УДК 004.89
DOI: 10.18523/2617-3808.2019.2.38-42

Жежерун О. П., Рєпкін М. С.

КЛАСИФІКАЦІЙНА СИСТЕМА З ПІДБОРУ ПЕРСОНАЛУ

У статті представлено класифікаційну систему, побудовану у вигляді бази знань, яка допомагає розв'язати задачу класифікації під час підбору персоналу. Ця система порівнюється із системами на основі нейронних мереж. Проведено огляд засобів побудови онтологій, доречних для застосування у подібних системах.

Описано функціонал та реалізацію класифікаційної системи на основі фреймворку Protégé, що співпрацює з Telegram-ботом і дає змогу ефективно відбирати кандидатів, які найбільше відповідають вимогам вакансії.

Ключові слова: класифікаційна система, база знань, онтологія, Protégé, Telegram-бот.

Вступ

Основною метою класифікації є розбиття множини елементів на категорії таким чином, щоб ці елементи мали достатню кількість однакових ознак, що дає змогу не брати до уваги їхні індивідуальні відмінності [1].

Існують системи класифікації, що працюють на базі нейронних мереж, зокрема використовують метод «навчання з вчителем». Такі системи є достатньо ефективними, популярними і найчастіше використовуються за наявності великих масивів даних. Робота нейронної мережі є ефективнішою за бази знань, оскільки натренована мережа обчислює результати швидше. Проте такі системи мають певні недоліки, наприклад у випадку роботи зі структурою, де часто змінюються вимоги до даних. Треба мати велику кількість даних для повноцінного тренування нейронних мереж. Для задач, в яких об'єм даних для тренування неможливо здобути, або де вимоги до даних часто змінюються, використовують бази знань.

У цій роботі описано класифікаційну систему, що використовує онтології, області реалізації таких систем і технології їх побудови.

Методи класифікації

Головна ідея класифікатора на основі нейронної мережі – це патерн, що подається до мережі як активація набору певних нейронів. Ця активація поширюється мережею через зв'язки, внаслідок чого активовані фінальні нейрони видають класифікаційний результат [5].

Іншим методом класифікації є використання онтологій і різонерів, що працюють з онтологіями. Більшість OWL різонерів, таких як «Pellet», «FaCT++», «RacerPro», вирішують проблему класифікації алгоритмом «Enhanced Traversal». Enhaced Traversal – алгоритм, який дає змогу будувати представлення відношення класів в онтології та додавати очевидні відношення [4].

Онтології для побудови баз знань, засоби реалізації та застосування

Сучасні бази знань працюють разом із системами пошуку інформації. Для цього необхідна певна модель класифікації понять і визначений формат представлення знань. Ієрархічний спосіб представлення набору понять та відношень між ними в базі знань називають онтологією [2]. Крім того, що бази знань та бази даних будуються різними способами, їхня головна відмінність – використання в базах знань методів логічного виводу дискриптивної логіки з припущенням про відкритість світу. Припущенням про відкритість світу називають припущення в формальній логіці про те, що істинність твердження не залежить від того, чи «відома» будь-якому спостерігачу правильність цього твердження. Це впливає на те, які дані вважають такими, що логічно випливають із бази знань [7].

Для представлення онтологій використовують мову під назвою OWL (Web Ontology Language). На цей момент вважають актуальною другу версію мови OWL, яка має декілька різновидів: OWL 2 DL, OWL EL, OWL QL, OWL RL, OWL 2 FULL. У роботі було використано різновид OWL DL, префікс якого походить від «Descriptive Logic», він створений для ефективного використання алгоритмів із виводу даних, що явно не прописані в онтології, але логічно випливають з неї [5].

Завдяки можливості побудови ієрархії та алгоритмам логічного виводу, бази знань часто використовують у рекомендаційних, класифікаційних, пошукових системах тощо.

Опис функціоналу та реалізації бази знань для підбору персоналу

Ідея системи полягає у відборі найкращих кандидатів із великої кількості резюме, що надсилають користувачі системи на певні вакансії компанії. Найкращий кандидат – це той, хто має найбільшу кількість таких навичок, що вказані в описі вакансії, або навичок із суміжних галузей знань. Навички у вакансії ранжуються за важливістю. Як приклад предметної області взято веб-розробку, зокрема фреймворки, мови програмування та технології, що використовують у цій сфері.

Protégé є найпопулярнішим редактором онтологій, який вміє експортувати побудовані ієрархії у декілька форматів, серед яких RDF/XML, OWL/XML та JSON-LD. RDF/XML – це заданий консорціумом «W3C» синтаксис серіалізації графа RDF у вигляді XML документа. «RDF» – модель даних для представлення ресурсів семантичної павутини [3]. OWL/XML – синтаксис для представлення онтології на мові OWL у вигляді XML документа [7]. JSON-LD (JavaScript Object Notation for Linked Data) – один із методів передання зв'язаних даних із використанням текстового формату JSON. Формат використовує поняття контексту для підтримки моделі даних RDF [6].

Telegram є зручним, і його часто використовують менеджери компаній для отримання анкет кандидатів на свої вакансії. Цей месенджер є найбільш захищеним із наявних, а тому популярним. Доданки у вигляді ботів мають широкий функціонал, а контроль над ними здійснюється через секретний ключ під назвою «API-TOKEN», який можна отримати в головного бота «BotFather», що робить здійснення несанкціонованого доступу до бота майже неможливим. Зважаючи на ці факти, було вирішено використати як постачальник даних Telegram-бот, що працює з онтологією, яка була експортована з Protégé у форматі OWL/XML.

У системі існує адміністратор та користувач. Звичайний користувач має можливість переглядати всі вакансії компанії та відправляти на кожну своє резюме з переліком власних навичок. Адміністратор може створювати нові вакансії, переглядати, редагувати (змінювати список необхідних навичок та їх пріоритет) та видаляти їх, а також переглядати список кандидатів на відкриті вакансії, що відсортовані за схожістю навичок вакансії та навичок із кожного резюме.

Визначення найкращого кандидата проходить таким чином:

- Якщо користувач знає таку саму технологію, що є в вакансії, то до загальної кількості балів додається таке число:

`(загальна кількість навичок у вакансії – скільки вже було проаналізовано) * k1.`

- Якщо користувач знає технологію із суміжної області знань, то

`(загальна кількість навичок у вакансії – скільки вже було проаналізовано) * k2.`

На основі проведених експериментів було виведено такі значення коефіцієнтів: $k1 = 2$, $k2 = 1.5$.

Для того щоб отримати відсортований список, необхідно виконати такі кроки:

1. Адміністратор має створити нову вакансію з описом та навичками.
2. Користувачі мають надіслати своє резюме, де вони матимуть можливість обрати свої навички зі списку. Або, якщо такої немає, створити нову, що буде класифікована базою знань та записана в необхідне місце в ієрархії.
3. Адміністратори мають запустити програму класифікації.

Для можливості класифікації навичок та сортування кандидатів на вакансію було розроблено онтологію за допомогою редактора онтологій Protégé 5.5.0: http://www.semanticweb.org/maksim/ontologies/2019/2/getTheBestCandidate. Вона задає ієрархію всієї бази знань, зокрема навичок, які використовують для підбору персоналу. Ієрархія має такий вигляд:

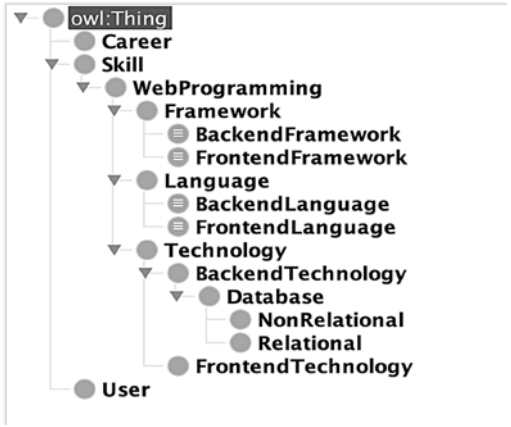

Рис. 1. Побудована онтологія

Career

У цьому класі описані характеристики вакансій, що подає адміністратор. Цей клас має object property під назвою «requireSkill», в якому

зберігається інформація про необхідні навички для вакансії.

User

Цей клас описує резюме, які залишили користувачі на обрану вакансію. Має object property з назвою «hasSkill», в якому описані всі навички, які має кандидат.

Skill

Клас зберігає в собі ієрархію навичок, які стосуються веб-програмування. Має такі object property: «usesFramework», «usesLanguage», які є інверсивними та зберігають інформацію про відношення між навичками.

Логіка системи зумовила таку архітектуру проекту (рис. 2).

Рис. 2. Архітектура проекту

app

Python-клас, у якому здійснюється основна частина, де прописані взаємодії бота та користувача. Після запуску цього Python-класу завантажується вся онтологія, перевіряється на помилковість, робляться логічні висновки з неї та записуються назад до файлу з розширенням owl.

constants

У цьому Python-класі містяться всі необхідні дані про доданок, які можна змінити в будь-який час. Тут можна знайти:

- Bot API token;
- Інтерфейси взаємодії для різних типів користувачів;
- Методи динамічного наповнення інтерфейсів взаємодії;
- Масиви для роботи з даними різних класів онтології.

user

У цьому Python-класі описано логіку для взаємодії з класом User в онтології. Тут можна знайти методи з генерації унікального номера для користувача, отримання даних у необхідному форматі для виведення на екран через Telegram-додаток.

skill

Тут можна знайти логіку взаємодії з класом Skill в онтології. У цьому класі наявні методи генерації унікального номера для навичок, отримання даних у двох форматах (1 – в масиві Python-класів, 2 – в масиві owl-онтології).

sorter

Тут зберігається логіка сортування кандидатів на вакансії за схожістю вимог та навичок кандидата.

ontologyInteraction

У цьому Python-класі міститься основна логіка взаємодії Python-класів та побудованої в Protégé онтології.

career

У цьому Python-класі здійснюється повний контроль над класом онтології Career, зокрема отримання даних про цей клас у потрібному форматі, запис до онтології новий вакансій та інше.

owlClasses

Цей Python-клас із переліком усіх класів бази знань було створено для уникнення проблем зі зчитуванням даних методами Python з онтології.

Для реалізації заданого функціоналу було використано pyTelegramBotApi 0.3.0 – бібліотека для програмування Telegram боті на мові Python, owlready2 – бібліотека для онтологічно-орієнтованого програмування на Python [8].

Під час тестування системи було розглянуто вакансію, яка має такі вимоги: знання фреймворку Laravel та технології Blockchain. Вона була занесена до бази знань через Telegram-бот, для цього були використані Python-класи «ontologyInteraction» та «career». З Python-об'єкта Protégé перетворив вакансію у формат OWL/XML і зберіг її в такому вигляді в базу знань:

```
<Career rdf:about="#BackendDeveloperCareer">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
  <requireSkill rdf:resource="#Blockchain"/>
  <requireSkill rdf:resource="#Laravel"/>
  <career_id rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</career_id>
  <career_name>Backend developer</career_name>
</Career>
```

Рис. 3. Розглянута вакансія

Після застосування різонеру HermiT було автоматично додано нову навичку до цієї вакансії, яка логічно випливає з неї, тому що фреймворк «Laravel» написаний на мові «PHP» і це відображено в онтології, і тепер вакансія виглядає у базі знань таким чином:

```
<Career rdf:about="#BackendDeveloperCareer">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
  <requireSkill rdf:resource="#Blockchain"/>
  <requireSkill rdf:resource="#Laravel"/>
  <requireSkill rdf:resource="#PHP"/>
  <career_id rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</career_id>
  <career_name>Backend developer</career_name>
</Career>
```

Рис. 4. Оновлена вакансія

На цю вакансію було подано 300 анкет, більшість із яких було згенеровано за допомогою Python. Крім штучно згенерованих резюме було створено декілька через Telegram-бот, одна з таких анкет має повний збіг навичок із даними з вакансії. Приклад анкети з точним збігом із вакансією, що була відправлена за допомогою Telegram-бота (рис. 5).

Після генерації анкет було застосовано Python-клас «sorter», який, за описаною вище формулою, визначив найкращих кандидатів. У результаті роботи системи анкета, з такими самими навичками, посіла третє місце з найбільш релевантних. Вище за неї опинилися анкети, в яких окрім точного збігу з вакансією вказані навички з суміжних галузей знань. Нижчі місця посіли анкети, що не мають точних збігів, але максимально схожі на вказані в вакансії.

Було протестовано змінення важливості навичок у вакансії. Кожного разу найвищі позиції були розподілені між однаковими анкетами. Вакансія з точним збігом не змінювала своєї позиції у списку. Гірші, на думку системи, анкети завжди займали різні місця.

Під час розробки цієї системи для підбору персоналу було виявлено недолік у вигляді отримання вищої позиції у списку резюме, в яких вказано велику кількість навичок, жодна з котрих точно не збігається з навичками в вакансії, тобто вони з суміжних галузей, порівняно з резюме з невеликою кількістю навичок, але з наявністю точних збігів із вакансією.

Рис. 5. Приклад анкети

Висновки

Після аналізу можливостей онтології можна зробити висновок, що така технологія є доречною у вирішенні розглянутої проблеми, завдяки своїм класифікаційним можливостям та можливостям логічного виводу. Також із цього аналізу випливає, що онтологія має великі переваги над нейронними мережами у контексті розглянутої задачі, бо нам не потрібно заново тренувати модель кожного разу, коли змінюються дані. На великих об'ємах даних це є критичним.

Список літератури

1. Глибовець А. М. Інтелектуальні мережі / А. М. Глибовець, М. М. Глибовець, М. В. Поляков. – Дніпропетровськ, 2014. – 462 с.
2. Рассел С. Искусственный интеллект: современный подход / С. Рассел, П. Норвиг ; пер. с англ. – Москва, 2006.
3. A Semantic Web Primer for Object-Oriented Software Developers [Electronic resource] / H. Knublauch, D. Oberle, P. Tetlow, P. Wallace // W3C Working Group Note. – 2006. – Mode of access: https://www.w3.org/TR/sw-oosd-primer/. – Title from the screen.

4. Cudre-Maroux P. The Semantic Web – ISWC 2012 / P. Cudre-Maroux, J. Heflin, E. Sirin. – Boston, MA, USA : 11th International Semantic Web Conference, 2012.
5. Dubey A. A Study of Classification Techniques Using Soft-Computing / Abhishek Dubey. – Bilaspur : Guru Ghasidas Vishwavidyalaya, 2011.
6. JSON-LD 1.1 [Electronic resource] // Draft Community Group Report. – 2019. – Mode of access: https://json-ld.org/spec/latest/json-ld/. – Title from the screen.
7. Lamy B. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies / B. Lamy. – Bobigny, France: Sorbonne University.
8. OWL Web Ontology Language Semantics and Abstract Syntax [Electronic resource] // W3C Recommendation. – 2004. – Mode of access: https://www.w3.org/TR/2004/REC-owl-semantics-20040210/. – Title from the screen.

References

Cudre-Maroux, P., Heflin, J., & Sirin E. (2012). *The Semantic Web – ISWC 2012*. Boston, MA, USA: 11th International Semantic Web Conference.

Draft Community Group Report. (2019). Retrieved from https://json-ld.org/spec/latest/json-ld/.

Dubey, A. (2011). *A Study of Classification Techniques Using Soft-Computing*. Bilaspur: Guru Ghasidas Vishwavidyalaya.

Hlybovets, A., Hlybovets, M., & Poliakov, M. (2014). *Intelektualni merezhi*. Dnipropetrovsk [in Ukrainian].

Knublauch, H., Oberle, D., Tetlow, P., & Wallace, P. (2006). A Semantic Web Primer for Object-Oriented Software Developers. *W3C Working Group Note*. Retrieved from https://www.w3.org/TR/sw-oosd-primer/.

Lamy, B. (2017). *Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies*. Bobigny, France: Sorbonne University.

OWL Web Ontology Language Semantics and Abstract Syntax. (2004). Retrieved from https://www.w3.org/TR/2004/REC-owl-semantics-20040210/.

Rassel, S., & Norvyh, P. (2006). *Iskusstvennyi intellekt: sovremennyi podkhod*. Moskva [in Russian].

O. Zhezherun, M. Riepkin

CLASSIFICATION SYSTEM FOR STAFF SELECTION

The main goal of classification is to split a plurality of elements into categories so that these elements have a sufficient number of identical features, which allows to ignore their individual differences. There are classification systems based on neural networks, which use the method of "supervised learning". Such systems are effective and popular, and they are usually used on big amounts of data. The work of such a network is more effective than the knowledge base, because a well-trained network calculates results faster. Nevertheless, neural networks have a problem of necessity to retrain the network on the structure change. Ontology-based classification systems solve this problem and allow to work with it right away due to semantic reasoning. Such systems are also used when it is impossible to get the data-set for the neural-network training. Due to these reasons the ontology-based classification system is more suitable for resolving the described problem.

Protégé is the most popular ontology editor which is able to export ontologies in various formats (RDF/XML, OWL/XML, JSON-LD) and provide a number of semantic reasoners (Pellet, HermiT, FaCT). Telegram is one of the most secure and popular messengers. Moreover, Telegram bots have a wide functionality. Taking into consideration these facts, it was decided to implement the system which helps company managers to select the best candidates for their vacancies, according to the required skills and the skills of a candidate. The system was implemented by using the ontology exported from the Protégé in OWL/XML format to define the hierarchy, semantic reasoner HermiT to classify data, and Python programming language to implement the bot's functionality.

A description of the structure, functionality, test-case, and the main problems of the system is given in the article.

Keywords: classification system, knowledge base, ontology, Protégé, Telegram-bot.

Матеріал надійшов 16.05.2019