням, станом, успішністю і забирати результат. Оскільки перевірки мають виконуватись за розкладом, потрібен компонент, який би це робив: для цього обрано Celery-Redbeat. Як чергу повідомлень обрано Redis як надшвидку базу даних, що зберігається в оперативній пам'яті: між компонентами системи будуть передаватись повідомлення малого розміру, тому зберігання даних в оперативній пам'яті є допустимим. Для адміністрування Celery було обрано інструмент Flower: він надає можливість через вебінтерфейс дивитись за станом черг, задач, їхнім виконанням і можливими помилками.

Замість застосування планувальника Redbeat для Celery також розглядали вибір іншого планувальника або оркестратора задач, наприклад Airflow або Dagster, але через складність цих інструментів, трудоємність інтеграції та малу утилізацію їхнього потенціалу для поставленої задачі ці опції були визначені як надлишкові та відкинуті.

Для демонстрації алгоритмів визначення аномалій було обрано такі бібліотеки: Statsmodels для виявлення за допомогою STL декомпозиції, Prophet як бібліотеку аналізу часових рядів за допомогою неконтрольованого навчання.

Для відправки електронних листів було вирішено обрати Gmail як найбільш популярну систему електронної пошти. Аутентифікація в Gmail проводиться через протокол OAuth 2.0. Відправка повідомлень в Slack відбувається за допомогою Slack API. Запити до обидвох систем відправляються через бібліотеку httpx.

Як спосіб реалізації вебчастини системи було вирішено вибрати стратегію змішаного рендерингу: серверний рендеринг відбувається за допомогою Jinja2, а клієнтський рендеринг за допомогою JavaScript бібліотек. Для комунікації між сервером і клієнтом при клієнтському рендерингу обрано концепцію RESTful API. Стисло кажучи, ця концепція полягає в атомарності і незалежності один від одного запитів між клієнтом та сервером: головним є те, що запит до сервера має мати всю необхідну інформацію, щоб його опрацювати, тобто запит не має контексту.

На боці клієнта реалізацію вебсторінок було зроблено за допомогою мови розмітки HTML 5 та мови стилів CSS 3, з використанням JavaScript для динамічності сторінок. Інтерфейс зроблено на базі бібліотеки компонентів Tabler: вона постачається у вигляді CSS та JS файлів, які під'єднуються до потрібного вебсайту, і надає широкий набір заготовлених компонентів, зовнішній вигляд яких зроблений в одному стилі. Tabler дуже популярний через те, що побудований на базі Bootstrap — CSS-бібліотеки стилів, яка повсюдно використовується в вебсторінках, оскільки дозволяє максимально швидко розробляти красиві вебсторінки, адаптовані до різних пристроїв і форматів екранів. Для клієнтського рендерингу було обрано бібліотеку Alpine.js, яка надає можливість декларативно описувати залежності компонентів, і при зміні цих залежностей компоненти будуть перебудовуватись з новими даними, що позбавляє потреби робити оновлення вмісту і вигляду компонентів вручну.

Керування кодовою базою відбувається через систему контролю версій Git, а як віддалене сховище для Git репозиторію обрано GitHub як стандарт де-факто в індустрії. Опис до Git комітів зроблений за допомогою концепції Conventional Commits.

Для виконання поставлених вимог на базі обраних технологій було розроблено ПС автоматизованого виявлення аномалій. Архітектурним патерном виконання системи є мікросервісна архітектура.

Система складається з трьох компонентів: вебсервер (web server) — відповідає за роботу вебінтерфейсу системи; планувальник (scheduler) — планує виконання перевірок аномалій за розкладом; виконавець (worker) — виконує задачі, що приходять від планувальника та вебсервера. Схему взаємодії компонентів системи подано на рис. 1.

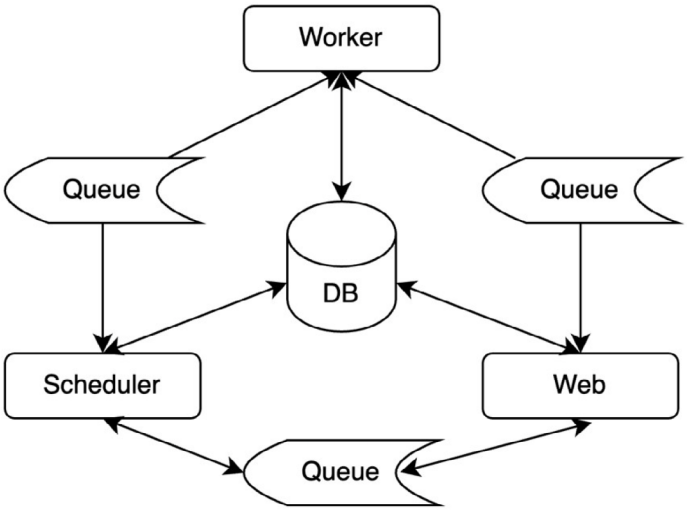


Рис. 1. Схема взаємодії компонентів системи

Компоненти взаємодіють один з одним за допомогою черги задач (task queue): вебсервер і планувальник надсилають у цю чергу задачі, які потрібно виконати, виконавець виконує їх і сигналізує про виконання результатом, який надсилає в цю саму чергу. Черга відповідає лише за комунікацію про завдання між компонентами системи; за зберігання даних відповідає БД, яка доступна всім компонентам.

Система складається з п'яти функціональних модулів: конфігурації виявлення, запуски перевірок, під'єднання до БД, контакти та повідомлення. Для того, щоб налаштувати систему, користувачу потрібно зробити такі кроки: додати під'єднання до бази даних, додати контакт для надсилання сповіщень, створити конфігурацію виявлення.

Головний елемент системи — менеджер налаштування конфігурацій виявлення. Ключовою особливістю системи є її інтерактивність: користувач має в інтерактивному режимі налаштовувати виявлення аномалій. Цю особливість було реалізовано в менеджері налаштування.

Налаштування складається з чотирьох кроків, які користувач проходить послідовно:

1. *Вибір під'єднання до БД*. Користувач обирає попередньо додане ним під'єднання до бази даних.
2. *Вибір даних для відстеження*. Менеджер сканує обрану базу даних і пропонує користувачу обрати потрібну таблицю та колонки, з яких система братиме дані. Система сповіщає користувача, якщо певну таблицю чи колонку неможливо обрати.
3. *Вибір і налаштування моделі*. На цьому етапі користувач має можливість вибрати модель, налаштувати часові інтервали і параметри моделі та одразу перевірити вибрану конфігурацію. Таким чином можна швидко налаштувати конфігурацію так, як це потрібно користувачу. Користувач бачить, що визначає модель і про що вона сповіщає, на графіку.
4. *Налаштування розкладу і сповіщень*. Користувач вибирає розклад запуску перевірки за допомогою Cron Expression (нотація для запису періодичності), та вибирає канал, в який потрібно буде надіслати сповіщення. Після проходження всіх цих кроків створюється конфігурація запуску визначення аномалій, яка зберігається в базі даних і додається до розкладу планувальника. При цьому перезавантажень планувальника для додавання чи видалення конфігурації не потребується.

Планувальник працює постійно і порівнює поточний час з розкладом виконання моделі, за збігу — планує виконання визначень, конфігурує виконання і надсилає в чергу повідомлення з задачею на виявлення. Ці повідомлення отримує виконавець та робить саму перевірку. Таким чином, за

допомогою черги задач процес оброблення даних, який є дуже вартісним із погляду ресурсів, відділений від інших функцій системи, і можна окремо керувати його ресурсами.

Виконання перевірки складається з чотирьох етапів: вибір даних з бази, оброблення даних моделлю, запис результатів до БД, надсилання сповіщення (за потреби). Якщо при виконанні перевірки відбулась програмна помилка, користувач буде сповіщений про це в обраному каналі комунікації, також це буде відображено на сторінці запуску. Якщо при надсиланні сповіщення виникла проблема, через яку це неможливо зробити, користувач побачить це тільки на сторінці запуску.

Для надсилання сповіщень про помилку при виконанні перевірки або про знайдені аномалії в системі підтримується два механізми — Slack і Gmail.

Для запуску перевірок розроблено таку систему часових інтервалів: інтервал спостереження, інтервал виявлення та інтервал сповіщення, які обраховують з урахуванням зсуву. Інтервал спостереження — це проміжок часу, який використовують для тренування моделі аномалій. Його початком є різниця моменту запуску аномалій, зсуву в часі та тривалість інтервалу спостереження, його кінцем — різниця моменту запуску аномалій і зсуву в часі. Для знайдених на цьому проміжку аномалій сповіщення не надсилаються. Інтервал виявлення — це проміжок часу, який використовують для оброблення моделлю аномалій. Його початком є різниця моменту запуску аномалій, зсуву в часі та тривалість інтервалу виявлення, його кінцем — різниця моменту запуску аномалій та зсуву в часі. Для знайдених у цьому проміжку аномалій сповіщення надсилаються (навіть з урахуванням того, що точки в цьому інтервалі також належать інтервалу спостереження). Зсув у часі — це різниця між часом виконання перевірки та кінцем інтервалів спостереження і виявлення. Аномалії, які лежать поза межами інтервалу виявлення, називають *аномаліями без сповіщень*, а ті, що лежать в межах інтервалу, — *аномаліями зі сповіщеннями*.

На сторінці конфігурації можна побачити статистику виконання перевірок для цієї конфігурації: відсоток виконаних перевірок зі сповіщеннями, без сповіщень і без аномалій. Також можна побачити статус виконання 30 останніх перевірок, вибрані налаштування перевірки, вибрані під'єднання до БД і канал сповіщення, і внизу сторінки доданий список останніх запусків для моделі.

На сторінці запуску (рис. 2) можна побачити статуси запуску, графік, побудований для даних, список аномалій, які викликали сповіщення, список аномалій без сповіщень, конфігурацію запуску перевірки, розміри часових інтервалів та інші параметри запуску.

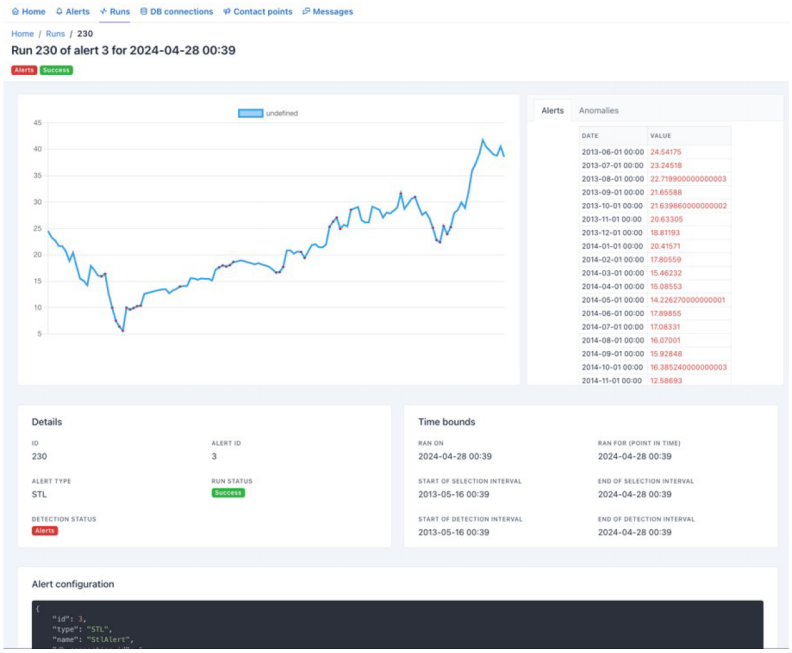


Рис. 2. Сторінка запуску

Модулі під'єднань до БД та контакти схожі між собою: обидва модулі містять сторінки для додавання і перегляду таких під'єднань. Кожне під'єднання користувачу обов'язково треба перевірити перед додаванням за допомогою кнопки «Перевірити», некоректні під'єднання неможливо додати в систему.

Важливо зазначити, що ключовим моментом реалізації цих модулів є архітектурно і інтерфейсно передбачена можливість додавання нових механізмів (провайдерів). У коді системи створені абстракції, для додавання нових провайдерів потрібно унаслідуватись від цих абстракцій у новому класі і перевизначити методи абстракції, додавши логіку, що стосується саме цієї системи.

При цьому інтерфейс користувача автоматично підлаштовується під кожного провайдера: для цього достатньо лише визначити нову модель даних, і системою буде створений інтерфейс, що матиме всі потрібні поля без потреби в додаткових налаштуваннях.

Подібний механізм створений і для моделей визначення аномалій: для додавання нових достатньо унаслідуватись від абстракцій і перевизначити логіку, що відмінна саме для цієї моделі, так само тут зроблене автоматичне генерування налаштувань. Підтримка різних моделей і провайдерів, легкість додавання нових імплементацій цих компонентів, і тим самим висока адаптивність є ключовою особливістю системи, що особливо важлива в індустрії даних, де використовують багато різних технологій, які часто змінюються. Це дає можливість використовувати систему в різних умовах.

Розроблена система повністю задовольняє поставлені раніше вимоги: легкість у налаштуванні забезпечується інтерфейсом користувача і інтерактивним процесом, гнучкість і легкість кастомізації — обраними технологіями та архітектурними абстракціями, надійність — розділенням компонентів через чергу задач, функціональні вимоги — розробленими складовими модулями. Система виконує поставлену задачу автоматизованого виявлення аномалій в бізнес-даних і відповідає сучасним стандартам у галузі даних.

5. Перспективи розвитку системи

Одним із обмежень на етапі формулювання вимог була мінімальна достатня функціональність продукту (MVP), оскільки реалізація додаткового функціоналу потребувала б більше часу і ресурсів. Саме тому система має великий потенціал до розвитку і може бути допрацьована в багатьох моментах.

Передусім, до системи може бути додано підтримку інших баз і сховищ даних (наприклад, AWS Athena, Amazon Redshift), підтримку API для отримання даних та інтеграцію з іншими системами комунікації (наприклад, Microsoft Teams). Це забезпечить можливість використання системи в інших умовах і компаніях, оскільки в галузі одночасно використовується велика кількість різних технологій. Велику практичну користь матиме і додавання інших моделей виявлення аномалій, наприклад, тих, що базуються на машинному навчанні.

Одним із потрібних доопрацювань є додавання вимірів для даних (data dimensions) і аналіз аномалій усередині цих вимірів, як індивідуально, так і в комбінації (наприклад, відстежуються продажі певного продукту для 10 різних платформ, у цьому випадку ці платформи будуть вимірами даних, і можна буде аналізувати аномалії для кожної з таких платформ).

Також до системи можна буде додати попереднє оброблення вхідних даних, наприклад фільтрацію, агрегацію, або написання власного SQL-запиту для вибору даних із бази. Для надсилання сповіщень можливо додати шаблони, за допомогою яких задаватиметься текст таких сповіщень, що підвищить їхню інформативність і цінність для певного користувача.

Функціональність статистики в системі може бути розширено для перегляду перевірок і їхніх результатів, профілювання даних на основі різних перевірок, або й таким чином, який би дав можливість визначати стан даних на основі всіх перевірок.

Висновки

У дослідженні проведено огляд і аналіз наукової літератури за темою, що дозволив систематизувати таксономію аномалій і технік їхнього визначення. Було наведено основні види і класифікації для аномалій і технік визначення аномалій, що дає уявлення про практичну цінність виявлення аномалій в обробленні даних.

Ми дали визначення поняття бізнес-даних, розглянули застосування цього підходу в цій галузі і перелічили проблеми, з якими сьогодні стикається галузь при аналізі даних. На основі проведеного дослідження підходу виявлення аномалій і поставленої проблеми було зроблене припущення про можливість використання автоматизованої ПС для вирішення знайдених проблем аналізу даних. Результатом дослідження стала побудова автоматизованої системи виявлення аномалій.

Для ефективної реалізації ПС проведено дослідження наявних інструментів, що близькі до тематики роботи. Було знайдено подібні рішення і проведений їхній аналіз із погляду відповідності їх потребам бізнесу і якості вирішення цих проблем. Аналіз показав, що наявні системи неактуальні, не підтримуються і не виконують задачі, потрібні для розв'язання проблеми. Таким чином було підтверджено потребу в системі, що вирішувала б розглянуту в роботі проблему.

На основі зробленого дослідження підходу виявлення аномалій, знайдених при аналізі рішень переваг і недоліків наявних систем, і власного досвіду авторів у галузі було розроблено вимоги, яким має відповідати розроблена система. На виконання цих вимог було обрано і спроєктовано архітектуру системи та проаналізовано і обрано технології для її програмної імплементації.

Практичним результатом роботи є автоматизована ПС виявлення аномалій. Система складається з кількох програмних модулів, що разом забезпечують виконання задач аналізу бізнес-даних. Ця ПС має високу адаптивність, є легкою до модифікації і зручною у використанні. Легкість налаштування забезпечується інтерактивним вебінтерфейсом, а надійність використання — розмежованою архітектурою з обмеженою відповідальністю компонентів.

Отже, в результаті роботи вдалося підтвердити поставлену раніше гіпотезу про можливість ефективного використання програмних систем вирішення проблеми аналітики бізнес-даних.

Список літератури

1. Arthur C. Tech giants may be huge, but nothing matches big data [Electronic resource] / C. Arthur // The Guardian. — Mode of access: https://www.theguardian.com/technology/2013/aug/23/tech-giants-data (date of access: 15.05.2025).
2. Brownlow J. Data-Driven Business Models: A Blueprint for Innovation [Electronic resource] / J. Brownlow et al. — 2015. — Mode of access: https://doi.org/10.13140/RG.2.1.2233.2320 (date of access: 15.05.2025).
3. Chandola V. Anomaly Detection: A Survey [Electronic resource] / V. Chandola, A. Banerjee, V. Kumar // ACM Computing Surveys. — 2009. — Vol. 41, no. 3. — Pp. 1–58. — Mode of access: https://doi.org/10.1145/1541880.1541882 (date of access: 15.05.2025).
4. Edgeworth F. Y. XLI. On discordant observations [Electronic resource] / F. Y. Edgeworth // The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. — 1887. — Vol. 23, no. 143. — Pp. 364–375. — Mode of access: https://doi.org/10.1080/14786448708628471 (date of access: 15.05.2025).
5. Features — FastAPI [Electronic resource] // FastAPI. — Mode of access: https://fastapi.tiangolo.com/features/ (date of access: 15.05.2025).
6. GitHub — chaos-genius/chaos_genius: ML powered analytics engine for outlier detection and root cause analysis [Electronic resource] // GitHub. — Mode of access : https://github.com/chaosgenius/chaos_genius (date of access: 15.05.2025).
7. GitHub — cuebook/CueObserve: Timeseries Anomaly detection and Root Cause Analysis on data in SQL data warehouses and databases [Electronic resource] // GitHub. — Mode of access: https://github.com/cuebook/CueObserve (date of access: 15.05.2025).
8. GitHub — elementary-data/elementary: The dbt-native data observability solution for data & analytics engineers. Monitor your data pipelines in minutes. Available as self-hosted or cloud service with premium features [Electronic resource] // GitHub. — Mode of access: https://github.com/elementary-data/elementary (date of access: 15.05.2025).
9. Grubbs F. E. Procedures for Detecting Outlying Observations in Samples / F. E. Grubbs // Technometrics. — 1969. — Vol. 11, no. 1. — Pp. 1–21.
10. Hawkins D. M. Identification of Outliers [Electronic resource] / D. M. Hawkins. — Dordrecht : Springer Netherlands, 1980. — Mode of access: https://doi.org/10.1007/978-94-015-3994-4 (date of access: 15.05.2025).
11. Ravishankar D. T. N. Application of Time Series Analysis for Better Decision Making in Business [Electronic resource] / D. T. N. Ravishankar // TechnoareteTransactions on Intelligent Data Mining and Knowledge Discovery. — 2022. — Vol. 2, no. 4. — Mode of access: https://doi.org/10.36647/ttidmkd/02.04.a005 (date of access: 15.05.2025).

References

Arthur, C. (2013). Tech giants may be huge, but nothing matches big data. *The Guardian*. https://www.theguardian.com/technology/2013/aug/23/tech-giants-data.

Brownlow, J., et al. (2015). *Data-driven business models: A blueprint for innovation*. https://doi.org/10.13140/RG.2.1.2233.2320.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41* (3), 1–58. https://doi.org/10.1145/1541880.1541882.

Edgeworth, F. Y. (1887). On discordant observations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *23* (143), 364–375. https://doi.org/10.1080/14786448708628471.

Features — FastAPI. *FastAPI*. https://fastapi.tiangolo.com/features.

GitHub — chaos-genius/chaos_genius: ML powered analytics engine for outlier detection and root cause analysis. *GitHub*. https://github.com/chaosgenius/chaos_genius.

GitHub — cuebook/CueObserve: Timeseries anomaly detection and root cause analysis on data in SQL data warehouses and databases. *GitHub*. https://github.com/cuebook/CueObserve.

GitHub — elementary-data/elementary: The dbt-native data observability solution for data & analytics engineers. Monitor your data pipelines in minutes. Available as self-hosted or cloud service with premium features. *GitHub*. https://github.com/elementary-data/elementary.

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics*, *11* (1), 1–21.

Hawkins, D. M. (1980). *Identification of outliers*. Springer Netherlands. https://doi.org/10.1007/978-94-015-3994-4.

Ravishankar, D. T. N. (2022). Application of time series analysis for better decision making in business. *Technoarete Transactions on Intelligent Data Mining and Knowledge Discovery*, *2* (4). https://doi.org/10.36647/ttidmkd/02.04.a005.

M. Postnikov, S. Gorokhovsky

AUTOMATED SYSTEM FOR DETECTION OF ANOMALIES IN BUSINESS DATA

The article describes the analysis of the anomaly-detecting process in business data, known software solutions, formulated requirements for the system and developed an automated software system for detecting anomalies. The developed system consists of software modules, has high adaptability, ease of modification, and ease of use. The system fully satisfies the previously set requirements: ease of configuration is provided by the user interface and interactive process, flexibility and ease of customization – by selected technologies and architectural abstractions, reliability – by separation of components through a queue of tasks, functional requirements – by developed component modules. The system performs the task of automated detection of anomalies in business data and meets modern standards in the field of data. The main types and classifications for anomalies and anomaly detection techniques were given, which give an idea of the practical value of anomaly detection in data processing. The analysis of the problem showed that existing systems are irrelevant, not supported, and do not perform the tasks required to solve the problem. Based on the research, the requirements that the system must meet were developed. To meet these requirements, the system architecture was selected and designed, and technologies for its software implementation were analyzed and selected. The practical result of the work is an automated software system for anomaly detection. The system consists of several software modules that together provide the performance of business data analysis tasks. The developed software system has high adaptability, ease of modification, and ease of use. Ease of configuration is ensured by an interactive web interface, and reliability of use is ensured by a delimited architecture with limited component liability. Thus, because of the work, it was possible to confirm the previously formulated hypothesis about the possibility of effective use of software systems to solve the problem of business data analysis. The system has great potential for development and can be developed in many directions, for example, support for other databases and data warehouses, as well as integration with other communication systems.

Keywords: anomaly detection, software system, system requirements, architecture, technologies, web application, FastAPI, Celery, REST API, Pandas.

Матеріал надійшов 25.06.2025

УДК 004.415.5
DOI: 10.18523/2617-3808.2025.8.167-173

Франків О. О.

АВТОМАТИЗОВАНЕ ВИЯВЛЕННЯ ВАД АРХІТЕКТУРИ ПРОГРАМНОГО МОДУЛЯ З ВИКОРИСТАННЯМ ГРАФОВОЇ МОДЕЛІ ВІЗУАЛІЗАЦІЇ

У цій статті описано новий підхід до автоматизованого виявлення вад проєктування програмного модуля з використанням моделі візуалізації архітектури у формі графа із застосуванням додаткових алгоритмів аналізу. Запропонований спосіб дозволяє виявити поширені вади та інформацію про них у легкому для сприйняття поданні з використанням лише вихідного коду програми. У статті розглянуто чотири ознаки наявності помилок при проєктуванні в контексті ООП, зокрема порушеної згуртованості, зв'язності і циклічних залежностей, а також способи їх виявлення та представлення у моделі архітектури. На реальних прикладах виконано автоматизоване виявлення вад із використанням розробленого інструменту для мови Swift з підтвердженням їх наявності у процесі детального аналізу кодової бази.

Ключові слова: програмування, програмний модуль, архітектура програмного забезпечення (ПЗ), оцінювання якості ПЗ, візуалізація архітектури, автоматизований статичний аналіз, шаблони проєктування, метрики, зв'язність програмного коду, згуртованість програмного коду.

Вступ

На сьогодні важливим викликом у галузі розроблення програмного забезпечення є підтримка належної якості продукту. Попри те, що наразі доступна велика кількість інструментів автоматичного виявлення помилок у програмному коді, значна частка з них концентрується на виявленні доволі простих помилок на рівні синтаксису та іноді семантики [6; 8]. Іншою проблемою наявних інструментів є обмежені можливості представлення інформації про програму, що робить процес обробленя результатів нетривіальним.

Раніше в статтях [1; 2] було описано підхід до моделювання архітектури програми у формі графа, в якій вузли відображають класи та їхні компоненти, а ребра — зв'язки між ними, такі як належність, використання, наслідування тощо. Естетично прийнятне розміщення цієї моделі в тривимірному просторі за силовим алгоритмом і з відображенням у доповненій реальності дає можливість спростити сприйняття складних залежностей на інтуїтивному рівні на основі стереометричних метафор. У процесі дослідження було розроблено програмний комплекс A.D.A.R. — Architecture Displayer in Augmented Reality, для автоматичної генерації та відображень моделі архітектури.

Попри те, що розроблений інструмент вдало будував візуалізацію архітектури, питання виявлення помилок проєктування залишалося поза контекстом. У цій статті ми розглянемо автоматизацію виявлення вад, що стала можливою завдяки значним удосконаленням візуальної моделі, а також автоматизації аналізу аномалій з використанням вбудованих статистичних прийомів. Важливою є також можливість індикації потенційно проблемних місць у структурі програми безпосередньо в моделі.

Вади проєктування в ООП і способи їх візуалізації

Парадигма об'єктно-орієнтованого програмування не є новою. На сьогодні існує велика кількість досліджень [4; 9], у яких запропоновано та перевірено різноманітні метрики, що дозволяють із певною точністю перевірити дотримання базових принципів ООП. Використання метрик дає змогу не лише отримати конкретне значення рівня дотримання принципів, а й розробляти ефективні рішення для їх обчислення. З іншого боку, числове представлення складних концепцій не є інтуїтивним, знач-

но залежить від технологій, а ймовірний некоректний результат може ще більше ускладнити детальний аналіз [10].

Розглянемо деякі поширені помилки проєктування, що виникають унаслідок порушення принципів ООП, та можливі підходи до їх інтуїтивної візуалізації.

Концентрація логіки програми в межах одного класу є небажаною з огляду на утворення сильної зв'язності і, як наслідок, надлишкових залежностей, що своєю чергою призводить до підвищення ризику аномалій при зміні коду та крихкості логіки загалом. Такі метрики, як CBO (Coupling Between Object) — зв'язність між об'єктами і LOCM (Lack of Cohesion in Methods) — відсутність згуртованості методів, можна просто обчислити, провівши статичний аналіз вихідного коду, та виявити ознаки вад проєктування. Однак у класичному випадку вони обчислюються для цілого проєкту і радше свідчать про якість коду в цілому. Очевидно, їх можна обчислювати і для окремих класів, що значно звужує область пошуку проблемних місць, але в такому разі слабким місцем такого підходу є потреба в гнучкому визначенні порогу критичності для кожної з метрик, що не може бути універсальним, адже залежить від специфіки конкретної програми.

На противагу такому підходу розглянемо велику зв'язність і низьку згуртованість у контексті графової моделі архітектури. Вочевидь ознакою великої зв'язності в області конкретного компонента (не обов'язково класу, а й властивості чи метода) є велика кількість ребер (зв'язків), що сполучають цей компонент з іншими. Отже вузли, що мають аномально великий степінь у графовій моделі, потенційно є центрами областей великої зв'язності.

З іншого боку, низька згуртованість також є проблемою, що може свідчити про невдалу декомпозицію логіки. З урахуванням того, що граф архітектури є зваженим, а ваги відображають інтенсивність зв'язку між компонентами, аномально мала вага зв'язку структурної частини (властивості чи метода) з класом порівняно з іншими свідчить про низьку згуртованість і, як наслідок, ставить під сумнів правильність структурної належності. В екстремальних випадках порушення згуртованості призводить до таких ситуацій, як «заздрісні методи» (feature envy) [5].

Слід зауважити, що, на противагу числовим представленням метрик, у графовій моделі зв'язність і згуртованість легко відображаються відстанями між компонентами, а також кольорами для додаткової індикації (рис. 1).

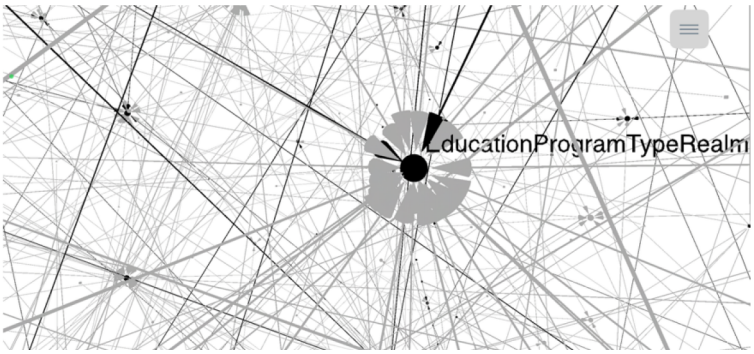


Рис. 1. Приклад вузла з великим степенем

Ще однією проблемою при проєктуванні є виникнення циклічних залежностей між компонентами. У деяких випадках такий підхід може бути виправданим, але загалом є небажаним, адже в такому разі відсутність циклу на етапі виконання часто досягається шляхом роботи в багатопотоковому середовищі, а отже є вразливим до можливого «змагального стану» (race condition).

Очевидно, що граф є вдалою структурою для пошуку циклічних залежностей, що також дозволяє використовувати ефективні підходи до пошуку циклів із погляду продуктивності. Окрім того, візуальна індикація ребер, що формують цикл, дозволяє легше сприйняти циклічну залежність незалежно від кількості ланок у ланцюзі.

Ураховуючи те, що для розміщення графу у тривимірному просторі для забезпечення високого рівня естетичності застосовується адаптація силового алгоритму, серйозні порушення в контексті естетичності подання також можуть свідчити про надмірну зв'язність компонентів. Одним із базових критеріїв естетичності, які за задумом забезпечує алгоритм, є неперетин ребер, або, менш строго, мінімізація перетинів ребер. Відповідно індикація ребер, що перетинаються чи близькі до перетину з певною похибкою, дозволяє звернути увагу на потенційно проблемну ділянку.

Тут варто зауважити, що силовий алгоритм не є стабільним, а якість відображення графа у просторі сильно залежить і від додаткових налаштувань алгоритму, зокрема критеріїв зупинки. З огляду на це порушення естетичності в частині перетину ребер є дотичною ознакою і корелює як зі структурними особливостями графа, так і з особливостями роботи алгоритму в конкретному частковому випадку.

Виявлення аномалій архітектури програми у графовій моделі

Виявлення вад проєктування за умови відсутності метаданих проєкту, таких як проєктна документація, не може передбачати перевірку шляхом встановлення відповідності еталону. Серед альтернативних варіантів можна визначити перевірку за метриками з порівнянням із загальноприйнятими рекомендованими значеннями якісних архітектур і пошук аномалій.

Сучасні програмні рішення можуть значно відрізнятись залежно від доменної області, зокрема у складності та об'ємі. Окрім того, на сьогодні широко використовують спеціалізації та видозміни ООП, як, до прикладу, ПОП (протокольно-орієнтоване програмування) чи РП (реактивне програмування). Через це еталонні значення метрик, встановлені до появи деяких сучасних технологій або ж на конкретного виду програмах, можуть виявитись нерелевантними. В такому разі користувач системи повинен самостійно налаштувати критерії, що своєю чергою негативно впливає на автоматизацію як таку, адже потребує додаткових ресурсів, зокрема для глибокого аналізу особливостей проєкту. Варто також зазначити, що точково тонко налаштовані еталони стають досить умовним критерієм якості.

Саме з огляду на ці проблеми при розробленні нашого рішення ми сконцентрувались на гнучкому виявленні аномалій як ознак імовірних порушень загальної архітектури програми. Варто зауважити, що для сценаріїв, у яких загальна якість архітектури низька в принципі, ймовірно, більш оптимальним рішенням може бути гібридний підхід із використанням узагальнених еталонних значень для мінімального порогу критерію у поєднанні з пошуком аномалій.

Для пошуку аномалій у нашому інструменті реалізовано окремий модуль AnomalyDetector. Він містить однойменний клас, екземпляри якого можуть аналізувати наявну готову графову модель і виявляти в ній ознаки потенційних порушень структури.

AnomalyDetector паралельно виконує пошук таких аномалій:

1) вузли з аномально великим степенем;
2) ребра, що позначають належність до класу, але мають аномально малу вагу;
3) цикли;
4) надмірне зближення ребер, у тому числі перетин.

На підставі цих даних програма-переглядач додає візуальну індикацію для потенційно проблемних елементів.

Вузли, що мають аномально великий степінь, очевидно, є областями великої зв'язності коду. Важливою проблемою тут є визначення, які значення степеня є аномальними. Для забезпечення гнучкості одночасно з низьким впливом на швидкодію було вирішено використати простий статистичний інструментарій. Для кожного виду вузлів у графі (класи, властивості, методи) окремо обраховується математичне сподівання, яке в цьому випадку збігається з середнім арифметичним, та дисперсія.

Поняття аномального значення вводиться класично відповідно до формули:

$$anomaly(x) = \begin{cases} true, if\ |x - \text{m}| > k * \text{s} \\ flase, otherwise \end{cases},$$

де x — значення степеню певного вузла, μ — математичне сподівання, k — пороговий коефіцієнт (за замовчуванням $k = 2$), σ — дисперсія.

Для виявлення ознак низької згуртованості між компонентами класу було значно вдосконалено частину проєкту A.D.A.R. — аналізатор. В оригінальній версії при побудові графа вага ребра, що відображає зв'язок «є методом» (класу) або «є властивістю» (класу), завжди безумовно становила 1 при області значень ваг [0;1]. Таким чином модель ніколи не відображала реальної згуртованості, а лише максимальне значення.

Для обчислення реального рівня згуртованості компонентів було взято спрощений варіант відповідних метрик, що обчислювався для кожного компонента окремо. Окрім цього, це значення обчислюється по-різному для методів і властивостей.

Слід зауважити, що поняття властивостей у сучасних мовах програмування може бути дуже відносним. До прикладу, у Swift існують обчислювані властивості (computed properties), що окрім синтаксису мало відрізняються від методів без параметрів, або ж властивості, що ініціалізуються шляхом виконання анонімної функції (closure). У цьому дослідженні ми опустили ці особливості і сконцентрувались на роботі з більш класичними поняттями методів і властивостей.

Для обчислення ваги ребра, що поєднує властивість із класом, якому вона належить, з огляду на те, що властивість не використовує метод, але метод може звертатись до неї у своєму тілі, ми використовували формулу:

$$w(p, c) = \sigma(12 * \frac{m_{p,c}}{Mc} - 6),$$

де w (p, c) — вага ребра між вузлом властивості p та класу c; $m_{p,c}$ — кількість методів класу c, що використовують властивість p; M_c — загальна кількість методів класу c.

Для обчислення ваги ребра, що поєднує метод з класом, якому він належить, ми використовували таку формулу:

$$w(\mu, c) = (\frac{m_{\mu,c}+p_{\mu,c}}{M_c+P_c}),$$

де w (μ, c) — вага ребра між вузлом методу μ та класу c; $m_{\mu,c}$ — кількість методів класу c, що використовує метод μ; M_c — загальна кількість методів класу c; $p_{\mu,c}$ — кількість властивостей класу c, що використовує метод μ; P_c — загальна кількість властивостей класу c.

Для визначення того, чи вага ребра є аномально малою, ми використовували аналогічний підхід, як для визначення аномального значення степеня вузла.

Задача виявлення циклів у цьому контексті є тривіальною. AnomalyDetector просто долучає до звіту інформацію про знайдені цикли.

У контексті пошуку порушення критеріїв естетичності, зокрема значного наближення чи перетину ребер, важливо підкреслити, що у реальних випадках візуалізації точний перетин відрізків малоймовірний. Тому ми зробили акцент на пошуку наближень відрізків, адже перетин є частковим екстремальним випадком. Відрізки вважаються аномально наближеними, якщо найменша відстань між ними є в межах деякої невеликої відстані, що у нашому випадку становить 0,0001 умовної одиниці розміру за загального розміру куба, в який вписано граф 500 * 500 * 500 умовних одиниць.

Виявлення та локалізація проблемних частин архітектури

Із метою перевірки здатності запропонованого підходу виявляти та локалізовувати потенційно проблемні місця в архітектурі програмного модуля було проведено експерименти на реальних проєктах iOS додатків на платформі GitHub, написаних мовою Swift.

Для ілюстрації розглянемо невеликий застосунок — гру 2048 [3]. В першу чергу було проведено автоматизовану побудову архітектури програми з використанням нашого інструменту з додатковим автоматизованим аналізом аномалій. Отриману візуальну модель можна побачити на рис. 2. Чорним кольором позначено потенційно проблемні елементи, а сірим — всі інші.

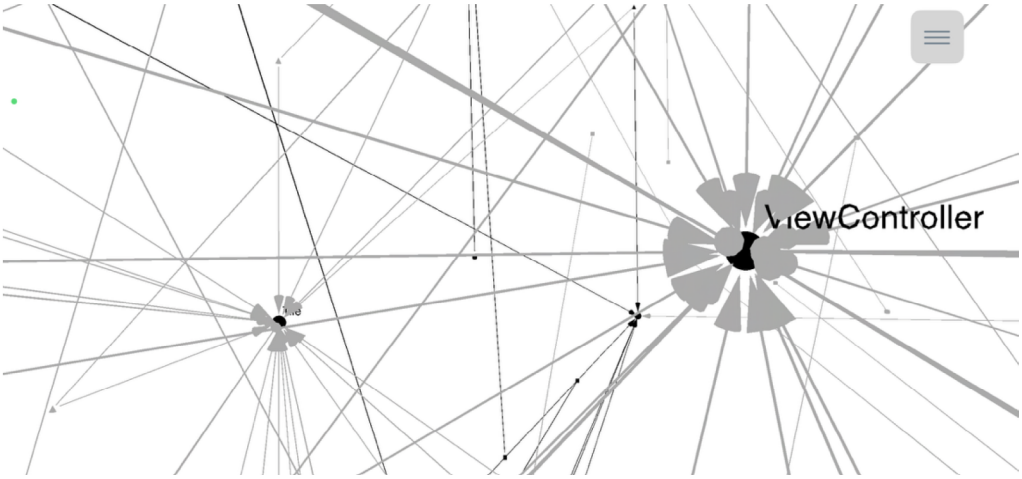


Рис. 2. Частина візуалізації проєкту 2048

Варто зауважити, що циклічних залежностей у цьому випадку не було виявлено, що зумовлено тим, що, по-перше, проєкт є досить простим, а по-друге, наявність таких залежностей є досить рідкісним явищем.

Натомість виявлено п'ять вузлів з аномально великим степенем, чотири властивості з аномально низькою згуртованістю і 10 ребер, що розміщені досить близько в умовах цієї візуалізації.

```
class ViewController: UIViewController {
      var manager: GameLogicManager!
      var score: Score!
      var highScore: HighScore!
      var renderer: GameBoardRenderer!
      var restartButton: RestartButton!
      // …
}
```

Лістинг 1. Властивості класу ViewController

Одним із вузлів із найбільшим степенем є вузол, що позначає клас ViewController. За детальнішого аналізу (лістинг 1) добре видно, що цей клас справді поєднує в собі всі частини функціоналу застосунку, що, як наслідок, робить цю частину коду крихкою.

Іншим прикладом вузла з великим степенем є вузол, що позначає метод save(dimension: Int, tiles: [Tile]). Як видно з лістингу 2, в тілі цього методу справді є ознаки порушення інкапсуляції, зокрема в частині створення екземпляру TileModel.

```
func save(dimension: Int, tiles: [Tile]) {
      deleteAllTiles()
      for tile in tiles {
            let tileModel = TileModel(context: ResistanceService.context)
            tileModel.positionX = Int16(tile.position.x)
            tileModel.positionY = Int16(tile.position.y)
            tileModel.tileValue = Int32(tile.value ?? 0)
            ResistanceService.saveContext()
      }
}
```

Лістинг 2. Порушення інкапсуляції в методі save(dimension: Int, tiles: [Tile])

Усі чотири виявлені властивості з низьким рівнем згуртованості зі своїм класом належать до класу Tile. Цей клас містить лише декларації цих властивостей без методів. Натомість клас GameLogicManager містить метод isGameOver() -> Bool (Лістинг 3), що порушує інкапсуляцію властивостей класу Tile, працюючи безпосередньо з ними і, як наслідок, ці властивості мають вищу згуртованість із ним, аніж із власним класом.

```
private func isGameOver() -> Bool {
      if tiles.filter({$0.value == nil}).count != 0 {
            return false
      }
      for tile in tiles {
            let v = tile.value!
            let neighbours = [tile.upTile, tile.rightTile, tile.
bottomTile, tile.leftTile]
                              .filter { $0?.value == v }
            if neighbours.count != 0 {
                  return false
            }
      }
      return true
}
```

Лістинг 3. Порушення інкапсуляції в методі isGameOver() -> Bool

Що стосується виявлених наближень ребер графу, то вони за детального аналізу здебільшого є ознакою загальної великої зв'язності проєкту.

Отже, з використанням автоматизованої побудови візуалізації архітектури програми з виявленням вад проєктування можна інтуїтивно легко виявити проблемні області коду програми.

Висновки

У цій статті розглянуто вдосконалення інструменту для автоматизованої побудови візуальної графової моделі архітектури програмного модуля з акцентом на виявлення та локалізацію потенційних вад проєктування. Пошук порушення зв'язності та згуртованості між компонентами програми дозволяє скоротити процес пошуку областей, вразливих до змін, і сконцентрувати ресурс для детального аналізу відповідно.

Новий модуль AnomalyDetector, працюючи лише з моделлю архітектури, дозволяє швидко та гнучко виявляти ознаки проблемного коду завдяки використанню простого статистичного інструментарію.

На прикладі реального застосунку проілюстровано відповідність між виявленими аномаліями в моделі та реальними проблемами безпосередньо в коді, що демонструє результативність, ефективність та практичну цінність запропонованого рішення.

Список літератури

1. Франків О. О. Використання доповненої реальності для візуалізації архітектур програмних модулів [Електронний ресурс] / О. О. Франків // Наукові записки НаУКМА. — 2023. — Т. 5. — С. 26–30. — Режим доступу: https://doi.org/10.18523/2617-3808.2022.5.26-30.
2. Франків О. О. Автоматизована візуалізація компонентів архітектури програми для мови Swift [Електронний ресурс] / О. О. Франків, М. М. Глибовець // Кібернетика та системний аналіз. — 2024. — Т. 60 (6). — С. 190–198. — Режим доступу: https://doi.org/10.1007/s10559-024-00737-9.
3. Bobrov A. 2048 [Electronic resource] / A. Bobrov. — GitHub, 2020. — Mode of access: https://github.com/aibobrov/2048.
4. Chidamber S. R. A metrics suite for object oriented design / S. R. Chidamber, C. F. Kemerer // IEEE Transactions on Software Engineering. — 1994. — Vol. 20, iss. 6. — Pp. 476–493. — Mode of access: https://doi.org/10.1109/32.295895.
5. Fowler M. Refactoring: Improving the design of existing code / M. Fowler, K. Beck, J. Brant, W. Opdyke, D. Roberts. — Boston, MA: Addison Wesley Professional, 1999. — 464 p.
6. Lenarduzzi V. A Critical Comparison on Six Static Analysis Tools: Detection Agreement and Precision [Electronic resource] / V. Lenarduzzi, F. Pecorelli, N. Saarimäki, S. Lujan, F. Palomba // SSRN Electronic Journal. — 2022. — Mode of access: https://doi.org/10.2139/ssrn.4044439.
7. Li R. Understanding software architecture erosion: A systematic mapping study [Electronic resource] / R. Li, P. Liang, M. Soliman, P. Avgeriou // Journal of Software: Evolution and Process. — 2022. — Vol. 34 (3). — e2423. — Mode of access: https://doi.org/10.1002/smr.2423.
8. Schneider S. Comparison of static analysis architecture recovery tools for microservice applications [Electronic resource] / S. Schneider, A. Bakhtin, X. Li, J. Soldani, A. Brogi, T. Černý, R. Scandariato // Empirical Software Engineering. — 2025. — Vol. 30. — P. 128. — Mode of access: https://doi.org/10.1007/s10664-025-10686-2.
9. Tang M.-H. An empirical study on object-oriented metrics [Electronic resource] / M.-H. Tang, M.-H. Kao, M.-H. Chen // Proceedings of the Sixth International Software Metrics Symposium. — 1999. — Pp. 242–249. — Mode of access: https://doi.org/10.1109/metric.1999.809745.
10. Voas J. What happened to software metrics? [Electronic resource] / J. Voas, R. Kuhn // Computer. — 2017. — Vol. 50, no. 5. — Pp. 88–98. — Mode of access: https://doi.org/10.1109/MC.2017.144.

References

Bobrov, A. (2020). *2048* [Source code]. GitHub. https://github.com/aibobrov/2048.

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, *20* (6), 476–493. https://doi.org/10.1109/32.295895.

Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). *Refactoring: Improving the design of existing code*. Addison Wesley Professional.

Frankiv, O. O. (2023). Application of augmented reality for software module architecture visualization. *Scientific Notes of NaUKMA*, *5*, 26–30. https://doi.org/10.18523/2617-3808.2022.5.26-30 [in Ukrainian].

Frankiv, O. O., & Hlybovets, M. M. (2024). Automated visualization of software architecture components for Swift. *Cybernetics and Systems Analysis*, *60* (6), 190–198. https://doi.org/10.1007/s10559-024-00737-9 [in Ukrainian].

Lenarduzzi, V., Pecorelli, F., Saarimäki, N., Lujan, S., & Palomba, F. (2022). A critical comparison on six static analysis tools: Detection agreement and precision. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4044439.

Li, R., Liang, P., Soliman, M., & Avgeriou, P. (2022). Understanding software architecture erosion: A systematic mapping study. *Journal of Software: Evolution and Process*, *34* (3), e2423. https://doi.org/10.1002/smr.2423.

Schneider, S., Bakhtin, A., Li, X., Soldani, J., Brogi, A., Černý, T., & Scandariato, R. (2025). Comparison of static analysis architecture recovery tools for microservice applications. *Empirical Software Engineering*, *30*, 128. https://doi.org/10.1007/s10664-025-10686-2.

Tang, M.-H., Kao, M.-H., & Chen, M.-H. (1999). An empirical study on object-oriented metrics. In *Proceedings of the Sixth International Software Metrics Symposium* (pp. 242–249). https://doi.org/10.1109/metric.1999.809745.

Voas, J., & Kuhn, R. (2017). What happened to software metrics? *Computer*, *50* (5), 88–98. https://doi.org/10.1109/MC.2017.144.

O. Frankiv

AUTOMATED DETECTION OF SOFTWARE MODULE ARCHITECTURE FLAWS USING A GRAPH-BASED VISUALIZATION MODEL

This paper introduces a novel approach to the automated detection of software module design flaws through a graph-based architectural visualization model, enhanced with specialized analysis algorithms. The method is designed to identify common structural and logical issues in codebases, relying solely on static information derived from the source code. Unlike traditional approaches that require extensive runtime instrumentation or manual inspection, this solution produces a visual and interpretable model of the software architecture, highlighting problematic areas automatically. The visualization is not merely illustrative — it serves as an analytical layer capable of surfacing critical design issues in a comprehensible and compact form.

The proposed method extends the capabilities of architectural visualization by integrating algorithms that automatically detect signs of architectural degradation. This includes the ability to pinpoint segments of the code that demonstrate structural inefficiencies, making the analysis both accessible and actionable. As a result, developers and architects are provided with immediate feedback about design quality, which can be used to guide refactoring efforts and architectural improvements.

The paper focuses on four specific indicators of poor design in the context of object-oriented programming: weakened cohesion, excessive coupling, the presence of cyclic dependencies, and architectural violations in module boundaries. Techniques for detecting and representing these patterns in the architectural model are discussed in detail. An anomaly detection-based approach is proposed to ensure the solution remains both performant and adaptable, allowing it to be applied across a diverse set of projects varying in size, domain, and technological stack.

The approach has been validated on real-world Swift codebases using a custom-built analysis tool. The tool successfully identified structural flaws, which were confirmed through manual inspection and deeper analysis of the source code, demonstrating its effectiveness and practical utility.

Keywords: programming, software module, software architecture, software quality assessment, architecture visualization, automated static analysis, design patterns, metrics, code coupling, code cohesion.

Матеріал надійшов 07.07.2025

УДК 004.056:37.018.43
DOI: 10.18523/2617-3808.2025.8.174-179

Глибовець А. М., Бабич Т. А.

РЕАЛІЗАЦІЯ ПРАКТИКО-ОРІЄНТОВАНОЇ СИСТЕМИ КІБЕРПОЛІГОНУ З ІНДИВІДУАЛІЗОВАНИМ ЗАПУСКОМ СЕРЕДОВИЩ

У статті описано архітектуру й технічну реалізацію контейнерно-орієнтованого кіберполігону, інтегрованого з LMS Moodle через LTI. Показано переваги підходу щодо швидкодії, масштабованості й безпеки. Проведено порівняння з альтернативними рішеннями та визначено подальші напрями розвитку платформи.

Ключові слова: кіберполігон, кібербезпека, практико-орієнтоване навчання, Capture the Flag (CTF), контейнеризація, Docker, LTI інтеграція, Moodle, інтерактивне навчання, імітаційні сценарії.

Вступ

Зростання кількості та складності кібератак у сучасному світі вимагає підвищення ефективності освіти у сфері кібербезпеки. Традиційні освітні підходи, що зосереджені на теорії, сертифікації та відповідності вимогам, не забезпечують достатнього практичного досвіду для протидії реальним загрозам. Як наслідок, виникає потреба у впровадженні практико-орієнтованих методів навчання, які надають слухачам можливість отримати практичні навички у безпечному, контрольованому середовищі. Саме таким інструментом є кіберполігони (англ. cyber ranges) — інтерактивні віртуальні середовища, що імітують мережі, системи та додатки і дозволяють відпрацьовувати навички кіберзахисту на реальних сценаріях. Кіберполігони сприяють формуванню практичних умінь, підвищують обізнаність і готовність фахівців шляхом моделювання реальних інцидентів без ризику для продуктивних систем.

Однак розробка та підтримка власного кіберполігону є ресурсномісткими: потрібні значні фінансові витрати, обчислювальні ресурси й час на розгортання складної інфраструктури. Традиційні лабораторні середовища на базі фізичних чи віртуальних машин також мають обмеження: вони важко масштабуються на велику кількість студентів, потребують ручного налаштування. У стандартних навчальних програмах часто бракує реалізму та інтерактивності: студенти вивчають теорію і здають тести, але майже не отримують практичного досвіду роботи з актуальними атаками чи інцидентами. Це призводить до розриву між теоретичними знаннями та реальними навичками, необхідними для кібербезпеки [3].

Відповіддю на ці виклики стало впровадження кіберполігонів як навчального інструменту. Вони забезпечують реалістичне навчання через імітацію кібератак та захисних заходів у безпечному середовищі. Дослідження показують, що практичні вправи суттєво підвищують мотивацію і рівень підготовки учасників [3; 4].

У цьому контексті актуальним є розроблення практико-орієнтованих платформ, що легко інтегруються в освітній процес. У статті представлено реалізацію такої системи кіберполігону, побудованої за контейнерним підходом з індивідуалізованим запуском середовищ для кожного користувача. Платформа інтегрована в навчальну LMS (Moodle) через стандарт LTI, що забезпечує єдиний освітній простір для теорії та практики. Нижче розглянуто сучасні підходи до створення кіберполігонів, детально описано архітектуру нашого рішення, проаналізовано технічні виклики і шляхи їх вирішення, а також обговорено переваги і обмеження підходу [3; 1; 6].

Огляд підходів до побудови кіберполігонів

Існує декілька концептуальних підходів до організації навчальних кіберполігонів. Розглянемо ключові з них:

- Полігони з теоретичними матеріалами та симуляційними завданнями. У таких системах навчальна платформа поєднує текстові або мультимедійні матеріали з кібербезпеки і інтерактивні симуляції або вправи. Студенти спочатку вивчають теорію, після чого виконують практичні завдання в середовищі, що моделює певні ситуації (наприклад, пошук вразливостей, аналіз трафіку тощо). Подібна інтеграція теорії з практикою покликана забезпечити міцніший зв'язок між знаннями і навичками. Дослідження показують, що інтерактивне навчання, яке підкріплює теоретичний матеріал практичними вправами, підвищує засвоєння і зацікавленість студентів. Прикладом можуть бути онлайн-лабораторії, де кожен розділ курсу супроводжується вбудованою практичною вправою або симулятором.
- Системи з інтерактивними підказками (віртуальні асистенти): більш просунуті кіберполігони впроваджують віртуальних наставників — інтерактивні підказки чи боти, що допомагають користувачам у процесі виконання завдань. Такий підхід часто використовує елементи штучного інтелекту або сценарних підказок, які реагують на дії учня. Змагання Capture The Flag (CTF) традиційно застосовують систему підказок із вирахуванням балів, що, як показує досвід, мотивує учасників знаходити рішення самостійно, але може стримувати тих, хто потребує допомоги. Інтерактивні асистенти всередині навчального середовища здатні адаптивно скеровувати студента, пояснювати помилки і пропонувати додаткові ресурси, що підвищує ефективність навчання. Подібні можливості починають з'являтися у сучасних платформах кібербезпеки, поєднуючи практику з елементами інтелектуальної підтримки [3; 4].

Історично перші кіберполігони будували з використанням віртуалізації на рівні гіпервізора. Кожному студенту надається одна або кілька віртуальних машин (ВМ), що містять необхідні інструменти або вразливі системи. Перевагою цього підходу є повна ізоляція середовища для кожного користувача та можливість емулювати різні операційні системи (Linux, Windows тощо). ВМ забезпечують високий рівень реалізму: студент працює ніби з окремим фізичним комп'ютером. Проте недоліками є значні витрати ресурсів і часу на розгортання таких середовищ. Кожна ВМ містить повноцінне ядро ОС та усі служби, тому десятки паралельних ВМ швидко споживають ресурси сервера. Час запуску ВМ також довший порівняно з контейнерами. За даними досліджень, Docker-контейнери демонструють принаймні на 50 % швидший старт і обслуговування запитів, ніж еквівалентні віртуальні машини. Контейнеризація загалом забезпечує кращу ефективність виконання сценаріїв порівняно з класичними ВМ за рахунок меншого накладного навантаження. Отже, ВМ-полігони добре підходять для складних сценаріїв, що потребують повної емуляції систем, але їх масштабування на велику кількість слухачів обмежене та потребує суттєвих потужностей або хмарних ресурсів [3; 1].

Сучасним трендом у побудові кіберполігонів є використання контейнеризації — ізоляції середовищ на рівні операційної системи. Контейнери (наприклад, Docker) дають можливість запускати декілька ізольованих середовищ на спільному ядрі ОС, що значно знижує витрати ресурсів на кожне середовище. На відміну від ВМ, контейнер не містить власного ядра, а використовує ядро хостової системи, завдяки чому він «легший» і запускається майже миттєво. Це дає змогу на одному фізичному сервері розгорнути набагато більше одночасних ізольованих середовищ без втрати продуктивності. Контейнерні образи легко переносити між різними машинами, що спрощує розгортання навчальних сценаріїв на різних майданчиках. Таким чином, контейнерний підхід здатний забезпечити як високу щільність розміщення середовищ, так і швидке їх оновлення чи скидання до початкового стану. Відомим прикладом є фреймворк Labtainers, що пропонує набір навчальних лабораторій із кібербезпеки на базі Docker-контейнерів; він спрощує підготовку середовищ для викладача, пакуючи всі налаштування в образ, однак не підтримує деякі розширені можливості (командну роботу, аналітику прогресу тощо) [3; 1; 5].

Рівень автоматизації і масштабованості платформ

Кіберполігони також класифікують за ступенем автоматизації розгортання середовищ і управління ними. Прості реалізації можуть потребувати ручного налаштування кожного середовища: наприклад, інструктор заздалегідь готує образ ВМ чи контейнера і видає його студентам. Такі рішення не масштабуються на великі аудиторії, оскільки адміністрування десятків / сотень копій середовища стає трудомістким і схильним до помилок. Натомість сучасні платформи впроваджують оркестрацію — автоматизоване управління життєвим циклом середовищ: їх розгортання, налаштування, запуск, зупинку та очищення. Дослідження показують, що рішення, побудовані з опорою на інфра-

структурні платформи та оркестрацію, забезпечують кращу ізоляцію і надійність середовищ, тоді як полігони на основі ad-hoc інфраструктури без оркестрації можуть мати слабші механізми ізоляції і потребують більше ручного втручання. Отже, за критерієм автоматизації платформи варіюються від локальних скриптів розгортання окремих лабораторій до комплексних хмарних сервісів, здатних автоматично масштабуватися під навантаження. При виборі підходу важливо врахувати баланс між контролем над середовищем і витратами часу на його підтримку: високоавтоматизовані рішення знижують навантаження на адміністраторів і мінімізують людський фактор, проте можуть вимагати складнішого початкового налаштування [3; 1].

Опис реалізованої системи

Розроблена нами система кіберполігону призначена для інтеграції в освітній процес і забезпечення кожному студенту ізольованого практичного середовища. Архітектурно рішення складається з вебзастосунку (серверної частини), що реалізований на мові PHP із використанням фреймворку Yii, та системи контейнеризації Docker для запуску навчальних середовищ. Вебзастосунок розгорнуто окремо від LMS, але інтегрований у Moodle за допомогою LTI (Learning Tools Interoperability). LTI — це стандарт IMS Global, який дозволяє підключати зовнішні навчальні інструменти до платформ на кшталт Moodle, Canvas тощо. У нашому випадку Moodle виступає як LTI-споживач (Platform), а вебзастосунок кіберполігону — як LTI-інструмент (Tool). Така інтеграція забезпечує єдину автентифікацію: користувачі заходять до Moodle під своїми обліковими записами і запускають кіберполігон через активність зовнішнього інструменту, без потреби окремого введення логіна та пароля на стороні полігону. Moodle передає нашому застосунку необхідні дані про користувача та курс (через підписаний LTI-запит), завдяки чому відбувається SSO (Single Sign-On) — студент одразу потрапляє у свій особистий кібер-лаб без додаткової авторизації. Перевагою такого підходу є безшовний користувацький досвід: із інтерфейсу LMS студент одним кліком відкриває практичну лабораторію, а викладач може за необхідності отримувати зворотні дані (результати, оцінки) назад у Moodle через сервіси LTI Advantage, зокрема Assignment and Grade Services [3; 1; 6].

Після переходу користувача до завдання кіберполігону система автоматично розгортає для нього індивідуальне середовище у вигляді Docker-контейнера. Для кожного завдання підготовлено шаблонний контейнерний образ, що містить усе необхідне програмне забезпечення, вразливі вебдодатки, скрипти тощо — залежно від сценарію. Вебзастосунок через Docker API запускає новий контейнер із цього образу, ізолюючи його мережеві та файлові ресурси. В результаті студент отримує власний екземпляр середовища для виконання завдання. Індивідуалізація реалізована через запуск кожного контейнера з унікальними параметрами, згенерованими для конкретного користувача. Зокрема, реалізовано механізм передання у контейнер секретного прапорця (flag) та інших змінних оточення. Прапорець — це спеціальний рядок символів, який слугує індикатором успішного виконання завдання (типово використовується у CTF-завданнях). У нашій системі для кожного запуску генерується випадковий унікальний flag, який передається в контейнер як змінна середовища (FLAG=<значення>). Всередині контейнера цей прапорець може бути автоматично записаний у певне місце (наприклад, файл у системі або рядок у банері програми) чи використовуватися сервісом перевірки. Студент, успішно виконавши завдання (знайшовши вразливість або експлуатувавши її), повинен виявити цей секретний рядок і подати його як доказ виконання. Модель прапорців гарантує, що кожен студент працює з унікальним розв'язком: навіть якщо двоє користувачів спробують обмінятися відповідями, у кожного прапорець інший, отже чужий результат не зарахується. Такий підхід широко використовується в навчальних CTF-іграх і довів ефективність у запобіганні шахрайству, дозволяючи при цьому співпрацю (студенти можуть обмінюватися ідеями, але не готовими відповідями) [3; 1; 6].

Технічні виклики та рішення

У типовому навчальному сценарії десятки студентів можуть запускати свої контейнерні середовища одночасно (наприклад, під час лабораторної роботи або заліку). Це висуває підвищені вимоги до серверної інфраструктури. Контейнери значно ефективніше використовують ресурси порівняно з ВМ, тому наш сервер здатен підтримувати в рази більше паралельних середовищ без деградації продуктивності. Проте апаратні обмеження все одно існують — насамперед за оперативною пам'ят-

тю та процесорними ядрами. Щоб запобігти перевантаженню, у системі встановлено ліміти на ресурси для кожного контейнера (через cgroups Docker конфігурується максимальний обсяг RAM і доля CPU).

Наприклад, типовому вебзастосунку може виділятися не більше ніж 512 МБ пам'яті і 20 % від одного процесорного ядра. Це унеможливлює ситуацію, коли один некоректно працюючий або зламаний контейнер вичерпує всі ресурси хоста. В перспективі, інтеграція з оркестраторами на кшталт Kubernetes або OpenStack дозволить автоматично додавати вузли при збільшенні числа користувачів, тобто реалізувати еластичне масштабування кіберполігону [3; 1].

Управління життєвим циклом контейнерів

Автоматизований запуск персонального середовища має доповнюватись так само автоматизованим зняттям цих середовищ по завершенні роботи. Якщо залишати контейнери запущеними невизначений час, це призведе до витоку ресурсів (невикористані контейнери займатимуть пам'ять і портів), а також потенційних ризиків безпеки. В нашій реалізації для кожного контейнера встановлено таймаут сесії — 2 години з моменту запуску. Після спливу цього часу контейнер автоматично зупиняється і видаляється, якщо студент не запросив продовження. Крім того, при добровільному виході користувача із завдання (натисканні кнопки «Завершити лабораторію») його середовище також згортається, щоб уникнути залишення неактивних або незавершених контейнерів у системі. Для реалізації цього механізму серверний застосунок має планувальник завдань, який періодично перевіряє час життя активних контейнерів. Управління здійснюється через Docker API: виконується команда зупинки і видалення контейнера. Важливо, що усі дані всередині контейнера після його видалення втрачаються (система побудована за принципом ephemeral environments — кожен запуск дає чистий стан). Якщо студенту потрібно повторити завдання, запускається новий екземпляр із початковим станом. Такий підхід спрощує підтримку (не треба зберігати проміжний стан) і усуває проблему накопичення змінених середовищ. Автоматизація більшості таких операцій значно знижує навантаження на адміністратора полігону та забезпечує стабільну роботу платформи [1; 2].

Одним із пріоритетів при розробці було мінімізувати час між натисканням студентом кнопки «Запустити лабораторію» і готовністю середовища до роботи. За рахунок використання контейнерів цей час вдалося звести до десятків секунд або менше (для порівняння, запуск повноцінної VM може тривати кілька хвилин). Щоб прискорити cold start (перший запуск образу), всі необхідні контейнерні образи завчасно завантажуються та кешуються на сервері. Зокрема, перед початком курсу адміністратор розгортає (або оновлює) останні версії образів завдань, тож під час заняття не витрачається час на скачування образу з реєстру. Для великих образів (>1 GB) ми використовували шарувату структуру: спільні базові шари (наприклад, ОС Ubuntu) шаряться між різними образами і завантажуються одноразово. Крім того, Docker демон підтримує копію-на-запис (CoW), що дозволяє запускати багато контейнерів із одного образу, використовуючи пам'ять ефективно: незмінні частини образу займають пам'ять тільки раз, а для кожного контейнера виділяються тільки його унікальні змінені блоки. Це особливо корисно, коли десятки студентів працюють з ідентичним середовищем: RAM витрачається значно менше, ніж якби кожен мав окрему VM. Warm start — повторні запуски того самого образу — здійснюються майже миттєво завдяки кешуванню. Таким чином, продуктивність системи в реальному часі оптимізована для інтенсивної роботи у класі [1].

Особливу увагу приділено безпеці ізольованих середовищ. Хоча Docker-контейнери ізольовані від хостової системи, теоретично існують ризики escape-атак, коли зловмисник усередині контейнера може отримати доступ до системи хоста через уразливості ядра або неправильну конфігурацію. Зважаючи на це, ми дотримувалися принципу найменших привілеїв: усі контейнери запускаються від непривілейованого користувача (відсутність root-доступу всередині контейнера, якщо це не потрібно за сценарієм), без підвищених привілеїв Docker (опція --cap-drop=ALL відсікає небезпечні привілеї ядра), без монтування зовнішніх директорій хоста. Мережевий рушій налаштовано так, що кожен контейнер працює в окремому Docker network, і між контейнерами різних користувачів немає прямого трафіку. Це унеможливлює атаки між студентами або перехоплення трафіку. Кожне середовище міститься у своєму ізольованому просторі адрес, що моделює реальну мережеву сегментацію. Усі ці заходи підвищують рівень ізоляції і забезпечують, що навчальні атаки залишаються в межах «пісочниці» полігону. Практика показала, що за належних налаштувань контейнерний підхід

є достатньо безпечним для освітніх цілей, тоді як його переваги у швидкодії та легкості розгортання суттєво переважують залишкові ризики [1].

Інтеграція з Moodle та підтримка безперервного користувацького досвіду

При впровадженні системи важливо було зробити її максимально непомітною для кінцевого користувача — студента. Інтеграція через LTI забезпечила єдиний вхід, але окрім цього ми подбали про узгодженість інтерфейсу та навігації. Застосунок кіберполігону вбудовується у вікно Moodle за допомогою iframe, що забезпечує безперервність інтерфейсу для студента та збереження контексту LMS. Верхня панель полігону містить навігаційні елементи Moodle (логотип, назву курсу тощо) для збереження контексту. LTI-запит передає інформацію про курс і завдання, тому наш застосунок може відображати назву поточної лабораторної роботи, інструкції і навіть оцінку, отриману студентом, синхронно з курсом. Викладачі інтегрують кіберполігон як звичайний модуль курсу Moodle (через активність "External Tool"), що дозволяє використовувати всі стандартні можливості LMS — обмеження за датою, умови відкриття, журнали оцінок тощо. З технічного боку, нам довелося вирішити питання відповідності протоколу LTI останньої версії (1.3 / LTI Advantage) і сумісності з Moodle. Ми реалізували необхідний OAuth 2.0 потік і перевірку підписів JWT, якими Moodle постачає дані про користувача при запуску інструменту. Безперервність користувацького досвіду також означає швидке повернення з полігону до інших матеріалів курсу. Цього досягнуто завдяки тому, що полігон відкривається в тій самій вкладці браузера і має кнопку виходу, яка просто закриває фрейм. Із погляду студента, робота в кіберполігоні нічим не відрізняється від роботи з вбудованим модулем курсу [3; 6].

Висновки та подальші напрями роботи

У статті описано реалізацію практико-орієнтованої системи кіберполігону, інтегрованої в освітнє середовище через Moodle і LTI, що дозволяє надавати кожному студенту персональне ізольоване середовище на основі Docker-контейнерів. Такий підхід вирішує одразу кілька задач: студенти отримують реалістичний досвід роботи з вразливими системами та атаками у безпечних умовах; викладачі можуть масово розгортати вправи без ручної підготовки кожної машини; навчальний процес стає більш інтерактивним і наближеним до реальних викликів кібербезпеки. Інтеграція через LMS забезпечує зручність для кінцевих користувачів і вписує практичні заняття в навчальні плани без швів [3; 1; 6].

Перспективи розвитку системи охоплюють декілька напрямів. Сценарії командної гри (Red/Blue team) — наступний логічний крок у еволюції полігону. Як показує досвід, ефективна підготовка фахівців із кібербезпеки має передбачати командну взаємодію і змагальний елемент. Ми плануємо реалізувати можливість створення спільних віртуальних середовищ для двох і більше груп студентів, де одна команда виконує роль атакуючих (Red Team), а інша — захисників (Blue Team). Технічно це потребуватиме розгортання багатокомпонентних середовищ: декількох контейнерів (або й комбінації контейнерів із повноцінними ВМ) в одній віртуальній мережі, з розподілом доступу між командами. В рамках такої розширеної моделі ми розглядаємо інтеграцію з інструментами оркестрації і моніторингу, щоб масштабні навчальні кібербитви залишалися керованими і відтворюваними [1].

Другим напрямом є розробка модуля аналітики для викладача. Зараз інструктор отримує досить обмежений зворотний зв'язок. Такий інструмент міг би збирати анонімізовані дані з контейнерів (наприклад, командні логи або журнали подій системи) і візуалізувати їх у зручній формі [1].

На завершення, розвиток кіберполігонів як навчального інструменту відкриває нові горизонти для ІТ-освіти. Представлена система показує, як сучасні технології контейнеризації та стандарти інтеграції (LTI) дають можливість створити гнучке, масштабоване і персоналізоване середовище для здобуття практичних навичок. Подальше вдосконалення платформи — впровадження командних сценаріїв, розумної аналітики і глибшої інтеграції з інфраструктурою безпеки — зробить навчальні кіберполігони ще більш наближеними до реальних умов роботи кіберфахівців. Це сприятиме підготовці нового покоління спеціалістів, озброєних не лише теоретичними знаннями, а й досвідом реагування на сучасні кіберзагрози у безпечному навчальному просторі [3; 1; 7; 6].

Список літератури

1. Chouliaras N. A novel autonomous container-based platform for cybersecurity training and research / N. Chouliaras, N. Karanikolas, T. Diamantopoulos, S. Gritzalis // PeerJ Computer Science. — 2023. — Vol. 9. — e1574.
2. Cloud Range. What is a Cyber Range? [Electronic resource]. — FAQ. — 2023. — Mode of access: https://cloudrangecyber.com/ (date of access: 25.06.2025).
3. Lazarov W. Lessons Learned from Using Cyber Range to Teach Cybersecurity at Different Levels of Education [Electronic resource] / W. Lazarov // Technology, Knowledge and Learning. — 2025. — Mode of access: https://doi.org/10.1007/s10758-025-09840-y.
4. O'Rourke G. Unique Challenges for CTFd – GitHub repository plugin [Electronic resource] G. / O'Rourke. — 2021. — Mode of access: https://github.com/ (date of access: 25.06.2025).
5. Thompson M. F. Labtainers: A Docker-based framework for cybersecurity labs / M. F. Thompson, C. E. Irvine // Proceedings of the 2021 ACM SIGCSE. — 2021.
6. Virginia Cyber Range. Cyber Range LTI Integration. — Knowledge Base. — 2023.
7. What Is Missing in Traditional Cybersecurity Training Programs? [Electronic resource]. — Cloud Range Cyber. — 16.08.2024. — Mode of access: https://cloudrangecyber.com/ (date of access: 25.06.2025).

References

Chouliaras, N., Karanikolas, N., Diamantopoulos, T., & Gritzalis, S. (2023). A novel autonomous container-based platform for cybersecurity training and research. *PeerJ Computer Science*, *9*, e1574. https://doi.org/10.7717/peerj-cs.1574.

Cloud Range. (2023). What is a cyber range? — FAQ. *Cloud Range Cyber*. https://cloudrangecyber.com/.

Cloud Range. (2024, August 16). What is missing in traditional cybersecurity training programs? *Cloud Range Cyber*. https://cloudrangecyber.com/.

Lazarov, W. (2025). Lessons learned from using cyber ranges to teach cybersecurity at different levels of education. Technology, Knowledge and Learning. Advance online publication. https://doi.org/10.1007/s10758-025-09840-y.

O'Rourke, G. (2021). Unique challenges for CTFd [Computer software]. *GitHub*. https://github.com/CTFd/CTFd.

Thompson, M. F., & Irvine, C. E. (2021). Labtainers: A Docker-based framework for cybersecurity labs. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 123–129). Association for Computing Machinery. https://doi.org/10.1145/3408877.3432403.

Virginia Cyber Range. (2023). Cyber Range LTI integration — Knowledge base. https://virginiacyberrange.org/knowledgebase/lti.

A. Hlybovets, T. Babych

IMPLEMENTATION OF A PRACTICE-ORIENTED CYBER RANGE SYSTEM WITH INDIVIDUALIZED ENVIRONMENT DEPLOYMENT

The growing complexity and frequency of cyberattacks require a transformation in cybersecurity education. Traditional approaches, which focus primarily on theoretical knowledge, certification, and compliance, often fail to provide learners with sufficient hands-on experience. To address this gap, cyber ranges have emerged as an effective tool for practical, scenario-based training in a safe and controlled environment. This paper presents the implementation of a modern cyber range system designed specifically for educational purposes. The platform is container-based and integrates seamlessly into the learning process via the LTI standard, enabling single sign-on and data exchange with the Moodle LMS. Each student receives an isolated environment on demand, with auto-generated flags embedded for individual task verification. The system automatically deploys and destroys containers based on session activity, ensuring efficient use of server resources and maintaining a secure, ephemeral environment for practice. The solution also includes network segmentation and privilege restrictions for each container, enhancing security while simulating real-world conditions. The proposed architecture supports scalability and rapid deployment, making it suitable for use in classrooms, exams, and self-paced learning. In this article, we review existing approaches to building cyber ranges, discuss the design and implementation of our system, and highlight the technical and pedagogical benefits. Future developments includes support for team-based Red/Blue training, instructor analytics modules, and orchestration integration. The presented platform bridges the gap between theoretical training and real-world cybersecurity skills development, helping to prepare students for modern digital threats.

Keywords: cyber range, cybersecurity education, practical training, Capture the Flag (CTF), containerization, Docker, Moodle, LTI integration, interactive learning, educational platform.

Матеріал надійшов 01.07.2025

УДК 004.9
DOI: 10.18523/2617-3808.2025.8.180-189

Гаврилюк В. Д., Гороховський К. С., Соболевська Л. Г.

КРОС-ЧЕЙН ІНФРАСТРУКТУРА ДЛЯ ВЕРИФІКОВАНОГО СТИМУЛЮВАННЯ РЕСАЙКЛІНГУ: ДОСВІД ВПРОВАДЖЕННЯ НА ICP ТА SOLANA

У цій статті розглянуто принципи створення вебплатформи для підвищення екологічної свідомості у громадян України. Продемонстровано, як використання блокчейн-технологій надає можливість побудувати надійний додаток, який гарантує коректність даних, пов'язаних з переробленням, і дозволяє розробити прозору систему винагород за допомогою utility-токенів і NFT. У роботі детально розглянуто процес розроблення екологічно корисної ініціативи, враховуючи аргументацію обраних технологічних рішень, аналіз вимог до системи, а також масштабованість і перспективи створеного у результаті вебдодатка в майбутньому.

Ключові слова: екологія, блокчейн, ICP, Internet Identity, Solana, переробка сміття, децентралізація.

Вступ

Сортування і перероблення сміття сьогодні — це не просто тренд, а справжній спосіб зберегти навколишнє середовище і зменшити руйнівний вплив цивілізації на природні екосистеми. Перероблення відіграє вирішальну роль у збереженні природних ресурсів, оскільки зменшує потребу у видобуванні первинної сировини. Крім того, перероблення також сприяє зменшенню обсягів відходів, що потрапляють на сміттєзвалища, і тих, що спалюють.

Проблеми з переробленням вторсировини в нашій державі існують протягом довгого часу і мають серйозний характер. До прикладу, за статистикою видання «USA today», Україна є однією з десяти країн, які виробляють найбільшу кількість сміття на одну особу [13]. Відповідно до даних Міністерства розвитку громад та територій України [1], у 2021 р. було утворено близько 51 млн м^3 побутових відходів, з яких 2 % спалили і приблизно 3–4 % відправили на перероблення. Проблема полягає не лише у великій кількості виробленого сміття, а у відсотку його перероблення. Вітчизняна статистика рециклінгу є критично низькою порівняно з країнами ЄС, де в середньому 40 % сміття використовують повторно. Отже, збільшення обсягів переробки в Україні є важливим з огляду євроінтеграції.

Існує декілька основних причин, чому в Україні не є розвиненою культура сортування і рециклінгу. Серед них можна назвати такі [3]:

1. Хоча в Україні на законодавчому рівні передбачено вимоги щодо поводження з побутовими відходами (зокрема, відповідно до Закону України «Про відходи»), на практиці ці норми часто не виконують або не контролюють належним чином. На відміну від більшості європейських країн, де за порушення правил сортування передбачено реальні санкції для громадян, в Україні немає ефективного механізму моніторингу та притягнення до відповідальності. Окрім того, через обмежену кількість пунктів приймання, переробних підприємств і сміттєсортувальних станцій, процес перероблення залишається складним і малодоступним для більшості населення.
2. Серед значної частини українців досі немає усталеної практики роздільного збирання сміття. Це пов'язано, зокрема, з браком чіткої системи заохочення: громадяни не мають зрозумілої вигоди від сортування відходів, а наявні механізми стимулювання здебільшого не дають очікуваного результату. У результаті процес сортування сприймається як додатковий клопіт без реального сенсу чи користі.

Запропоноване рішення

Враховуючи попередньо наведені факти, можна стверджувати, що платформа з елементами гейміфікації, яка може мотивувати громадян долучатись до екологічної спільноти, надаючи винаго-

роду за перероблення, є справді актуальною. Як рішення було розроблено вебзастосунок, що надає змогу отримати винагороду у формі токенів, завантаживши фотографію з переробної станції зі сміттям як підтвердження своєї екологічно корисної діяльності. Перевірка валідності даних користувача про рециклінг буде здійснюватися на основі API-інтеграції зі штучним інтелектом. Токени, зі свого боку, є валютою, за яку можна придбати право на знижку у мережі магазинів партнерів, шаблони цих купонів (NFT-токенів) партнер може добровільно розміщати в застосунку. Списання знижки відбувається з використанням згенерованих на основі NFT QR-коду. Стан купону (активний або неактивний) зберігається on-chain, що унеможливлює повторне використання знижки. Елементи гейміфікації у формі щоденних квестів і опитувань допомагають учасникам застосунку більше дізнатись про принципи сортування та отримати додаткові бонуси, що викликає зацікавленість з боку користувачів у регулярному відвідуванні додатку. Створена платформа має назву **Proof of recycling**, або скорочено **PoR** («Доказ передання на перероблення»).

На сьогодні застосунок має дві основні ролі — Recycler і Partner. Recycler — користувач, який надає докази перероблення сміття і отримує за це токени. За токени він може купувати купони на знижку в партнерів. Partner — користувач, який може розміщувати NFT, що дає право на знижку на платформі. Він також може списувати купони, придбані користувачами.

Аналіз подібних рішень

Проаналізовані українські аналоги платформи мають серйозні недоліки. До прикладу, мобільний застосунок «Кліматичні краплі» [2] має схожу концепцію: за різноманітні корисні екологічні дії, і рециклінг також, користувач отримує «краплі», якими він може розплатитись у закладах партнерів. Однак, із технічного погляду, додаток має багато недоліків. Наприклад, інформація про зароблені токени зберігалась на пристрої користувача, отже, після зміни смартфона він втрачав свої бали. Більше того, таке рішення дозволило зловживати можливостями додатка отримати «краплі» через технічні неточності в коді платформи. Також нагорода за більшість активностей надавалась вручну модераторами. Це, своєю чергою, призводить до потреби у залученні додаткового людського ресурсу, а також може викликати певну недовіру у користувачів, тому що з таким підходом немає гарантії, що вони отримають обіцяну компенсацію за свої зусилля. Беззаперечно, існують і закордонні аналоги цієї ідеї, як-от іспанський Reciclos [10] або італійський Junker app [8], що пропонують бонуси за перероблення відходів, однак вони не є популярними в Україні, і, на відміну від них, наша платформа забезпечує прозору фіксацію факту перероблення з використанням блокчейну, що унеможливлює шахрайство та дозволяє автоматично нараховувати винагороди у формі токенів.

Використання блокчейн-технологій

Застосування децентралізованого підходу дає можливість позбутись багатьох недоліків конкурентів. PoR написаний на смарт-контрактах — самостійно виконуваних програмах, що зберігаються в блокчейні і виконують транзакції на основі попередньо заданих умов. Відповідно, учасники платформи можуть бути впевнені у гарантованому отриманні бенефітів за свої старання, і для цього не потрібна зовнішня регуляція з боку людини. Окрім того, блокчейн дає змогу побудувати прозору і гнучку систему винагород на основі utility-токенів і NFT, що буде працювати на децентралізованих серверах, це робить її стійкою до відмов і унеможливлює зловживання. Таким чином, партнери, котрі надають можливість отримати знижку в додатку, можуть бути впевнені у достовірності права на бонуси, що є критично важливим у контексті ризику фінансових втрат.

Internet computer protocol

Internet computer protocol (ICP) є ключовою технологією, що лягла в основу проєкту. ICP — це інноваційна блокчейн-мережа, розроблена фондом DFINITY, яка дозволяє повноцінно розробляти і розгортати клієнтську та серверну частини вебзастосунків на децентралізованих серверах. Вона є у мільйони разів потужнішою і здатна замінити хмарні сервіси й традиційні IT-системи [5]. Мета ICP — розширити публічний інтернет, додавши до нього вбудовану функціональність хмарних обчислень. Серед головних переваг ICP, на які ми спирались при виборі технології, можна виділити:

1. Смарт-контракти в блокчейні ICP можуть використовувати сотні гігабайтів пам'яті з використанням stable memory [12] та обчислюватися з повною швидкістю сучасного процесора, що на кілька порядків перевищує можливості смарт-контрактів Ethereum.
2. На відміну від більшості децентралізованих систем, смарт-контракти в ICP можуть самостійно надсилати HTTPS-запити до зовнішніх сервісів, тим самим вирішуючи The oracle problem [9]. Усунення потреби в зовнішніх ораклах надає значну кількість покращень, дозволяючи напряму взаємодіяти з API, базами даних і смарт-контрактами інших блокчейнів.
3. ICP застосовує reverse gas model, що пропонує розробникам попередньо оплачувати комісії за газ, наповнюючи свої смарт-контракти так званими циклами. Це дає змогу користувачам взаємодіяти з додатком без потреби мати токени чи гаманець. Такий підхід спрощує вхід у Web3, забезпечуючи користувацький досвід, подібний до традиційних вебзастосунків.
4. Смарт-контракти, що працюють на ICP, є потужною еволюцією традиційних смарт-контрактів, і їх називають каністрами (рис. 1). Це обчислювальні одиниці, які поєднують у собі як код, так і дані. Каністри можна використовувати для найрізноманітніших цілей — від надання вебсторінок для клієнтського застосунку, написаних на JS-фреймворку, до реалізації повноцінної децентралізованої системи. Каністри можуть бути написані різними мовами, які здатні компілюватись до стандарту WASM; основними є Rust і Motoko.

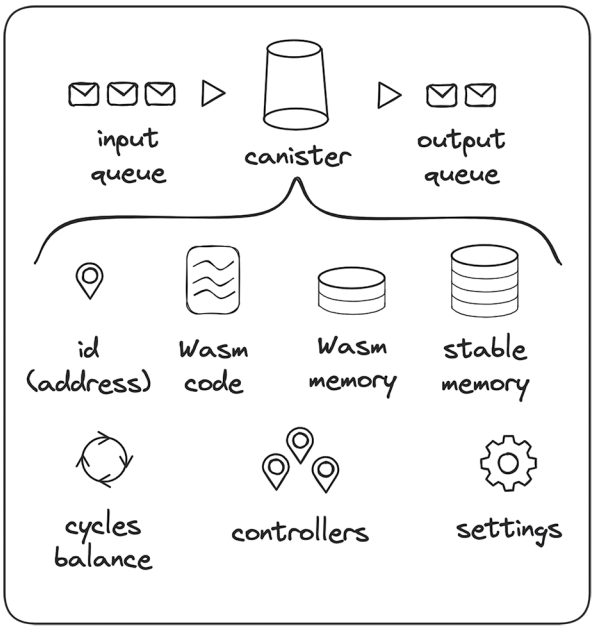


Рис. 1. Основні компоненти смарт-контракту («каністри») в ICP

5. Internet computer protocol має вбудовану систему автентифікації — Internet Identity, що керує сесіями на основі криптографії. Коли користувач підключається до децентралізованого застосунку, створюється нова пара відкритого та закритого ключів для сесії, яку Internet Identity засвідчує за допомогою поєднання WebAuthn [6] і порогового підпису (threshold signing).
6. Використовуючи інструмент збірки dfx, процес розгортання каністр у локальному середовищі є простим, що дозволяє розробникам швидко демонструвати результати їхньої роботи. Водночас, перехід між локальною мережею і mainnet потребує виконання лише декількох термінальних команд і поповнення каністр циклами, що дає змогу легко переконфігурувати проєкт для промислового використання.

Solana

Зі свого боку, Solana — це більш класична блокчейн-платформа, створена організацією Solana labs у 2017 р. Вона не надає можливості розробляти повноцінні вебзастосунки з клієнтською частиною, проте пропонує модель побудови швидких децентралізованих додатків на базі смарт-контрактів і DAO.

Алгоритм консенсусу Proof of History [4] є ядром блокчейну Solana. Цей механізм відіграє центральну роль у забезпеченні високої пропускної здатності блокчейну ефективним способом. Solana теоретично може здійснювати оброблення 65 тисяч паралельних операцій.

Написані на ній програми доцільно використати у розробленій на ICP платформі, продемонструвавши можливість інтернет-комп'ютера робити інтеграції з іншими децентралізованими системами. Крім того, інтеграція Solana в проєкт уможливлює порівняння блокчейнів із різними моделями газу, адже в ньому користувач самостійно платить за комісію для здійснення транзакцій.

Загальна архітектура

Вебзастосунок використовує класичний варіант клієнт-серверної взаємодії, маючи окремий SPA-застосунок, що працює в браузері, а також розділені сервіси для серверної та блокчейн логіки. Основу платформи було написано на ICP та розгорнуто в каністрах (рис. 2).

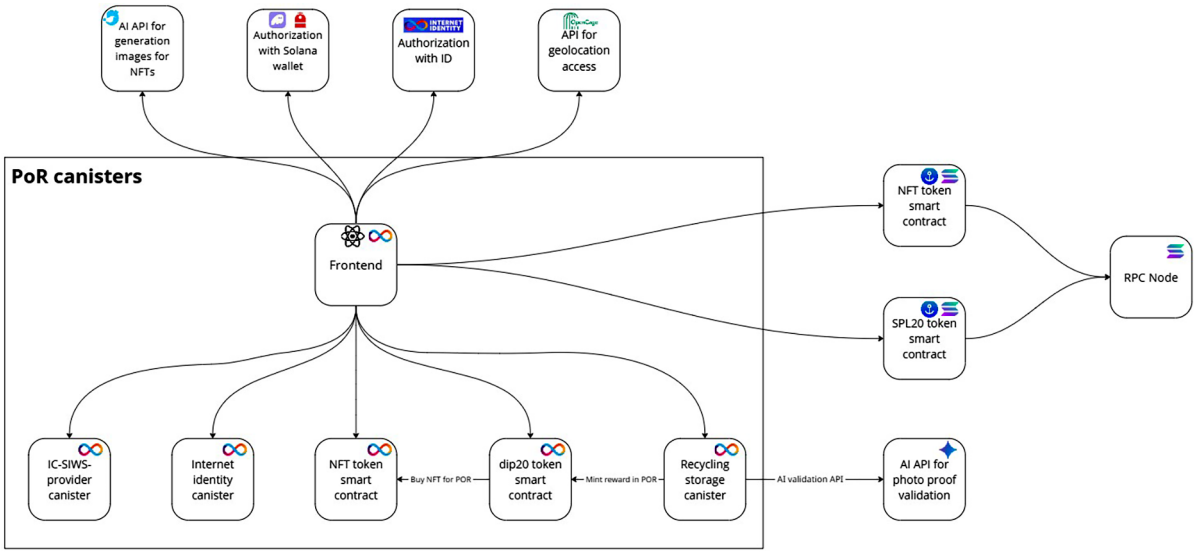


Рис 2. Загальна архітектура застосунку

Клієнтська частина, написана на React.js, також розміщена в мережі інтернет-комп'ютера, хоча сьогодні більшість децентралізованих застосунків, побудованих на інших блокчейнах, покладаються на централізованих хмарних провайдерів (наприклад, AWS, GCP) для хостингу фронтенду, що створює єдину точку відмови і порушує саму концепцію децентралізованої і незалежної платформи. Написані смарт-контракти на інтернет-комп'ютері, окрім логіки розподілення токенів, додатково дають змогу створювати повноцінні бекенди зі сховищем даних, використовуючи Stable memory.

Каністри в ICP

Для застосунку було розроблено три основні каністри (і використано дві готові). Каністра **storage** (рис. 3) відповідає за збереження інформації користувача про рециклінг, а також робить запити до інших контрактів і до Gemini API для валідації фотографії переробки. Смарт-контракт **dip20** імплементує стандарт взаємозамінних токенів для ICP, що застосовуються у платформі у вигляді POR-токенів, які є внутрішньою валютою додатку. Каністра **nft** реалізує типовий смарт-контракт для логіки невзаємозамінних токенів, що використовуються як доказ користувача на право знижки у партнера. Для розроблення додатка на ICP ми використали мову програмування Rust разом із Rust Canister Development Kit (CDK). Для імплементації постійного сховища даних у каністрах ми застосовуємо Rust-крейт ic-stable-structures, що робить можливим використання звичайних структур даних, однак при цьому загальний ліміт стабільної пам'яті в одній каністрі може досягати 500 гігабайтів.

Як уже зазначено раніше, одним із найбільших плюсів інтернет-комп'ютера є його здатність робити HTTPS-виклики поза власною мережею, не використовуючи оракли. Ми використовуємо цю функцію, щоб делегувати перевірку фотодоказу користувача, коли він надає інформацію про переробку; як наслідок, якщо користувач надав будь-яке невідповідне зображення, то він не зможе отримати винагороду.

```rust
use ic_cdk::api::call::call;
use lazy_static::lazy_static;
use candid::Principal;

type PrincipalId = String;


#[derive(CandidType, Deserialize, Clone)]
4 implementations
struct RecycleData {
    image: Vec<u8>,
    comment: String,
    location: String,
    principal_id: PrincipalId,
    created_at: u128 ,
}

#[update]
async fn store_data(image: Vec<u8>, comment: String, location: String, created_at: u128) -> Result<String, String> {
    if !validate_image_with_gemini(image.clone(), &location, &comment).await? {
        return Err("AI rejected the image as invalid.".to_string());
    }
    let principal_id: String = caller().to_string();
    let recycle_data: RecycleData = RecycleData { image, comment, location, principal_id: principal_id.clone(), created_at };

    let mut storage: MutexGuard<'_, HashMap<String, …>> = STORAGE.lock().unwrap();
    let user_records: &mut Vec<RecycleData> = storage.entry(principal_id.clone()).or_insert(vec![]);
    user_records.push(recycle_data);
    reward_user(caller()).await;
    Ok(principal_id)
}
```

Рис. 3. Використання IC CDK і функція зберігання даних про переробку

У своїй роботі ми застосовуємо вбудовану систему автентифікації Internet Identity, адже ми хочемо надати доступ до смарт-контрактів лише авторизованим користувачам. Для інтеграції з Internet Identity потрібно розгорнути в проєкті pre-built каністру, додати деякі конфігурації і, до того ж, встановити на JS-клієнті бібліотеку @dfinity/auth-client.

Інтеграція Solana в застосунок

Доповненням до основної частини платформи є написання Solana-програм та їх інтеграція в застосунок. Для написання смарт-контрактів на Solana було обрано фреймворк Anchor на мові Rust. Anchor — це провідна платформа для розроблення програм на Solana. З використанням цього фреймворку вдалось написати функціональність винагород, аналогічну до тієї, що спроєктована в ICP каністрах **dip20** і **nft**. Таким чином, якщо при вході на платформу обрати спосіб авторизації з криптогаманцем (Phantom, Backpack чи іншим), то клієнтська частина вебзастосунку буде звертатись до Solana-програми, і, як наслідок, нагорода буде нараховуватись на баланс обраного криптогаманця.

Для того щоб взаємодіяти з каністрами, користувач повинен бути авторизованим і мати прив'язану до його сесії Internet Identity. Однак якщо користувач буде авторизовуватись через криптогаманець Solana, то він не зможе автоматично викликати методи ICP смарт-контрактів, для цього потрібен мапінг публічної адреси гаманця до новоствореної Internet Identity. Саме тому в проєкті використано в застосунку рішення від ic-siws [7] — це проєкт, який забезпечує автентифікацію на основі гаманця Solana для застосунків на ICP (Internet Computer). Застосування цієї бібліотеки гарантує, що вхід через Solana-гаманець завжди використовує той самий Principal (ідентифікатор ICP), незалежно від клієнта або пристрою, створюючи однозначну відповідність між Solana-адресами і Principal-ідентифікатором.

Клієнтська частина

Для створення користувацького інтерфейсу обрано популярний фреймворк ReactJS. Щоби полегшити написання стилів для компонентів, було обрано бібліотеку TailwindCSS. Цей CSS-фреймворк дотримується принципу mobile-first системи брейкпоінтів. Використовуючи цей принцип, вдалось створити адаптивний вебінтерфейс, що враховує особливості розмірів різних типів пристроїв.

Важливим також є механізм поєднання фронтенд-каністри з іншими. Це можливо за допомогою спеціальних Candid-інтерфейсів. Candid — це мова опису інтерфейсів (interface description language), призначена для визначення публічного контракту сервісу. Згенерувати їх для каністри можна за допомогою інструменту збірки dfx.

Інтеграції із зовнішніми сервісами

Для автоматизації деяких процесів на платформі були налаштовані інтеграції з API сторонніх сервісів на клієнтській і серверній частинах.

Для процесу валідації зображення на основі використовується API моделі Gemini 2.0 Flash. Gemini — це сімейство мультимодальних великих мовних моделей (LLMs), розроблених компанією Google DeepMind.

Для автоматизації процесу визначення геолокації було використано сервіс Opencagedata, що надає способи геокодування (перетворення координат у адреси або назви місць) по всьому світу на основі відкритих даних через REST API.

Ще одним сервісом, який було впроваджено у застосунку, є DeepAI — сервіс, що надає REST API для роботи з Text-to-image моделлю для створення зображення на основі наданого промпт-запиту.

Для того щоб зберігати метадані і зображення NFT у децентралізованому сховищі і мати доступ до них за URL-адресою, було обрано сервіс Pinata, який допомагає керувати файлами в IPFS (децентралізована система зберігання файлів). IPFS є корисним для Web3-платформ, бо дозволяє зберігати NFT-метадані та зображення без ризику втрати через централізований сервер.

Розгортання

Для того щоб зібрати програму, потрібно створити файл dfx.json. Файл dfx.json використовується для налаштування параметрів проєкту, зокрема дозволяє вказати перелік каністр у застосунку, їхні назви, типи та вихідні файли, параметри збірки за замовчуванням і багато інших опцій. Можна порівняти файл dfx.json із файлами docker-compose, що так само декларативно визначають список сервісів системи.

Відповідно, для деплою каністр потрібно скористатись командою dfx deploy. Якщо розміщення відбувається в mainnet, то розробник попередньо має поповнити баланс каністри «циклами». Цикли (cycles) — це одиниця обчислювального ресурсу. Вони виконують роль «палива» для каністр, подібно до того, як газ використовується в Solana. Лише за один долар можна придбати близько одного трильйона циклів (1,000,000,000,000 циклів). Такої кількості вистачить, щоби зберігати 1 гігабайт даних у каністрі протягом року, при цьому виконуючи мільйони викликів у межах платформи [11]. Після розгортання застосунку в mainnet, він стає публічно доступним і отримує доменну адресу, що сформована на основі ідентифікатору каністри. Для прикладу, представлена в статті платформа після деплою отримала адресу https://2dwgp-naaaa-aaaan-qzynq-cai.icp0.io.

Принципи роботи платформи

Початково користувач має авторизуватись, для цього він має обрати один із двох можливих методів: напряму через Internet Identity або використовуючи Solana-гаманець. Обираючи Solana wallet,

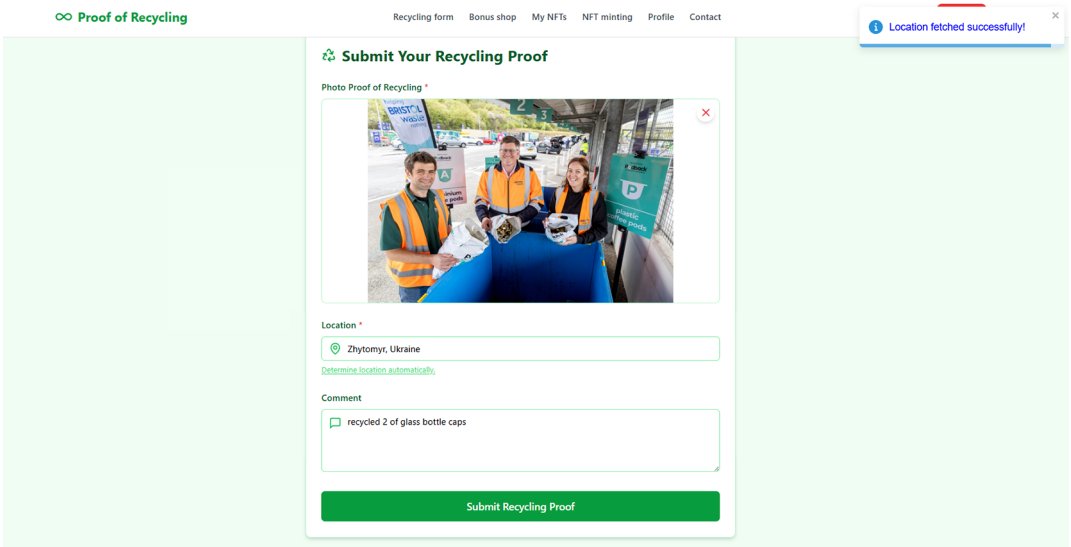


Рис 4. Форма для перероблення сміття

користувач отримуватиме NFT-знижку на обраний Solana-акаунт. При авторизації через Internet Identity відбувається редірект на нову сторінку, цей механізм подібний до OAuth2 авторизації з провайдерами Google, Facebook та іншими. На цій сторінці потрібно обрати або створити новий principal ID. Далі користувач, що здає сміття на перероблення, має надати фотодоказ із локацією і описом своєї екологічно корисної активності (рис. 4). Після перевірки зображення користувач одразу отримає 1000 POR-токенів.

Зі свого боку, користувач-партнер може створити шаблон NFT-знижки, вказавши назву, опис, фото (яке він може згенерувати також за допомогою штучного інтелекту), категорію і зберегти результати. Для нових купонів ціна визначається автоматично і становить 2500 POR.

Коли користувач назбирає принаймні 2500 токенів, у нього з'явиться опція придбати знижку в магазині бонусів. Якщо він не має приєднаного Solana-гаманця, то NFT буде відображатись на сторінці NFT-колекції, інакше NFT можна буде побачити і на криптогаманці. На прикладі здійснюється купування купону для знижки на масаж (рис. 5).

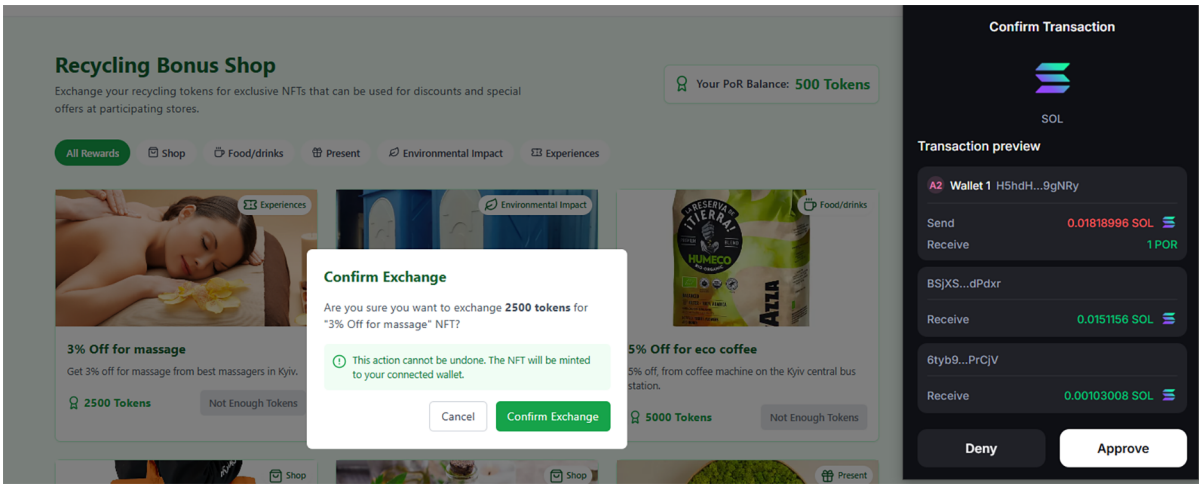


Рис 5. Покупка знижки за POR-токени з Solana wallet

Учасник платформи, що купує знижку з Solana-гаманцем, має сплатити комісію за проведення транзакції, однак за аналогічну дію, що здійснюється на ICP, оплата за транзакцію не береться. Це можливо через reverse gas model.

Щоб користувач мав змогу скористатися своєю знижкою, він має перейти на сторінку з власною NFT-колекцією і показати QR-код партнеру, який створив шаблон цього купона. Лише він здатний списати знижку. Для синхронного оновлення змін було використано технологію WebSocket. QR-код генерується за допомогою бібліотеки react-qr-code (рис. 6).

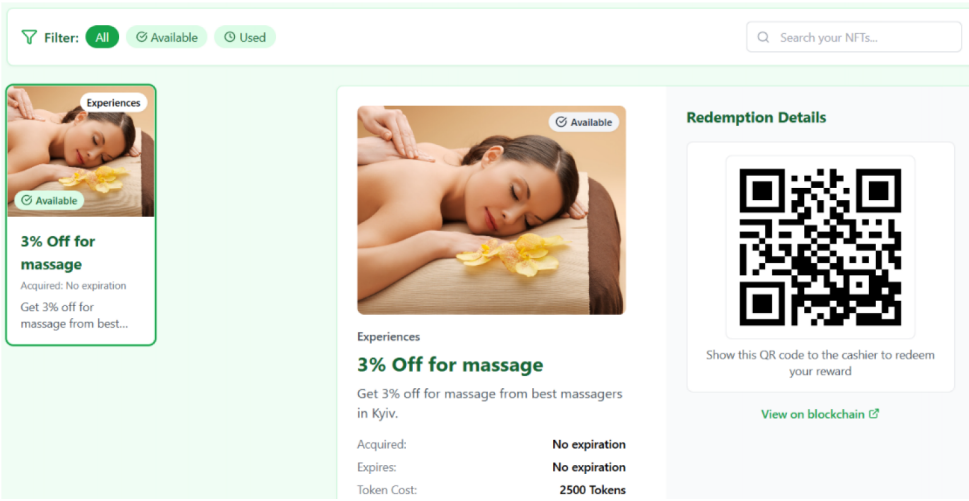


Рис 6. NFT-колекція знижок перед списанням

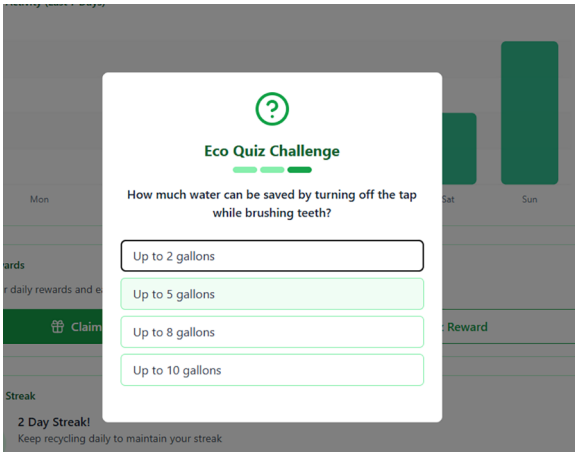


Рис 7. Проходження щоденного опитування

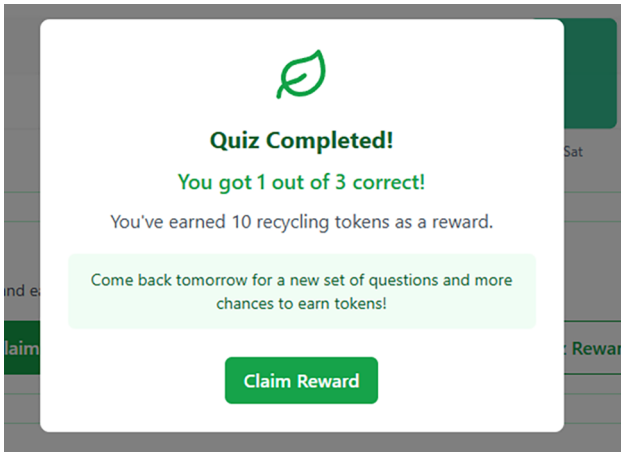


Рис 8. Результат опитування

Крім того, існує можливість заробляти POR-токени за певну активність на платформі. Валюту можна отримати за щоденне відвідування застосунку, а також за невелике опитування, що складається з трьох екологічних питань (рис. 7). Чим більше правильних відповідей буде набрано, тим більше токенів учасник може отримати (рис. 8). До того ж, розроблено функціональність рейтингу спільноти, що може бути додатковим способом заохочення учасників вкладати більше зусиль у перероблення за рахунок фактора змагальності.

Висновок

У процесі роботи над проєктом вдалось дослідити серйозність і масштаби екологічних проблем в Україні. Крім того, визначено головні переваги ICP для побудови децентралізованих застосунків, а також проаналізовано відмінності між блокчейнами з різними моделями газу. Було проведено детальний аналіз і надано аргументацію використання Web3-архітектури у проєкті.

Як результат, було реалізовано гнучку платформу для мотивації користувачів переробляти відходи. Вебдодаток має адаптивний інтерфейс, розгорнутий і повноцінно працює у блокчейні-мережі ICP. Також удалось додати кросчейнову функціональність, побудувавши Solana-програми на Anchor і впровадивши їх у платформу. Було додано декілька способів авторизації, залежності від вибору та потреб користувача. Логіка отримання винагород написана і працює на обох блокчейнах. Разом з тим, розроблені інтеграції з API зовнішніх сервісів, що значно автоматизують деякі процеси під час роботи в додатку. Елементи гейміфікації надають змогу зацікавити користувача у щоденному відвідуванні платформи, а також допомагають дізнатись більше про проблематику перероблення, що існує в нашій державі.

Ідея застосунку вже отримала ствердні відгуки від експертів. Проєкту Proof of Recycling удалось перемогти в хакатоні Let's try ICP і посісти друге місце на Mohyla startup accelerator. Це означає, що платформа має високий потенціал завдяки своїй важливій соціальній місії і інноваційному поєднанню децентралізованих технологій.

Список літератури

1. Проєкт національного плану управління відходами України до 2033 року. Міністерство захисту довкілля та природних ресурсів України — офіційний сайт [Електронний ресурс]. — 2021. — Режим доступу: https://mepr.gov.ua/wp-content/uploads/2023/12/proyekt-Natsionalnyj-plan-upravlinnya-vidhodamy-23.11-002.docx (дата звернення: 25.06.2025).
2. Сімончук О. Як мобільний додаток «Кліматичні краплі» вчить дітей ековідповідальності [Електронний ресурс]/ О. Сімончук // Освіторія Медіа. — 2019. — Режим доступу: https://osvitoria.media/experience/yak-mobilnyj-dodatok-klimatychni-krapli-vchyt-ditej-ekovidpovidalnosti (дата звернення: 25.06.2025).
3. Скільки українців готові сортувати сміття. Active Group — Маркетингові та соціологічні дослідження [Електронний ресурс]. — 2021. – Режим доступу: https://activegroup.com.ua/2021/11/15/skilki-ukraynciv-gotovi-sortuvati-smittya (дата звернення: 25.06.2025).
4. Albus J. Solana deep dive: Unpacking proof-of-history [Electronic resource] / J. Albus // Blockworks. — Mode of access: https://blockworks.co/news/solana-proof-of-history (date of access: 25.06.2025).
5. Blockchain and Applications, 4th International Congress (Modelling of the Internet Computer Protocol Architecture: The Next Generation Blockchain; pp. 3–12) [Electronic resource] / ed. by J. Prieto et al. — Cham : Springer International Publishing, 2023. — Mode of access: https://doi.org/10.1007/978-3-031-21229-1 (date of access: 25.06.2025).

6. Guide to Web Authentication. WebAuth docs [Electronic resource]. — Mode of access: https://webauthn.guide/#about-webauthn (date of access: 25.06.2025).
7. Kristofer L. SIWS, Sign in with Solana for ICP, the Internet Computer. Build cross chain Solana apps on ICP [Electronic resource] / L. Kristofer. — GitHub. — Mode of access: https://github.com/kristoferlund/ic-siws (date of access: 25.06.2025).
8. Recycling is easy in Italy, just a blip away with Junker. La differenziata in un blip [Electronic resource]. — Mode of access: https://junker.app/recycling-is-easy-in-italy-just-a-blip-away-with-junker (date of access: 25.06.2025).
9. Stable memory | Internet Computer. Internet Computer docs [Electronic resource]. — Mode of access: https://internetcomputer.org/docs/motoko/main/stable-memory/stablememory (date of access: 25.06.2025).
10. The app that rewards you for recycling — Reciclos [Electronic resource]. — Mode of access: https://www.reciclos.com/en (date of access: 25.06.2025).
11. The concept behind canister smart contracts. DFINITY Foundation. Terminology [Electronic resource]. — 2025. — Mode of access: https://support.dfinity.org/hc/en-us/articles/360057605571-What-are-canister-smart-contracts (date of access: 25.06.2025).
12. The oracle problem in smart contracts: data privacy, security, and solutions. MediaLaws [Electronic resource]. — Mode of access: https://www.medialaws.eu/rivista/the-oracle-problem-in-smart-contracts-data-privacy-security-and-solutions (date of access: 25.06.2025).
13. Waste recycling market size, share, growth & trend by 2032. Allied Market Research. (2024) [Electronic resource]. — 2024. — Mode of access: https://www.alliedmarketresearch.com/waste-recycling-market-A144607 (date of access: 25.06.2025).

References

Albus, J. (2025). Solana deep dive: Unpacking proof-of-history. *Blockworks*. https://blockworks.co/news/solana-proof-of-history.

Blockchain and Applications, 4th International Congress (Modelling of the Internet Computer Protocol Architecture: The Next Generation Blockchain; pp. 3–12). (2023) / ed. by J. Prieto et al. Springer International Publishing. https://doi.org/10.1007/978-3-031-21229-1.

Guide to Web Authentication. *WebAuth docs*. https://webauthn.guide/#about-webauthn.

Kristofer, L. SIWS, Sign in with Solana for ICP, the Internet Computer. Build cross chain Solana apps on ICP. *GitHub*. https://github.com/kristoferlund/ic-siws.

Proyekt natsionalnoho planu upravlinnya vidkhodamy Ukrayiny do 2033 roku. Ministerstvo zakhystu dovkillya ta pryrodnykh resursiv Ukrayiny — ofitsiynyy sayt. (2021). https://mepr.gov.ua/wp-content/uploads/2023/12/proyekt-Natsionalnyj-plan-upravlinnya-vidhodamy-23.11-002.docx [in Ukrainian].

Recycling is easy in Italy, just a blip away with Junker. *La differenziata in un blip*. https://junker.app/recycling-is-easy-in-italy-just-a-blip-away-with-junker.

Simonchuk, O. (2019) Yak mobilnyy dodatok "Klimatychni krapli" vchyt ditey ekovidpovidalnosti. *"Osvitoriya" Media*. https://osvitoria.media/experience/yak-mobilnyj-dodatok-klimatychni-krapli-vchyt-ditej-ekovidpovidalnosti[in Ukrainian].

Skilky ukrayintsiv hotovi sortuvaty smittya. Active Group — Marketynhovi ta sotsiolohichni doslidzhennya. (2021). https://activegroup.com.ua/2021/11/15/skilki-ukraynciv-gotovi-sortuvati-smittya [in Ukrainian].

Stable memory | Internet Computer. *Internet Computer docs*. https://internetcomputer.org/docs/motoko/main/stable-memory/stablememory.

The app that rewards you for recycling — Reciclos. *Reciclos*. https://www.reciclos.com/en.

The concept behind canister smart contracts. *DFINITY Foundation*. *Terminology*. (2025). https://support.dfinity.org/hc/en-us/articles/360057605571-What-are-canister-smart-contracts.

The oracle problem in smart contracts: data privacy, security, and solutions. *MediaLaws*. https://www.medialaws.eu/rivista/the-oracle-problem-in-smart-contracts-data-privacy-security-and-solutions.

Waste recycling market size, share, growth & trend by 2032. *Allied Market Research*. (2024). https://www.alliedmarketresearch.com/waste-recycling-market-A144607.

V. Havryliuk, K. Gorokhovskyi, L. Sobolevska

CROSS-CHAIN INFRASTRUCTURE FOR VERIFIED RECYCLING INCENTIVIZATION: IMPLEMENTATION EXPERIENCE ON ICP AND SOLANA

This article explores the principles of developing a web platform aimed at increasing environmental awareness among Ukrainian citizens through digital rewards for recycling waste. The continuous increase in polluted areas across Ukraine has led to a range of ecological issues. In this context, the creation of a digital tool that draws public attention to the problem of waste management and motivates responsible environmental behaviour is both relevant and necessary.

The use of blockchain technologies provides the opportunity to build a reliable application that ensures the integrity of data related to waste recycling. It also enables the creation of a transparent reward system based on utility tokens and NFTs. The integration of two blockchain ecosystems — Internet Computer Protocol (ICP) and Solana — demonstrates the strengths of both technologies in terms of decentralisation, user accessibility, and scalability.

The ICP blockchain offers a unique environment for hosting complete web applications on decentralised nodes without relying on traditional centralised servers. Through its innovative reverse gas model, end users are not required to hold tokens to interact with the platform, as the computational costs are covered

by the developer or organisation behind the application. This allows for a seamless and user-friendly experience, particularly for non-technical participants in the recycling initiative.

Meanwhile, Solana adds flexibility for the development of smart contracts and fast transaction processing. By combining these two technologies, the article provides a comparative overview of different blockchain architectures and gas models, contributing to a deeper understanding of their practical implications.

Additionally, the article presents a brief explanation of the integration between the ICP and Solana blockchains, including token-based interactions, NFT issuance, and synchronisation of environmental data. The development process is supported by deployment tools such as DFX (for ICP canister management) and the Anchor framework for Solana, ensuring an efficient and structured development workflow.

Overall, the work highlights the reasoning behind the selected technological stack, outlines the key requirements for the platform, and explores its potential for future growth and scalability. The project exemplifies how modern decentralised technologies can be harnessed to promote sustainable development and environmental responsibility in Ukrainian society.

Keywords: ecology, blockchain, ICP, Internet Identity, Solana, waste recycling, decentralisation.

Матеріал надійшов 01.07.2025

УДК 004.9
DOI: 10.18523/2617-3808.2025.8.190-196

Яременко П. В., Гороховський К. С.

ОМНІЧЕЙН-ІНТЕГРАЦІЯ ТОКЕНА MOR НА ОСНОВІ СТАНДАРТІВ LAYERZERO OFT І WORMHOLE NTT

У статті представлено проєкт MORSOL — омнічейн-рішення для токена MOR, що поєднує стандарти LayerZero OFT і Wormhole NTT. Архітектура інтегрує Solana (Anchor-програма) та EVM-смартконтракти з мережевими компонентами LayerZero і Wormhole. Для контролю емісії використано мультипідпис: Squads Vault на Solana та Gnosis Safe на EVM. Описано реалізацію механізму burn/mint під час передання токенів між мережами, підтвердження VAA, а також безпекові функції, як-от пауза контрактів і контроль загальної емісії. Результати демонструють, що поєднання OFT і NTT дозволяє створити нативний мультиланцюговий токен MOR без обгортання, зі збереженням єдиної емісії та високою децентралізацією.

Ключові слова: омнічейн, крос-ланцюгова інтеграція, LayerZero OFT, Wormhole NTT, мультипідпис (multisig), Solana, Ethereum, інтероперабельність, децентралізація.

Вступ

У сучасних умовах розвитку блокчейн-екосистем дедалі більшої ваги набуває проблема інтероперабельності — можливості активів і додатків взаємодіяти між різними ланцюгами. Зокрема, існує потреба у тому, щоб криптовалютні токени могли вільно переміщуватися між кількома блокчейнами, залишаючись при цьому єдиним цілісним активом (без створення дублюючих «обгорнутих» версій).

Існують різні міжланцюгові протоколи для передання даних і активів. Одними з провідних рішень у сфері мостів є LayerZero та Wormhole — обидва надають інфраструктуру для обміну повідомленнями між блокчейнами, але реалізують різні моделі безпеки та довіри. LayerZero пропонує стандарт OFT (Omnichain Fungible Token) для випуску токенів одразу на декількох ланцюгах з використанням модульної моделі перевірки повідомлень незалежними верифікаторами (DVN). Wormhole, своєю чергою, запровадив фреймворк NTT (Native Token Transfers) для *нативного мультиланцюгового токена* з довіреною перевіркою повідомлень через децентралізовану мережу вартових (Guardians). Обидва підходи використовують модель burn-and-mint (спалення на вихідному ланцюгу та карбування на цільовому) замість обміну через ліквідні пули, що дозволяє переносити цінність без створення вторинних токенів.

Morpheus (MOR) — нативний токен екосистеми Morpheus (мережі персональних AI-агентів) — було задумано як омнічейн-актив, доступний на кількох платформах Web3.

У цій статті докладно описано реалізацію омнічейн інтеграції MOR, що отримала назву MORSOL. MORSOL забезпечує зв'язок між мережею Solana та EVM-ланцюгами з використанням гібридної архітектури: на Solana розгорнуто смарт-контракт (Anchor-програму), який інтегрується з LayerZero та Wormhole, тоді як на кожному EVM-ланцюгу працює стандартний контракт MOR (ERC-20 з розширенням OFT). Для управління критичними функціями (емісія, налаштування мереж тощо) залучено мультипідписні акаунти: Squads Vault на Solana та Gnosis Safe на Ethereum і сумісних ланцюгах. У розділі **Методологія** розглянуто алгоритми міжланцюгового переказу MOR, зокрема послідовність операцій burn/mint та механізми підтвердження (DVN у LayerZero та VAA у Wormhole). У розділі **Архітектура** описано основні компоненти системи та їхню взаємодію, зокрема роль LayerZero endpoint-контрактів і Wormhole-інфраструктури. Далі, у розділі **Результати**, аналізуються досягнуті показники та досвід використання запропонованої системи. **Висновки** підсумовують отримані результати та окреслюють внесок роботи в галузі мультиланцюгових технологій.

Методологія

Підхід до інтеграції: Проєкт MORSOL ставить за мету зробити токен MOR повноцінним омнічейн-активом, який циркулює одночасно на кількох блокчейнах без створення дублікативних wrapped-версій. Методологія побудована на принципі єдиного глобального балансу: незалежно від того, в якій мережі перебувають монети MOR, загальна кількість токенів в обігу залишається сталою. Для цього всі міжланцюгові трансфери виконуються через спалення та карбування: токени вилучаються (спалюються) з обігу на вихідному ланцюгу і створюються (карбуються) на цільовому ланцюгу у такій самій кількості. Ця модель підтримується як стандартом Wormhole NTT, так і LayerZero OFT, і дозволяє уникнути появи конкурентних копій токена та необхідності довіряти стороннім пулам ліквідності.

Підсумовуючи, методологія реалізації MORSOL базується на поєднанні найкращих практик двох різних протоколів: ми використовуємо стандартні, аудовані компоненти LayerZero та Wormhole без модифікацій ядра, додаючи при цьому свій управлінський шар (мультипідпис) для максимального контролю. Це дає змогу досягти мети — зробити MOR повністю мультиланцюговим токеном — із мінімальними змінами логіки токена.

Архітектура

Огляд архітектури: Система MORSOL складається з компонентів, розгорнутих у двох середовищах — мережі Solana та низці EVM-сумісних мереж, які тісно взаємодіють через повідомлення LayerZero і Wormhole. На рис. 1 (схема архітектури MOR на Solana) представлено загальну структуру рішення.

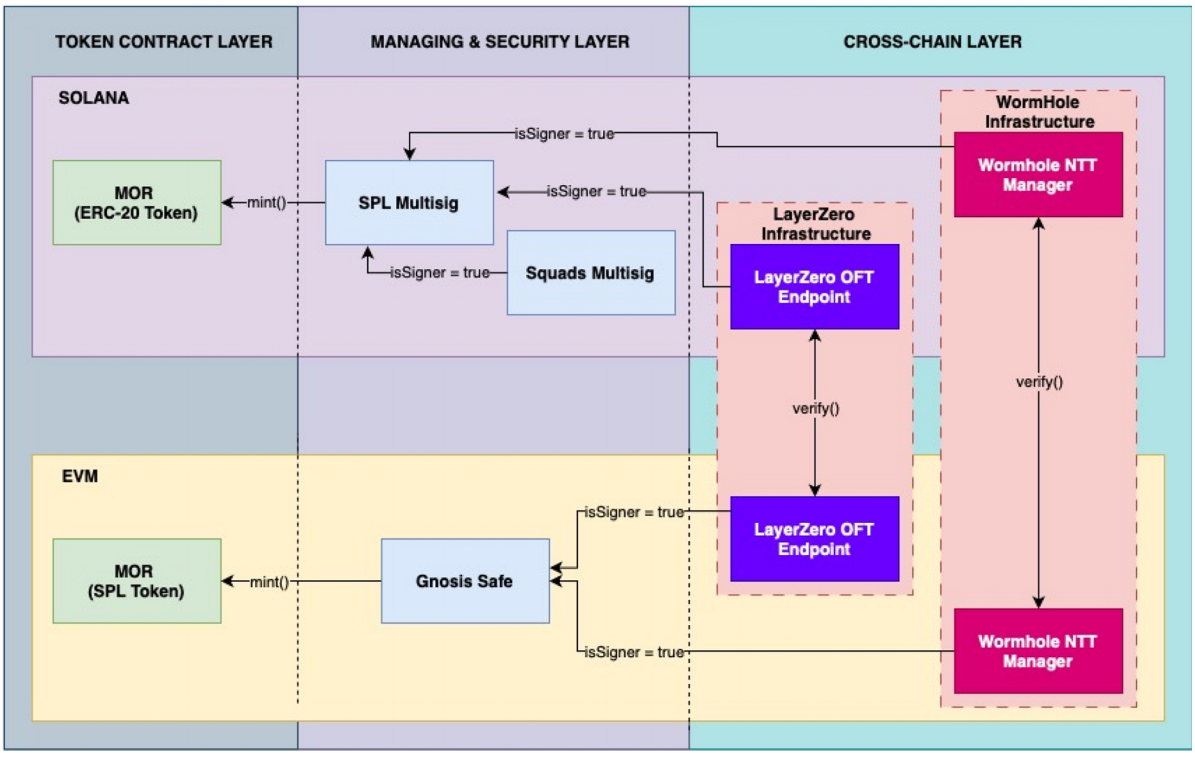


Рис. 1. Архітектура MORSOL

Основні складові можна розділити на кілька рівнів:

- **Рівень токен-контрактів.** Це контракти, що безпосередньо керують балансами MOR у певній мережі. На Solana роль токен-контракту виконує SPL-мінт MOR, а на EVM-ланцюгах — контракт ERC-20 MOR-OFT. Важливо, що всі ці контракти представляють *той самий* токен MOR, тобто загальна емісія розподілена між ними (сумарний оборот постійно 42 млн MOR відповідно до токеноміки). Контракт MOR на EVM успадковує функціонал LayerZero: він інтегрований з Endpoint-контрактом LayerZero і охоплює логіку відправки / отримання крос-ланцюгових

повідомлень. На Solana ж SPL-мінт MOR взаємодіє із Anchor-програмою MORSOL, яка виступає контролером випуску токенів. Той самий підхід реалізовано і для WormHole NTT.

- **Рівень керування і безпеки.** До нього належать *мультипідписні гаманці* та рольові налаштування, що контролюють критичні операції контрактів. У архітектурі MOR реалізовано дворівневу модель контролю: (1) багатопідписний контроль емісії токена на Solana через SPL Multisig та Squads Multisig для додаткової можливості контролю емісії, та (2) мультисиг-контроль адміністративних функцій контрактів на EVM через Gnosis Safe. На Solana SPL Multisig виступає власником (mint authority) токена MOR і має право дозволяти чи блокувати карбування нових монет. Крім того, цей самий мультипідпис може бути призначений повноважним підписантом для критичних інструкцій Anchor-програми (напряму або опосередковано). На EVM-ланцюгах мультипідписні гаманці Gnosis Safe (5 із 9 підписів) контролюють права адміністратора контрактів MOR, тобто можуть затверджувати оновлення логіки, зміну конфігурацій безпеки LayerZero (наприклад, список довірених DVN) або активацію екстрених механізмів на кшталт паузи. У сукупності, мультипідписна інфраструктура додає рівень децентралізованого управління поверх автоматизованих мостів: жоден окремий підписант, окрім контрактів, не здатен одноосібно випустити нові токени або змінити налаштування, не отримавши згоду кількох незалежних сторін.
- **Рівень міжланцюгової логіки.** Це компоненти, що забезпечують надсилання і прийом повідомлень між блокчейнами. У LayerZero це Endpoint-контракти (по одному на кожному ланцюгу); у Wormhole — Core (Core Bridge) контракти та пов'язані з ними NTT Manager контракти. LayerZero Endpoint являє собою універсальний комунікаційний модуль: всі звернення OMNI-додатків (таких як OFT) проходять через нього. Endpoint відповідає за формування стандартного пакета повідомлення, його доставку (разом з обраними DVN (Decentralized Verifier Networks) веріфаєрами) на інший ланцюг та виклик потрібної функції там. Wormhole Core — аналогічний за призначенням модуль, але працює інакше: він записує повідомлення на вихідному ланцюгу та ініціює збір підписів Guardians; на цільовому ланцюгу він перевіряє підписи в VAA (Verified Action Approvals) і передає дані в NTT Manager для виконання трансферу. NTT Manager — це окремий контракт, що розгортається на кожному ланцюгу для конкретного токена; він містить бізнес-логіку роботи з Wormhole (ініціювання burn, прийом VAA, карбування) і, як правило, належить інтегратору токена. У випадку MOR, Anchor-програма MORSOL на Solana підтримує NTT Manager (і одночасно OFT-адаптера) на стороні Solana, а для EVM-ланцюгів цю роль може відігравати або сам контракт MOR (через впровадження інтерфейсу Wormhole), або допоміжний контракт.

Компоненти Solana. В екосистемі Solana токен MOR представлений у вигляді SPL-токена. Його випуск і переміщення контролюються Anchor-програмою MORSOL. Основні інструкції програми охоплюють спалення MOR на Solana та надсилання повідомлення в іншу мережу; карбування MOR на Solana на основі отриманого підтвердження з іншого ланцюга та інші інфраструктурні задачі.

З погляду міжмережевої комунікації, Solana взаємодіє як із LayerZero, так і з Wormhole. MORSOL підключає Endpoint через CPI (Cross-Program Invocation): при відправці токенів у EVM вона викликає функцію Endpoint send() з необхідними параметрами, а при отриманні — Endpoint сам викликає зареєстрований хендлер програми після верифікації повідомлення (аналогічно до того, як це відбувається в EVM контрактах OFT). Взаємодія з Wormhole здійснюється через вбудовані програми Solana: Wormhole Core Bridge (Program Id відповідного моста) та, за необхідності, Wormhole Token Bridge. Для NTT-режиму Token Bridge може працювати в спеціальному режимі: MOR може бути зареєстрований як *нативний токен* замість стандартного обгорнутого. В будь-якому разі, мультипідпис на Solana є центральною ланкою: він зберігає mint-authority MOR та визначає, кому довірено право карбування. Так, при розгортанні MOR було створено обліковий запис-мультисиг (структура Solana, де задано M підписантів із N і поріг 1) і призначено Authority для MOR-мінта. Потім до списку підписантів цього мультисіг-акаунту були додані: (1) PDA скарбниці Squads (Vault PDA), (2) PDA самої Anchor-програми MORSOL як потенційний співпідписант, (3) зарезервоване місце для Wormhole NTT (буде додано після розгортання). Порог підписів встановлено таким чином, щоб для карбування MOR був потрібен підпис програми + підтвердження Vault (умовно 2 з 2) або, на перехідний період, 1 з 2 (де достатньо підпису програми за наявності VAA). Це означає, що навіть коли Anchor-програма виконує інструкцію карбування токенів (наприклад, при надходженні повідомлення з Ethereum), фінальна дія mint_to виконується токен-програмою Solana тільки за умови,

що підпис мультисиг-акаунту валідовано [1]. Практично, SquadsVault може *наперед схвалювати* певні дії в межах встановлених політик, наприклад дозволити програмі MORSOL карбувати до X токенів на добу, але в будь-якому разі програма не зможе перевищити ті ліміти без згоди кворуму ключів. Цей багатопідписний механізм істотно підвищує безпеку: навіть у разі теоретичного злому міжланцюгового мосту зловмисник не зможе самостійно «надрукувати» MOR на Solana — потрібна дія, схвалена колективно.

Компоненти EVM. На стороні EVM до архітектури входять один або кілька контрактів MOR (ERC-20/OFT) і потенційно допоміжні контракти-мости. Під час запуску MOR було вирішено максимально використовувати типовий контракт OFT від LayerZero без модифікацій протоколу. Контракт MOROFT.sol успадковує клас OFTCore (для нових токенів) і реалізує стандартні методи ERC-20 (transfer, balanceOf тощо) разом з новими (sendFrom, estimateSendFee тощо). Особливістю MOR є те, що його випуск здійснювався за схемою *Fair Launch* із розподілом на кількох мережах, тому контракт має фіксовану загальну кількість 42,000,000 MOR, але на початку не всі монети були випущені одразу на всіх ланцюгах. Контракт MOR на Ethereum був розгорнутий як основний мінт, із початковим випуском певної частини токенів, інші мережі почали з нульовим балансом і отримували токени в міру їх претендування або перенесення з Ethereum. Контракт MOROFT налаштований таким чином, що право карбування (mint) на ньому жорстко закріплене за окремим обліковим записом — L2MessageReceiver (або іншим контрактом-дистриб'ютором). Це зроблено для розмежування ролей: сам токен не може бути *довільно* накарбований навіть власником — випуск нових монет відбувається лише за наявності валідного крос-ланцюгового запиту (через контракт-посередник). Таким посередником в екосистемі MOR слугує контракт розподілу, що отримує підтвердження про зарезервовані для користувача токени з «головного» ланцюга і викликає mint на локальному MOR (це стосується фази емісії токена). У контексті міжланцюгових переказів функцію посередника виконує LayerZero механізм: коли MOR надходить з іншого ланцюга, повідомлення потрапляє прямо у функцію onOFTReceived контракту MOR, який і карбує монети для отримувача. Адміністраторські права контракту MOR (як-от призначення нового власника чи оновлення Proxy-логіки, якщо така використовується) закріплені за мультисиг-гаманцем Gnosis Safe. Для MOR був налаштований Gnosis Safe 5/9 на кожному основному EVM-ланцюгу, що включає ключі декількох членів спільноти Morpheus. Таким чином, усі незвичайні дії з контрактом MOR (наприклад, активація *pause* у LayerZero Endpoint або зміна налаштувань Trusted Remote) потребують колективного рішення декількох довірених осіб. Поточна реалізація контрактів MOR на EVM *без проксі*, тобто вони не можуть бути оновлені (immutable); однак безпечний власник (Safe) може у разі надзвичайної ситуації застосувати функції паузи (в LayerZero Endpoint є функція блокування виходу / входу повідомлень) або встановити ліміти переказів. Цими можливостями, втім, планується користуватися лише в разі критичної необхідності (наприклад, у випадку експлойту) — у звичайному режимі мультипідпис не втручається в роботу токена.

Особливу увагу при проєктуванні EVM-частини приділено інтеграції Wormhole. На момент написання статті MOR ще не повністю задіяв Wormhole на Ethereum, але в архітектурі передбачено розгортання окремого контракту MOR-NTT Manager (можливо, як розширення наявного контракту або окремого модуля). Цей NTT Manager на Ethereum буде уповноважений спалювати MOR (з балансу користувача, який ініціює переказ) та карбувати MOR (на адресу отримувача після підтвердження VAA). Для цього мультисиг Safe, що володіє контрактом MOR, додасть NTT Manager як мінтер / бернер до токена MOR. Конфігурація Wormhole тоді віддзеркалить solana-сценарій: NTT Manager (Ethereum) + NTT Manager (Solana, тобто MORSOL програма) будуть зареєстровані одне в одного як довірені адреси, а Squads Vault слугуватиме кінцевим арбітром, який підтверджує випуск токенів на Solana.

Отже, архітектура EVM-складової є достатньо простою з точки зору користувача — це стандартний контракт токена MOR, сумісний з будь-якими гаманцями та біржами, тоді як «під капотом» він містить логіку LayerZero і прив'язку до мультисиг. Завдяки цьому MOR може безшовно переміщуватись між Ethereum та іншими мережами другого рівня, залишаючись під захистом налаштованої нами моделі безпеки.

[1] Squads — популярний мультисиг-сервіс у Solana, де Vault PDA — це програмно-генерована адреса, яка може тримати активи і підписувати транзакції після проходження голосування учасників мультипідпису. У нашому випадку Squads Vault не зберігає токени напряму, а використовується як довірений підписант операцій з мінтом MOR.

Результати

Функціональність мосту. Реальні тести переказів підтвердили коректність роботи як LayerZero-, так і Wormhole-компонентів. З Ethereum у Arbitrum та Base токен переміщується майже миттєво: завдяки тому, що LayerZero сплачує комісію на вихідному ланцюгу і оптимізує доставлення, користувач за декілька хвилин бачить токени на рахунку в цільовій мережі (з урахуванням часу фіналізації блоку). Перенесення між Ethereum та Solana (через Wormhole) займає більше часу, оскільки передбачає підтвердження від Guardian-мережі та очікування підтверджень в обох блокчейнах; проте і в цьому випадку транзакції проходять успішно, а затримка типово не перевищує 2-3 хвилини. Користувачі Morpheus отримали змогу безпечно переміщувати MOR, наприклад із Solana (де токен може використовуватися в Solana-програмах або DEXах) до Ethereum (для торгівлі на Uniswap та участі в Ethereum-діях) і назад. Це значно підвищує ліквідність та утилітарність токена, адже усуває розрив між ізольованими екосистемами — MOR функціонує як міст між спільнотами Solana та EVM.

Дотримання єдиної емісії. Одним із головних результатів є підтвердження того, що контроль за загальною кількістю MOR діє коректно. У процесі тестування спеціально моделювалися крайні ситуації — одночасні перекази великих сум через різні мости, спроби повторного відправлення того самого повідомлення тощо. У всіх випадках механізми безпеки (зокрема, Global Accountant Wormhole та перевірка nonce повідомлень у LayerZero) запобігли будь-яким дублюванням чи розбалансуванню постачання. Загальна кількість MOR на всіх ланцюгах після серії тестових транзакцій залишалася рівно 42,000,000, що підтверджує принцип збереження маси в нашій моделі.

Висновки

У цій роботі представлено комплексне рішення для омнічейн-інтеграції токена MOR, яке поєднує дві провідні міжланцюгові технології — LayerZero OFT та Wormhole NTT. Проєкт MORSOL показав, що мультиланцюговий токен може бути реалізований безпечним і прозорим чином, якщо грамотно скомбінувати сильні сторони різних протоколів і додати рівень децентралізованого контролю. Архітектура MOR охоплює Anchor-програму на Solana, мережу смарт-контрактів на EVM-ланцюгах, а також інфраструктурні компоненти LayerZero (endpoint-и, DVN) та Wormhole (Core, Guardians, VAA). Ключовою особливістю є використання мультипідписної моделі управління: всі операції карбування токена MOR на Solana здійснюються за участі Squads-мультисиг, а критичні налаштування контрактів на Ethereum перебувають під контролем Gnosis Safe. Це гарантує, що жодна транзакція переміщення або випуску MOR не відбувається без колективної згоди, — потужний запобіжник проти зловживань або експлойтів.

Запропоноване рішення забезпечує уніфікований обіг токена MOR на різних платформах: замість ізольованих wrapped-версій, MOR існує всюди як той самий актив, дотримуючись єдиної економіки і загального максимального обсягу. Завдяки механізму burn-and-mint та перевіркам LayerZero / Wormhole, вдалося досягти цього без втрати швидкості або зручності для користувачів. Аналіз безпеки показав, що комбінована модель має високу стійкість: навіть у разі гіпотетичного прориву одного рівня захисту інший рівень (наприклад, мультисиг) візьме на себе запобігання негативним наслідкам.

Практичні результати впровадження MORSOL підтвердили ефективність підходу. MOR успішно функціонує на мережах Solana, Ethereum, Arbitrum, Base, забезпечуючи користувачам свободу переміщувати свої токени між ними і використовуючи там, де це найбільш вигідно. Це підвищує ліквідність токена і сприяє інтеграції спільнот різних блокчейнів у єдину економічну систему проєкту Morpheus. Водночас було досягнуто високого рівня довіри з боку спільноти: прозора мультипідписна інфраструктура та публічні аудити дали впевненість, що з токеном не станеться непередбачуваних емісій чи блокувань.

Отже, проєкт MORSOL робить вагомий внесок у розвиток концепції омнічейн-токенів. На практичному прикладі продемонстровано, що децентралізований крос-ланцюговий контроль є можливим без істотних втрат у продуктивності чи юзабіліті. Отримані знання можуть бути використані іншими розробниками токенів, які прагнуть максимальної міжланцюгової сумісності своїх проєктів. Поєднання LayerZero OFT та Wormhole NTT, доповнене механізмами DAO-управління, може стати новим стандартом для випуску активів у багатоланцюговому світі Web3.

У підсумку, інтеграція MOR засвідчує, що бачення One Token, Many Chains уже реалізується на практиці. Очікується, що з часом все більше проєктів ітимуть цим шляхом, розмиваючи межі між блокчейнами і створюючи по-справжньому універсальні криптоактиви. Досвід MOR підтверджує: правильна архітектура та орієнтація на безпеку здатні подолати бар'єри сумісності, відкривши шлях до більш зв'язаного і децентралізованого криптосередовища.

Список літератури

1. Blockchain Technology for Tracing Drug with a Multichain Platform: Simulation Method [Electronic resource] / E. Fernando et al. // Advances in Science, Technology and Engineering Systems Journal. — 2021. — Vol. 6, no. 1. — Pp. 765–769. — Mode of access: https://doi.org/10.25046/aj060184 (date of access: 25.06.2025).
2. Documentation | Morpheus [Electronic resource]. — Mode of access: https://gitbook.mor.org/smart-contracts/documentation (date of access: 25.06.2025).
3. Hwang W.-Y. A Study on the Interoperable Method of Security Tokens Between Permissioned and Public Blockchains [Electronic resource] / W.-Y. Hwang, H.-K. Kim // Korea Industrial Technology Convergence Society. — 2025. — Vol. 30, no. 1. — Pp. 23–29. — Mode of access: https://doi.org/10.29279/jitr.k.2025.30.1.23 (date of access: 25.06.2025).
4. Isaac D. Technical Deep Dive: Multichain Tokens [Electronic resource] / D. Isaac // Medium. — Mode of access: https://defi-isaac.medium.com/technical-deep-dive-multichain-tokens-452cbb8c82cc (date of access: 25.06.2025).
5. Nelson D. Crypto Bridge LayerZero Connects to Solana Blockchain [Electronic resource] / D. Nelson // CoinDesk. — Mode of access: https://www.coindesk.com/tech/2024/05/29/crypto-bridge-layerzero-connects-to-solana-blockchain (date of access: 25.06.2025).
6. Omnichain Tokens | LayerZero [Electronic resource]. — Mode of access: https://docs.layerzero.network/v2/concepts/applications/oft-standard (date of access: 25.06.2025).
7. Wormhole: Native Token Transfers (NTT) [Electronic resource] — Mode of access: https://app.blockworksresearch.com/unlocked/wormhole-native-token-transfers-ntt (date of access: 25.06.2025).

References

Blockworks Research. (n. d.). Wormhole : Native Token Transfers (NTT). https://app.blockworksresearch.com/unlocked/wormhole-native-token-transfers-ntt.

Fernando, E., Rizal, M., Ramadhan, I., Setiawan, A., & Sulistyono, A. (2021). Blockchain technology for tracing drug with a multichain platform: Simulation method. *Advances in Science, Technology and Engineering Systems Journal*, *6* (1), 765–769. https://doi.org/10.25046/aj060184.

Hwang, W.-Y., & Kim, H.-K. (2025). A study on the interoperable method of security tokens between permissioned and public blockchains. *Korea Industrial Technology Convergence Society*, *30* (1), 23–29. https://doi.org/10.29279/jitr.k.2025.30.1.23.

Isaac, D. (n .d.). Technical deep dive: Multichain tokens. Medium. https://defi-isaac.medium.com/technical-deep-dive-multichain-tokens-452cbb8c82cc.

LayerZero. (n. d.). Omnichain tokens. https://docs.layerzero.network/v2/concepts/applications/oft-standard.

Morpheus. (n. d.). Documentation | Morpheus. https://gitbook.mor.org/smart-contracts/documentation.

Nelson, D. (2024, May 29). Crypto bridge LayerZero connects to Solana blockchain. CoinDesk. https://www.coindesk.com/tech/2024/05/29/crypto-bridge-layerzero-connects-to-solana-blockchain.

P. Yaremenko, K. Gorokhovskyi

OMNICHAIN INTEGRATION OF THE MOR TOKEN BASED ON LAYERZERO OFT AND WORMHOLE NTT STANDARDS

This article presents the architecture, implementation, and interoperability strategy of the MORSOL system — a cross-chain solution for deploying the MOR token across multiple blockchain ecosystems. The system is based on two widely adopted omnichain token standards: LayerZero's OFT (Omnichain Fungible Token) and Wormhole's NTT (Native Token Transfers), enabling seamless transfers of MOR tokens between Solana and EVM-compatible blockchains.

A distinguishing feature of the architecture is that the MOR token exists in native form across chains, without relying on traditional wrapped token models. The system ensures that MOR has a single, globally tracked total supply, maintained through a burn-and-mint mechanism. This method guarantees that when MOR is transferred from one chain to another, it is burned on the source chain and freshly minted on the destination, eliminating the risk of inflation or double-spending.

The cross-chain functionality is facilitated by a combination of smart contracts and protocol-specific infrastructure. On Solana, the project uses an Anchor-based program, and on EVM networks, Solidity contracts are deployed. LayerZero endpoints handle message relaying and verification through Decentralized Verifier Networks (DVNs), while Wormhole utilizes its Guardians to provide VAA (Verified Action Approval) proofs that validate token transfers.

Governance and token emission are controlled by decentralized multisignature infrastructure: Squads Vault (via PDA accounts) on Solana and Gnosis Safe on EVM chains. These multisigs are used to authorize minting and burning operations and are critical for maintaining control over token supply across chains.

Security features include the ability to pause token transfers, upgrade contracts when needed, and enforce strict validation through the LayerZero and Wormhole relayer networks. A comparative analysis of OFT and NTT protocols is included, showing how their integration leverages the advantages of both approaches—such as LayerZero's modularity and Wormhole's proven track record in VAA validation.

The result is a robust, scalable, and secure omnichain token infrastructure for MOR, capable of supporting decentralized applications (dApps) across multiple ecosystems. This implementation serves as a reference model for other projects seeking to maintain synchronized assets in a multichain environment while preserving decentralization and security.

Keywords: omnichain, MOR token, LayerZero OFT, Wormhole NTT, cross-chain transfer, burn-mint, multisig governance, Squads Vault, Gnosis Safe, decentralized interoperability.

Матеріал надійшов 27.06.2025

УДК 004.415.5
DOI: 10.18523/2617-3808.2025.8.197-204

M. Piz, A. Nahirna

DEVELOPMENT OF AN iOS APPLICATION FOR TASK PLANNING WITH CONSIDERATION OF THE USER'S EMOTIONAL STATE

The article focuses on the development of a digital tool designed to address the problem of decreased productivity caused by emotional exhaustion. The main objective of the study is to create an iOS application for task planning that takes into account the user's emotional state, offers mindful breaks for emotional awareness and recovery, and provides analytics on emotional trends.

The research includes a comparative analysis of existing software solutions in the areas of time management and mental well-being. During the development process, modern frameworks, tools, and architectural patterns for iOS development were analyzed. An adaptive planning algorithm was implemented, that takes into accountboth the user's emotional feedback and the attributes of tasks.

As a result of the research, a mobile application named Moodpace was developed using Swift, with SwiftUI for building the user interface, SwiftData for data persistence, and the MVVM architectural pattern to ensure maintainable code structure. During the development process, SwiftLint was used for static code analysis, and SwiftFormat was integrated for automatic code formatting. The app was localized into Ukrainian using String Catalog.

The developed application is designed to help users manage their tasks while maintaining balance with their mental well-being. It is suitable for everyday use and especially beneficial for individuals with flexible schedules.

Keywords: task management, productivity, emotion, adaptive planning, mobile application, iOS, Swift, SwiftUI, SwiftData.

Introduction

Today's reality is shaped by constant military conflicts, economic challenges, and information overload. In times of this global instability, emotional burnout has become a common problem, which can lead to a significant drop in personal productivity. Studies have shown that employees who engage in emotional regulation exercises and manage their mental health have 20-30% higher productivity compared to those who rely only on time management techniques [7].

At the same time, the rise of digital technology has made popular tools that help users manage their time, track their mood, and general well-being. Though there are a lot of apps in the fields of time management and mood-tracking, there is currently no widespread solution that integrates both productivity and mental health into a unified system. This creates a gap for a digital tool that adapts schedules to users' emotional states and promotes mindful time management.

This paper presents the design and development of a mobile app that addresses this gap. The objective is to create a user-friendly tool that offers a personalized task planner that adjusts to a user's current emotions, encourages mindfulness, and supports analytics to reflect on trends over time and improve future planning. Additionally, the application supports the Ukrainian language. The proposed system effectively combines adaptive planning, emotion tracking, and mindfulness exercises.

Overview of Existing Solutions

To gain a better understanding of the subject area, four mobile applications were analyzed: two focused on emotion tracking (DailyBean [4] and Bearable [3]) and two on task management (TickTick [12] and Notion [8]).

As shown in Table 1, existing tools tend to focus on isolated aspects of the problem domain. None of them provide an automated way to integrate time management with emotional state, which highlights the need for a new approach.

Table 1. **Comparison of Existing Solutions**

App	DailyBean	Bearable	TickTick	Notion
Primary Purpose	Daily journaling of activities and mood	Tracking health metrics, well-being, and mood	Task management using productivity techniques	Information organization and project planning
Time Planning	No	No	Yes	Yes (via templates)
Emotion Tracking	Yes	Yes	No	Yes (via templates)
Customization Level	Low	High	Medium	High
Account Creation	Yes, optional	Yes, required	Yes, optional	Yes, required
Integration with Other Services	No	Yes (Fitbit / Apple Health)	Yes (Calendars, Notion, Apple Health, Reminders etc.)	Yes (Slack, Jira, Google Calendar, Figma etc.)
Reminders	Yes	Yes	Yes	Yes
Ukrainian Language Support	No	Yes	Yes	No
Adaptive Task Planning	No	No	No	No

Additionally, it was discovered that all tools support reminders; only half of them offer Ukrainian language support and require account creation, and most apps enhance usability by having integrations with other services.

Technologies and Tools Used

The mobile application, called Moodpace, was developed for the iOS platform because of increased user engagement on iPhones [13] and higher trust in Apple's privacy policies [2].

Swift 6 was used for development, as it is Apple's recommended programming language for modern iOS development [9].

For building the user interface, the declarative SwiftUI framework was used. It reduces boilerplate code and improves rendering efficiency [5].

For data persistence, the SwiftData framework was chosen as a native solution that combines the advantages of CoreData and Realm and provides easy integration with SwiftUI [10, 11].

To ensure code quality, the static code analyzer SwiftLint was integrated as an Xcode Run Script Build Phase, and the automatic code formatter SwiftFormat was added as a Git pre-commit hook.

Application Architecture and Structure

The application follows the MVVM (Model–View–ViewModel) architectural pattern, which provides clear separation of concerns, where the Model handles data, the View presents the UI, and is responsible for user interaction, and the ViewModel acts as an intermediary that handles business logic. This pattern was selected for its balance of simplicity and scalability, which makes it suitable for medium-sized applications [6].

As shown in Figure 1, the project's file structure is organized in a way that makes it easier to find necessary files and expand the project with new features.

Classification of Tasks

In the proposed system, tasks can be either fixed or flexible.

Fixed tasks have predetermined start and end times, for example, a university lecture or a scheduled daily work meeting.

Flexible tasks need to be completed during the day, but the exact time is not critical; examples include watching a recorded lecture or watering houseplants.

In addition to the type, each task has the following attributes:

– title;
– description;

– category: work, study, personal, home, or other;
– difficulty: very easy, easy, medium, difficult, or very difficult;
– start and end time (for fixed tasks);
– estimated duration (for flexible tasks).

All attributes, except for title, have default values.

Regardless of its type, each task is assigned a time block and is displayed on an interactive calendar. This aligns with the time-blocking technique, encouraging the user to focus on one task at a time.

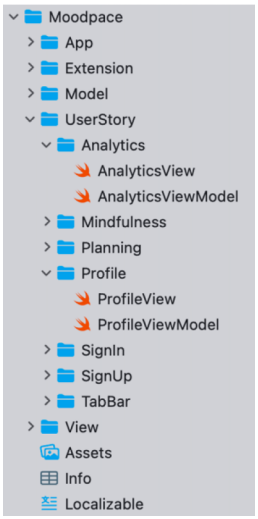


Figure 1. File structure of the app

Emotional Feedback

Once a task is marked as completed, the user is prompted to select the emotion they are experiencing now. This encourages emotional awareness and supports mindful productivity. Emotions are represented by five emojis: very sad, sad, neutral, happy, and very happy. As shown in Figure 2, these emotions are internally converted into numerical values from 0 to 4 – emotionScore.

Emotion.swift

```swift
enum Emotion: String, Codable, CaseIterable {
    case verySad
    case sad
    case neutral
    case happy
    case veryHappy

    var emotionScore: Int {
        switch self {
        case .verySad:
            0
        case .sad:
            1
        case .neutral:
            2
        case .happy:
            3
        case .veryHappy:
            4
        }
    }
}
```

Figure 2. Emotion enumeration and numerical mapping for emotion score values

Adaptive Planning Algorithm

The system uses an adaptive planning algorithm that dynamically adjusts the tasks depending on the user's current emotion. Naturally, the algorithm can only rearrange flexible tasks. In some cases, it also suggests mindfulness exercises.

As illustrated in Figure 3, based on the selected emotion, the algorithm adjusts the user's schedule accordingly:

– if the emotion is neutral or positive (emotion score 2–4), the system assumes that the user is in a good state and can handle more challenging tasks first; so, flexible tasks are sorted by difficulty in descending order;
– if the emotion is negative (emotion score 0–1), the algorithm sorts flexible tasks in ascending order of difficulty so that the user could do something easy and recover mentally. Additionally, the system evaluates the upcoming task. If it is flexible, or fixed, but there is at least a 10-minute gap before it, the short mindfulness practice is suggested. Otherwise, the app just shows a supportive message.

Then the system waits for the next task to be completed, continuously adjusting the workflow to balance productivity and emotional well-being.

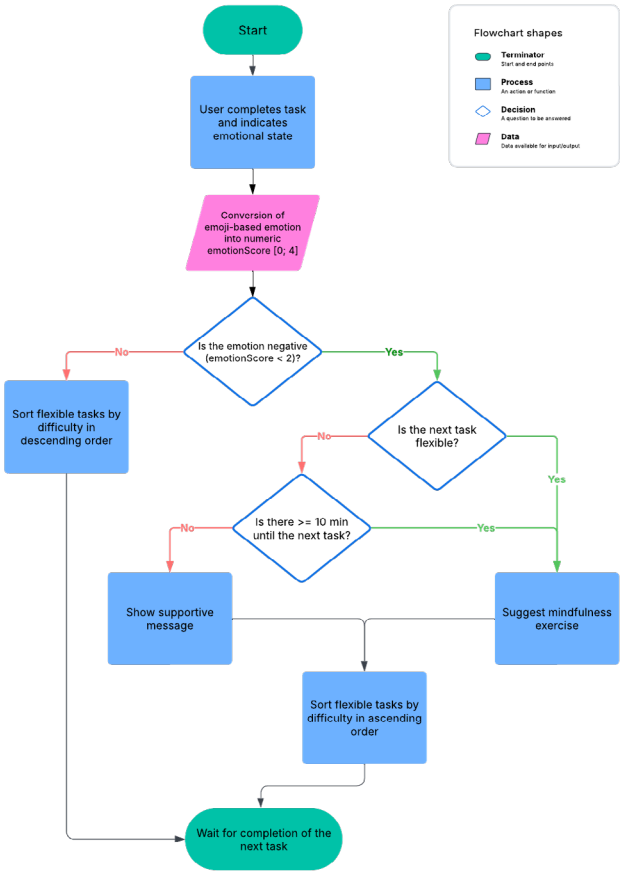


Figure 3. Flowchart of the adaptive planning algorithm

Results

As a result of the research, the iOS app Moodpace was developed. The core functionality of the app is organized into four primary sections, accessible via a tab bar (Figure 4):

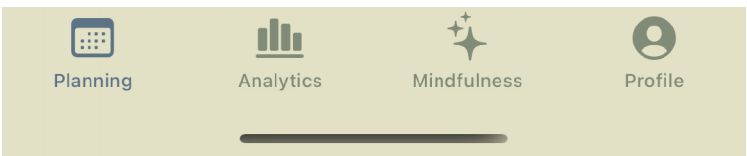


Figure 4. The tab bar of the Moodpace app

1) planning: Using an interactive calendar, users can create, view, edit, and delete tasks (Figure 5). After marking a task as completed, users are asked to rate their emotional state. Based on this input, an

adapted schedule is created and, if needed, a mindfulness exercise is suggested, or a supportive message is shown;

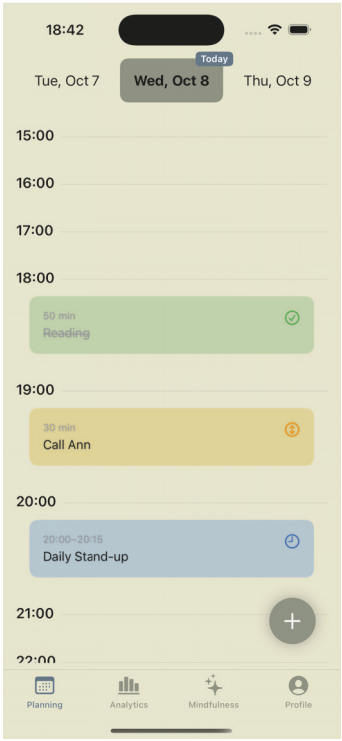


Figure 5. Planning screen

Figure 6. Analytics screen

2) analytics: Users can view statistics on completed and not completed tasks, as well as charts displaying average emotions by category and day over a selected period (Figure 6). These charts are implemented using the Swift Charts framework. The presented analytics help to identify patterns in emotional state based on time and categories;
3) mindfulness: Users can select one of five short interactive mindfulness exercises: 4–7–8 Breathing, Emotional Journaling, 5–4–3–2–1 Grounding, Gratitude Practice, or Emotional Drawing. These practices are recommended after negative emotional feedback but also are available on demand (Figure 7);

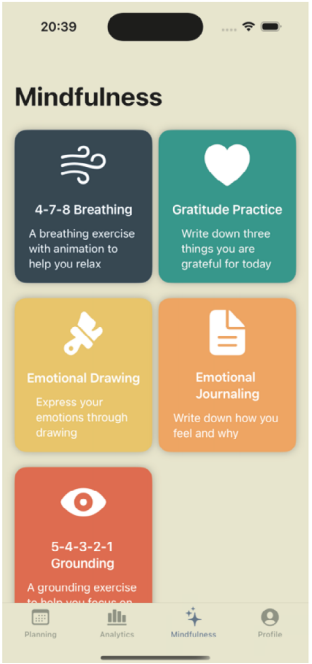


Figure 7. Mindfulness practices screen

4) profile: Users can manage their account and configure preferences, such as receiving notifications for fixed tasks and setting day start/end times. Authorized users can also request a CSV export of their tasks data.

All use cases of the app are shown in the use case diagram in Figure 8.

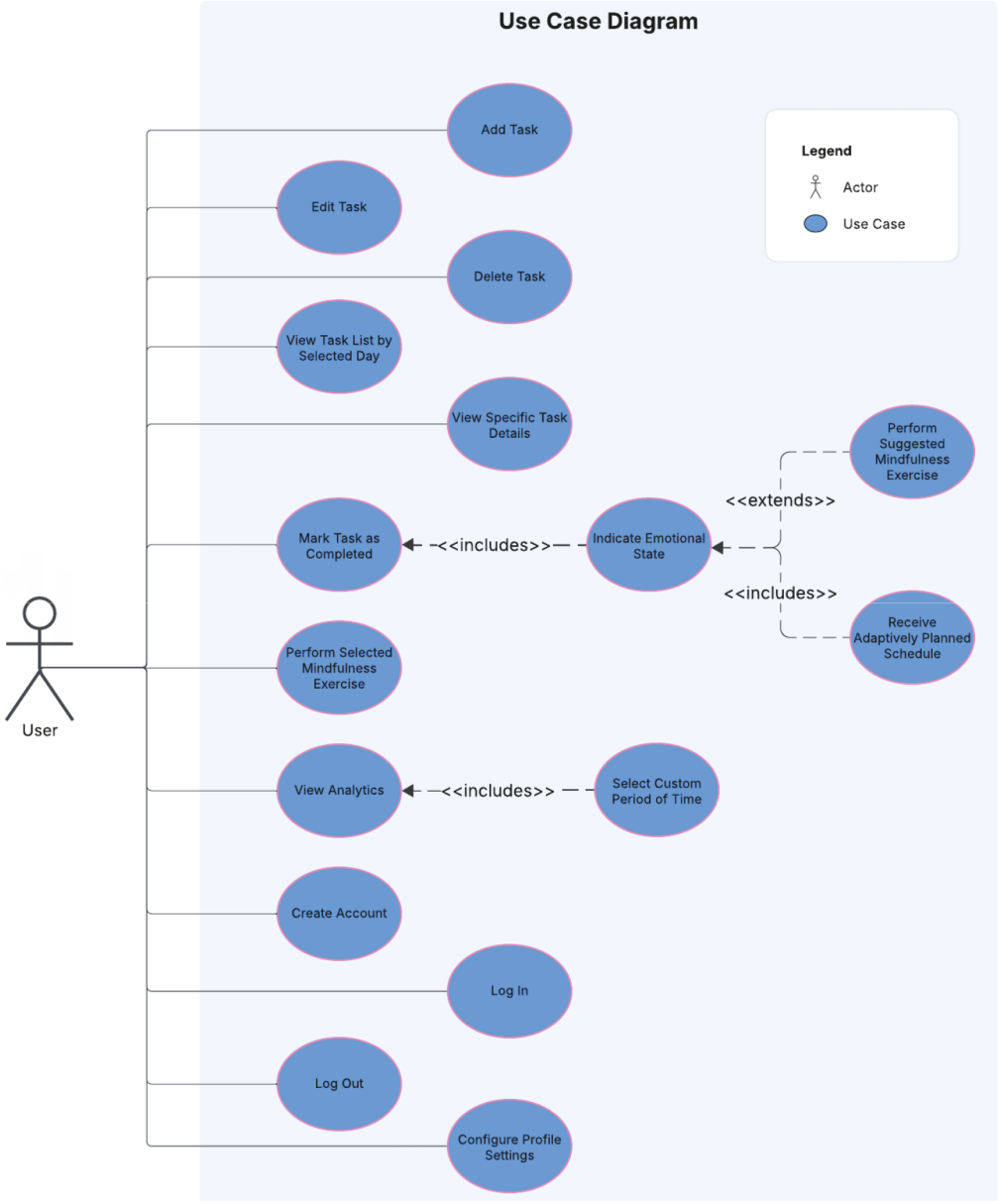


Figure 8. Use case diagram of a system

To help users avoid missing fixed tasks, the notification mechanism was implemented using the UserNotifications framework. The system sends local reminders 15 minutes before and at the start time of fixed tasks.

The app is localized into Ukrainian using the String Catalog, which is the preferred localization approach starting with Xcode 15 [1]. The app's language is automatically selected based on the iPhone's system language.

Conclusions

This study addressed the lack of integrated solutions that combine productivity with consideration of emotional state. A comparative analysis of existing applications showed that none of them offered adaptive planning based on a user's current emotions. This identified a gap and defined the direction for a new solution.

To bridge this gap, tasks were categorized as either flexible or fixed, emotions were quantified, and, finally, the adaptive planning algorithm was developed. This algorithm dynamically adjusts the order of flexible tasks based on the user's mood and tasks' attributes.

As a result, an iOS application was implemented using modern technologies for iOS development such as Swift, SwiftUI, and SwiftData. Code quality and format were ensured by SwiftLint and SwiftFormat. The project followed the MVVM architecture.

The application includes such core features as task management and planning, marking emotional state after completing a task, analyzing emotional trends across time and categories, and suggesting mindfulness exercises when needed. The key innovation of the system lies in the adaptive planning mechanism, which helps users stay productive while being mindful of their mental well-being. Additional features include account management, reminders for fixed tasks, and Ukrainian localization.

The results of this research can be applied in the domains of time management and mental health. Future development may be conducted by applying machine learning to personalize the algorithm based on individual patterns. Another promising direction is integration with health tracking solutions such as Apple HealthKit. This would enable the system to consider both mental and physical health data in the task planning process.

References

1. Apple Inc. Localizing and varying text with a string catalog [Electronic resource]. — Mode of access: https://developer.apple.com/documentation/xcode/localizing-and-varying-text-with-a-string-catalog (date of access: 25.06.2025). — Title from screen.
2. Apple Inc. Privacy [Electronic resource]. — Mode of access: https://www.apple.com/privacy (date of access: 25.06.2025). — Title from screen.
3. Bearable [Electronic resource]. — Mode of access: https://apps.apple.com/ua/app/bearable-symptom-tracker/id1482581097 (date of access: 25.06.2025). — Title from screen.
4. DailyBean [Electronic resource]. — Mode of access: https://apps.apple.com/us/app/dailybean-simplest-journal/id1553223828 (date of access: 25.06.2025). — Title from screen.
5. Dhruv S. SwiftUI vs. UIKit: iOS Development Comparison [Electronic resource] / Stuti Dhruv // Aalpha. — 15.04.2025. — Mode of access: https://www.aalpha.net/blog/swiftui-vs-uikit-comparison (date of access: 25.06.2025). — Title from screen.
6. Gronek J. MVC Swift vs MVVM on iOS: Key Differences With Examples [Electronic resource] / Jedrzej Gronek // Netguru. — 24.02.2025. — Mode of access: https://www.netguru.com/blog/mvc-vs-mvvm-on-ios-differences-with-examples (date of access: 25.06.2025). — Title from screen.
7. Hülsheger U. R. Benefits of Mindfulness at Work: The Role of Mindfulness in Emotion Regulation, Emotional Exhaustion, and Job Satisfaction [Electronic resource] / Ute R. Hülsheger, Hugo J. E. M. Alberts, Alina Feinholdt, Jonas W. B. Lang // Journal of Applied Psychology. — 2013. — Vol. 98, № 2. — C. 310-325. — Mode of access: https://doi.org/10.1037/a0031313.
8. Notion [Electronic resource]. — Mode of access: https://www.notion.com (date of access: 25.06.2025). — Title from screen.
9. Rafalski K. Top iOS Programming Languages for App Development in 2025 [Electronic resource] / Kacper Rafalski // Netguru. — 31.03.2025. — Mode of access: https://www.netguru.com/blog/top-ios-programming-languages (date of access: 25.06.2025). — Title from screen.
10. Shanmugam K. Core Data vs. Swift Data: A Detailed Comparison with Real-Time Solutions [Electronic resource] / Kaligoss Shanmugam // Medium. — 25.06.2024. — Mode of access: https://medium.com/@kalidoss.shanmugam/core-data-vs-swift-data-a-detailed-comparison-with-real-time-solutions-54ba8142c100 (date of access: 25.06.2025). — Title from screen.
11. SwiftlyNomad. CoreData vs Realm: What to Choose as a Database for iOS [Electronic resource] // Medium. — 13.05.2024. — Mode of access: https://swiftlynomad.medium.com/coredata-vs-realm-what-to-choose-as-a-database-for-ios-7c1d09911d8f (date of access: 25.06.2025). — Title from screen.
12. TickTick [Electronic resource]. — Mode of access: https://ticktick.com (date of access: 25.06.2025). — Title from screen.
13. Twarogal P., Hanzel S. 136% More People Have Androids Than iPhones [Electronic resource] / Paulina Twarogal, Szymon Hanzel // Neontri. — 04.02.2025. — Mode of access: https://neontri.com/blog/android-iphone-statistics-report (date of access: 25.06.2025). — Title from screen.

References

Apple Inc. *Localizing and varying text with a string catalog*. (n. d.). Apple Inc. https://developer.apple.com/documentation/xcode/localizing-and-varying-text-with-a-string-catalog.

Apple Inc. *Privacy*. (n. d.). Apple Inc. https://www.apple.com/privacy.

Bearable. (n. d.). Apple App Store. https://apps.apple.com/ua/app/bearable-symptom-tracker/id1482581097.

DailyBean. (n. d.). Apple App Store. https://apps.apple.com/us/app/dailybean-simplest-journal/id1553223828.

Dhruv S. (2025, April 15). *SwiftUI vs. UIKit: iOS Development Comparison*. Aalpha. https://www.aalpha.net/blog/swiftui-vs-uikit-comparison.

Gronek J. (2025, February 24). *MVC Swift vs MVVM on iOS: Key Differences With Examples*. Netguru. https://www.netguru.com/blog/mvc-vs-mvvm-on-ios-differences-with-examples.

Hülsheger U. R., Alberts H. J. E. M., Feinholdt A., Lang J. W. B. (2013*). Benefits of Mindfulness at Work: The Role of Mindfulness in Emotion Regulation, Emotional Exhaustion, and Job Satisfaction*. Journal of Applied Psychology, 98(2), 310–325. https://doi.org/10.1037/a0031313.

Notion. (n. d.). Notion. https://www.notion.com.

Rafalski, K. (2025, March 31). *Top iOS Programming Languages for App Development in 2025*. Netguru. https://www.netguru.com/blog/top-ios-programming-languages.

Shanmugam, K. (2024, June 25). *Core Data vs. Swift Data: A Detailed Comparison with Real-Time Solutions*. Medium. https://medium.com/@kalidoss.shanmugam/core-data-vs-swift-data-a-detailed-comparison-with-real-time-solutions-54ba8142c100.

SwiftlyNomad. (2024, May 13). *CoreData vs Realm: What to Choose as a Database for iOS*. Medium. https://swiftlynomad.medium.com/coredata-vs-realm-what-to-choose-as-a-database-for-ios-7c1d09911d8f.

TickTick. (n. d.). TickTick. https://ticktick.com.

Twarogal, P., & Hanzel, S. (2025, February 4). *136% More People Have Androids Than iPhones.* Neontri. https://neontri.com/blog/android-iphone-statistics-report.

Пізь М. А., Нагірна А. М.

РОЗРОБКА iOS-ЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ ЗАВДАНЬ З УРАХУВАННЯМ ЕМОЦІЙНОГО СТАНУ КОРИСТУВАЧА

У статті розглянуто створення цифрового інструмента для вирішення проблеми зниження продуктивності через емоційне виснаження. Метою дослідження є розробка iOS-застосунку для планування справ, який враховує емоційний стан користувача й відповідно до нього коригує графік справ і надає можливість виконувати вправи для емоційного усвідомлення й відновлення. У ході роботи проаналізовано наявні програмні рішення в нішах управління часом і ментального здоров'я, а також розроблено алгоритм адаптивного планування залежно від емоцій користувача та властивостей завдань.

Ключові слова: управління справами, продуктивність, емоції, адаптивне планування мобільний застосунок, iOS, Swift, SwiftUI, SwiftData.

Матеріал надійшов 28.06.2025

УДК 004.451.5
DOI: 10.18523/2617-3808.2025.8.205-212

Левченко А. С., Франків О. О., Петелєв Є. Р.

ДОСЛІДЖЕННЯ ТА ОПТИМІЗАЦІЯ МЕТОДІВ ОЦІНЮВАННЯ РОЗМІРУ ФАЙЛОВОЇ ІЄРАРХІЇ В APFS (APPLE FILE SYSTEM)

Цю статтю присвячено дослідженню та оптимізації процесів сканування файлової системи APFS (Apple File System). Розглянуто ключові інструменти доступу до APFS та алгоритмічні стратегії, зокрема верхньорівневий обхід, повний обхід, фільтрацію за стоп-словами та інтерактивний підхід. Реалізовано методи оброблення файлових ієрархій, які передбачають послідовне та паралельне оброблення з використанням Grand Central Dispatch (GCD) і Swift Concurrency. Розроблено застосунок для сканування APFS, який демонструє практичне застосування запропонованих підходів. Проведено тестування й порівняльний аналіз методів сканування APFS.

Ключові слова: Apple File System, APFS, сканування файлової системи, macOS.

Вступ

Файлові системи є невід'ємною складовою будь-якої сучасної операційної системи. Вони відіграють важливу роль у швидкому доступі до даних, надійності їх збереження та ефективному використанні простору. Apple File System (APFS) було представлено 14 червня 2016 р. як файлову систему наступного покоління для пристроїв Apple [12]. Вона прийшла на зміну Hierarchical File System Plus (HFS+), яку використовували з 1998 р. [3]. Станом на 2024 рік macOS є другою за популярністю десктопною операційною системою, а загальна частка операційних систем Apple на ринку становить близько 25 % [8].

Зростання популярності пристроїв Apple зумовлює потребу в глибшому розумінні внутрішніх механізмів роботи файлової системи цих пристроїв, а також оптимізації процесу її сканування, який лежить в основі таких задач, як резервне копіювання, індексація, антивірусне сканування, відновлення даних і цифрова криміналістика.

Apple File System принесла значні зміни, спрямовані на покращення доступу до даних, забезпечення їхньої цілісності, ефективності управління сховищем та підвищення безпеки. На відміну від HFS+, де кожен том має фіксований розмір і не ділиться вільним простором з іншими розділами, APFS запровадила спільне використання простору [11]. Це дозволяє кільком томам у контейнері динамічно розподіляти сховище, гнучко змінюючи розмір без необхідності переформатування (рис. 1).

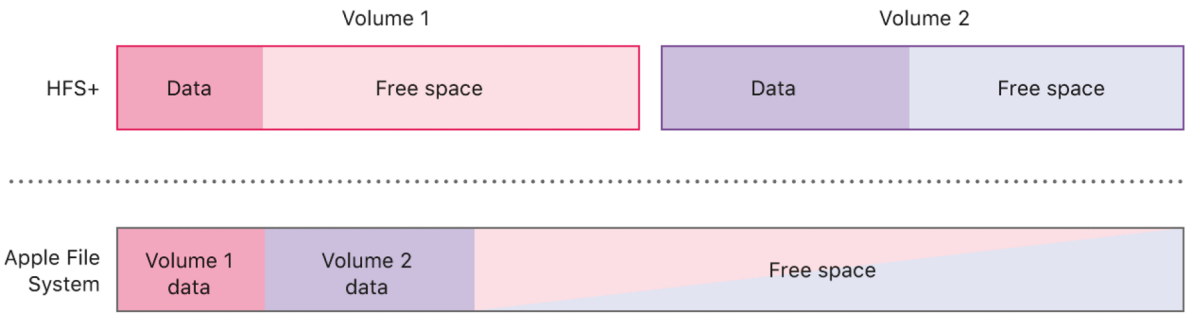


Рис. 1. Розподіл простору між томами в HFS+ та APFS

APFS використовує механізм copy-on-write, який гарантує, що дані не перезаписуються, поки зміни не буде повністю завершено. Це допомагає мінімізувати ризик пошкодження даних і підвищує

загальну стабільність файлової системи [1]. Для керування файлами та каталогами APFS використовує високооптимізовану структуру B-дерева, яка прискорює процеси копіювання, видалення та перейменування файлів [2].

Однією з ключових особливостей файлової системи APFS є можливість створення знімків (snapshots), які фіксують стан файлової системи у певний момент часу. Вони допомагають швидко відновити дані без використання додаткового місця для дублікатів даних [6].

Усі ці особливості Apple File System роблять її високопродуктивною та надійною файловою системою, але водночас ускладнюють процес її сканування. Саме тому оптимізація сканування APFS є важливою темою сучасних досліджень, оскільки швидкість і точність цього процесу безпосередньо впливають на загальну ефективність роботи системи. У цій статті ми зосередимося на аналізі методів і алгоритмів, які дають змогу максимально ефективно сканувати APFS, та визначимо оптимальне рішення.

Інструменти для роботи з APFS

У межах цього дослідження ми зосереджуємо увагу на визначенні розміру файлової ієрархії. У файловій системі APFS кожен елемент характеризується двома типами розмірів: логічним і фізичним. Логічний розмір (actual size) відображає обсяг даних, фактично записаних у файл або директорію, тоді як фізичний розмір (allocated size або size on disk) визначає кількість дискового простору, зарезервованого файловою системою для зберігання цього елемента. У нашому дослідженні для сканування APFS ми використовуватимемо саме фізичний розмір, оскільки він дає змогу точніше оцінити структуру та використання простору.

Для роботи з APFS існує декілька інструментів: NSFileManager, URLResourceKey і термінальна команда du. NSFileManager — це базовий API для роботи з Apple File System. Він забезпечує зручний доступ до файлів і каталогів. Метод attributesOfItem(atPath:) дозволяє отримати розмір файлу через атрибут '.size'. Однак для обчислення розміру директорії потрібно застосовувати рекурсивний обхід. Це робить процес сканування повільним і ресурсоємним, особливо для великих файлових структур. Крім того, за допомогою NSFileManager неможливо отримати дані про фізичний розмір файлів на диску.

URLResourceKey є більш сучасним підходом, який використовує об'єкти URL для безпечної роботи з файловою системою. Ключі fileSizeKey та fileAllocatedSizeKey дозволяють отримати логічний і фізичний розмір файлів відповідно. Для обчислення розміру директорій також потрібно застосовувати рекурсивний підхід, проте URLResourceKey оптимізує цей процес, опрацьовуючи лише необхідні атрибути. Це підвищує продуктивність сканування порівняно з NSFileManager. URLResourceKey зменшує кількість помилок, пов'язаних із неправильним форматом шляхів до елементів файлової системи, і є більш універсальним інструментом.

Термінальна команда du ("disk usage") ефективно оцінює використання дискового простору файлами та директоріями. Вона відображає інформацію у блоках, кожен з яких — це 512 байтів [9]. Команда du не потребує використання рекурсивного підходу для сканування файлової ієрархії. Також цей інструмент підтримує такі параметри, як -a (all — відображає розмір для кожного файла у ієрархії), -s (summarize — відображає загальний розмір файлу або директорії без деталізації по піддиректоріях), -d (depth — відображає розмір для всіх елементів лише на вказану глибину файлової структури) [10]. Для інтеграції цього інструменту в Swift використовується клас Process. Проте оброблення даних можливе лише після завершення виконання команди, і це може сповільнити процес сканування. Також під час використання цього інструменту виникають проблеми з доступом до певних директорій, що вимагає додаткового оброблення помилок.

Порівняльний аналіз інструментів для роботи з Apple File System демонструє, що NSFileManager зручний API для роботи з окремими файлами, але досить повільний для директорій. Команда du швидка у скануванні файлової ієрархії, але складна в інтеграції та обробленні. URLResourceKey — найефективніший інструмент для роботи з APFS, який поєднує швидкість, безпеку та гнучкість.

Алгоритми обходу файлової системи APFS

Для ефективного сканування Apple File System варто застосувати різні алгоритмічні стратегії для обходу файлової ієрархії. Вони допоможуть врахувати складність структури і мінімізувати зайві опе-

рації. Основні алгоритми, які застосовувались, передбачають верхньорівневий підхід, повний обхід, інтерактивний обхід і фільтрацію за стоп-словами.

Верхньорівневий підхід сканує лише елементи верхнього рівня. Це забезпечує високу швидкість завдяки мінімальним зверненням до файлової системи. Реалізація за допомогою команди du з параметром -d 1 забезпечує мінімальні витрати ресурсів. Однак NSFileManager та URLResourceKey потребують використання рекурсивного обходу, який знижує ефективність, особливо для NSFileManager. URLResourceKey залишається продуктивним завдяки оптимізованим запитам до APFS.

Повний обхід файлової системи рекурсивно сканує файлову ієрархію і надає детальну інформацію про розміри файлів та директорій. Команда du з параметром -a надає повний список елементів файлового дерева, а NSFileManager та URLResourceKey використовують ключі .size та fileAllocatedSizeKey відповідно. Цей метод більш точний, але час його виконання значний. Тому це може призвести до певних незручностей для користувача.

Інтерактивний обхід файлової системи дозволяє отримувати результати поступово та покращує взаємодію з користувачем. Цей метод реалізовано за допомогою AsyncStream [4]. Файли та директорії додаються в асинхронний потік, використовуючи метод yield(_:), а після завершення викликається метод finish(). Інтерактивний підхід дозволяє розпочати оброблення даних ще до завершення процесу сканування, а також робить сканування більш гнучким і зручним для користувача.

Фільтрація за стоп-словами оптимізує сканування APFS шляхом вилучення непотрібних елементів. Для її реалізації використовуються алгоритми DFS і BFS. Алгоритм DFS [5] виконує рекурсивний обхід директорій та пропускає елементи, що містять стоп-слова, та має вищу швидкість для глибоких структур. Алгоритм BFS [7] дозволяє швидше проаналізувати елементи верхніх рівнів файлової ієрархії, оскільки елементи файлової системи обробляються по рівнях за допомогою черги. На практиці DFS переважає BFS за швидкістю сканування APFS, але BFS усе ж є досить ефективним алгоритмом.

Отже, вибір алгоритму для сканування Apple File System залежить від задачі: верхньорівневий підхід — для швидкого аналізу, повний обхід — для детального аналізу, інтерактивний обхід — для зручності користувача, а фільтрація за стоп-словами з використанням DFS або BFS — для оптимізації ресурсів.

Багатопотокове оброблення файлової ієрархії APFS

Послідовне оброблення не завжди є ефективним, особливо якщо структура даних є досить складною та багаторівневою. Тому для підвищення швидкості сканування файлової системи APFS застосовується багатопотокове оброблення, реалізоване за допомогою Swift Concurrency та Grand Central Dispatch (GCD).

Послідовне оброблення виконує операції в одному потоці. Воно просте і зручне для невеликих директорій або окремих файлів. Також, оскільки немає потреби в управлінні потоками та синхронізації, послідовне оброблення легко відлагоджувати. Однак для складних файлових ієрархій цей метод є недостатньо швидким, що стало основною причиною для застосування методів багатопотокового оброблення.

Swift Concurrency впроваджений починаючи з версії Swift 5.5. Він використовує async/await, Task Groups та Actors для структурованого паралелізму. Task у Swift Concurrency — це асинхронні блоки коду, які виконуються без блокування основного потоку. Task Groups дозволяють створювати й керувати кількома завданнями, які виконуються паралельно. Функція withThrowingTaskGroup дозволяє створювати та керувати групами асинхронних завдань, які виконуються паралельно, одночасно обробляючи потенційні помилки. Actors запобігають проблемам типу data race, автоматично керуючи доступом до ресурсів, що усуває потребу в ручних блокуваннях, таких як NSLock, і знижує ризик взаємного блокування (deadlock).

Grand Central Dispatch (GCD) — це низькорівневий API для управління паралельними операціями. GCD керує виконанням завдань за допомогою DispatchQueue, які можуть бути послідовними та паралельними, що дозволяє ефективно розподіляти системні ресурси відповідно до поточного навантаження. Для координації виконання декількох задач використовується DispatchGroup, що дає можливість запускати асинхронні операції паралельно, але синхронізувати момент завершення групи. Функція withCheckedThrowingContinuation забезпечує інтеграцію GCD із сучасним підходом async/await. Хоча GCD є старішим механізмом, він і досі залишається ефективним.

Отже, для багатопотокового оброблення APFS Swift Concurrency є більш ефективним, ніж GCD, завдяки швидкості, безпеці та масштабованості. Для простих задач допускається використання послідовного оброблення.

Архітектура розробленого рішення

Розроблене рішення для сканування файлової системи APFS базується на модульній архітектурі, яка забезпечує гнучкість і масштабованість. Архітектура, зображена на рис. 2, включає ключові компоненти: протокол FS API для взаємодії з файловою системою через різні API (NSFileManager, URLResourceKey або термінальна команда du), модуль Concurrency для послідовного або багатопотокового оброблення елементів за допомогою Swift Concurrency або GCD. Також є модуль Techniques для стратегій обходу файлової ієрархії: верхньорівневої, повної, інтерактивної та фільтрації за стоп-словами з використанням алгоритмів DFS та BFS. Модульна структура дозволяє легко адаптувати рішення та змінювати алгоритми чи API залежно від вимог для забезпечення ефективного сканування Apple File System.

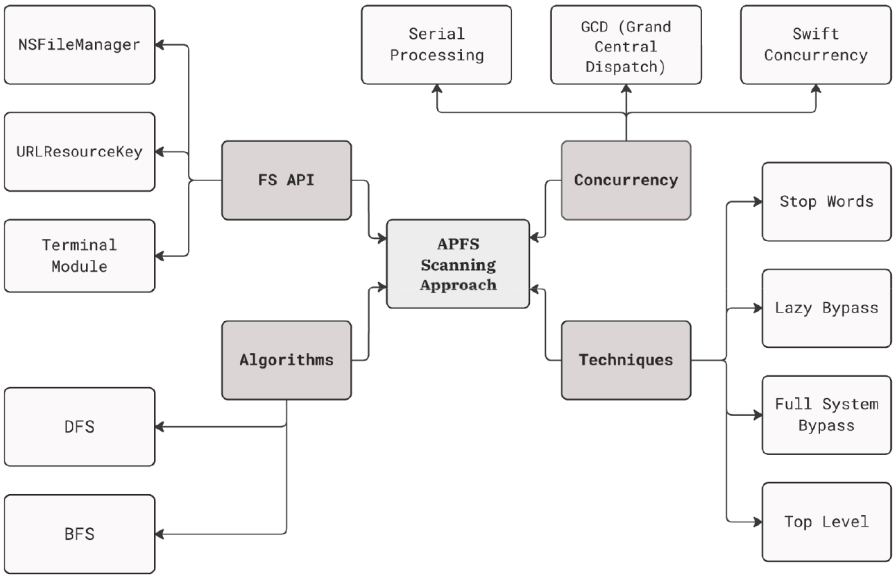


Рис. 2. Структура розробленого рішення

Оцінка ефективності

Для оцінки ефективності методів сканування APFS було проведено тестування з використанням спеціальних наборів даних, які моделюють різні файлові ієрархії. Це дало можливість проаналізувати продуктивність у типових і крайніх випадках. Зокрема було застосовано чотири шаблони:

- Breadth-First Structure (BFS) — шаблон для широких структур, який горизонтально розширюється перед поглибленням;
- Depth-First Structure (DFS) — шаблон, який моделює глибокі файлові структури;
- Balanced Tree Structure — шаблон для рівномірних і впорядкованих структур;
- Unbalanced Tree Structure — шаблон, який моделює непередбачувані та нерегулярні сценарії, які наближені до реальних користувацьких систем.

Тестування проводили на комп'ютері MacBook Air із чипом Apple M2 (8-ядерний процесор: 4 ядра продуктивності і 4 ядра ефективності), 16 GB оперативної пам'яті та SSD-накопичувачем, що працює з файловою системою APFS. Операційна система — macOS Sequoia 15.4.1. Кожен алгоритм випробовували на чотирьох шаблонах даних із різними розмірами файлів. Алгоритми оцінювали за часом виконання та піковим використанням оперативної пам'яті. Результати тестування проілюстровано графіками, які порівнюють ефективність різних методів сканування APFS.

Спочатку було проведено порівняння інструментів для сканування APFS за часом виконання на чотирьох типах файлових ієрархій: BFS, DFS, Balanced і Unbalanced (рис. 3). Графік відображає значні відмінності в продуктивності залежно від структури ієрархії. Найскладнішою виявилась BFS

структура через її широку організацію з великою кількістю дочірніх елементів, що ускладнює та подовжує сканування. Натомість найпростішою була Balanced Tree Structure завдяки своїй рівномірній і передбачуваній структурі. Найефективнішим інструментом став URLResourceKey, який значно перевершив NSFileManager і du для всіх типів ієрархій. Другою за продуктивністю є команда du, що показала стабільні результати, поступившись NSFileManager лише під час сканування Balanced структури.

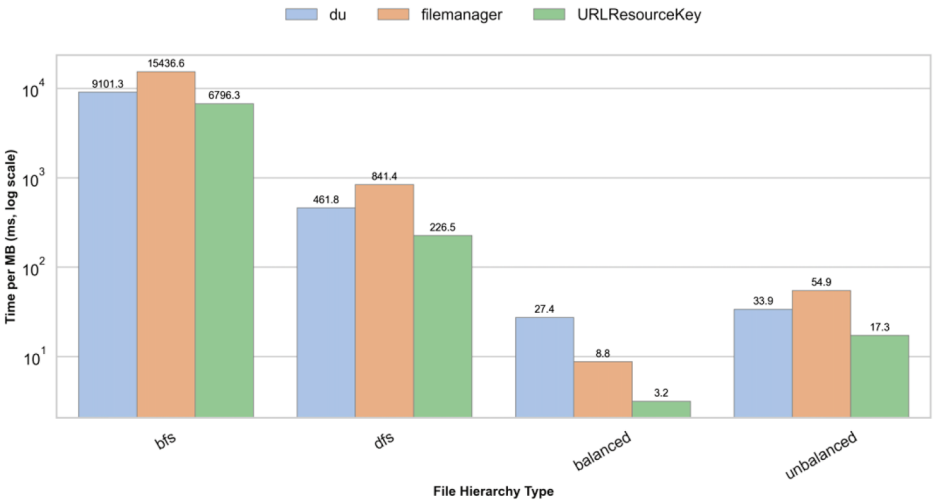


Рис. 3. Порівняння часу виконання сканування за допомогою різних інструментів для чотирьох типів файлових ієрархій

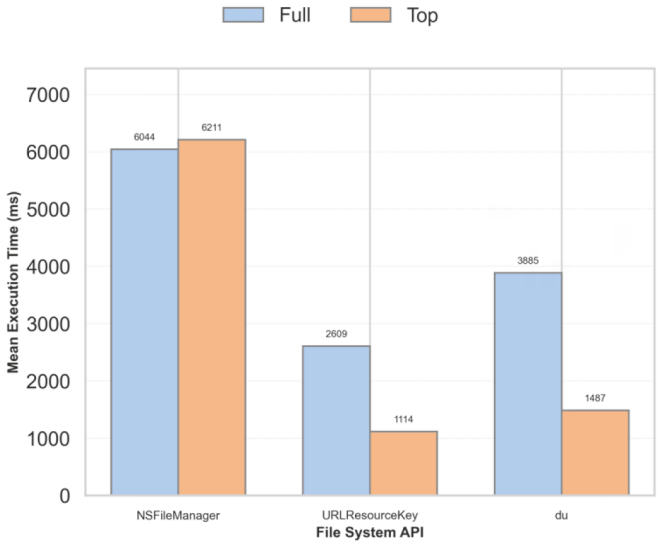


Рис. 4. Порівняння часу виконання сканування за допомогою повного та верхньорівневого підходу та різних інструментів

На рис. 4 подано діаграму, що порівнює середній час сканування за допомогою NSFileManager, URLResourceKey і команди du для верхньорівневого та повного обходу файлової ієрархії. Верхньорівневий підхід скорочує час сканування в 2–3 рази для URLResourceKey і du. Проте для NSFileManager продуктивність не змінюється через необхідність застосування рекурсивного обходу в обох методах.

Також було досліджено інтерактивний обхід і фільтрацію за стоп-словами, які показали помітне покращення ефективності сканування файлової системи. Інтерактивний підхід забезпечує більш швидкий доступ до перших результатів без втрати точності, що значно покращує користувацький досвід при роботі з великими ієрархіями. Фільтрація за стоп-словами, своєю чергою, зменшує час сканування та навантаження на системні ресурси, обробляючи лише необхідні елементи файлової ієрархії.

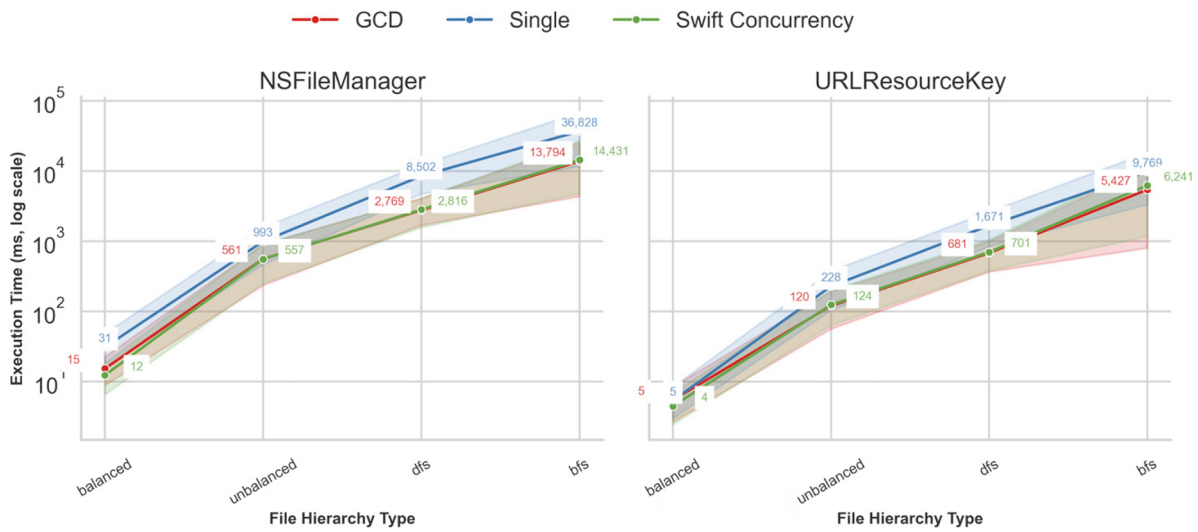


Рис. 5. Порівняння середнього часу виконання для послідовного оброблення, GCD і Swift Concurrency

Рисунок 5 демонструє графік порівняння багатопотокових підходів для чотирьох типів файлових структур, розділених на дві частини: ліва відображає результати для NSFileManager, а права — для URLResourceKey. Багатопотокові методи GCD та Swift Concurrency значно перевершують послідовне оброблення і прискорюють сканування APFS у 2–3 рази. Натомість GCD і Swift Concurrency демонструють майже однаковий рівень продуктивності.

Таблиця 1. **Середнє використання оперативної пам'яті різними багатопотоковими підходами та API**

File System API	GCD (MB)	Single (MB)	Swift concurrency (MB)
NSFileManager	222.0	226.5	241.1
URLResourceKey	96.4	70.2	113.9
du	59.9	43.5	65.7

Ще одним важливим параметром для оцінювання ефективності розробленого рішення було використання оперативної пам'яті під час сканування APFS. Таблиця 1 демонструє споживання пам'яті інструментами NSFileManager, URLResourceKey та командою du, а також різними підходами до багатопотокового оброблення. NSFileManager виявився найбільш ресурсоємним, тоді як команда du показала найменше споживання пам'яті. Серед багатопотокових методів оброблення Swift Concurrency споживає найбільше пам'яті, Grand Central Dispatch — дещо менше, а послідовне оброблення створює мінімальне навантаження. Хоча послідовне оброблення є найменш вимогливим до ресурсів, багатопотокові методи значно прискорюють сканування і компенсують додаткове використання пам'яті суттєвим підвищенням продуктивності.

Висновки

Для дослідження ефективності методів оптимізації сканування Apple File System було створено застосунок із модульною архітектурою, яка забезпечує гнучкість, масштабованість і можливість комбінувати різні підходи. Було проведено тестування на чотирьох типах штучних файлових ієрархій (BFS, DFS, Balanced, Unbalanced), що охоплюють як типові, так і крайні випадки файлових структур. Результати показали високу продуктивність і точність запропонованих методів.

Застосування паралельного оброблення за допомогою GCD і Swift Concurrency скоротило час сканування у 2–3 рази порівняно з послідовним обробленням файлових ієрархій. Особливо ефективною виявилась комбінація Swift Concurrency та URLResourceKey, яка показала найкращі результати для типових користувацьких структур. Реалізовані стратегії обходу файлової системи, зокрема верхньорівневий, повний, інтерактивний обхід і фільтрація за стоп-словами, оптимізують сканування APFS, забезпечуючи гнучке налаштування процесу під конкретні завдання.

Дослідження показало потенціал для подальшого вдосконалення, а запропоновані методи можуть бути застосовані не лише для оптимізації сканування APFS, а й для поглиблення розуміння управління даними в сучасних файлових системах macOS.

Список літератури

1. Apple Inc. Apple File System Guide [Electronic resource] / Apple Inc. — 2018. — Mode of access: https://developer.apple.com/library/archive/documentation/FileManagement/Conceptual/APFS_Guide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40016999-CH1-DontLinkElementID_15.
2. Apple Inc. Apple File System Reference [Electronic resource] / Apple Inc. — 2020. — Mode of access: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf.
3. Apple Inc. HFS Plus Volume Format / Apple Inc. — Technical Note TN1150, 2010.
4. Apple Inc. Streams, Sockets, and Ports [Electronic resource] / Apple Inc. — Mode of access: https://developer.apple.com/documentation/foundation/streams_sockets_and_ports.
5. Garg D. Analysis of the Depth First Search Algorithms / D. Garg, N. Kaur. — Thapar University, 2012.
6. Hansen K. H. Decoding the APFS file system [Electronic resource] / K. H. Hansen, F. Toolan // Digital Investigation. — 2017. — Vol. 22. — Pp. 107–132. — Mode of access: https://doi.org/10.1016/j.diin.2017.07.003.
7. Holdsworth H. The Nature of Breadth-First Search / H. Holdsworth. — School of Computer Science Mathematics and Physics, James Cook University, Australia, 1999.
8. Kosisochukwu H. U. Exploring operating system diversity: A comparative analysis of Windows, Mac OS, Android and IOS / H. U. Kosisochukwu, M. I. Abdullahi // Systematic and Modern Science Research. — 2024. — Vol. 5, no. 9. — Pp. 23–40.
9. Nordvik R. APFS [Electronic resource] / R. Nordvik // Mobile Forensics — The File Format Handbook. — 2022. — Pp. 3–39. — Mode of access: https://doi.org/10.1007/978-3-030-98467-0.
10. Platt D. Tweak Your Mac Terminal [Electronic resource] / D. Platt. — Berkeley (CA) : Apress, 2021. — Mode of access: https://doi.org/10.1007/978-1-4842-6171-2_1.
11. Rane R. Demystifying File Systems: A Comprehensive Exploration of Data Organization [Electronic resource] / R. Rane, A. Singh – 2024. — Mode of access: https://doi.org/10.13140/RG.2.2.31160.35845.
12. Tamura E. Introducing Apple File System / E. Tamura, D. Giampaolo. — 2016.

References

Apple Inc. (2018). *Apple File System Guide.* Apple Developer. https://developer.apple.com/library/archive/documentation/FileManagement/Conceptual/APFS_Guide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40016999-CH1-DontLinkElementID_15.

Apple Inc. (2020). *Apple File System Reference.* Apple Developer. https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf.

Apple Inc. (2010). *HFS Plus Volume Format — Technical Note TN1150*.

Apple Inc. (n. d.). *Streams, Sockets, and Ports.* Apple Developer Documentation. https://developer.apple.com/documentation/foundation/streams_sockets_and_ports.

Garg, D., & Kaur, N. (2012). *Analysis of the Depth First Search Algorithms.* Thapar University.

Hansen, K. H., & Toolan, F. (2017). Decoding the APFS file system. *Digital Investigation*, *22*, 107–132. https://doi.org/10.1016/j.diin.2017.07.003.

Holdsworth, H. (1999). *The Nature of Breadth-First Search.* School of Computer Science Mathematics and Physics, James Cook University.

Kosisochukwu, H. U., & Abdullahi, M. I. (2024). Exploring operating system diversity: A comparative analysis of Windows, Mac OS, Android and IOS. *Systematic and Modern Science Research (JSMSR)*, *5* (9), 23–40.

Nordvik, R. (2022). APFS. In *Mobile Forensics — The File Format Handbook* (pp. 3–39). Springer. https://doi.org/10.1007/978-3-030-98467-0.

Platt, D. (2021). *Tweak Your Mac Terminal*. Apress. https://doi.org/10.1007/978-1-4842-6171-2_1.

Rane, R., & Singh, A. (2024). *Demystifying File Systems: A Comprehensive Exploration of Data Organization. https://doi.org/10.13140/RG.2.2.31160.35845*.

Tamura, E., & Giampaolo, D. (2016). *Introducing Apple File System*.

A. Levchenko, O. Frankiv, Y. Peteliev

INVESTIGATION AND OPTIMIZATION OF FILE HIERARCHY SIZE ESTIMATION METHODS IN APFS (APPLE FILE SYSTEM)

File systems are an integral part of modern operating systems, providing the foundation for organizing, storing, and accessing data. The efficiency of a file system plays a critical role in determining software performance, particularly when handling large volumes of data. This study focuses on the research and analysis of optimization methods for scanning the Apple File System (APFS), a modern file system developed by Apple to enhance data access, integrity, and storage management. APFS introduces advanced features such as shared space allocation, B-tree structures, and support for snapshots, which, while improving performance, also pose challenges for efficient scanning.

The article explores a range of scanning strategies, including top-level traversal, full system bypass, interactive bypass, and stop-word filtering, as well as serial and parallel processing approaches using Swift Concurrency and Grand Central Dispatch (GCD). Various tools for accessing APFS, such as NSFileManager, URLResourceKey, and du, were utilized to facilitate this analysis. To enable a systematic evaluation of these methods across various file hierarchies, a specialized tool for scanning APFS was developed. The research aims to assess key performance aspects such as speed, scalability, and resource utilization, offering insights into optimizing APFS scanning for improved efficiency.

Testing was conducted on four types of file hierarchies: Breadth-First Structure (BFS), Depth-First Structure (DFS), Balanced Tree Structure, and Unbalanced Tree Structure. The results demonstrated the effectiveness of the proposed methods, highlighting their ability to adapt to different structural complexities while maintaining high performance. This validation underscores the practical utility of the developed tool and the potential for these optimization techniques to enhance APFS scanning in real-world applications.

Keywords: Apple File System, APFS, file system scanning, macOS.

Матеріал надійшов 19.06.2025

УДК 004.777:[811.161.2:801.66]
DOI: 10.18523/2617-3808.2025.8.213-224

Смиш О. Р., Швець Д. В.

АНАЛІЗУВАННЯ УКРАЇНСЬКОМОВНИХ ВІРШІВ ЗАСОБАМИ ОБРОБКИ ПРИРОДНОЇ МОВИ

У статті описано розроблення та застосування парсера для комплексного аналізування українськомовних віршів. Створено механізми визначення віршового розміру, рим і способів римування, що ґрунтуються на побудові схем наголошування. Парсер дає змогу виявляти входження 16 художніх засобів на трьох рівнях оброблення тексту. На фонологічному рівні застосовано rule-based підходи в аналізуванні тексту, на морфосинтаксичному — оброблення даних формату CoNLL-U, сформованих із результатів роботи трьох мовних аналізаторів, а для виявлення тропів на семантичному рівні — використано велику мовну модель.

Створено вебзастосунок для наочної взаємодії з функціями парсера. Розроблений інструмент застосовано для аналізування трьох українськомовних збірок віршів.

Ключові слова: обробка природної мови, парсер, мовний аналізатор, великі мовні моделі, вебзастосунок, українська мова, вірш, аналіз вірша, троп, художній засіб, віршовий розмір, рима, спосіб римування.

Вступ

У сучасних умовах інтенсивного зростання обсягів текстової інформації та стрімкого розвитку технологій обробка природної мови стає одним із провідних напрямів у галузі інформатики та комп'ютерної лінгвістики. Сьогодні близько 30 млн осіб знають українську мову і вважають її рідною [15], тож існує потреба в інструментах для оброблення та різнобічного аналізування українськомовних текстів.

Необхідною частиною освітнього процесу в Україні є вивчення художньої літератури, зокрема поезії. У програмі зовнішнього незалежного оцінювання (ЗНО) з української мови та літератури зазначено, що учні повинні знати та розуміти поняття «ліричний вірш» і «художні засоби» [3]. Тому необхідно, щоб учителі української літератури надавали учням достатню кількість прикладів аналізу віршів.

Перед студентами гуманітарних спеціальностей, філологами, літературознавцями й іншими фахівцями, що досліджують фольклор і різноманітні епохи та течії в українській поезії, виникає потреба в обробленні значного обсягу віршованих текстів, що вимагає часу та зусиль. Проте наразі немає готових рішень, які б уможливили програмне аналізування віршів, написаних українською мовою.

Розроблений парсер дає змогу здійснювати різнобічне аналізування українськомовних віршів, а саме:

- визначати віршовий розмір (метр) кожного рядка та всього вірша;
- визначати риму та способи римування рядків;
- виявляти входження 16 тропів: алітерація, анафора, асонанс, гіпербола, епітет, епіфора, інверсія, літота, метафора, оксиморон, паралелізм, персоніфікація, порівняння, рефрен, риторичне звертання, риторичне питання.

Основні терміни в аналізі віршів

Автоматизоване аналізування вірша потребує комплексного підходу, що охоплює звукову та смислову структуру поетичного тексту. Через метр і риму організовано ритміку тексту та його звукову впорядкованість. Натомість тропи унаочнюють змістову й естетичну наповненість тексту, художнє мислення автора. Ці компоненти можна визначити однозначно, на відміну від теми вірша, емоційного забарвлення слів тощо. Об'єднання їх дає змогу різнобічно характеризувати поетичний

текст, уможливлюючи подальше порівнювання віршів, визначення закономірностей у межах поетичної збірки чи в усій творчості поета, глобальне аналізування поезії певного мистецького напряму чи епохи.

Стопа — визначальний метричний період, найкоротший відрізок співмірності певного віршового метра, сконцентрованого в групі складів із відносно незмінним наголосом [2, с. 435]. Із наголошених і ненаголошених складів сформовано стопи, а зі стоп — ритмічну схему, за якою можна визначити метр вірша. Наголошені склади позначають символом «–», а ненаголошені — «∪».

Таблиця. **Стопи**

Назва	Схема	Наголошено склад
Хорей	– ∪	перший із двох
Ямб	∪ –	другий із двох
Дактиль	– ∪ ∪	перший із трьох
Амфібрахій	∪ – ∪	другий із трьох
Анапест	∪ ∪ –	третій із трьох

Віршовий розмір, або метр — особливість стопи, покладеної в основу певної віршової структури, що фіксує відповідний тактометричний період [1, с. 192]. До основних розмірів належать двоскладові (хорей і ямб) і трискладові (дактиль, амфібрахій, анапест) стопи [2, с. 387]. Віршовий розмір складається з кількості повторення стопи та її назви: п'ятистопний хорей, тристопний анапест тощо.

Рима — композиційно-звуковий засіб суголосся закінчень, що має фонетичне та метричне значення, об'єднує останні слова суміжних або близько розташованих віршових рядків (починаючи з останнього наголошеного складу) [2, с. 322]. Рядки є римованими, якщо їхні останні наголошені склади містять однакові голосні звуки (або близькі за звучанням звуки [и] й [і]) та розташовані на однаковій відстані до кінця рядка.

Римування — особливість розташування рим у вірші, інтервал між ними. Суміжне римування означає, що римовані слова розташовані в сусідніх рядках, перехресне — через один рядок (у другому та четвертому, п'ятому та сьомому тощо). У кільцевому римуванні два римовані рядки розташовані через два рядки, які також між собою римовані [2, с. 324].

Художні засоби (тропи) — сукупність зображально-виражальних засобів, прийомів, способів діяльності письменника, за допомогою яких він творить нову художню дійсність [2, с. 565].

Визначення віршового розміру

Сформовано програмний механізм визначення віршового розміру:

1) розставити наголоси в тексті вірша;
2) створити список списків булевих значень, де True позначає наголошені склади, а False — ненаголошені;
3) визначити для кожного рядка, скільки разів схема якої стопи в ньому повторюється;
4) визначити найчастішу стопу серед рядків і перевірити для рядків, у яких визначена інакша стопа (або не визначено жодної), чи підходить найчастіша стопа їм;
5) скоригувати створений у п. 2 список, якщо деякі наголоси суперечать визначеній схемі.

Для розставлення наголосів у тексті вірша використано Python-пакет Ukrainian Word Stress [8]. У цьому модулі передбачено оброблення омографів, можна обрати стратегію наголошування слів із паралельними наголосами. Вирішено використовувати стратегію розставлення всіх можливих паралельних наголосів, оскільки на етапі коригування залишаються лише ті, що не суперечать визначеній схемі.

Для кожного булевого списку, що відповідає складам одного рядка вірша, застосовано метод, який перевіряє перші n елементів (складів), де або n кратне двом чи трьом (оскільки стопа може бути двоскладовою чи трискладовою), або останній з n наголосів є наголошеним. Ці n елементів мають бути рівними n елементам списку, сформованого відповідно до однієї зі схем стоп. Якщо це так, то кількість стоп дорівнює частці від цілочисельного ділення n на довжину стопи, інакше розмір є невизначеним.

У віршах деякі слова можуть мати наголос, відмінний від нормативного. Окрім того, емпірично виявлено, що модуль для наголошування не завжди правильно розставляє наголоси. Із цих причин реалізовано методи для перевірки віршового розміру та коригування наголосів.

Після формування схем наголошення безпосередньо з наголошеного тексту, програма визначає стопу, що відповідає найбільшій кількості рядків. Якщо для деяких рядків визначено іншу стопу або не визначено жодної, програма перевіряє, чи відповідає такий рядок найчастішій стопі.

Якщо принаймні половина схем стоп у рядку відповідає схемі найчастішої стопи, відбувається коригування наголосів у цьому рядку. Для двоскладових стоп відбувається вилучення наголосів, що суперечать схемі, через що можуть утворитися стопи без наголошених складів. Двоскладова стопа без наголошених складів — пірихій — не суперечить схемам з ямбом і хореєм [2, с. 216]. Проте в трискладових стопах пропуски наголошених складів недопустимі, оскільки так виникає по п'ять ненаголошених складів поспіль. Тому для коригування наголосів у рядках із трискладовими стопами програма перестворює їхні схеми наголошування.

Визначення рим і способів римування

Для кожного рядка програма створює його репрезентацію, що є кортежем (a, b, c, d), де a — від'ємний індекс останнього наголошеного складу (останнього значення True в булевому списку наголошених і ненаголошених складів), b — голосний звук останнього наголошеного складу, c — решта голосних звуків після останнього наголошеного, d — усі голосні звуки рядка.

Рими між рядками позначено списком пар їхніх індексів. Пошук рим за їхніми репрезентаціями відбувається ітеративно зі зменшенням строгості зіставлення репрезентацій. Значення строгості спадає на кожній ітерації на одиницю від 3 до 1. Якщо для більш ніж половини рядків не знайдено рим після пошуку зі строгістю 2, програма визначає цей вірш таким, що не є римованим, тобто є верлібром [1, с. 165].

При строгості зі значенням 3 програма визначає римованими лише ті рядки, у репрезентаціях яких збігаються a, b і c; зі значенням 2 — a і b. Строгість зі значенням 1 передбачена для випадків, коли в одному з двох римованих рядків програма не поставила наголос на склад, який насправді є останнім наголошеним. Тоді для кожної пари зі списку рядків, для яких не знайдено рими, програмний механізм знаходить найбільший індекс останніх наголошених складів (і далі до кінця рядка) та перевіряє, чи збігаються в них голосні за цим індексом. Якщо так, то ці рядки є римованими, елементам із відповідним індексом у булевих списках римування цих рядків надається значення True.

Щоби визначити види римування для груп римованих рядків, використано словник, у який програма додає пари індексів римованих рядків відповідно до їхнього розташування у вірші.

Виявлення художніх засобів

Для аналізування парсером обрано тропи, що входять до списку художніх засобів у програмі ЗНО з української мови та літератури [3]: алітерація, анафора, асонанс, гіпербола, епітет, епіфора, інверсія, літота, метафора, оксиморон, паралелізм, персоніфікація, порівняння, рефрен, риторичне звертання, риторичне питання.

Наведені 16 тропів розділено на три групи, що відповідають певним рівням NLP [10], на яких можна визначити тропи відповідної групи. Алітерацію, анафору, асонанс, епіфору, рефрен і риторичне питання можна виявити на фонологічному рівні аналізування тексту. На морфосинтаксичному рівні можна визначити епітет, інверсію, паралелізм, порівняння та риторичне звертання. Решта тропів — гіпербола, літота, метафора, оксиморон і персоніфікація — стосуються семантичного рівня.

Щоб уніфікувати вигляд результатів аналізування тропів і точно вказувати їхні розташування в тексті вірша, знайдені входження тропів вирішено представляти як діапазони (a, b), де a — індекс символу в тексті вірша, де починається входження тропа (охопно), b — індекс символу, де закінчується входження тропа (не охопно). Тобто якщо перший символ входження має індекс 7, а останній — 14, то діапазоном є (7, 15).

Тропи на фонологічному рівні

На фонологічному рівні аналізування тексту існують три види правил: фонетичні, фонемні та просодичні [10], тобто на цьому рівні достатньо застосовувати правилозалежні (rule-based) підходи.

Алітерація [1, с. 53] і асонанс [1, с. 101] є стилістичними прийомами, що полягають у повторенні однакових звуків: алітерація — приголосних, асонанс — голосних. Для виявлення цих тропів засто-

совано однаковий підхід, що базується на підрахунку кількості входжень відповідних літер у тексті вірша, де йотовані голосні замінені відповідними голосними звуками (я → а, ю → у тощо). Якщо кількість однакових приголосних у рядку не менша за 3, то програма визначає такі символи як входження асонансу. Аналогічно з алітераціями, проте для них мінімальною кількістю повторів є 4, оскільки кількість різних голосних звуків в українській мові менша за кількість приголосних.

Рефрен є композиційним прийомом, що полягає в повторенні групи слів, рядка або кількох віршових рядків у строфах [2, с. 318]. Метод виявлення рефренів реалізує механізм, оснований на пошуку слів або їхніх послідовностей, які повторюються в тексті.

Аналізування починається з токенізації — виокремлення слів, для кожного з яких програма фіксує діапазони розташування у вихідному тексті. Це дає змогу не лише встановити факт повторення, а й точно визначити позицію кожного входження. Наступним кроком є побудова словника, де кожному унікальному слову відповідає список індексів його входжень. Після цього програма аналізує лише ті слова, які входять у текст більше ніж один раз.

Для кожного з повторів допоміжна функція визначає підмножину індексів, що позначає дубльовані слова, сусідні слова яких також повторюються. Ідея полягає в тому, щоби знайти ті входження слова, у яких найближче сусіднє слово (зліва або справа) збігається з аналогічним елементом в інших входженнях того самого слова. Такий підхід дає змогу вилучити дублікати одного слова без повтору принаймні одного сусіднього.

Анафора [1, с. 66] і епіфора [1, с. 342] — це стилістичні фігури, що полягають у повторенні однакових слів або зворотів на початку (анафора) чи наприкінці (епіфора) рядків. Застосовано метод, який визначає позиції перших і останніх літер кожного рядка відповідно. Далі використано метод виявлення рефренів, у який передано список індексів перших (для анафор) або останніх (для епіфор) літер у рядках. Це змінює поведінку методу визначення рефренів так, що він повертає входження повторів, які розташовані на початку чи в кінці рядків відповідно.

Риторичне питання — це стилістична фігура, виражена питанням, яке не потребує відповіді [2, с. 329]. Риторичне питання має бути словами автора (не належати героєві твору), тобто бути поза прямою мовою.

Питальні речення в українській мові позначають знаком питання, тож використано підхід знаходження таких речень за умови, що речення не починається з тире та не розташоване між парою лапок.

Тропи на морфосинтаксичному рівні

На морфосинтаксичному рівні аналізування тексту основну увагу зосереджено на морфологічних ознаках слів, граматичних зв'язках між ними та синтаксичній структурі речень. Для виявлення тропів на цьому рівні використано залежні від синтаксичних зв'язків правила, що спираються на розбір тексту у форматі CoNLL-U.

CoNLL-U (або Universal Dependencies CoNLL format) — це формат представлення лінгвістично розмічених корпусів, який використано в рамках проєкту UD для зберігання синтаксичних дерев і морфологічної інформації про слова. Він є частиною ініціативи CoNLL (Conference on Computational Natural Language Learning) для забезпечення уніфікованого підходу до оброблення мовних даних [5].

Дані CoNLL-U про вірш вирішено формувати на основі результатів оброблення тексту вірша трьома мовними аналізаторами: UDPipe [14], Stanza [13] та spaCy [12]. Для кожного слова програма обирає ті варіанти значень стовпців, які збігаються принаймні для двох аналізаторів, щоби запобігти ситуації, де один аналізатор неправильно визначає певні ознаки слів.

Епітет — це троп, виражений переважно прикметниками, що образно наголошує на характерній ознаці, одиничній якості певного предмета або явища. У ролі епітета може вживатися також іменник [1, с. 342]. Для його виявлення парсер шукає слова, які є прикметниками (*ADJ*) або прикладками (залежність *appos*).

Інверсія — стилістична фігура художнього мовлення, що полягає в порушенні прямого порядку слів у реченні [1, с. 419]. Для виявлення інверсій програма шукає інверсійні розміщення слів (див. рис. 1).

Паралелізм — аналогія, уподібнення, спільність ознак або дій, художній і композиційний прийом, що полягає в зіставленні або протиставленні двох чи кількох подій. Зазвичай паралелізм виражений однорідними синтаксичними конструкціями [2, с. 182]: присудками в межах однієї граматич-

ної основи або сурядними частинами. Метод визначення паралелізмів шукає однорідні конструкції (залежність *parataxis*), де послідовні речення або частини речення містять присудки з однаковим способом дії (*Mood*): дійсний, умовний або наказовий. Програма доповнює виявлені групи зв'язаними частками, якщо такі наявні для охоплення всього входження тропа.

```
if (tok[F.deprel] in ('nsubj', 'amod', 'det',
                      'advmod:det', 'det:numgov', 'det:nummod', 'nummod', 'nummod:gov')
        and tok[F.head] < tok[F.id]):
    inversions.append((head_of(tok, sent)[F.misc], tok[F.misc]))
elif (tok[F.deprel] in ('obj', 'iobj', 'obl', 'nmod', 'acl:relcl',
                        'xcomp', 'xcomp:pred')
      and tok[F.upos] in ('NOUN', 'PROPN', 'PRON')
      and tok[F.head] > tok[F.id]):
    adp = [t[F.misc] for t in sent[:tok[F.id]]
           if t and t[F.deprel] == 'case' and t[F.head] == tok[F.id]]
    if adp:
        inversions.append((adp[-1], tok[F.misc], head_of(tok, sent)[F.misc]))
    else:
        inversions.append((tok[F.misc], head_of(tok, sent)[F.misc]))
```

Рис. 1. Фрагмент методу виявлення інверсій

Порівняння — троп, що полягає в поясненні одного предмета через подібний до нього інший, зіставленні їх за допомогою компаративної зв'язки, тобто єднальних сполучників: *як*, *наче*, *неначе*, *мов*, *немов*, *мовби*, *немовби*, *ніби*, *нібито* [2, с. 248]. Програма шукає такі сполучники, а також залежність *advcl*, що в CoNLL-U позначає розгорнуті обставини, до яких належать порівняльні звороти, після чого поєднує пов'язані слова з відповідним сполучником.

Риторичне звертання — стилістична фігура, прийом звертання в якій ужитий для надання мовленню необхідної інтонації [2, с. 329]. Парсер знаходить цей троп як іменник у кличному відмінку (ознака *Case* має значення *Voc*) або за залежністю *vocative*, що в CoNLL-U позначає звертання.

Тропи на семантичному рівні

Семантичний рівень аналізування передбачає інтерпретацію значення тексту та виявлення смислових зв'язків. На цьому рівні реалізовано виявлення тропів, значення яких неможливо встановити за формальними ознаками, а лише через осмислення контексту.

```
У поданому тексті знайди всі входження вказаних тропів, тільки якщо такі наявні, і дай відповідь
чітко у форматі JSON без додаткової інформації:
{results: {metaphor: [метафори], personification: [персоніфікації], hyperbole: [гіперболи],
litotes: [літоти], oxymoron: [оксиморони]}}
Метафора — троп, що полягає в перенесенні значення одного слова (словосполучення) на
інше, розкрити сутності одних явищ чи предметів
через інші за схожістю і контрастом.
Персоніфікація — троп, що полягає в наданні предметам чи явищам рис живих істот.
Гіпербола — троп, для якого характерне надмірне перебільшення особливостей чи ознак предмета,
явища або дії.
Літота — троп, що полягає в художньому зменшенні величини, сили, значення зображуваного предмета
чи явища.
Оксиморон — троп, особливістю якого є поєднання контрастних, протилежних за значенням слів,
внаслідок чого утворюється парадоксальна семантична сполука.
Кожне входження має бути взяте з тексту без розривання тексту.

Текст:
Синиця в шибку вдарила крильми.
Годинник став. Сіріють німо стіни.
Над сизим смутком ранньої зими
Принишкли хмари, мов копиці сіна.
Пливе печаль. Біліють смолоскипи
Грайливо пофарбованих ялин —
Вони стоять, немов у червні липи,
Забрівши в сивий і густий полин.
Полин снігів повзе до видноколу,
Лоскоче обрій запахом гірким.
Лапаті, білі і колючі бджоли
Неквапно кружеляють понад ним...
```

Рис. 2. Приклад сформованого запиту для виявлення тропів у вірші В. Симоненка «З вікна»

На відміну від попередніх рівнів, тут застосовано велику мовну модель Claude 3.5 Sonnet, яка здатна інтерпретувати художні засоби, спираючись на попереднє тренування на великих корпусах текстів. Для цього програма формує запит (prompt) до LLM, у якому зазначено текст вірша та перелік тропів, які потрібно знайти. До вказаних у запиті тропів програма додає їхні визначення: метафора [1, с. 35], персоніфікація [2, с. 208], гіпербола [1, с. 227], літота [1, с. 581], оксиморон [2, с. 149]. Модель має надати відповідь строго у форматі JSON, де для кожного вказаного тропа наведено список його входжень. Приклад сформованого запиту наведено на рис. 2. У запиті також указано, що кожне входження має бути взято з тексту, не розриваючи його.

Для під'єднання до LLM використано Python-модуль g4f. Він містить засоби для взаємодії з переліком LLM [9]. Щоби запобігти галюцинаціям, кожен запит надсилається в окремому чаті.

Отриману відповідь програма перетворює на словник, після чого шукає вказані входження в тексті та замінює їх відповідними діапазонами індексів. Це потрібно для однакового представлення входжень усіх тропів, які виявляє парсер, а також для перевірки, що надані моделлю входження дійсно наявні в тексті вірша.

Створення вебзастосунку

Розроблення вебзастосунку для автоматичного аналізування українськомовних віршів зумовлено необхідністю створення інтерфейсу доступу до функцій парсера.

Серверна частина вебзастосунку реалізована з використанням вебфреймворку FastAPI, що забезпечує асинхронне оброблення HTTP-запитів і підтримання типізованого обміну даними через моделі Pydantic [6]. Ця частина відповідає за отримання тексту користувача, виклик відповідних модулів аналізування вірша та формування структурованої відповіді. Реалізовано два основні маршрути.

Перший маршрут — «/tropes» із методом POST — приймає в тілі запиту текст вірша та список тропів, які потрібно виявити. Згідно з рівнем аналізування, програма обирає відповідний клас для кожного тропа, створює по одному екземпляру класу для однієї категорії тропів і викликає потрібні методи. Після завершення аналізування тексту та виявлення тропів, програма об'єднує результати в один об'єкт і повертає його клієнтові у форматі JSON.

Другий маршрут — «/metre-rhyme» із методом POST — приймає в тілі запиту текст вірша та повертає результати аналізування віршового розміру, рими та способів римування. У відповіді клієнт отримує:

- бінарну схему наголошування (де 1 — наголошений склад, 0 — ненаголошений);
- представлення віршового розміру у форматі словника, де ключами на верхньому рівні є скорочені назви стоп, їхніми значеннями — об'єкти з ключами, що відповідають кількості стопи, та значеннями, представленими як списки індексів рядків (індекс відповідає номерові рядка в тексті);
- представлення способів римування у форматі словника, де ключами є назви способів римування, а їхніми значеннями — списки з парами індексів римованих рядків. Якщо деякі рими не відповідають жодному з трьох способів римування, то вони розміщені в списку за ключем «rest».

Клієнтська частина вебзастосунку реалізована з використанням бібліотеки React, що дає змогу створювати динамічний односторінковий інтерфейс із реактивною зміною станів. Основне завдання клієнтської частини полягає в збиранні вхідних даних від користувача, формуванні HTTP-запитів до серверної частини й інтерактивному відображенні результатів аналізування тропів, віршового розміру та римування.

Інтерфейс поділено на дві основні вкладки: «Тропи» та «Метр і рима». У першій вкладці користувач має змогу (див. рис. 3):

- увести текст вірша;
- обрати один або кілька тропів для виявлення;
- запустити виявлення тропів;
- переглядати результати через візуально виділені фрагменти тексту — кожен троп марковано відповідним кольором;
- перемикатися між кнопками тропів для фокусування на кожному з них (сірі кнопки позначають тропи, яких не виявлено в тексті).

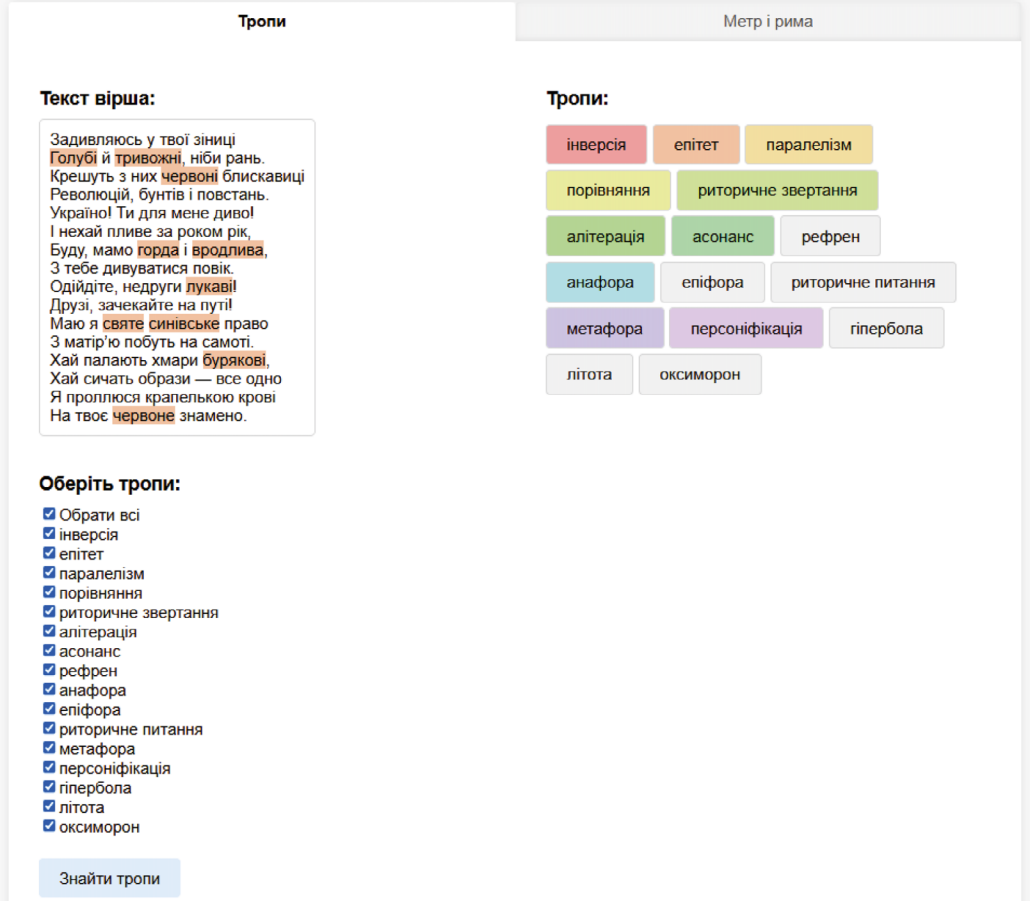


Рис. 3. Вкладка «Тропи» (вірш «Задивляюсь у твої зіниці...» В. Симоненка)

У вкладці «Метр і рима» відображаються результати аналізування віршового розміру та римування (див. рис. 4). Користувач може ввести текст вірша та натиснути відповідну кнопку для аналізування. На лівій половині вікна, праворуч від тексту розміщені кольорові позначки римованих рядків (між собою римовані ті рядки, що мають однакові позначки). Під текстом вірша — знайдені види римування відповідних кольорів. На правій половині — аналіз метра: схема наголошування, метр кожного рядка, основний віршовий розмір — тобто такий, що відповідає більшості рядків вірша.

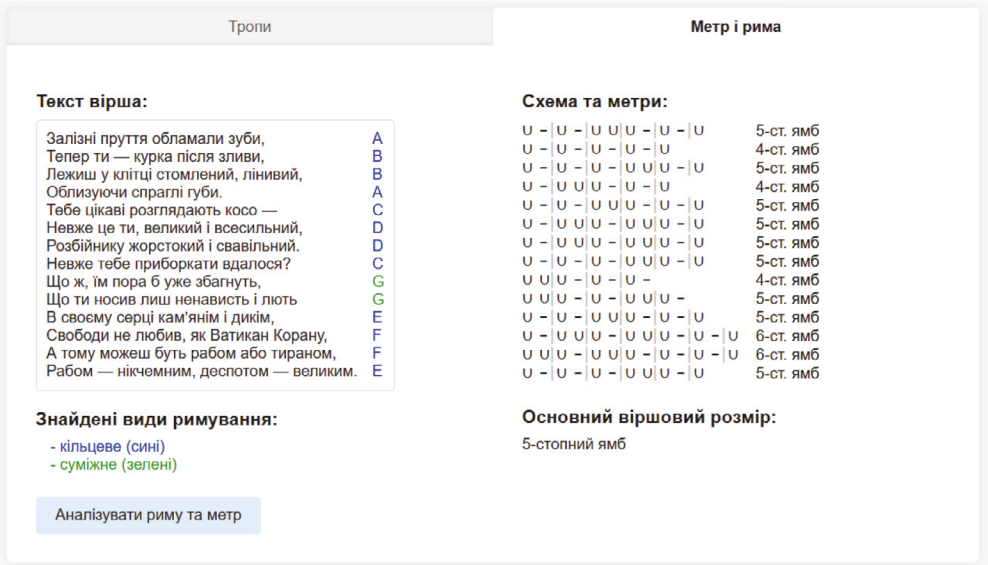


Рис. 4. Вкладка «Метр і рима» (вірш «Лев у клітці» В. Симоненка)

Точність аналізування віршів парсером і великими мовними моделями

Щоби надати оцінку точності визначення віршового розміру (метра) та рими парсером, сформовано збірку тестів, представлену як таблицю, де стовпцями є: текст вірша; назва стопи; кількість стоп; список пар індексів римованих рядків.

Зі 107 текстів віршів парсер правильно визначив метр (тобто назву стопи та її кількість одночасно) для 101 тексту, а риму — для 103. Тому точність визначення метра парсером становить 94,39 %, а рими — 96,26 %.

Щоб оцінити точність виявлення тропів парсером, створено збірку тестів у форматі таблиці, де стовпцями є: текст вірша або його фрагмент; назва тропа, який потрібно виявити; слово чи набір слів, що є входженням тропа в тексті. Збірка містить загалом 1808 таких тестів. Результати тестування наведено на рис. 5.

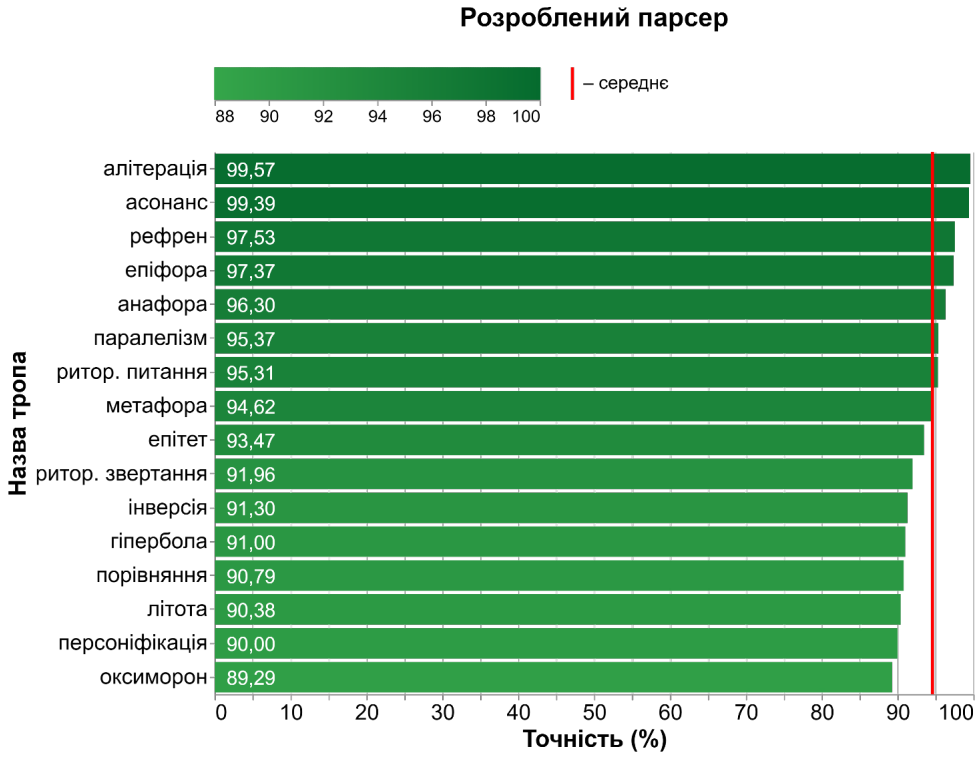


Рис. 5. Точність виявлення тропів розробленим парсером

З огляду на широкі можливості LLM, вирішено оцінити їхню здатність визначати метр і риму, а також виявляти художні засоби (тропи) в українськомовних віршах. У рамках дослідження протестовано три мовні моделі: ChatGPT-4 [11], Claude 3.5 Sonnet [4] і Gemini 2.0 Flash [7].

На основі результатів тестування великих мовних моделей і розробленого парсера на точність виявлення тропів, визначення метра та рими, складено графіки порівняння точностей, зображені на рис. 6.

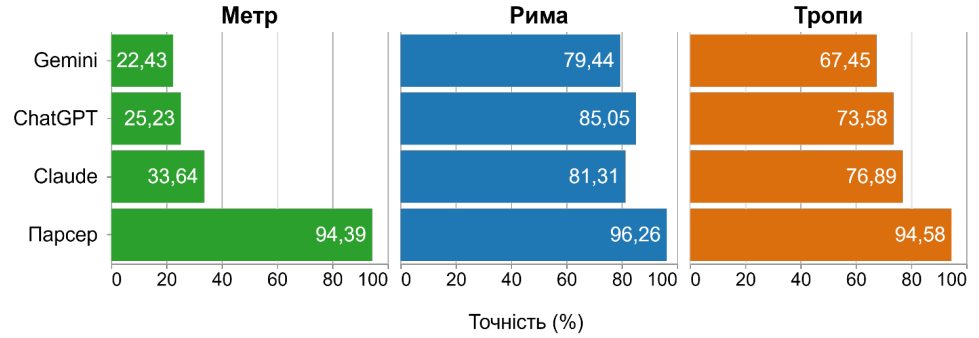


Рис. 6. Графіки точностей визначення метра, рими і тропів великими мовними моделями та розробленим парсером

На рисунку можна побачити відсоткові значення точностей. Розроблений парсер має точність визначення метра вірша на 67,26 % краще за середнє значення для LLM, рими — на 14,33 %, а тропів — на 21,94 %.

Застосування парсера для аналізування збірок віршів

Щоби наочно продемонструвати можливості розробленого парсера, з його використанням проаналізовано українськомовні вірші з трьох поетичних збірок: «Кобзар» Т. Шевченка — 66 віршів, «Відгуки» Лесі Українки — 25 віршів, «Земне тяжіння» В. Симоненка — 52 вірші.

На рисунку 7 зображено розподіл віршів зі збірки «Кобзар» за роками написання з урахуванням метричних стоп. Можна бачити, що до 1850 р. поет використовував різні стопи: переважно двоскладові (ямб і хорей), проте наявні також два вірші, написані анапестом, і один — амфібрахієм. Після перерви, з 1857 р. основною стопою у віршах Шевченка є лише ямб.

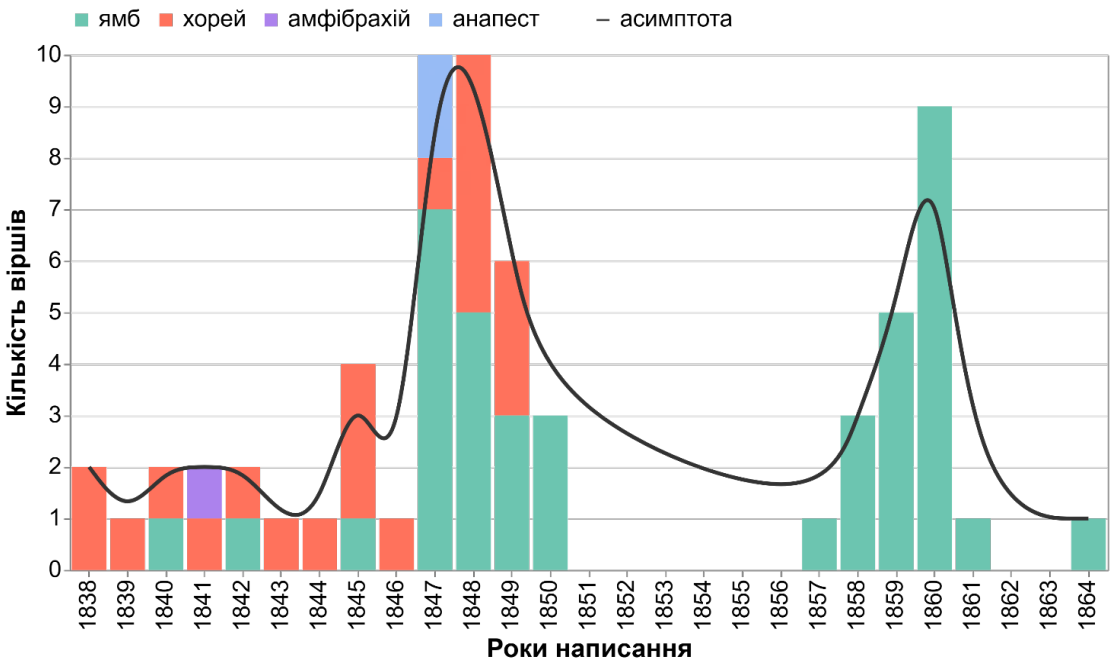


Рис. 7. Розподіл віршів зі збірки «Кобзар» за роками написання з урахуванням стоп

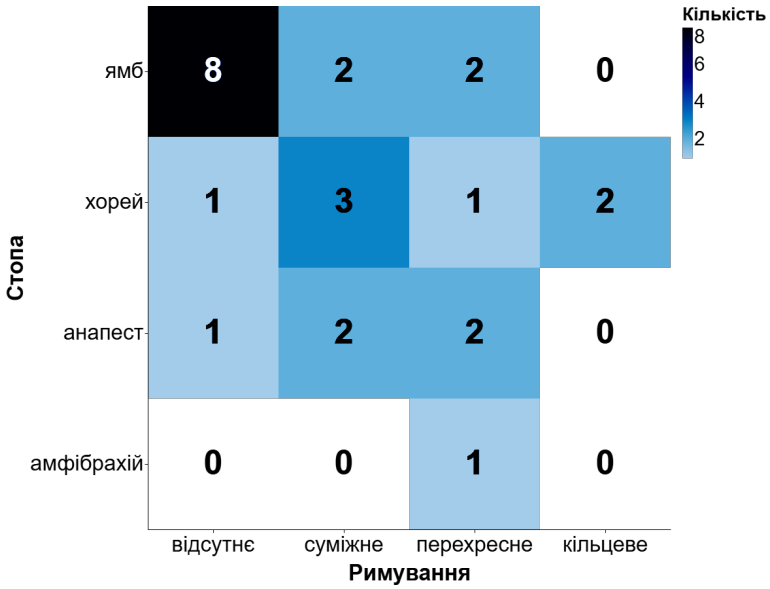


Рис. 8. Згруповані за використаною стопою та способом римування вірші зі збірки «Відгуки»

Наступною візуалізацією, поданою на рис. 8, є теплова мапа, де відображено кількості віршів зі збірки «Відгуки», згруповані за використаною стопою та способом римування. Найбільшу кількість віршів у збірці (8) написано ямбом без римування.

Для кожної з трьох проаналізованих збірок створено двовимірні KDE-графіки щільності розподілу тропів.

Для кожного вірша збірки, за допомогою парсера, визначено індекси символів, що відповідають входженням тропів, після чого координати цих ділянок нормалізовано від 0 до 100 у двох вимірах. Горизонтальна вісь відповідає позиції тропа в межах кожного рядка, а вертикальна — положенню рядка у вірші. У результаті для всієї сукупності віршів відповідної збірки сформовано список точок, що вказують розміщення тропів у тексті. На основі таких списків побудовано графіки, що відображають щільність розподілу тропів на площині, утвореній нашаруванням віршів збірки. Мапа щільності дає змогу виявити тенденції в композиційному розміщенні художніх засобів.

Першим на рис. 9 зображено графік для збірки «Кобзар». Можна побачити, що в правій верхній частині графіка є ділянка, де щільність точок найвища. Це свідчить про те, що в збірці в більшості віршів значна частина тропів розташована в кінці перших рядків.

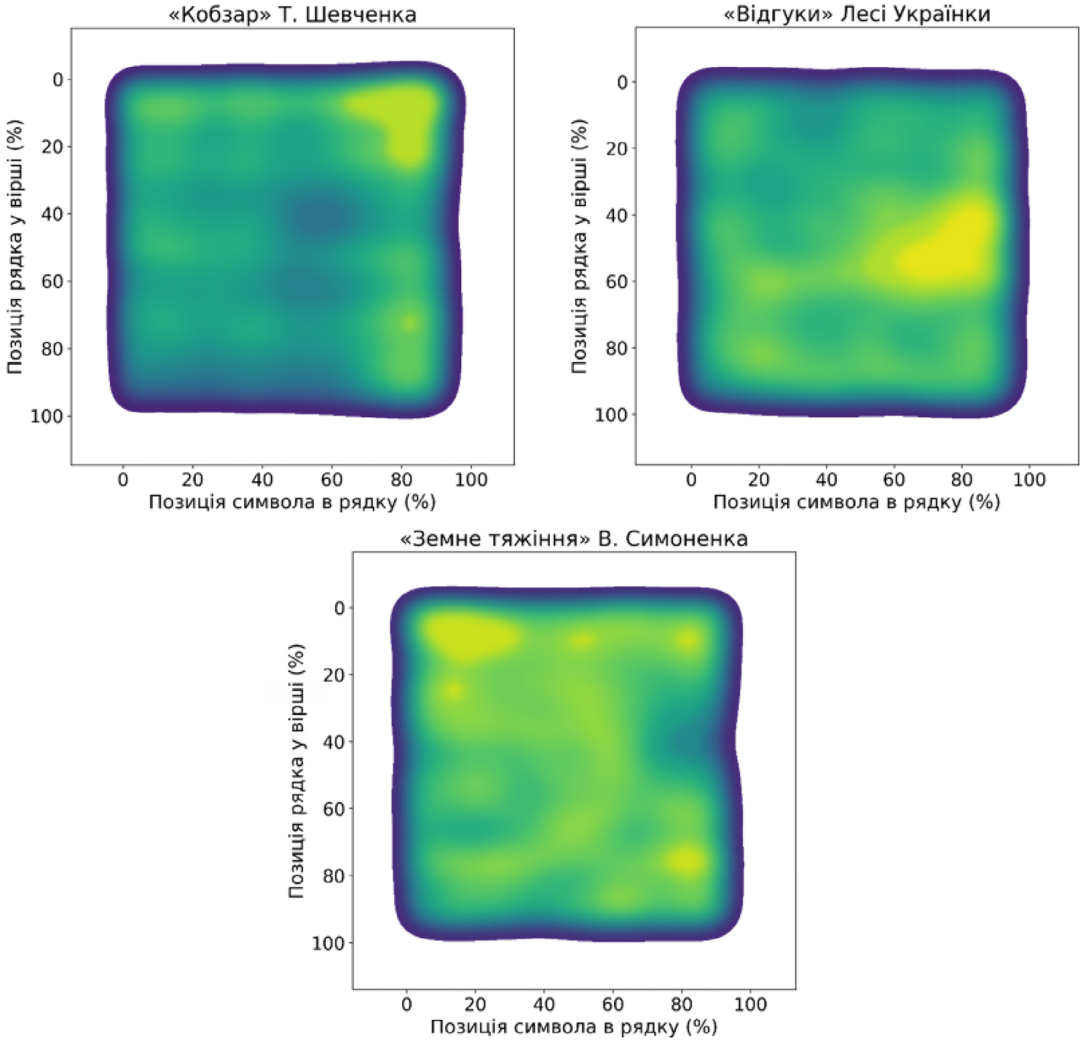


Рис. 9. KDE-графіки щільності розподілу тропів у проаналізованих збірках

На графіку для збірки «Відгуки» Лесі Українки видно, що найбільше тропів зосереджено в кінці середніх рядків віршів.

Якщо подивитися на KDE-графік для збірки «Земне тяжіння», можна помітити, що тропи у віршах збірки розташовані більш рівномірно, ніж у двох інших, а також те, що поет використовував тропи переважно на початку перших рядків вірша.

Висновок

У статті описано розроблення та застосування парсера для визначення метра, рими, способів римування та виявлення художніх засобів (тропів) в українськомовних віршах.

Запропоновано механізми визначення віршового розміру, рим і способів римування, що ґрунтуються на побудові схем наголошування. Передбачено методи коригування цих схем. Розроблено методи виявлення тропів на трьох рівнях оброблення тексту: фонологічному, морфосинтаксичному та семантичному. На фонологічному рівні застосовано правилозалежні підходи в аналізуванні тексту, на морфосинтаксичному — оброблення даних формату CoNLL-U, сформованих із результатів роботи трьох мовних аналізаторів, а для виявлення тропів на семантичному рівні — використано велику мовну модель. Реалізовано вебзастосунок, що уможливлює візуалізацію результатів роботи парсера, з використанням FastAPI та React.

Здійснено оцінювання точності парсера на основі сформованих збірок тестів. За результатами тестування встановлено точності компонентів парсера: 94,39 % — визначення метра, 96,26 % — визначення рими, 94,58 % — виявлення тропів. Порівняння з результатами роботи LLM показало вищу точність створеного інструмента. Також у статті оглянуто застосування парсера для аналізування віршів із трьох поетичних збірок та наведено візуалізації, що унаочнюють отримані результати.

Отже, розроблений парсер уможливлює комплексне аналізування українськомовних віршів, охоплюючи визначення віршового розміру, рим, способів римування та виявлення художніх засобів.

Список літератури

1. Ковалів Ю. І. Літературознавча енциклопедія : у двох томах. Т. 1 / Ю. І. Ковалів. — Київ : ВЦ «Академія», 2007. — 608 с.
2. Ковалів Ю. І. Літературознавча енциклопедія : у двох томах. Т. 2 / Ю. І. Ковалів. — Київ : ВЦ «Академія», 2007. — 624 с.
3. Програма ЗНО з української мови і літератури 2025 року [Електронний ресурс] // Освіта.UA. — Режим доступу: https://osvita.ua/test/program_zno/946.
4. Claude [Electronic resource] // Anthropic. — Mode of access: https://claude.ai.
5. CoNLL-U Format [Electronic resource] // Universal Dependencies. — Mode of access: https://universaldependencies.org/format.html.
6. FastAPI [Electronic resource] // FastAPI. — Mode of access: https://fastapi.tiangolo.com.
7. Gemini [Electronic resource] // Google AI. — Mode of access: https://gemini.google.com.
8. GitHub - lang-uk/ukrainian-word-stress: Adds word stress to Ukrainian texts [Electronic resource] // GitHub. — Mode of access: https://github.com/lang-uk/ukrainian-word-stress.
9. GitHub - techwithanirudh/g4f: The official gpt4free repository // GitHub. — Mode of access: https://github.com/techwithanirudh/g4f.
10. Liddy E. Natural Language Processing / E. Liddy // Library and Information Science. — 2nd ed. — NY, 2001.
11. OpenAI. GPT-4 Technical Report [Electronic resource] / OpenAI // Computation and Language. — 2023. — Mode of access: https://doi.org/10.48550/arXiv.2303.08774.
12. SpaCy · Industrial-strength Natural Language Processing in Python [Electronic resource] // SpaCy. — Mode of access: https://spacy.io.
13. Stanza Overview [Electronic resource] // Stanza. — Mode of access: https://stanfordnlp.github.io/stanza.
14. UDPipe [Electronic resource] // LINDAT/CLARIAH-CZ. — Mode of access: https://lindat.mff.cuni.cz/services/udpipe.
15. Ukrainian speaking countries [Electronic resource] // Worlddata.info. — Mode of access: https://www.worlddata.info/languages/ukrainian.php.

References

Claude. Anthropic. https://claude.ai.

CoNLL-U Format. Universal Dependencies. https://universaldependencies.org/format.html.

FastAPI. https://fastapi.tiangolo.com.

Gemini. Google AI. https://gemini.google.com.

GitHub - lang-word-stress: uk/ukrainianAdds word stress to Ukrainian texts. GitHub. https://github.com/lang-uk/ukrainian-word-stress.

GitHub - techwithanirudh/gpt4free: The official gpt4free. GitHub. https://github.com/techwithanirudh/gpt4free.

Kovaliv, Yu. I. (2007a). *Literaturoznavcha entsyklopediia* (T. 1). VTs “Akademiia” [in Ukrainian].

Kovaliv, Yu. I. (2007b). *Literaturoznavcha entsyklopediia* (T. 1). VTs “Akademiia” [in Ukrainian].

Liddy, E. (2001). Natural Language Processing. *Library and Information Science* (2nd ed.). Marcel Decker, Inc, NY, USA.

OpenAI. (2023). GPT-4 Technical Report. *Computation and Language*. https://doi.org/10.48550/arXiv.2303.08774.

Prohrama ZNO z ukrainskoi movy i literatury 2025 roku. (2025). Osvita.UA. https://osvita.ua/test/program_zno/946 [in Ukrainian].

SpaCy · Industrial-strength Natural Language Processing in Python. SpaCy. https://spacy.io.

Stanza Overview. Stanza. https://stanfordnlp.github.io/stanza.

UDPipe. CZ. LINDAT / CLARIAH. https://lindat.mff.cuni.cz/services/udpipe.

Ukrainian speaking countries. Worlddata.info. https://www.worlddata.info/languages/ukrainian.php.

O. Smysh, D. Shvets

UKRAINIAN POETRY ANALYSIS USING NLP METHODS

This paper presents the development and usage of a parser designed for the comprehensive analysis of Ukrainian-language poetry. The parser integrates techniques from natural language processing to determine metrical patterns, identify rhymes and rhyme schemes, and detect the presence of 16 tropes across three levels of linguistic processing: phonological, morphosyntactic, and semantic.

At the phonological level, rule-based methods are employed to detect stylistic devices such as alliteration, assonance, anaphora, epiphora, refrain, and rhetorical questions. For morphosyntactic analysis, the parser processes CoNLL-U formatted data generated by three language analysers (UDPipe, Stanza, spaCy) to ensure robustness in identifying epithets, inversions, parallelisms, similes, and rhetorical appeals. Semantic-level analysis is conducted using a large language model (LLM), Claude 3.5 Sonnet, which interprets metaphors, hyperbole, litotes, personification, and oxymora.

The parser includes a metrical analysis module that identifies stressed syllables and constructs rhythmic patterns to classify each poetic line into one of the canonical meters (iamb, trochee, dactyl, amphibrach, or anapest). It also identifies rhyme pairs and classifies rhyme schemes as tail, ring, or internal.

A web application was developed using FastAPI and React, enabling users to input poems and interactively visualize analytical results. This tool has been applied to three canonical Ukrainian poetry collections — by Taras Shevchenko, Lesia Ukrainka, and Vasyl Symonenko — demonstrating its capacity to uncover stylistic patterns across different periods and poetic styles.

Evaluation results show high accuracy: 94.39% for metre detection, 96.26% for rhyme identification, and 94.58% for trope recognition, outperforming three reviewed LLMs. The parser enables nuanced analysis of Ukrainian poetic texts, supporting educational, philological, and literary research.

Keywords: natural language processing, parser, language analyser, large language models, web application, Ukrainian language, poem, poem analysis, trope, metre, rhyme, rhyme scheme.

Матеріал надійшов 19.06.2025

УДК 004.415.3
DOI: 10.18523/2617-3808.2025.8.225-231

Ліпський Д. О.

АРХІТЕКТУРА НОВОЇ ВДОСКОНАЛЕНОЇ ПЛАТФОРМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБЗАСТОСУНКІВ

Вдосконалення автоматизації тестування вебзастосунків є особливо актуальним напрямом у сучасному процесі розроблення програмного забезпечення. У цій статті здійснено аналіз сучасних підходів та інструментів автоматизації тестування, їхніх переваг та недоліків. Розглянуто шляхи усунення цих недоліків, а також можливість інтеграції технологій штучного інтелекту.

У статті представлено архітектуру запропонованої платформи автоматизованого тестування, реалізованої у формі бібліотеки, що легко інтегрується в наявні проєкти. Архітектуру побудовано за модульним принципом, що забезпечує гнучкість, масштабованість і можливість поетапного розширення функціональності. Основні компоненти платформи — конфігураційний модуль, менеджер драйверів, модулі взаємодії з елементами, логування, API-викликів та роботи з локальним сховищем браузера — працюють як єдиний узгоджений механізм, забезпечуючи прозоре, стабільне та ефективне виконання тестів.

Окрему увагу приділено аналізу таких аспектів сучасних технологій автоматизації, як зниження вартості впровадження та підтримки тестових рішень, масштабованість, гнучкість налаштувань, а також інтеграція з іншими компонентами життєвого циклу програмного забезпечення.

Ключові слова: автоматизація, вебтехнології, платформа, тестування.

Вступ

Завдання тестування вебзастосунків полягає у створенні та виконанні тестових сценаріїв на вебплатформах за допомогою спеціалізованого програмного забезпечення (ПЗ). Така технологія перевірки правильності ПЗ спрямована на виявлення помилок і невідповідностей до вимог у вебзастосунках [1]. Автоматизація дозволяє виконувати повторювані тестові процедури без залучення тестувальника, що значно підвищує ефективність перевірки якості та надійності програмних продуктів [2].

Основні компоненти автоматизованого тестування охоплюють: визначення тестових даних і наперед заданих умов, встановлення очікуваних результатів і безпосереднього виконання тестових сценаріїв. Це забезпечує виявлення та вирішення таких проблем у вебзастосунках, як функціональне тестування, перевірка навантаження, безпеки, користувацького інтерфейсу та інтеграції систем [5].

Під час функціонального тестування перевіряють вибрані функції вебзастосунку через імітацію реальних сценаріїв користувача, включаючи тестування форм, аутентифікації та пошукових запитів [1]. Також здійснюється навантажувальне тестування для оцінки стійкості застосунку під великою кількістю користувачів, використовуючи віртуальне середовище для генерації численних запитів до системи [5].

Безпека є однією з критичних областей, де тестування зосереджене на виявленні потенційних уразливостей, таких як SQL ін’єкції і крос-сайтовий скриптинг [4]. Це забезпечує захист вебзастосунку від шкідливих дій. Тестування користувацького інтерфейсу перевіряє, що візуальні елементи добре інтегровані та інтуїтивно зрозумілі для користувачів, тоді як інтеграційне тестування забезпечує, що всі модулі системи ефективно працюють разом [5].

Серед сучасних інструментів для автоматизації тестування, таких як Selenium, QTP і Ranorex, розробники можуть вибрати ті, що підтримують широкий спектр мов програмування та можливість інтеграції з різними середовищами розробки [2]. Ці інструменти забезпечують гнучкість і адаптивність, необхідні для адекватного реагування на специфічні вимоги проєкту. Однак наявні методи і технології автоматизації тестування лише частково покращують якість вебзастосунків і знижують можливі ризики. Тому актуальними є розроблення і впровадження нових, більш ефективних рішень на основі AI [5].

Переваги та недоліки сучасних платформ автоматизованого тестування

Автоматизація тестування забезпечує численні переваги, які значно покращують процес розроблення програмного забезпечення: збільшення швидкості виконання тестів, підвищення точності, економія ресурсів та покращення покриття тестування [1].

Одна з найважливіших переваг автоматизації тестування — здатність швидко проводити великі обсяги тестів, виконуючи тестові сценарії значно оперативніше, ніж це могли б зробити люди. Це призводить до прискорення циклу розробки, оскільки зворотний зв'язок отримується майже відразу, що дає змогу швидше вносити корективи [5].

Автоматизація вивільняє цінні ресурси, зокрема час тестувальників, які тепер можуть бути спрямовані на більш складні стратегічні завдання. Замість витрат часу на рутинне виконання тестів тестувальники можуть зосередитися на аналізі складних випадків, вдосконаленні тестових сценаріїв та оптимізації загальної стратегії якості [5].

Автоматизація мінімізує помилки тестувальників, що часто трапляються при ручному виконанні через втому або неуважність. Кожен автоматизований тест виконується однаково при кожному запуску, забезпечуючи стабільність та повторюваність результатів, що гарантує виявлення та усунення помилок до того, як продукт досягне кінцевого користувача [2].

Оскільки автоматизація потребує меншого втручання людей, витрати на трудові ресурси зменшуються, роблячи процес тестування більш економічно вигідним. Автоматизація також дає можливість працювати неперервно, що збільшує загальний обсяг виконаних робіт за певний період [5].

Автоматизація дає змогу включати у тестування значно більшу кількість тестових сценаріїв, ніж у ручному режимі, що забезпечує ширше покриття і детальнішу перевірку програмного забезпечення [5]. Використання штучного інтелекту (AI) в платформах для автоматизації тестування вебзастосунків є ключовим напрямком підвищення якості, оскільки AI та машинне навчання (ML) дають можливість розширити можливості традиційного автоматизованого тестування, забезпечуючи більшу точність, ефективність і гнучкість процесів тестування [5].

Інтеграція AI в автоматизацію тестування сприяє суттєвому покращенню різних аспектів цього процесу. По-перше, AI підвищує ефективність тестування за рахунок автоматичного аналізу результатів тестів, допомагаючи ідентифікувати закономірності та передбачати потенційні проблеми, зменшуючи потребу в ручному аналізі [1]. Важливим є також оптимізований вибір тестових сценаріїв: використовуючи методи ML, платформа може визначати, які сценарії найкраще підходять для конкретних змін у коді, що робить тестування більш ефективним і цілеспрямованим [4].

Крім того, штучний інтелект сприяє автоматичному виявленню та класифікації дефектів. Це допомагає швидко знаходити помилки у програмному забезпеченні та оцінювати їх серйозність і тип, що значно спрощує процес виправлення [5].

Нарешті, існуючі AI рішення аналізують історичні дані тестування, виявляючи слабкі місця в тестових сценаріях і пропонуючи можливі покращення. Це підвищує загальну якість тестових процедур і сприяє створенню більш надійного програмного забезпечення [2].

Загалом, автоматизація тестування пропонує значні переваги, які забезпечують вищу продуктивність та ефективність процесів розроблення програмного забезпечення, зменшуючи при цьому витрати та забезпечуючи високу якість продуктів [1].

Автоматизація тестування потребує значних початкових інвестицій, що передбачають витрати на придбання та налаштування спеціалізованого програмного забезпечення, необхідного для реалізації автоматизованих процесів [1]. Наприклад, платформи на зразок BrowserStack можуть бути особливо дорогими для невеликих команд або індивідуальних розробників, що створює додаткове фінансове навантаження [5].

Також для створення систем тестування потрібні відповідні інструменти, що підтримуватимуть автоматизацію на всіх етапах. Важливим аспектом є розвиток компетенцій: необхідно підвищити кваліфікацію наявного персоналу або залучити нових фахівців із потрібними знаннями та навичками [4]. Такі спеціалісти повинні мати досвід розробки й підтримки автоматизованих систем, що забезпечить стабільне функціонування та розвиток автоматизації тестування в майбутньому [5].

Автоматизовані тестові системи вимагають регулярного оновлення, щоб відповідати змінам у програмному забезпеченні, яке вони тестують. Завдання такого оновлення може бути складним і ресурсоємним. Наприклад, платформи, що часто оновлюють свій UI, як-от Browserling, вимагають постійного оновлення тестових скриптів для взаємодії з новими елементами інтерфейсу [1].

Автоматизація тестування може бути особливо складною при імітації людської взаємодій, часто необхідної у комплексних тестових сценаріях. Такі інструменти, як Appium або Robot Framework, пропонують рішення для деяких сценаріїв, але вони не відтворюють складні аспекти користувацького досвіду, що обмежує покриття тестів [5].

Зміни в користувацькому інтерфейсі вебзастосунків відбуваються часто, що своєю чергою вимагає реалізації можливості постійного оновлення тестових скриптів, особливо залежних від специфічних атрибутів елементів, таких як ідентифікатори, класи, або стилі. Ці оновлення можуть бути частими та потребувати значних зусиль для забезпечення того, щоб тестові скрипти залишалися актуальними та ефективними [4].

Ці аспекти підкреслюють, що автоматизація тестування хоча й є могутнім інструментом для покращення ефективності та об'єму тестування, та має свої значні виклики і недоліки, які вимагають уважного розгляду та планування перед розробленням і впровадженням.

Інтеграція штучного інтелекту (AI) в платформи для автоматизації тестування вебзастосунків стає ключовим напрямом розвитку в галузі якості програмного забезпечення. Один із прикладів цього — партнерство Microsoft та Leapwork, що спрямоване на надання можливостей для автоматизації тестування користувачам Microsoft Dynamics 365 і Microsoft Power Platform. Вони використовують платформу Leapwork, яка є AI-підсиленою, візуальною та забезпечує безпеку, для спрощення процесу створення тестів і зниження ризиків збоїв під час щомісячних оновлень програмного забезпечення [2]. Ще один приклад — компанія Tricentis, яка використовує технологію швидкого оптичного розпізнавання символів (OCR) для автоматизації тестування, це покращує точність та швидкість процесу за допомогою візуального аналізу [5].

Архітектура та особливості нової платформи для автоматизованого тестування вебзастосунків

На основі проведеного в попередньому розділі аналізу сильних і слабких сторін сучасних платформ запропоновано нову архітектуру автоматизованої тестової системи, реалізованої у вигляді бібліотеки, як спробу подолати ключові обмеження, що виникають під час використання типових підходів. У розробленій платформі зроблено акцент на модульність, простоту конфігурації, підтримку паралельного тестування, централізоване керування параметрами середовища та гнучку інтеграцію з різними рівнями системи, зокрема через API.

Система побудована з використанням мови C# та принципів об'єктно-орієнтованого програмування. Основу архітектури становить WebDriverManager, що реалізує патерн Singleton для уникнення надмірного створення браузерних сесій, зберігаючи стабільність тестів і оптимізуючи споживання ресурсів. Підтримка роботи як у локальному, так і у віддаленому середовищі (через Selenium Grid) забезпечує масштабованість платформи. Конфігурація параметрів тестування здійснюється через зовнішні JSON-файли, що дає можливість змінювати налаштування без втручання у код, що є особливо актуальним у багатокомандних або мультисередовищних проєктах.

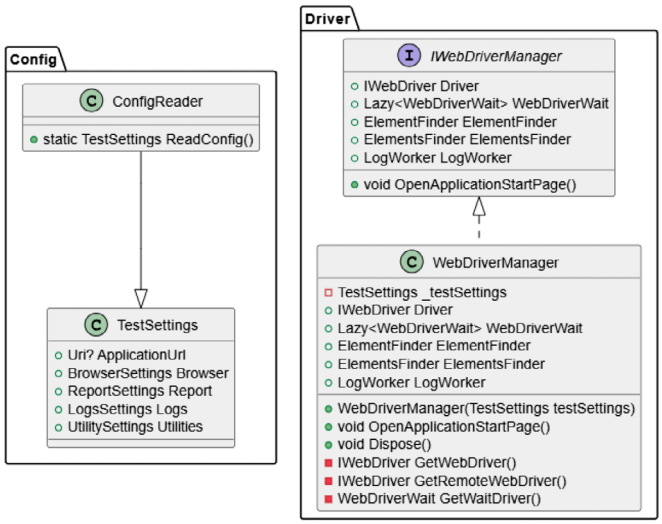


Рис. 1. Конфігурація і драйвери

Платформа підтримує як UI-, так і API-тестування, що дозволяє реалізовувати комплексні перевірки багаторівневих вебзастосунків. Для роботи з вебелементами створено набір класів (Element, ElementFinder, ElementsFinder), які розширюють функціональність Selenium WebDriver завдяки вбудованим механізмам автоматичного скролінгу, підсвічуванням елементів, розширеним перевіркам атрибутів і динамічним очікуванням (через інтеграцію з власним модулем Wait). Це суттєво підвищує стабільність автоматизованих тестів, особливо у випадках із динамічним інтерфейсом, що активно використовує AJAX-запити або асинхронне динамічне відображення контексту на вебсторінках.

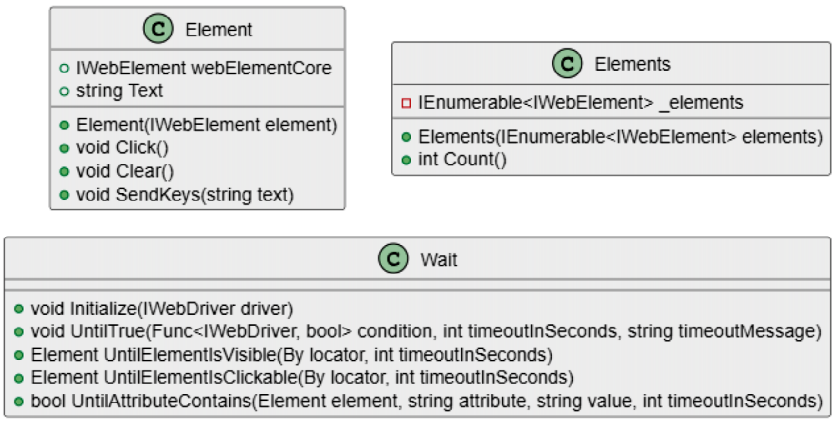


Рис. 2. Робота з вебелементами

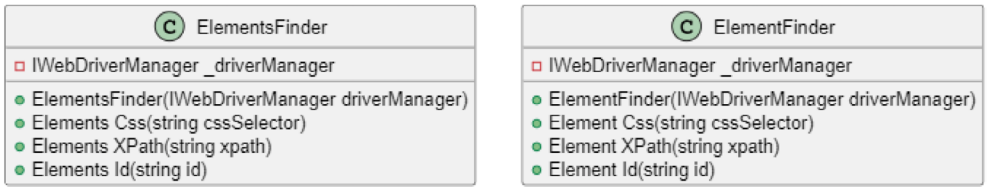


Рис. 3. Пошук вебелементів

Модуль ApiWorker забезпечує повноцінну підтримку API-тестування, дозволяючи надсилати HTTP-запити, обробляти й перевіряти відповіді. Платформа орієнтована на асинхронне виконання запитів, що дозволяє виконувати паралельні перевірки та значно скорочує час проходження тестів. Додатково реалізовано LocalStorageWorker для роботи з локальним сховищем браузера — важливого для тестування додатків, які зберігають користувацький стан безпосередньо в клієнті.

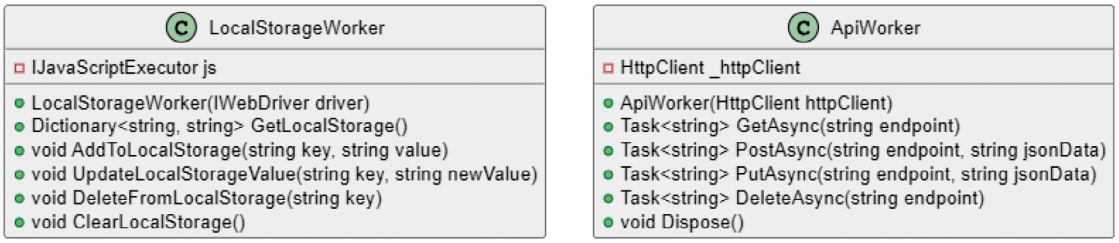


Рис. 4. Допоміжні модулі платформи — частина 1

Для підвищення гнучкості й варіативності сценаріїв застосовується TestDataWorker, який генерує унікальні дані (рядки, числа, дати, email, телефонні номери тощо), що дозволяє тестувати поведінку системи в умовах реалістичних змінних. Результати тестування перевіряються за допомогою модулів CheckWorker та VerifyWorker, які підтримують як базові, так і складні перевірки (наприклад, порівняння списків незалежно від порядку, перевірка регулярних виразів або повторне виконання перевірок через задані проміжки часу).

Для фіксації перебігу тестування та спрощення аналізу результатів реалізовано LogWorker, який записує структуровані лог-файли з часовими мітками, типами повідомлень і джерелом виклику. Це дозволяє легко ідентифікувати джерело помилки та надає прозорість під час перевірки якості програмного забезпечення.

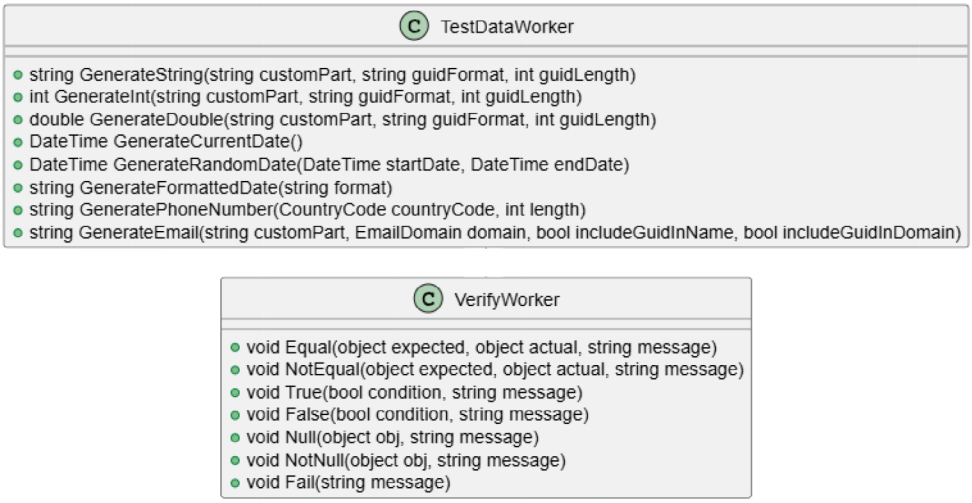


Рис. 5. Допоміжні модулі платформи — частина 2

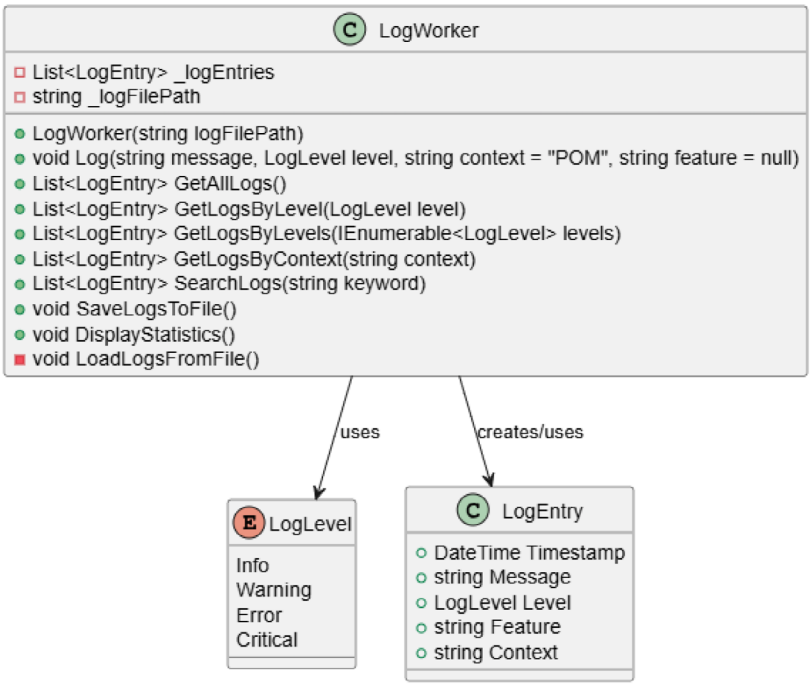


Рис. 6. Логування подій

Для візуалізації результатів використовується HtmlReportGenerator, який генерує інтерактивні HTML-звіти з маркуванням кроків і деталями кожного сценарію.

Однією з важливих переваг архітектури запропонованої платформи є її відкритість до інтеграції інтелектуальних рішень. Розвиток системи передбачає поетапне впровадження сучасних моделей машинного навчання, зокрема логістичної регресії, методів кластеризації, дерев рішень, випадкових лісів, штучних нейронних мереж та машин опорних векторів (SVM), які вже зарекомендували себе як ефективні інструменти для задач класифікації, виявлення аномалій, побудови моделей ризику та аналізу структурованих логів [6].

У рамках реалізації цієї стратегії вже впроваджено окремий модуль для оброблення логів, який забезпечує можливість інтеграції алгоритмів машинного навчання з метою автоматизованої класифікації дефектів, визначення їх пріоритетності та прогнозування потенційних збоїв. Завдяки модульності архітектури, функціональність системи може розширюватися без внесення змін до її основної логіки. Оскільки зазначений напрям охоплює окрему дослідницьку область, детальний опис використаних підходів, алгоритмів та результатів їхнього застосування буде представлено в подальших публікаціях.

Загальна ефективність платформи забезпечується не лише підтримкою інтелектуальних рішень, а й продуманою архітектурною будовою її основних модулів, кожен з яких виконує чітко визначену функцію та має низку технічних переваг.

Перевагою ініціалізаційного блоку є його гнучкість і конфігурованість. Компонент ConfigReader зчитує параметри тестового середовища та передає їх у TestSettings, що дозволяє швидко адаптувати платформу до різних середовищ і сценаріїв без потреби змінювати код. Це значно підвищує масштабованість та повторне використання тестів.

Модуль WebDriverManager забезпечує централізоване створення екземпляра браузера та відкриття стартової сторінки. Його головна перевага — уніфікований механізм роботи з різними браузерами, що підвищує стабільність тестових запусків і спрощує організацію крос-браузерного тестування.

Сильним боком архітектури є чіткий поділ відповідальностей. Окремі модулі для пошуку та взаємодії з елементами забезпечують підвищену надійність та простоту супроводу тестів. Завдяки цьому зменшується ризик помилок під час виконання та спрощується оновлення тестів у разі змін в інтерфейсі.

Ще однією перевагою платформи є інтеграція перевірок, логування та очікувань у режимі реального часу. Це забезпечує високу прозорість процесу тестування, полегшує аналіз результатів і дає змогу оперативно виявляти проблемні місця.

Модуль ApiWorker підвищує функціональну повноту платформи, дозволяючи реалізовувати інтеграційні сценарії через REST-запити. Це розширює сферу застосування платформи та дозволяє поєднувати UI- та API-рівні перевірок у межах одного тесту.

Компонент LocalStorageWorker забезпечує роботу з локальним сховищем браузера, що є особливо корисним для тестування сценаріїв зі збереженням стану. Його інтеграція дозволяє реалізовувати перевірки, які імітують поведінку реального користувача, що є додатковою перевагою при тестуванні складних вебзастосунків.

Основні модулі розробленої платформи пройшли успішну апробацію при виконанні лабораторних робіт з автоматизованого тестування вебзастосунків студентами Київського національного університету імені Тараса Шевченка.

Створена цілісна інтеграція модулів є надійним середовищем для автоматизованого тестування, яке легко адаптується до вимог швидкого впровадження, підтримки та масштабування. Таким чином, описані в роботі результати демонструють переваги запропонованої архітектури як універсального рішення для створення гнучких, розширюваних і ефективних систем автоматизованого тестування вебзастосунків, придатних для практичного використання в освітньому та промисловому середовищі.

Висновок

Створення нової платформи автоматизованого тестування зумовлене необхідністю усунення наявних недоліків, які заважають задовольнити вимоги, що стрімко зростають, до розроблення систем програмного забезпечення.

Реалізована платформа мінімізує початкові витрати, пов'язані з автоматизацією, та надає комплексне рішення «з коробки», усувається потреба вкладати кошти в розроблення власних інструментів для тестування. Дизайн платформи зосереджений на масштабованості, що дозволяє легко розширювати набори тестів або функціонал без значних додаткових інвестицій.

Інтеграція з інструментами типу Specflow підсилює адаптивність платформи, даючи змогу користувачам мінімальними зусиллями оновлювати тестові сценарії у відповідь на розвиток функціоналу додатків або вимог. Такий підхід не лише зменшує навантаження на підтримку, а й гарантує, що тести залишаються актуальними та ефективними у довгостроковій перспективі.

Модулі розроблюваної платформи долають загальні технічні обмеження, інтегруючи API та прямі запити до баз даних, що полегшує тестування складних сценаріїв, які можуть бути важкими для традиційної взаємодії через UI. Такий стратегічний вибір розширює обсяг тестування, гарантуючи всебічне покриття, яке охоплює процеси на рівні бекенду, цілісність даних і більше, незалежно від користувацького інтерфейсу.

Подальший розвиток платформи дасть можливість інтегрувати передові AI та ML моделі, такі як логістична регресія, методи кластеризації, дерева рішень, випадкові ліси, нейронні мережі та машини опорних векторів (SVM). Ці моделі можуть бути адаптовані для покращення процесів тестування шляхом передбачення потенційних збоїв, оптимізації вибору тестових кейсів та автоматичної ідентифікації дефектів.

Список літератури

1. Forgacs I. Modern Software Testing Techniques: A Practical Guide for Developers and Testers / I. Forgacs, A. Kovacs. — 1st ed. — Apress, 2024. — 350 p.
2. Homes B. Fundamentals of Software Testing / B. Homes. — 2nd ed., Revised and Updated. — Wiley-ISTE, 2024. — 280 p.
3. Lipskyi D. O. Development of a Platform for Web Application Testing Automation / D. O. Lipskyi // Bulletin of Taras Shevchenko National University of Kyiv. Series Physics & Mathematics. — 2023. — No. 1. — Pp. 45–50.
4. Loubser N. Software Engineering for Absolute Beginners: Your Guide to Creating Software Products / N. Loubser. — 1st ed. — Apress, 2021. — 330 p.
5. Mohan G. Full Stack Testing: A Practical Guide for Delivering High Quality Software / G. Mohan. — O'Reilly Media, 2022. — 420 p.
6. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems / A. Géron. — 3rd ed. — O'Reilly Media, 2022. — 856 p.

References

Forgacs, I., & Kovacs, A. (2024). *Modern software testing techniques: A practical guide for developers and testers* (1st ed.). Apress.

Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (3rd ed.). O'Reilly Media.

Homes, B. (2024). *Fundamentals of software testing* (2nd ed., Revised and updated). Wiley-ISTE.

Lipskyi, D. O. (2023). Development of a platform for web application testing automation. *Bulletin of Taras Shevchenko National University of Kyiv. Series Physics & Mathematics*, *1*, 45–50.

Loubser, N. (2021). *Software engineering for absolute beginners: Your guide to creating software products* (1st ed.). Apress.

Mohan, G. (2022). *Full stack testing: A practical guide for delivering high quality software*. O'Reilly Media.

D. Lipskyi

ARCHITECTURE OF A NEW ENHANCED PLATFORM FOR AUTOMATED WEB APPLICATION TESTING

Improving the automation of web application testing is a particularly relevant and rapidly evolving area in the modern software development process. The growing complexity of web-based systems, the increasing frequency of release cycles, and the ever-rising demand for high software reliability and performance make the adoption of automated testing solutions not only desirable but essential. This paper analyses the current state of test automation technologies, with a focus on widely adopted tools, frameworks, and methodologies. It outlines their primary advantages, including enhanced speed, high repeatability, improved accuracy, and reduced human error. At the same time, it identifies common limitations, such as high initial setup and learning costs, challenges in test maintenance, and limited adaptability to rapidly changing or project-specific requirements.

To address these challenges, the article introduces a novel architecture for an automated testing platform, designed and implemented as a reusable, extensible software library. The platform is built on a modular architecture, ensuring flexibility, maintainability, scalability, and seamless integration into a wide range of existing web development projects. Its core modules include a configuration handler, a browser driver manager, components for interaction with UI elements, a centralised logging subsystem, API communication tools, and browser storage management capabilities. These modules function together as a cohesive unit, forming a reliable and transparent environment that facilitates efficient and robust test execution.

In addition to architectural innovations, the paper discusses strategies to enhance test maintainability and reduce long-term resource consumption. These include intelligent reuse of test components, support for parameterised configurations, and mechanisms for simplifying test orchestration and execution across different environments. The proposed solution provides a practical, scalable framework for improving the quality, reliability, and efficiency of web application testing.

Keywords: automation, web technologies, platform, testing.

Матеріал надійшов 08.06.2025

УДК 004.415.5
DOI: 10.18523/2617-3808.2025.8.232-237

Суліменко А. А., Франків О. О., Нагнибіда А. А.

ПРОГРАМНИЙ КОМПЛЕКС STABILITY ASSURANCE TOOL: ЕВОЛЮЦІЯ ТА РОЗВИТОК ДЛЯ АВТОМАТИЗОВАНОЇ ОЦІНКИ СТАБІЛЬНОСТІ ТА ЗРОЗУМІЛОСТІ КОДУ SWIFT

У статті розглянуто процес створення, еволюції та практичного застосування програмного комплексу Stability Assurance Tool (SAT), призначеного для статичного аналізу коду, написаного мовою Swift. Головна мета інструменту полягає у забезпеченні автоматизованої оцінки таких характеристик, як стабільність і зрозумілість програмного забезпечення, що розробляється. Описано використані метрики, архітектурні рішення, методи інтеграції з середовищем розробки Xcode та системами безперервної інтеграції (CI/CD), а також результати адаптації класичних метрик об'єктно-орієнтованого програмування до специфіки Swift. Представлені результати демонструють потенціал SAT як платформи для подальшого розвитку засобів оцінки якості ПЗ.

Ключові слова: SPM, статичний аналіз, стабільність коду, зрозумілість коду, метрики програмного забезпечення, архітектура ПЗ, автоматизована оцінка.

Вступ

Якість програмного забезпечення є ключовим фактором його довговічності, ефективності та здатності до масштабування. У сучасному контексті стрімкого розроблення мобільних і десктопних застосунків, особливо в екосистемі Apple, значну увагу приділяють мовам програмування, що дозволяють створювати безпечні, надійні та зручні у супроводі програми. Однією з таких мов є Swift [1].

Swift є мовою, яка поєднує елементи об'єктно-орієнтованого, протокольно-орієнтованого та функціонального стилів програмування. Це створює як нові можливості, так і виклики для автоматизованого аналізу коду, адже більшість сучасних метрик якості були розроблені з урахуванням класичних об'єктно-орієнтованих мов, таких як Java або C++.

Відповідно, виникає потреба в інструменті, що не лише розуміє синтаксис Swift, а й здатен коректно інтерпретувати його семантику, моделі успадкування, використання протоколів, структури, акторальну модель конкурентності та інші мовні особливості. Саме для задоволення цієї потреби і було створено Stability Assurance Tool.

Теоретичні засади та методи оцінки стабільності програмного забезпечення

Оцінка якості архітектури програмного забезпечення є невід'ємною та фундаментальною складовою процесу управління життєвим циклом будь-якого програмного продукту. Відповідно до визнаних міжнародних стандартів, зокрема ISO/IEC 9126, якість програмного забезпечення визначається через багатогранний набір характеристик [6], таких як функціональність, надійність, зручність використання, ефективність, супроводжуваність і портативність. У контексті цього дослідження центральне місце посідає характеристика супроводжуваності (Maintainability), а саме її ключові підхарактеристики: змінюваність, тестованість, зрозумілість і стабільність. Стабільність, у цьому визначенні, трактується як здатність програмної системи зберігати свою цілісність і коректність функціонування при внесенні модифікацій, що є критично важливим для довгострокового розвитку та еволюції проєкту.

Для кількісної та об'єктивної оцінки цих атрибутів у програмній інженерії застосовуються метрики програмного забезпечення — стандартизовані методи вимірювання, що дозволяють формалізувати та оцінити якість чи складність коду.

Таким чином, методологічною основою для розробленого інструменту Stability Assurance Tool (SAT) став фундаментальний набір об'єктно-орієнтованих метрик, запропонований у праці S. R. Chidamber і C. F. Kemerer (надалі — C&K) [4]. Цей набір, що містить такі відомі показники, як WMC, DIT, NOC, CBO, RFC та LCOM, довів свою ефективність і широко цитується в більшості сучасних досліджень, присвячених аналізу якості коду [7]. У своїй роботі C&K не лише запропонували, а й теоретично та емпірично валідували цей набір метрик для вимірювання якості дизайну об'єктно-орієнтованого програмного забезпечення.

Однак, як було зазначено у пізніших дослідженнях, зокрема у праці L. Etzkorn, C. Davis і J. Talburt, класичний підхід C&K не завжди повною мірою враховує динамічність сучасного процесу розроблення, де навіть незначні структурні зміни можуть суттєво вплинути на результати вимірювань. Цей фактор, у поєднанні з унікальними особливостями мови програмування Swift, такими як її мультипарадигменість, активне використання протокольно-орієнтованого підходу та сучасна модель конкурентності на базі акторів, зумовив нагальну необхідність глибокої адаптації класичних метрик. Відповідно, в рамках розробки SAT було здійснено ретельну адаптацію метрик C&K, уточнено алгоритми їх обчислення та розроблено систему гнучких шкал оцінки, що здатні враховувати масштаб і складність конкретного проєкту.

Адаптація метрик і розробка системи оцінки

Процес створення ефективної та об'єктивної методики оцінки стабільності коду, написаного мовою Swift, вимагав глибокого аналізу та адаптації кожної з обраних метрик. Для забезпечення контекстуальної коректності оцінки було впроваджено класифікацію програмних модулів за розміром (малі, середні, великі), що базується на кількості класів у проєкті. Такий підхід є обґрунтованим, оскільки показники зв'язності та складності мають різну вагу та інтерпретацію для проєктів різного масштабу. Нижче детально описано кожну метрику та її реалізацію в інструменті SAT.

WMC (Weighted Methods per Class — Зважені методи на клас)

В оригінальному визначенні ця метрика часто спрощується до простої кількості методів у класі. У розробленій адаптації запропоновано більш глибокий підхід, що фокусується на обчисленні комплексної складності класу. Значення WMC обчислюється як сума складностей (*Ci*) всіх методів (*Mi*), визначених у класі, за формулою:

$$WMC = i = 1 \sum nCi,$$

де *n* — це загальна кількість методів у класі. Якщо всі складності методів вважати рівними одиниці, то WMC дорівнюватиме *n*. У рамках SAT було запропоновано два способи обчислення складності (*Ci*):

1. **unity**: Складність кожного методу приймається за одиницю (*Ci* = 1). У цьому випадку WMC дорівнює кількості методів, що дає базову, кількісну оцінку складності класу.
2. **custom**: Складність методу обчислюється з використанням показника RFC (Response for a Class). Цей підхід дозволяє оцінити клас не лише за кількістю методів, а й за інтенсивністю та складністю його зовнішніх взаємодій. Таким чином, цей тип утилізує метрику RFC для більш точного та контекстуалізованого визначення WMC. Для метрики WMC були встановлені відносні порогові значення: відхилення до 10 % від середнього показника за проєктом вважається добрим результатом, до 15–20 % (залежно від розміру проєкту) — допустимим, тоді як більші значення сигналізують про потенційні проблеми зі складністю та супроводжуваністю класу.

NOC (Number of Children — Кількість кащадків)

Ця метрика дорівнює кількості безпосередніх дочірніх класів, що успадковуються від базового класу. Враховуючи, що мова Swift активно просуває альтернативи глибоким ієрархіям успадкування через використання протоколів та композиції, високе значення NOC може свідчити про надмірну складність і жорсткість архітектури. Для адекватної оцінки було розроблено допустимі відсоткові співвідношення кількості класів з високим NOC залежно від розміру проєкту: 10 % для малих, 30 % для середніх і 50 % для великих проєктів.

RFC (Response for a Class — Реакція на клас)

В оригінальній праці C&K максимальне значення RFC не було чітко визначено. SAT інтерпретує RFC згідно з фундаментальним визначенням авторів: як набір методів, що потенційно можуть бути виконані у відповідь на повідомлення, отримане об'єктом даного класу. Обчислення відбувається у два етапи:

1. Перший етап (прямі виклики):

$$RFC = M + R$$

де M — це кількість методів, визначених у самому класі, а R — кількість унікальних зовнішніх методів, що викликаються безпосередньо методами цього класу.

2. Повна оцінка (рекурсивні виклики):

$$RFC' = M + R'$$

де R' — це повний набір віддалених методів, викликаних рекурсивно по всьому дереву викликів, що виходять з класу. Численні дослідження, зокрема проведені NASA та іншими авторами, підтверджують, що високий показник RFC сильно корелює зі збільшенням щільності помилок та ускладненням тестування [9]. На основі цих даних було розроблено шкалу оцінки: середнє значення RFC до 50 вважається добрим, до 100 — допустимим, а вищі показники свідчать про надмірну складність та зв'язність класу.

LOC (Lines of Code — Розмірність файлів)

У розробленій системі ця метрика слугує не лише для прямої оцінки розміру програмних модулів, а і як важливий зв'язувальний фактор, що використовується для застосування коректних шкал оцінки до інших метрик. Також LOC опосередковано оцінює лаконічність архітектури в межах окреслених модулів.

LOCM (Lack of Cohesion of Methods — Показник недостатньої зчепленості методів)

Імлементація адаптації цієї метрики є важливим доповненням до першої версії аналізатора, в якій її не було реалізовано через складність обчислення та збору необхідних параметрів з архітектури програмного забезпечення.

Зчепленість є однією з ключових характеристик якісного об'єктно-орієнтованого дизайну. Вона відображає ступінь, до якого елементи всередині одного модуля взаємопов'язані та спрямовані на досягнення єдиної, чітко визначеної мети. Класи з високою зчепленістю є більш зрозумілими, легшими в супроводі та повторному використанні, оскільки їхня відповідальність є вузько сфокусованою.

LOCM, як випливає з її назви, вимірює протилежне — відсутність зчепленості. Високе значення LOCM свідчить про те, що клас, імовірно, виконує багато непов'язаних функцій. Його методи можуть бути розділені на групи, кожна з яких працює з власним набором атрибутів. Така фрагментована структура — ознака слабкого дизайну, і зазвичай вказує на необхідність рефакторингу через розбиття класу на декілька менших, більш сфокусованих одиниць [8]. Навпаки, низьке значення LOCM є ознакою сильної зчепленості: методи класу активно використовують спільні атрибути, що свідчить про якісну архітектуру.

З урахуванням синтаксичних особливостей мови Swift і ґрунтуючись на формальному визначенні метрики, в інструменті Stability Assurance Tool було реалізовано такий алгоритм обчислення LOCM.

Для класу C з множиною методів:

$$\{M1, M2, \ldots, Mn\}$$

та множиною атрибутів (змінних екземпляра)

$$\{A1, A2, \ldots, Am\}:$$

1. Визначення множин атрибутів, які використовує кожен метод.

Для кожного методу Mi визначається множина Ii — набір атрибутів, які він використовує (для читання або запису).

2. Підрахунок кількості пар методів.

Для кожної унікальної пари (*Mi*, *Mj*), *i* < *j* перевіряється перетин множин *Ii*∩*Ij*. Тоді:

– Якщо *Ii* ∩ *Ij* = ∅ — пара вважається незв'язною.
– Якщо *Ii* ∩ *Ij* ≠ ∅ — пара вважається зв'язною.

3. Обчислення LOCM.

$$LOCM = \{P - Q, 0, \text{якщо } P > Q \text{ інакше},$$

де *P* — кількість унікальних незв'язних пар методів, *Q* — кількість зв'язних пар.

Такий обрахунок фактично формалізує ступінь недостатньої зчепленості: якщо кількість незв'язаних пар перевищує кількість зв'язаних, LOCM буде позитивним і сигналізуватиме про проблему. Якщо ж навпаки, то значення LOCM буде нульовим — що вказує на належну внутрішню зв'язність.

Відповідно, інтеграція метрики LOCM у Stability Assurance Tool дозволила суттєво підвищити якість архітектурного аналізу класів, забезпечивши об'єктивний показник ступеня відповідності принципу єдиної відповідальності. LOCM слугує ефективним індикатором внутрішньої зчепленості та підґрунтям для виявлення архітектурних дефектів у дизайні об'єктно-орієнтованих систем.

Архітектура та еволюція комплексу

Архітектура Stability Assurance Tool пройшла кілька фаз трансформації: від простої CLI-програми до повноцінного багатокомпонентного інструменту, що інтегрується у виробничі процеси розроблення програмного забезпечення. Початкова версія SAT була реалізована у формі консольного застосунку, що виконував аналіз коду на основі синтаксичного дерева, сформованого за допомогою бібліотеки SwiftSyntax [2]. У ній реалізовувався обмежений набір метрик, а результати аналізу виводилися у формі звіту в терміналі або у форматі HTML.

У процесі розвитку проєкту було прийнято рішення перейти до модульної архітектури на основі Swift Package Manager. Це дозволило забезпечити розділення відповідальностей між компонентами: виконуваний модуль, бібліотека з логікою аналізу, плагін для зручного запуску з командного рядка. Однією з найважливіших переваг нового підходу стало те, що SAT може інтегруватися безпосередньо в Xcode за допомогою Build Phase Scripts [1]. Завдяки цьому аналіз коду виконується автоматично під час компіляції проєкту, а результати виводяться безпосередньо в Issue Navigator.

Ключовою зміною в архітектурі стала також підтримка паралельного обчислення метрик. Застосування механізмів Swift Concurrency дало можливість розпаралелити обчислення незалежних метрик, що значно зменшило час аналізу для великих кодових баз [10]. Використання структур Task зробило можливим незалежне виконання аналізу для кожного класу, модуля або метрики, що покращило масштабованість і продуктивність SAT.

Практичне застосування

Практичне використання Stability Assurance Tool охоплює як локальну оцінку проєктів під час розроблення, так і автоматизовану перевірку в процесах CI/CD. Інтеграція в Xcode відбувається за рахунок виконуваного скрипта у Build Phases, який викликає аналізатор перед компіляцією цільового модуля. Завдяки форматуванню виводу, що відповідає Xcode-стандартам, усі попередження та помилки автоматично відображаються в редакторі коду. Це забезпечує негайний зворотний зв'язок, дозволяє розробнику швидко виявити проблемні місця та своєчасно їх усунути.

Система конфігурації SAT основана на YAML-файлах, що дозволяє детально налаштовувати параметри аналізу. У YAML-налаштуваннях визначаються порогові значення для кожної метрики, їхні критичність, вагові коефіцієнти, активовані метрики, а також каталоги, які необхідно проаналізувати або вилучити. Такий підхід забезпечує гнучкість та адаптивність, дозволяючи застосовувати SAT до різноманітних проєктів — від невеликих open source-бібліотек до корпоративних рішень із сотнями тисяч рядків коду.

Особливо цінною є інтеграція з CI/CD-середовищами [3]. Аналізатор підтримує параметри типу maxAllowedWarnings, а також дозволяє призначати кожній метриці статус error чи warning при перевищенні порогу. Це забезпечує створення так званих quality gates, які автоматично зупиняють збірку у випадку погіршення якості коду. Таким чином, SAT виступає не лише інструментом діагностики, а й важливим елементом інфраструктури забезпечення якості програмного забезпечення.

Результати та обмеження розробленої системи

У процесі експлуатації SAT було підтверджено його ефективність у виявленні архітектурних недоліків і складних для підтримки модулів. Завдяки використанню кількісних метрик розробники отримують об'єктивну оцінку стабільності та зрозумілості коду, що дозволяє приймати обґрунтовані рішення щодо рефакторингу та покращення архітектури [5]. Інструмент продемонстрував здатність інтегруватися в реальні робочі процеси розробників, не порушуючи звичного циклу роботи.

Однак, попри всі переваги, SAT має і певні обмеження. Зокрема, поточна реалізація не підтримує глибокий аналіз конкурентного коду, зокрема взаємодії між акторами або асинхронних залежностей. Також метрики були адаптовані для класів, однак у Swift велике значення мають протоколи, структури та розширення, які не завжди охоплюються наявними правилами. У майбутньому SAT потребуватиме розширення метрик, орієнтованих саме на особливості Swift — як-от аналіз POP-конструкцій або акторальних сценаріїв.

Ще однією з проблем є продуктивність на надвеликих проєктах. Хоча паралелізація дозволяє прискорити аналіз, все ж час оброблення десятків тисяч класів або модулів може бути значним. Це створює потребу у додатковій оптимізації алгоритмів та побудові більш гнучких стратегій агрегації результатів.

Висновки

Stability Assurance Tool став важливою ініціативою у напрямі побудови надійної інфраструктури контролю якості Swift-проєктів. Його розвиток засвідчив можливість адаптації класичних метрик до сучасних мов програмування, а також продемонстрував ефективність глибокої інтеграції аналізу в процес розроблення. Архітектура SAT дозволяє зручно розширювати інструмент, додаючи нові метрики або вдосконалюючи наявні алгоритми.

Найперспективнішими напрямами подальшого розвитку є розширення підтримки для аналізу Swift Concurrency, зокрема виявлення потенційних блокувань і станів гонитви. Також доцільно розширити метрики для аналізу протокольно-орієнтованої архітектури, зокрема врахування розширень, композитних протоколів і реалізацій через розширення.

Не менш важливим є покращення взаємодії SAT з іншими інструментами. Розробка експортера у форматі SARIF дозволить інтегрувати результати SAT у загальні системи контролю якості, такі як GitHub Act, Azure DevOps або Jenkins. Підтримка Objective-C і змішаних Swift/ObjC-проєктів також відкриє можливості для аналізу складних, спадкових кодових баз.

У перспективі SAT має потенціал перетворитися не лише на інструмент перевірки, а й на рекомендаційний механізм, що не лише виявляє проблеми, а й пропонує шляхи їх вирішення. Залучення елементів машинного навчання та розроблення моделей, здатних на основі історичних даних прогнозувати стабільність змін, відкриє нові горизонти в автоматизованій оцінці якості коду.

Список літератури

1. Apple Inc. Swift Language Guide [Electronic resource]. — 2023. — Mode of access: https://docs.swift.org/swift-book.
2. Apple Inc. SwiftSyntax Documentation [Electronic resource]. — 2023. — Mode of access: https://github.com/apple/swift-syntax.
3. Beller M. How developers use static analysis tools in practice / M. Beller, G. Gousios, A. Zaidman, A. Van Deursen // Proceedings of the 37th International Conference on Software Engineering (ICSE). — IEEE, 2015. — Pp. 191–201.
4. Chidamber S. R. A Metrics Suite for Object-Oriented Design / S. R. Chidamber, C. F. Kemerer // IEEE Transactions on Software Engineering. — 1994. — Vol. 20, no. 6. — Pp. 476–493.
5. Fowler M. Refactoring: Improving the Design of Existing Code / M. Fowler. — Boston : Addison-Wesley, 2002. — 431 p.
6. ISO/IEC 9126-1:2001. Software Engineering — Product Quality. — Part 1: Quality Model. — Geneva : ISO, 2001.
7. Lanza M. Object-Oriented Metrics in Practice / M. Lanza, R. Marinescu. — Berlin : Springer, 2006. — 208 p.
8. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship / R. C. Martin. — Upper Saddle River, NJ : Prentice Hall, 2009. — 464 p.
9. NASA Software Assurance Technology Center. Software Quality Metrics Overview [Electronic resource]. — 2003. — Mode of access: https://ntrs.nasa.gov.
10. Swift.org. Concurrency in Swift: Structured Concurrency, Actors, and Async/Await [Electronic resource]. — 2023. — Mode of access: https://www.swift.org.

References

Apple Inc. (2023). Swift Language Guide. https://docs.swift.org/swift-book.

Apple Inc. (2023). SwiftSyntax Documentation. https://github.com/apple/swift-syntax.

Beller, M., Gousios, G., Zaidman, A., & Van Deursen, A. (2015). How developers use static analysis tools in practice. In *Proceedings of the 37th International Conference on Software Engineering* (pp. 191–201). IEEE.

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object-oriented design. *IEEE Transactions on Software Engineering*, *20* (6), 476–493.
Fowler, M. (2002). *Refactoring: Improving the design of existing code*. Addison-Wesley.
ISO/IEC. (2001). Software engineering — Product quality — Part 1: Quality model (ISO/IEC 9126-1:2001).
Lanza, M., & Marinescu, R. (2006). *Object-oriented metrics in practice*. Springer.
Martin, R. C. (2009). *Clean code: A handbook of agile software craftsmanship*. Prentice Hall.
NASA Software Assurance Technology Center. (2003). Software quality metrics overview. https://ntrs.nasa.gov.
Swift.org. (2023). Concurrency in Swift: Structured concurrency, actors, and async/await. https://www.swift.org.

A. Sulimenko, O. Frankiv, A. Nagnybida

SOFTWARE PACKAGE STABILITY ASSURANCE TOOL: EVOLUTION AND DEVELOPMENT FOR AUTOMATED EVALUATION OF SWIFT CODE STABILITY AND READABILITY

The development of robust and maintainable software systems is highly dependent on the architectural quality of source code. Swift, as a modern programming language developed by Apple, introduces unique challenges for static analysis due to its use of protocol-oriented and concurrent programming paradigms. Traditional quality metrics often fail to capture the nuanced characteristics of Swift codebases, creating a need for dedicated tooling.

This article presents the Stability Assurance Tool (SAT), a lightweight yet powerful static analysis system designed specifically for Swift. SAT applies a suite of object-oriented design metrics, such as those from the Chidamber & Kemerer framework, and adapts them for Swift using techniques based on abstract syntax trees generated via SwiftSyntax. The tool is engineered as a modular Swift package that integrates seamlessly with Xcode and continuous integration systems, providing developers with real-time feedback about the architectural soundness of their code.

The tool analyzes source code by computing metrics in parallel using Swift Concurrency. The results can be visualized via terminal reports or HTML dashboards and serve as input for quality gates in CI/CD pipelines. Special attention is given to YAML-based configuration that allows teams to calibrate metric weights, set custom thresholds, and fine-tune severity levels.

This article also explores SAT's extensible architecture and the practical results of its application to both open-source and enterprise Swift projects. Limitations of the current version include shallow semantic analysis and limited support for concurrency and protocol extensions, which will be addressed in future updates. Long-term plans include integration with SARIF, multi-language support, and machine learning-based prediction of unstable modules.

Keywords: static code analysis, Swift, software stability, code maintainability, CI/CD, SwiftSyntax, software metrics.

Матеріал надійшов 30.06.2025

УДК 005:32
DOI: 10.18523/2617-3808.2025.8.238-242

Хряпа О. І.

ЕМОЦІЙНИЙ АСПЕКТ БІЗНЕС-КОМУНІКАЦІЙ: ПЕРЕОСМИСЛЕННЯ РОЛІ НЕГАТИВНИХ ЕМОЦІЙ У ПРОФЕСІЙНОМУ ІТ-СЕРЕДОВИЩІ

Стаття досліджує роль емоцій в ІТ-бізнес-комунікаціях, зокрема аналізує місце так званих негативних емоцій у професійному середовищі. Авторка критично переосмислює традиційний підхід до розмежування бізнес-комунікацій та повсякденного спілкування, наголошуючи на важливості емоційного компонента в ефективній корпоративній взаємодії. На основі аналізу сучасних досліджень емоційного інтелекту та практичних кейсів демонструється, що прояв автентичних емоцій, і негативних також, може сприяти побудові довіри та покращенню організаційної ефективності в ІТ-компаніях.

Ключові слова: бізнес-комунікації, емоційний інтелект, негативні емоції, довіра, організаційна ефективність, автентичність, корпоративна культура.

Вступ

Традиційні підходи до бізнес-комунікації, що базувалися на жорсткому розмежуванні професійної та особистісної сфер, виявляються недостатньо ефективними в умовах сучасного суспільства, де людський капітал стає основним конкурентним ресурсом організації. Особливо це характерне для ІТ-сфери, де саме працівники є рушієм змін у компанії.

Дослідження в галузі емоційного інтелекту переконливо доводять, що емоції не є «шумом» у комунікативному процесі, а становлять його невід'ємну та функціонально важливу складову. Особливої актуальності набуває питання ролі так званих негативних емоцій у професійному ІТ-середовищі.

Попри зростання популярності концепції емоційного інтелекту, корпоративна культура більшості організацій досі характеризується стигматизацією прояву таких емоцій, як гнів, розчарування, тривога, сум тощо. Це створює парадоксальну ситуацію: від працівників вимагається розвиток емоційного інтелекту, але прояв природних емоційних реакцій розглядається як ознака непрофесіоналізму.

Теоретичні засади дослідження

Еволюція розуміння бізнес-комунікації

Історично розвиток теорії бізнес-комунікації відбувався в парадигмі механістичного підходу, що передбачав максимальну раціоналізацію комунікативних процесів та мінімізацію «людського фактора». Проте розвиток гуманістичного підходу в менеджменті поступово змінив розуміння ролі людських відносин в організації.

Сучасне розуміння бізнес-комунікації формується під впливом системного підходу, де емоції перестають сприйматися як перешкода для ефективної комунікації, натомість розглядаються як важливий ресурс для створення організаційної культури та досягнення стратегічних цілей.

Водночас традиційне протиставлення бізнес-комунікації та звичайного спілкування залишається поширеним у корпоративній практиці через структурні бар'єри: ієрархічні відносини, різні комунікативні стилі та особистісні особливості учасників.

Концептуальні основи емоційного інтелекту

Концепція емоційного інтелекту, теоретично обґрунтована П. Саловеєм та Дж. Мейєром [8] і популяризована Д. Гоулманом, революціонізувала розуміння ролі емоцій у професійній діяльності.

Модель емоційного інтелекту Гоулмана охоплює п'ять ключових компетенцій: самосвідомість, саморегуляція, мотивація, емпатія та соціальні навички [5].

Нейронаукові дослідження підтверджують фундаментальну роль емоцій у процесах прийняття рішень і соціальної взаємодії, що свідчить про те, що емоції не протиставляються раціональності, а є її необхідною складовою.

У бізнес-контексті емоційний інтелект виявляється особливо важливим для лідерських позицій. Дослідження показують, що успішність керівників на 90 % залежить від емоційного інтелекту, особливо на вищих управлінських рівнях.

Результати дослідження

Механізми стигматизації негативних емоцій

Аналіз корпоративних ІТ-практик виявляє системний характер стигматизації негативних емоцій через кілька механізмів.

Нормативний механізм — створення неписаних правил щодо «прийнятної» емоційної поведінки, що транслюються через систему заохочень і покарань.

Дискурсивний механізм — мовні практики, що надають негативну конотацію проявам емоцій («не треба емоціонувати», «це непрофесійно»).

Структурний механізм — використання звинувачень у надмірній емоційності як способу уникнення складних розмов або прикриття некомпетентності.

Гендерний механізм — різне ставлення до емоційних проявів чоловіків і жінок, де ті самі реакції інтерпретуються по-різному залежно від статі.

Функціональна роль негативних емоцій

Сучасні дослідження спростовують уявлення про виключно деструктивний характер негативних емоцій:

- *Гнів* сигналізує про порушення важливих цінностей та мобілізує енергію для їх захисту, може стимулювати інноваційні рішення.
- *Тривога* виконує функцію раннього попередження про потенційні проблеми, стимулюючи кращу підготовку.
- *Сум і розчарування* допомагають переосмислити цілі після невдач, сприяючи навчанню та адаптації.

Конструктивність негативних емоцій залежить не від їх придушення, а від способу вираження та інтеграції у комунікативний процес [13; 15].

Емоційний зв'язок як основа довіри

Емоційний зв'язок між учасниками організації є ключовим фактором формування довіри [4]. Парадоксально, але прояв уразливості через негативні емоції часто сприяє зміцненню такого зв'язку через:

- автентичність емоційних проявів як сигнал про чесність;
- спільне переживання складних емоцій, що створює відчуття спільності;
- готовність ділитися негативними емоціями як демонстрація довіри.

Аналіз українського кейсу

Особливої уваги заслуговує досвід українських громадських і волонтерських організацій, які продемонстрували винятковий рівень ефективності в умовах війни. Ключовою характеристикою їхньої комунікації є радикальна відкритість і автентичність.

Лідери відкрито діляться складними емоціями, що створює:

- відчуття автентичності та довіри;
- об'єднання спільноти навколо спільних цілей;
- конструктивне каналізування емоцій у продуктивні дії.

Важливим елементом є поєднання емоційної відкритості з високим рівнем відповідальності та професіоналізму, а також культура взаємної підтримки та визнання права на помилку.

Взаємозв'язок емоційного інтелекту та організаційної ефективності

Емпіричні дослідження демонструють статистично значущі кореляції між рівнем емоційного інтелекту та показниками продуктивності праці, задоволеності роботою, лідерської ефективності [6; 9]. Команди з емоційно інтелектуальними лідерами показують кращі результати у [14]:

- рівні довіри та згуртованості;
- здатності до адаптації в умовах змін;
- ефективності вирішення конфліктів;
- рівні інноваційності та креативності;
- показниках утримання персоналу.

Практичні рекомендації

Розвиток емоційної грамотності керівників

Програми розвитку мають передбачати [11]:

- теоретичну підготовку з основ психології емоцій;
- практичні вправи на розвиток емпатії та навичок активного слухання;
- тренінги з конструктивного вираження складних емоцій.

Створення культури психологічної безпеки

Необхідні системні зміни:

- перегляд системи оцінки працівників з включенням показників емоційного інтелекту;
- розроблення протоколів обговорення складних емоційних ситуацій;
- забезпечення каналів для висловлення незгоди без страху покарання.

Інтеграція емоційного компонента в корпоративні процеси

- врахування емоційних реакцій у процесах прийняття рішень;
- включення питань про емоційний досвід у системи зворотного зв'язку;
- особлива увага до емоційних аспектів при управлінні змінами.

Розвиток навичок конструктивного вираження емоцій

ІТ-компанії можуть розробляти:

- внутрішні протоколи для обговорення складних тем;
- техніки активного слухання та деескалації конфліктів;
- способи конструктивного вираження незгоди.

Висновки

Проведене дослідження дає можливість зробити висновок про необхідність кардинального перегляду традиційних підходів до ролі емоцій у бізнес-комунікаціях. Емоції становлять невід'ємну та функціонально важливу складову професійної взаємодії.

Стигматизація негативних емоцій у корпоративному середовищі є не лише етично проблематичною, а й економічно нераціональною практикою. Автентичний прояв емоцій, за умови їхнього конструктивного каналізування, може значно покращувати якість міжособистісних відносин і результативність спільної діяльності.

Український досвід демонструє альтернативну модель професійної взаємодії, що базується на відкритості, автентичності та взаємній підтримці. Ця модель може слугувати джерелом інсайтів для розвитку корпоративних комунікативних стратегій.

Подальші дослідження можуть бути спрямовані на розроблення інструментів оцінки емоційного клімату в організаціях, вивчення культурних особливостей сприйняття емоцій і аналіз довгострокових ефектів впровадження емоційно-орієнтованих комунікативних практик.

Список літератури

1. Андерсон К. Психологія довіри в організаціях / К. Андерсон. — Наука і освіта, 2019. — 342 с.
2. Бовє К. Л. Сучасні бізнес-комунікації / К. Л. Бовє, Дж. В. Тілл. — Пірсон, 2020. — 672 с.
3. Дамасіо А. У пошуках Спінози: радість, печаль і відчуття мозку / А. Дамасіо // Сучасні бізнес-комунікації. — Харкорт, 2003. — 355 с.
4. Дослідження емоційного інтелекту та задоволеності роботою [Електронний ресурс] // Українська правда. Життя. — 2017. — Режим доступу: https://life.pravda.com.ua/health/2017/10/15/226957/ (дата звернення: 20.05.2025).
5. Емоційний інтелект у бізнесі / за ред. Д. Гоулмана. — Наш формат, 2019. — 456 с.
6. Емоційний інтелект як фактор успіху в бізнесі [Електронний ресурс] // Дело.ua. — 2022. — Режим доступу: https://delo.ua/career/emotional-intelligence-business-success-390847/ (дата звернення: 22.05.2025).
7. Канеман Д. Думай швидко і повільно / Д. Канеман. — Фаррар, Страус і Жіру, 2011. — 499 с.
8. Мейєр Дж. Д. Емоційний інтелект: нова здатність чи еклектична суміш особистісних рис? / Дж. Д. Мейєр, П. Саловей, Д. Р. Карузо // Американський психолог. — 2008. — № 63 (6). — С. 503–517.
9. Психологічна безпека на робочому місці: український досвід [Електронний ресурс] // Mind.ua. — 2023. — Режим доступу: https://mind.ua/publications/20240156-psihologichna-bezpeka-na-robochomu-misciukrayinskij-dosvid (дата звернення: 25.05.2025).
10. Психологія управління персоналом / за ред. О. І. Бондарчук. — Логос, 2021. — 425 с.
11. Теорія та практика бізнес-комунікацій / за ред. С. М. Ромата. — Кондор-Видавництво, 2020. — 356 с.
12. Фрейд З. Психологія мас та аналіз Я / З. Фрейд. — Фоліо, 2018. — 192 с.
13. Joseph D. L. Emotional intelligence: An integrative meta-analysis and cascading model / D. L. Joseph, D. A. Newman // Journal of Applied Psychology. — 2010. — № 95 (1). — С. 54–78.
14. The New Science of Customer Emotions [Electronic resource] // Harvard Business Review. — 2015. — Mode of access: https://hbr.org/2015/11/the-new-science-of-customer-emotions (date of access: 15.05.2025).
15. Utilizing Emotional Intelligence in the Workplace [Electronic resource] // Verywell Mind. — 2023. — Mode of access: https://www.verywellmind.com/utilizing-emotional-intelligence-in-the-workplace-4164713 (date of access: 18.05.2025).

References

Anderson, K. (2019). *Psykholohiia doviry v orhanizatsiiakh.* Nauka i osvita [in Ukrainian].

Bove, K. L., & Till, J. V. (2020). *Suchasni biznes-komunikatsii.* Pirson [in Ukrainian].

Damasio, A. (2003). *U poshukakh Spinozy: radist, pechal i vidchuttia mozku.* Kharkort [in Ukrainian].

Doslidzhennia emotsiinoho intelektu ta zadovolennosti robotoiu. (2017). *Ukrainska pravda. Zhyttia.* https://life.pravda.com.ua/health/2017/10/15/226957/ (date of access: 20.05.2025) [in Ukrainian].

Emotsiinyi intelekt u biznesi. (2019). (D. Houlman, Ed.). *Emotsiinyi intelekt u biznesi.* Nash format [in Ukrainian].

Emotsiinyi intelekt yak faktor uspikhu v biznesi. (2022). *Delo.ua.* https://delo.ua/career/emotional-intelligence-business-success-390847/ (date of access: 22.05.2025) [in Ukrainian].

Freid, Z. (2018). *Psykholohiia mas ta analiz Ya.* Folio [in Ukrainian].

Joseph, D. L., & Newman, D. A. (2010). Emotional intelligence: An integrative meta-analysis and cascading model. *Journal of Applied Psychology*, *95* (1), 54–78.

Kaneman, D. (2011). *Dumai shvydko i povilno.* Farrar, Straus i Zhiru.

Meier, J. D., Salovei, P., & Karuzo, D. R. (2008). Emotsiinyi intelekt: nova zdatnist chy eklektychna sumish osobystisnykh rys?. *Amerikanskyi psykholoh*, *63* (6), 503–517 [in Ukrainian].

Psykholohichna bezpeka na robochomu mistsi: ukrainskyi dosvid. (2023). *Mind.ua.* https://mind.ua/publications/20240156-psihologichna-bezpeka-na-robochomu-misciukrayinskij-dosvid (date of access: 25.05.2025) [in Ukrainian].

Psykholohiia upravlinnia personalom. (2021). (O. I. Bondarchuk, Ed.). Lohos [in Ukrainian]

Teoriia ta praktyka biznes-komunikatsii. (2020). (S. M. Romata, Ed.). Kondor-Vydavnytstvo [in Ukrainian].

The New Science of Customer Emotions. (2015). *Harvard Business Review.* https://hbr.org/2015/11/the-new-science-of-customer-emotions (date of access: 15.05.2025).

Utilizing Emotional Intelligence in the Workplace. (2023). *Verywell Mind.* https://www.verywellmind.com/utilizing-emotional-intelligence-in-the-workplace-4164713 (date of access: 18.05.2025).

O. Khriapa

THE EMOTIONAL ASPECT OF BUSINESS COMMUNICATION: RETHINKING THE ROLE OF NEGATIVE EMOTIONS IN THE PROFESSIONAL ENVIRONMENT

This article examines the role of emotions in business communications, specifically analyzing the place of so-called "negative" emotions in the professional environments. The author critically reconsiders the traditional approach to distinguishing between business communications and everyday interaction, emphasizing the importance of the emotional component in effective corporate communication. Based on an analysis of contemporary emotional intelligence research and practical case studies, the study demonstrates that authentic emotional expression, including negative emotions, can contribute to building trust and improving organizational effectiveness.

The research employs an interdisciplinary approach, combining theoretical developments from business psychology, organizational behavior, and communication studies with a practical analysis of successful communication strategies. Particular attention is given to examining Ukrainian civil society and volunteer organizations that demonstrated exceptional effectiveness during the times of crisis, offering an alternative leadership model based on authenticity and emotional openness.

The study reveals the systematic stigmatization of negative emotions in corporate environments through normative, discursive, structural, and gender-based mechanisms. However, research shows that negative emotions perform important adaptive functions: anger mobilizes energy to overcome obstacles, anxiety signals potential threats and stimulates careful planning, while sadness facilitates goal reassessment and learning from failures. The key finding is that the constructiveness of negative emotions depends not on their suppression, but on the manner of their expression and integration into the communication process.

The analysis demonstrates that the emotional connection between organizational participants serves as a fundamental factor in building trust, which reduces transaction costs, increases readiness for cooperation, and promotes innovation. Companies that build emotional connections with clients based on shared values and authentic communication show higher loyalty rates and financial performance compared to competitors who are focused solely on functional product characteristics.

The study concludes that the traditional dichotomy between rational business communication and emotional personal interaction is artificial and counterproductive. Emotions are an integral component of any form of human communication, and their suppression makes their influence less predictable and harder to control. The research provides theoretically grounded recommendations for rethinking traditional approaches to emotional expression in corporate environments and for developing emotional literacy among managers.

Keywords: business communication, emotional intelligence, negative emotions, trust, organizational effectiveness, authenticity, corporate culture.

Матеріал надійшов 25.06.2025

ВІДОМОСТІ ПРО АВТОРІВ

Андрощук Максим Віталійович — здобувач ступеня доктора філософії, галузь «Комп'ютерні науки», факультет інформатики Національного університету «Києво-Могилянська академія», maxym.androshchuk@ukma.edu.ua

Афонін Андрій Олександрович — кандидат фізико-математичних наук, викладач кафедри мультимедійних систем факультету інформатики Національного університету «Києво-Могилянська академія», afonin@ukma.edu.ua

Бабич Трохим Анатолійович — студент PhD програми «Комп'ютерні науки», старший викладач кафедри інформатики, заступник декана факультету інформатики Національного університету «Києво-Могилянська академія», t.babych@ukma.edu.ua

Беймук Володимир Олегович — випускник бакалаврської програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», volodymyr.beimuk@ukma.edu.ua

Бублик Володимир Васильович — кандидат фізико-математичних наук, доцент кафедри мультимедійних систем факультету інформатики Національного університету «Києво-Могилянська академія», boublik@ukma.edu.ua

Бучко Олена Андріївна — PhD технічних наук, доцент факультету інформатики Національного університету «Києво-Могилянська академія», olena.buchko@ukma.edu.ua

Ванін Данило Олегович — випускник магістерської програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», danylo.vanin@ukma.edu.ua

Войтішин Микита Євгенович — студент бакалаврської програми «Прикладна математика» факультету інформатики Національного університету «Києво-Могилянська академія», mykyta.voitishyn@ukma.edu.ua

Волинець Євгеній Анатолійович — кандидат фізико-математичних наук, старший викладач факультету інформатики Національного університетут «Києво-Могилянська академія», ye.volynets@ukma.edu.ua

Гаврилюк Володимир Дмитрович — випускник бакалаврської програми «Інженерія програмного забезпечення» факультету інформатики Національного університету «Києво-Могилянська академія», vol.havryliuk@ukma.edu.ua

Глибовець Андрій Миколайович — професор, доктор технічних наук, декан факультету інформатики Національного університету «Києво-Могилянська академія», a.glybovets@ukma.edu.ua

Глибовець Микола Миколайович — доктор фізико-математичних наук, заслужений діяч науки і техніки України професор кафедри інформатики факультету інформатики Національного університету «Києво-Могилянська академія», glib@ukma.edu.ua

Гороховський Кирило Семенович — старший викладач кафедри мультимедійних систем факультету інформатики Національного університету «Києво-Могилянська академія», fintech.lab@ukma.edu.ua

Гороховський Семен Самуїлович — кандидат фізико-математичних наук, завідувач кафедри інформатики факультету інформатики Національного університету «Києво-Могилянська академія», gor@ukma.edu.ua

Дубовик Андрій Вікторович — студент бакалаврської програми «Інженерія програмного забезпечення» факультету інформатики Національного університету «Києво-Могилянська академія», andrii.dubovyk@ukma.edu.ua

Зважій Дмитро Володимирович — аспірант програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», старший викладач кафедри мережних технологій факультету інформатики Національного університету «Києво-Могилянська, d.zvazhii@ukma.edu.ua

Іващенко Дмитро Сергійович — студент PhD програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», d.ivashchenko@ukma.edu.ua

Кирієнко Оксана Валентинівна — старший викладач кафедри інформатики факультету інформатики Національного університету «Києво-Могилянська академія», o.kyriienko@ukma.edu.ua

Кузьменко Дмитро Олександрович — студент PhD програми «Комп'ютерні науки», старший викладач кафедри мультимедійних систем Національного університету «Києво-Могилянська академія», kuzmenko@ukma.edu.ua

Левченко Артем Сергійович — студент бакалаврської програми «Інженерія програмного забезпечення» факультету інформатики Національного університету «Києво-Могилянська академія», artem.levchenko@ukma.edu.ua

Ліпський Даниїл Олександрович — аспірант освітньо-наукової програми «Прикладна математика» факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка, lipsky@knu.ua

Марченко Олександр Олександрович — доктор фізико-математичних наук, професор кафедри математичної інформатики факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка, omarchenko@univ.kiev.ua

Махаммедов Жан Жанович — студент магістерської програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», zhan.makhammedov@ukma.edu.ua

Медвідь Сергій Олександрович — старший викладач, студент PhD програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», s.medvid@ukma.edu.ua

Микитишин Артем Павлович — студент бакалаврської програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», artem.mykytyshyn@ukma.edu.ua

Михайленко Олександр Ігорович — випускник магістерської програми «Інженерія програмного забезпечення» факультету інформатики Національного університету «Києво-Могилянська академія», oleksandr.mykhailenko@ukma.edu.ua

Моголівський Віталій Олегович — студент PhD програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», v.moholivskyi@ukma.edu.ua

Мокрий Михайло Вікторович — студент PhD програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», m.mokryy@ukma.edu.ua

Нагірна Алла Миколаївна — кандидат фізико-математичних наук факультету інформатики Національного університету «Києво-Могилянська академія», a.nahirna@ukma.edu.ua

Нагнибіда Андрій Андрійович — студент PhD програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», a.nahnybida@ukma.edu.ua

Олецький Олексій Віталійович — кандидат технічних наук, доцент, доцент кафедри мультимедійних систем факультету інформатики НаУКМА, oletsky@ukma.edu.ua

Петелєв Євгеній Русланович — магістр з комп'ютерних наук, заступник директора з технологічних досліджень та розробок, ТОВ «МАКПАУ», zhenya.peteliev@macpaw.com

Пізь Мар'яна Андріївна — випускниця бакалаврської програми «Інженерія програмного забезпечення» факультету інформатики Національного університету «Києво-Могилянська академія», marianapiz0412@gmail.com

Плахотна Дар'я Олександрівна — студентка бакалаврської програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», daria.plakhotna@ukma.edu.ua

Постніков Михайло Андрійович — бакалавр інженерії програмного забезпечення, Національний університет «Києво-Могилянська академія», mykhailo.postnikov@ukma.edu.ua

Проценко Володимир Семенович — кандидат фізико-математичних наук, доцент, доцент кафедри інформатики факультету інформатики Національного університету «Києво-Могилянська академія», v.protsenko@ukma.edu.ua

Сидорова Єлізавєта Олександрівна — студентка магістерської програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», yelizavieta.sydorova@ukma.edu.ua

Смиш Олег Русланович — доктор філософії з комп'ютерних наук, старший викладач кафедри мультимедійних систем факультету інформатики Національного університету «Києво-Могилянська академія», o.smysh@ukma.edu.ua, https://orcid.org/0000-0002-8074-9745

Соболевська Леся Георгіївна — асистент кафедри автоматизації технологічних процесів, Київський національний університет будівництва та архітектури, sobolevska.lg@knuba.edu.ua

Суліменко Андрій Андрійович — студент бакалаврської програми «Інженерія програмного забезпечення» факультету інформатики Національного університету «Києво-Могилянська академія», andrii.sulimenko@ukma.edu.ua

Ткаченко Владислав Олександрович — студент аспірантської програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», vo.tkachenko@ukma.edu.ua

Тригуб Олександр Семенович — кандидат фізико-математичних наук, доцент кафедри інформатики факультету інформатики НаУКМА, oleksandr.tryhub@ukma.edu.ua

Фітель Данило Романович — випускник магістерської програми «Прикладна математика» факультету інформатики Національного університету «Києво-Могилянська академія», головний інженер-програміст корпорації «Майкрософт», danylo.fitel@outlook.com

Франків Олександр Олександрович — студент PhD програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», o.frankiv@ukma.edu.ua

Франчук Олег Васильович — кандидат технічних наук, доцент кафедри мережних технологій факультету інформатики Національного університету «Києво-Могилянська академія»; старший науковий співробітник відділу створення та використання інтелектуальних мережних інструментів, НЦ «Мала академія наук України», м. Київ, Україна, o.franchuck@ukma.edu.ua

Хряпа Оксана Ігорівна — старша викладачка факультету інформатики Національного університету «Києво-Могилянська академія», Head of HR/ Head of security L&D EIS Ukraine company, okhriapa@ukma.edu.ua

Чоловський Сергій Олександрович — аспірант програми «Комп'ютерні науки» факультету інформатики Національного університету «Києво-Могилянська академія», s.cholovskyi@ukma.edu.ua

Швай Надія Олександрівна — доцент, кандидат фізико-математичних наук, факультет інформатики Національного університету «Києво-Могилянська академія», n.shvay@ukma.edu.ua

Швець Дмитро Віталійович — студент бакалаврської програми «Інженерія програмного забезпечення» факультету інформатики Національного університету «Києво-Могилянська академія», d.shvets@ukma.edu.ua, https://orcid.org/0009-0002-7681-9744

Яременко Петро Всеволодович — студент магістерської програми «Інженерія програмного забезпечення» факультету інформатики Національного університету «Києво-Могилянська академія», yyaremenko.p@gmail.com

ABOUT THE AUTHORS

Maksym Androshchuk — PhD student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, maxym.androshchuk@ukma.edu.ua

Andrii Afonin — Candidat of Physical and Mathematical Sciences, Department of Multimedia Systems, National University of Kyiv-Mohyla Academy, afonin@ukma.edu.ua

Trokhym Babych — PhD Student in Computer Science, Deputy Dean, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, t.babych@ukma.edu.ua

Volodymyr Beimuk — Bachelor's graduate in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, volodymyr.beimuk@ukma.edu.ua

Volodymyr Boublik — Candidate of Physical and Mathematical Sciences, Asociate Professor, Department of Multimedia Systems, National University of Kyiv-Mohyla Academy, boublik@ukma.edu.ua

Olena Buchko — PhD in Technical Sciences, Senior Lecturer, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, olena.buchko@ukma.edu.ua

Serhii Cholovskyi — PhD student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, s.cholovskyi@ukma.edu.ua

Andrii Dubovyk — Bachelor's student in Software Engineering, Faculty of Computer Science, National University of Kyiv-Mohyla Academy, andrii.dubovyk@ukma.edu.ua

Danylo Fitel — Master's graduate in Applied Mathematics, National University of Kyiv-Mohyla Academy; Principal Software Engineer at Microsoft Corporation, danylo.fitel@outlook.com

Oleg Franchuk — Candidate of Technical Sciences, Associate Professor, Department of Network Technologies, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy; Senior Researcher, Department of Creating and Using Intelligent Networking Tools, the NC "Junior Academy of Sciences of Ukraine", Kyiv, Ukraine, o.franchuck@ukma.edu.ua

Oleksandr Frankiv — PhD student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, o.frankiv@ukma.edu.ua

Mykola Glybovets — Doctor of Physical and Mathematical Sciences, Honored Worker of Science and Technology of Ukraine, Professor, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, glib@ukma.edu.ua

Kyrylo Gorokhovskyi — Senior Lecturer, Department of Multimedia Systems, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, fintech.lab@ukma.edu.ua

Semen Gorokhovskyi — Candidate of Physical and Mathematical Sciences, Head of the Department of Informatics, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, gor@ukma.edu.ua

Volodymyr Havryliuk — Bachelor's graduate in Software Engineering, Faculty of Computer Science, National University of Kyiv-Mohyla Academy, vol.havryliuk@ukma.edu.ua

Andrii Hlybovets — Professor, Doctor of Technical Sciences, Dean of the Faculty of Computer Science, National University of Kyiv-Mohyla Academy, a.glybovets@ukma.edu.ua

Dmytro Ivashchenko — PhD student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, d.ivashchenko@ukma.edu.ua

Oksana Khriapa — Senior Lecturer, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, Head of HR and Head of Security L&D, EIS Ukraine company, okhriapa@ukma.edu.ua

Dmytro Kuzmenko — PhD student in Computer Science, Senior Lecturer, Department of Multimedia Systems, National University of Kyiv-Mohyla Academy, kuzmenko@ukma.edu.ua

Oksana Kyriienko — Senior Lecturer, Department of Informatics, National University of Kyiv-Mohyla Academy, o.kyriienko@ukma.edu.ua

Artem Levchenko — Bachelor's student in Software Engineering, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, artem.levchenko@ukma.edu.ua

Danyil Lipskyi — PhD student in Applied Mathematics, Faculty of Computer Science and Cybernetics, Taras Shevchenko National University of Kyiv, lipsky@knu.ua

Zhan Makhammedov — Master's student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, zhan.makhammedov@ukma.edu.ua

Oleksandr Marchenko — Doctor of Physical and Mathematical Sciences, Professor, Department of Mathematical Informatics, Faculty of Computer Science and Cybernetics, Taras Shevchenko National University of Kyiv, omarchenko@univ.kiev.ua

Sergii Medvid — PhD student, Senior Lecturer, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, s.medvid@ukma.edu.ua

Vitalii Moholivskyi — PhD student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, v.moholivskyi@ukma.edu.ua

Mykhailo Mokryi — PhD student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, m.mokryy@ukma.edu.ua

Oleksandr Mykhailenko — Master's graduate in Software Engineering, National University of Kyiv-Mohyla Academy, oleksandr.mykhailenko@ukma.edu.ua

Artem Mykytyshyn — Bachelor's student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, artem.mykytyshyn@ukma.edu.ua

Alla Nahirna — Candidate of Physical and Mathematical Sciences, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, a.nahirna@ukma.edu.ua

Andrii Nahnybida — PhD student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, a.nahnybida@ukma.edu.ua

Oleksii Oletsky — PhD, Associate Professor, National University of Kyiv-Mohyla Academy, oletsky@ukma.edu.ua

Yevhenii Peteliev — Master's graduate in Computer Science, Deputy Director for Technological Research and Development, MacPaw LLC, zhenya.peteliev@macpaw.com

Mariana Piz — Bachelor's graduate in Software Engineering, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, marianapiz0412@gmail.com

Daria Plakhotna — Bachelor's student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, daria.plakhotna@ukma.edu.ua

Mykhailo Postnikov — Bachelor's graduate in Software Engineering, National University of Kyiv-Mohyla Academy, mykhailo.postnikov@ukma.edu.ua

Volodymyr Protsenko — Candidate of Physical and Mathematical Sciences, Associate Professor, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, v.protsenko@ukma.edu.ua

Nadiya Shvai — Candidate of Physical and Mathematical Sciences, Associate Professor, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, n.shvay@ukma.edu.ua

Dmytro Shvets — Bachelor's student in Software Engineering, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, d.shvets@ukma.edu.ua, https://orcid.org/0009-0002-7681-9744

Oleh Smysh — PhD in Computer Science, Senior Lecturer, Department of Multimedia Systems, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, o.smysh@ukma.edu.ua, https://orcid.org/0000-0002-8074-9745

Lesia Sobolevska — Assistant Professor, Department of Process Automation, Kyiv National University of Construction and Architecture, sobolevska.lg@knuba.edu.ua

Andrii Sulimenko — Bachelor's student in Software Engineering, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, andrii.sulimenko@ukma.edu.ua

Yelizavieta Sydorova — Master's student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, yelizavieta.sydorova@ukma.edu.ua

Vladyslav Tkachenko — PhD student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, vo.tkachenko@ukma.edu.ua

Oleksandr Tryhub — Candidate of Physical and Mathematical Sciences, Associate Professor, Department of Informatics, National University of Kyiv-Mohyla Academy, oleksandr.tryhub@ukma.edu.ua

Danylo Vanin — Master's graduate in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, danylo.vanin@ukma.edu.ua

Mykyta Voitishyn — Bachelor's student in Applied Mathematics, Faculty of Computer Science, National University of Kyiv-Mohyla Academy, mykyta.voitishyn@ukma.edu.ua

Yevhenii Volynets — PhD in Physics and Mathematics, Lecturer, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, ye.volynets@ukma.edu.ua

Pavlo Yaremenko — Master's student in Software Engineering, Faculty of Computer Sciences, the National University of Kyiv-Mohyla Academy, yyaremenko.p@gmail.com

Dmytro Zvazhii — PhD student in Computer Science, Faculty of Computer Sciences, National University of Kyiv-Mohyla Academy, d.zvazhii@ukma.edu.ua

ЗМІСТ

CONTENTS

НАУКОВІ ЗАПИСКИ НаУКМА
КОМП'ЮТЕРНІ НАУКИ

NaUKMA RESEARCH PAPERS
COMPUTER SCIENCE

Том 8

Редакторка і коректорка *О. Пазюк*
Комп'ютерна верстка *М. Кулікова*

Адреса редакції:
вул. Г. Сковороди, 2, м. Київ, 04070,
тел.: (044) 463-66-68

Свідоцтво про внесення до Державного реєстру видавців, виготівників і розповсюджувачів
книжкової продукції серія ДК № 3631 від 23.11.2009