УДК 004.89
DOI: 10.18523/2617-3808.2022.5.12-15

Жежерун О. П., Рєпкін М. С.

АВТОМАТИЧНА ГЕНЕРАЦІЯ ОНТОЛОГІЙ НА ОСНОВІ СТАТЕЙ УКРАЇНСЬКОЮ МОВОЮ

У статті представлено систему, яка здатна генерувати нові онтології або доповнювати наявні на основі статей українською мовою. Описано онтології та алгоритм, який доречний для використання у автоматизованому виокремленні концептів із текстів природною мовою.

Ключові слова: ontology engineering, база знань, онтологія, Protégé, аналіз природної мови, аналіз української мови.

Вступ

Після створення Semantic Web онтологія стала синонімом рішення багатьох проблем, пов'язаних із розумінням природної мови комп'ютерами. Якщо б існувала онтологія та за її допомогою було би проаналізовано кожен документ, то ми б мали системи, які можуть відповідати на дуже складні запити природною мовою. Через декілька років успіх Google показав, що завантажувати HTML-сторінки набагато простіше, ніж розмічати все семантичною розміткою, витрачаючи людський інтелектуальний ресурс. Щоб знайти рішення цієї проблеми, з'явився новий напрям в онтологічній сфері – онтологічна інженерія. Цей напрям почав вивчати шляхи автоматизації генерування знань, які консолідувались би онтологією з тексту [1].

У процесі цієї роботи було розглянуто задачу автоматизованої генерації онтології з використанням статей із української Вікіпедії, а як приклад предметної галузі було взято геометрію. Побудовано систему, яка збирає та аналізує дані й формує з них онтологію.

Огляд онтологій

Поняття онтології походить з античної філософії і описує «теорію існування природи» [3]. Аристотель запропонував свою онтологію з примітивними категоріями, як-от сутність та якість. У комп'ютерній сфері онтологія означає річ, яку було зроблено для моделювання знань із певної предметної галузі. Термін впровадили дослідники штучного інтелекту, які визнали працездатність математичної логіки. У 1980-х спільнота дійшла висновку, що онтологія не лише стосується теорії змодельованого світу, а і є одним із модулів систем знань [6].

Рекурентні нейронні мережі

Рекурентні нейронні мережі – це клас нейронних мереж, де лінки між нейронами формують направлений граф у часі. Це дає змогу досягти динамічної поведінки. Цей клас нейронних мереж використовує внутрішню пам'ять, щоб обробляти довгі вхідні дані, що дає можливість використовувати їх у мовному аналізі [2]. Такі мережі поділяють на два класи:

- фільтр із кінцевою імпульсною характеристикою – представлено у вигляді направленого ациклічного графу;
- фільтр із безкінечною імпульсною характеристикою – представлено у вигляді направленого графу.

Обидва види мають додаткові стани, що зберігаються у мережі. Сховище станів може бути замінено на іншу нейронну мережу або граф [5].

Одна з ідей застосування рекурентної нейронної мережі полягає в тому, що вони можуть зв'язувати попередню інформацію з поточною задачею, щоб минуле допомагало у розумінні теперішнього. Проте якщо між теперішнім і пояснювальним моментом у минулому пройшло достатньо часу, то рекурентні мережі втрачають можливість зв'язувати дані. Через це на допомогу приходить мережа LSTM (Long short-term memory) – архітектура рекурентних нейронних мереж, яку створено для запам'ятовування інформації на тривалий час. Уперше її представили Зеп Хохрайтер і Юрген Шмідхубер у 1997 р. [4].

Покроковий розбір роботи мережі:

© *Жежерун О. П., Рєпкін М. С.,* 2022

- Визначити, які дані можна відкинути. Це вирішує сигмоїдальна функція.
- Визначити нову інформацію, яка буде зберігатись. Це вирішує функція tanh.
- Заміна старих і нових даних для збереження в комірці.
- Останній крок – визначення інформації для виведення. Це вирішує сигмоїдальна та tanh функції.

У роботі використано нейронну мережу LSTM для POS(Part of speech) маркування.

Огляд даних сторінки Вікіпедії

Для здійснення першого пункту роботи системи, а саме – автоматизованого генерування онтологій на основі тексту, як джерело даних було використано україномовну Вікіпедію. Було вибрано саме цю енциклопедію, оскільки вона є відкритою, має достатній розмір, містить близько мільйона статей, поділених на категорії, які допомагали з побудовою ієрархії. Сторінку Вікіпедії умовно можна розбити на п'ять складових: назва, список категорій, до яких вона належить, стислий опис, повний опис і посилання на інші сторінки Вікіпедії. Під стислим описом маємо на увазі перший абзац, в якому лаконічно описано подію або концепт, що є темою статті. На момент написання системи було зібрано 972 918 статей, 234 837 категорій, 15 964 137 посилань однієї статті на іншу.

Огляд системи

Побудована система має архітектуру пайплайн (рис. 1). Це зроблено для можливості більш гнучкого горизонтального масштабування. Система має одне джерело даних, чотири кроки оброблення даних із джерела, одну вихідну онтологію, в якій консолідуються дані після оброблення, і крок із доповнення згенерованої онтології вже наявною. Всі п'ять сервісів написано мовою Python.

Етап 1. Data getters. Робиться http запит до Wikipedia. У відповідь програма отримує html-сторінку вибраної сторінки. За допомогою бібліотеки BeautifulSoup витягуються всі посилання на цій сторінці та додаються до черги посилань, які треба обробити, відбувається дедуплікація.

Етап 2. Data extract&filter. На цьому етапі витягується текст із таких HTML-тегів: h1, h2, h3, h4, h5, h6, p. У них зберігається повністю вся стаття без втрат і не міститься шуму, наприклад посилання на інші статті чи сторонні ресурси, хедер сайту та інше. На цьому ж кроці відбувається фільтрація слів за стоп-словами. Серед стоп-слів можна виокремити такі: а, в, буду, будь ласка, важливі, деякий та інші. Загалом це 1983 слова. Список цих слів було взято з відкритого джерела [7].

Етап 3. POS tagging. Тут відбувається тегування слів за частиною мови для покращення якості подальшого аналізу – за допомогою моделі машинного навчання з архітектурою Bidirectional LSTM. Окрім слів тут кодується пунктуація окремим кодом "PUNCT".

Етап 4. File storage. Акумуляція файлів у файловому сховищі у вигляді json документів. На кожну статтю створюється окремий файл зі структурою з одним рівнем вкладеності з такими полями: article_name – назва статті, categories – категорії статті, raw_html – необроблена html розмітка для можливості подальшого оброблення без звертання до Вікіпедії, redirects – метаінформація, що містить можливі синоніми до назви статті, text – виокремлений текст, links – посилання на інші статті Вікіпедії.

Етап 5. Hierarchy extraction. На цьому кроці ми зчитуємо всі файли зі сховища та будуємо наступні файли для подальшого оброблення: article->category.txt – відношення усіх статей до категорій, category->category.txt – опис категорій, які мають підкатегорії, article->article.txt – стаття, що посилається на інші статті.

Після цього будується зважений граф на основі файлів із відношенням категорій до категорій та з відношенням назв статей до категорій. Вага зв'язків у графі обраховується з кількості зв'язків у конкретного вузла. Через те, що категорії у Вікіпедії не формують чіткого дерева, то з цього неможливо побудувати ієрархії онтології. Для виокремлення структури з цього графу з циклами застосовуємо алгоритм «Max spanning tree», що знаходить дерево в графі, яке важить найбільше. Тут система має можливість обрати якусь тему, про що буде онтологія, наприклад «Математика», дати цьому вузлу велику вагу, що змусить алгоритм включити його в результу-

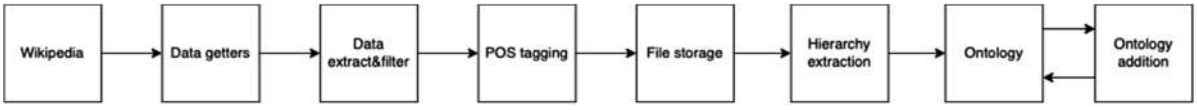

Рис. 1. Архітектура системи

юче дерево. У такий спосіб можна будувати ієрархію для онтології з будь-якої теми.

В онтологію також додаються терміни та зберігаються посилання на статті, де вони були використані. В термінів є зв'язок із «батьківським» терміном, який є назвою статті, а дочірній термін – це слово, під яким було додано гіперпосилання на статтю з батьківським терміном.

На виході ми отримуємо одну або декілька онтологій з ієрархією категорій та статей у них, словник термінів і посилання на статті, де вони використовуються.

Етап 6. Ontology addition. На цьому етапі ми отримуємо дані з попередніх кроків і робимо такі:

1. Розбиваємо статтю по словах і на двограми та триграми з урахуванням частин мов.
   a. Беруться тільки іменники, дієслова, прикметники, тому що вони з найбільшою ймовірністю будуть у складі термінів.
2. Перекладаємо всі дані англійською мовою, тому що ми будемо використовувати семантичну базу даних Wordnet, яка не має підтримки української. Переклад робиться після розбиття на ці групи, адже в такому випадку менше втрачається сенс речень.
3. Починаємо обробляти слово/двограм/триграм.
   a. Шукаємо це слово в наявній онтології, якщо знаходимо його, то вибираємо наступне слово.
   b. Якщо такого слова немає, дістаємо з Wordnet усі синоніми вибраного слова.
   c. Шукаємо це слово в наявній онтології, якщо знаходимо його, то беремо наступне слово.
   d. Якщо таких слів немає, то ми починаємо йти по графу слів Wordnet і шукаємо ієрархію слів, які мають задане слово як початковий кінцевий вузол, а слово, пов'язане зі словом «Математика», як початковий вузол. Максимальна дистанція між початковим і кінцевими вузлами має бути меншою ніж 4.
   e. Якщо таку ієрархію було знайдено, перевіряємо, чи вся ієрархія є новою для онтології. Доповнюємо наявну ієрархію або дописуємо її, починаючи з найвищого рівня.

Перед записом перекладаємо назад українською. Як приклад було взято тему з Wikipedia про математику. В результаті ми отримали потрібну онтологію. Загалом, із теми «Математика» вдалося отримати 8800 концептів із максимальною глибиною ієрархії 5. Всього виокремили 20 тис. термінів із посиланнями на різні статті. З Wordnet було дописано 700 концептів.

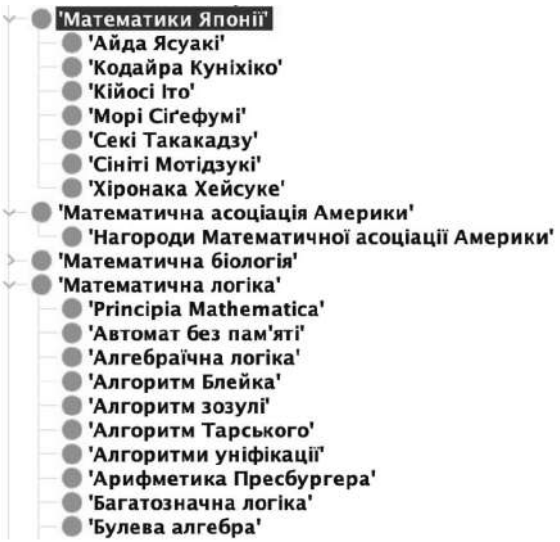

Рис. 2. Частина згенерованої онтології

Можливості розширення

У системи є декілька шляхів розвитку: 1) пошук альтернативи Wordnet для української мови або побудова власного тезаурусу; 2) удосконалення алгоритму виокремлення концептів із тексту та додавання їх до онтології; 3) прискорення часу роботи програми.

Висновки

Проаналізувавши історію онтологій та їхні можливості, можна сказати, що вона була б доречною для використання в пошукових системах через свої можливості ієрахізації знань і логічного виведення. Однак її не було використано через брак алгоритмів автоматизованої генерації онтологій із тексту. В статті запропоновано алгоритм, який вирішує цю проблему.

Список літератури

1. Biemann C. Ontology Learning from Text: A Survey of Methods [Electronic resource] / Chris Biemann. – 2005. – Mode of access: https://www.researchgate.net/publication/200044378_Ontology_Learning_from_Text_A_Survey_of_Methods.
2. Graves A. A Novel Connectionist System for Unconstrained Handwriting Recognition / A. Graves, M. Liwicki, S. Fernández et al. // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2009. – Vol. 31, Issue 5. – Pp. 855–868.

3. Liu L. Encyclopedia of Database Systems / L. Liu. – Boston, MA : Springer, 2009.
4. LSTM – сети долгой краткосрочной памяти [Електронний ресурс]. – 2017. – Режим доступу: https://habr.com/ru/company/wunderfund/blog/331310/.
5. Miljanovic M. Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction / Milos Miljanovic // Indian Journal of Computer Science and Engineering. – 2012. – Vol. 3 (1). – Pp. 180–191.
6. Sowa J. Conceptual Structures: Information Processing in Mind and Machine The Systems Programming Series / John Sowa. – Addison-Wesley, 1984.
7. Ukrainian Stopwords [Електронний ресурс]. – Режим доступу: https://github.com/skupriienko/Ukrainian-Stopwords/blob/master/stopwords_ua_list.txt.

References

Biemann, C. (1996). *Ontology Learning from Text: A Survey of Methods*. https://www.researchgate.net/publication/200044378_Ontology_Learning_from_Text_A_Survey_of_Methods.

Graves, Alex, Liwicki, Marcus, Fernández, Santiago, Bertolami, Roman, Bunke, Horst, & Schmidhuber, Jürgen. (2009). A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE transactions on pattern analysis and machine intelligence*, *31* (5), 855–868.

Liu, L. (2009). *Encyclopedia of Database Systems*. Springer.

LSTM – sety dolhoi kratkosrochnoi pamiaty. (2005). https://habr.com/ru/company/wunderfund/blog/331310/ [in Russian].

Miljanovic, Milos. (2012). Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction. *Indian Journal of Computer Science and Engineering*, *3* (1), 180–191.

Sowa, John. (1984). *Conceptual Structures: Information Processing in Mind and Machine The Systems Programming Series*. Addison-Wesley.

Ukrainian Stopwords. https://github.com/skupriienko/Ukrainian-Stopwords/blob/master/stopwords_ua_list.txt.

O. Zhezherun, M. Ryepkin

AUTOMATIC GENERATION OF ONTOLOGIES BASED ON ARTICLES WRITTEN IN UKRAINIAN LANGUAGE

The article presents a system capable of generating new ontologies or supplementing existing ones based on articles in Ukrainian. Ontologies are described and an algorithm suitable for automated concept extraction from natural language texts is presented.

Ontology as a technology has become an increasingly important topic in contemporary research. Since the creation of the Semantic Web, ontology has become a solution to many problems of understanding natural language by computers. If an ontology existed and was used to analyze documents, then we would have systems that could answer very complex queries in natural language. Google's success showed that loading HTML pages is much easier than marking everything with semantic markup, wasting human intellectual resources. To find a solution to this problem, a new direction in the ontological field, called ontological engineering, has appeared. This direction began to study ways of automating the generation of knowledge, which would be consolidated by an ontology from the text.

Humanity generates more data every day than yesterday. One of the main levers today in the choice of technologies for the implementation of new projects is whether it can cope with this flow of data, which will increase every day. Because of this, some technologies come to the fore, such as machine learning, while others recede to the periphery, due to the impossibility or lack of time to adapt to modern needs, as happened with ontologies. The main reason for the decrease in the popularity of ontologies was the need to hire experts for its construction and the lack of methods for automated construction of ontologies.

This article considers the problem of automated ontology generation using articles from the Ukrainian Wikipedia, and geometry was taken as an example of the subject area. A system was built that collects data, analyzes it, and forms an ontology from it.

Keywords: ontology engineering, knowledge base, ontology, Protégé, natural language analysis, Ukrainian language analysis.

Матеріал надійшов 12.09.2022

Creative Commons Attribution 4.0 International License (CC BY 4.0)