УДК 004.89 https://doi.org/10.15407/pp2020.04.034

О.П. Жежерун, М.С. Рєпкін

КЛАСИФІКАЦІЙНА СИСТЕМА З ПІДБОРУ ПЕРСОНАЛУ, БАЗОВАНА НА АНАЛІЗАТОРІ УКРАЇНСЬКОЇ МОВИ

У статті розглядається класифікаційна системах, яка базується на аналізі природньої мови. В багатьох таких системах використовуються нейронні мережі, проте вони потребують даних для навчання, які не завжди наявні. Автори пропонують використання онтологій в подібних системах аналізу природньої мови. В якості прикладу представлено класифікаційну систему, яка допомагає сформувати список найкращих кандидатів під час підбору персоналу. Представлено огляд методів побудови онтологій та мовних аналізаторів, доречних для класифікаційних систем, і побудовано систему у вигляді бази знань. Здійснена підтримка української та англійської мов у класифікаційній системі. Описані можливості розширення системи.

Ключові слова: класифікаційна система, база знань, онтологія, Protégé, аналіз природньої мови, аналіз української мови.

Вступ

Сьогодні у світі розробки програмного забезпечення та бізнесу зростає тенденція із зменшення присутності людини у повсякденних завданнях, особливо це стосується завдань монотонних або легко алгоритмізованих. Одна з причин цієї тенденції – значний зріст обсягу робіт такого типу, які людям обробити стає все важче. Разом із такою проблемою прогрес приносить і часткові вирішення у вигляді зростання обчислювальної потужності сучасних комп'ютерів. Це дає інженерам можливості використання таких алгоритмів, які виконуватимуть монотонні завдання за розумний час [1].

У ході цієї роботи, яка є продовженням статті «Класифікаційна система з підбору персоналу» [2], було розглянуто одне з таких завдань та запропоновано метод його вирішення. Це завдання вибору кандидатів серед великого списку претендентів на одну з посад у компанії. Щодня рекрутери переглядають десятки чи сотні резюме та обирають найбільш релевантні. Ефективність такого пошуку не є високою, тому було вирішено автоматизувати саме цей процес. Як основу системи було побудовано онтологію, яка описує ієрархію та відношення навичок, навчальних закладів та компаній для найбільш ефективного та обґрунтованого вибору кандидатів.

Онтології для побудови баз знань, засоби реалізації та застосування

Поняття онтології з'явилось ще за часів античності, більше ніж дві тисячі років до появи інформаційних технологій. У ті часи поняття онтології з'явилось у межах філософської науки. Онтологія у філософії – це вчення про буття, у якому з'ясовуються фундаментальні проблеми існування [3].

В інженерії "онтологічні" методи зазвичай застосовуються для побудови моделей процесів. Інженерна модель процесу і є онтологією. Точніше, онтологіє описує цю модель. Такі описи моделей є формальними, тобто зроблені спеціальною для цього мовою, конструкції якої завжди інтерпретуються точно та однозначно. Це дозволяє, наприклад, перевірити чи не існує в цьому процесі логічних протиріч [4]. Для зручної побудови необхідної онтології було обрано інструмент Protégé. Цей додаток надає зручний інтерфейс та містить в собі багато допоміжного інструментарію: збереження побудованої онтології у восьми різних форматах, вісім найпопулярніших різонерів та автоматична генерація Java коду на основі онтології.

Для побудови онтології система Protégé оперує наступними поняттями:

- Class – головний елемент будь-якої онтології. За їх допомогою описується ієрархія предметної області.

- Individual – екземпляр класу.
- Data property – властивості класів. Зазвичай є простими типами даних, наприклад integer/string.
- Object property – зв'язки класів між собою.

Одним з необхідних інструментів, який надає Protégé, є різонер. Це програмний застосунок, який аналізує побудовану ієрархію та зв'язки між її сегментами. Він є універсальним, тому не обов'язково будувати власний різонер під свою онтологію. Проте, якщо заданого функціоналу та точності не вистачає, то, звичайно, доведеться дописувати свої модулі до існуючого рішення [5]. Ось їх список:

- ELK
- FaCT
- HermiT
- Mastro DL-Lite Reasoner
- Ontop
- Pellet
- Pellet(Incremental)
- Jcel

Алгоритми обробки та аналізу тексту

1. Стемінг. В інформаційному пошуку такий алгоритм як стемінг відомий дуже давно. Він є базою для цієї гілки програмної інженерії. Цей алгоритм дозволяє знаходити основу слова, при цьому вона може не збігатися з коренем слова, тому цей алгоритм є евристикою. Проте він є основним алгоритмом компанії Google. Морфологія на основі стемінгу має декілька переваг. Наприклад, завдяки зменшенню обсягу інформації зростає швидкість аналізу. Найголовнішою перевагою цього алгоритму є той факт, що навіть за відсутності словника основ слів ми отримуємо морфологічну базу необмеженого розміру. При цьому морфологія ніколи не скаже, що такого слова немає у словнику [6].

Серед недоліків – невисока точність методу та неможливість синтезу на базі без основ. Без основи ми не зможемо зрозуміти чи є слово дією, яку необхідно виконати над певним об'єктом, чи це об'єкт цієї дії і т. д.

У результатах роботи алгоритму стемінгу ми бачимо, що у всіх випадках просто відкидається закінчення слова: в англійській мові закінчення, що характеризує множину об'єкту – "s", дієслово в минулому часі – "ed". Для коректної роботи алгоритму треба оновлювати список таких закінчень в залежності від мови, яку ви аналізуєте.

2. Лематизація. Як зазначено вище, стемінг – це кит, на якому стоїть інформаційний пошук. Лематизація – це другий кит. Цей алгоритм схожий на стемінг за своєю метою, яка полягає у зменшенні кількості інформації та приведення до однієї основи [7]. На відміну від стемінгу, який не гарантує, що результатом його роботи буде корінь слова, лематизація гарантує виокремлення кореня, або трансформацію слова до кореня. Для роботи цього методу недостатньо зберігати закінчення та приставки слів у вибраній мові. Цей алгоритм вимагає збереження всього словника мови, тому що, наприклад, до слів у множині може не тільки додаватись відповідне закінчення, а ще повністю змінюватись слово. Теж саме можна сказати про дієслова у різних часах.

3. POS-tagging (Part Of Speech tagging) – алгоритм класифікації слів за частинами мови, який також є основним алгоритмом інформаційного пошуку. Сам по собі алгоритм є низького рівня, за допомогою якого можна побудувати свою пошукову систему. Алгоритм не має певної послідовності дій, адже в кожній мові існують свої правила морфології, але найчастіша імплементація – це виокремлення закінчення слова, його аналіз і класифікація всього слова [8].

Також використовується аналіз порядку слів для мов із фіксованим порядком у реченні. У роботі використано бібліотеку nltk для англійскьої та pullenti для української, які класифікують наступні типи слів:

- CC – координуючі сполучники
- DT – коми, крапки…
- FW – іноземне слово
- JJ – прикметник
- JJR – порівняльна форма прикметника
- JJS – вища порівняльна форма
- NN – іменник

- VB – дієслово (інфінітив)
- VBD – дієслово в минулому часі
- та інші…

Огляд побудованої онтології

Онтологія, яка була побудована у редакторі Protégé показана на рис. 1.

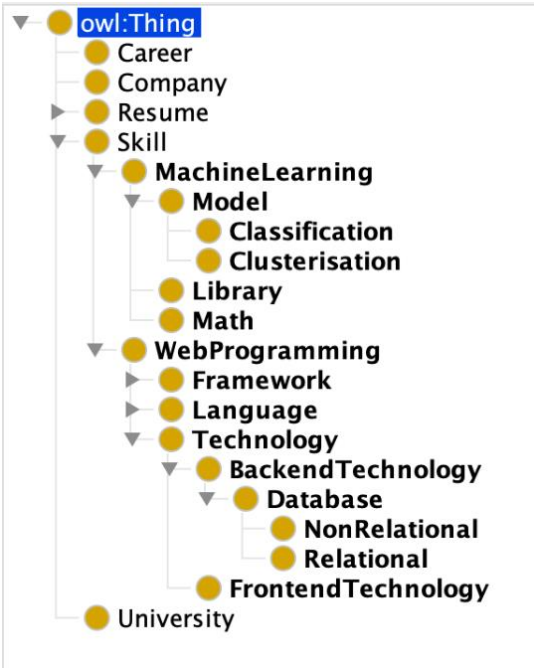

Рис. 1. Побудована онтологія

Далі опишемо тільки основні класи онтології.

Career. Клас використовується для опису вакансій в компанії та є підкласом головного класу “Thing”, тобто є класом найвищого рівня в цій онтології. Має object property “requireSkill”, що описує необхідні навички для посади. Та має такі data properties: “career_id”, “career_name”.

Resume. Є абстрактним класом та зберігає дані про користувачів, які лишили свої резюме на обрану вакансію. Має наступні data properties: “resume_id” – унікальний номер резюме, “resume_file_name” – шлях, де збережено файл із резюме, “resume_link” – всі покликання в файлі з резюме, “candidate_terms” – усі можливі терміни, які зустрілись в резюме, але вони ще не наявні в базі знань, щоб адміністратор зміг занести їх в один з класів онтології. Має object property – “resumeHasSkill”, що зберігає посилання на всі навички, що було знайдено у файлі з резюме.

Company. В цьому класі зберігаються усі IT компанії за 2020 рік . Має наступні data property – “company_id” – унікальний номер компанії, “company_name” – назва компанії.

University. В цьому класі зібрані всі університети України . Має data property – “university_id” – унікальний номер університету, “university_name” – назва університету.

Skill. Є абстрактним класом, він визначає можливості користувачів та необхідні навички для здобуття визначеної роботи. Має наступні data properties – “skill_id” – визначає унікальний номер навички та “skill_name” – ім’я навички, що буде показуватись користувачу.

Огляд розробленого аналізатора текста

Побудований аналізатор текста працює з файлами у формати “PDF”. Коли кандидат надсилає своє резюме, система зберігає його на диск в директорію, яка відповідає обраній кандидатом вакансії. Після цього аналізатор починає свою роботу. Далі будуть описані основні етапи роботи аналізатора з прикладами.

Етап 1. Прийняття резюме та попередня обробка (рис. 2).

Цей етап передбачає виконання наступних кроків.

1. Відкриття файлу.
2. Видалення всіх несуттєвих символів, які не містять інформації. Наприклад, серед них є “\n”, “\r”, “|” та інші.
3. Пошук та видалення в тексті всіх адрес електронної пошти та телефонних номерів українського стандарту. Пошук здійснюється за допомогою регулярних виразів. Все, знаходиться на цьому етапі, зберігається в онтології в класі “Resume” в data property “resume_link” та “resume_email”.
4. Переведення тексту до нижнього регістру.
5. Видалення “стоп слів”, які не містять інформації. Список цих слів було взято з бібліотеки “get_stop_words” (рис. 3).

Після виконання усіх зазначених кроків першого етапу надіслане резюме має вигляд (рис. 4).

Максим Рєпкін

ryepkin.maksym98@gmail.com | https://github.com/maxflexx

Навички

· Python, C++

· Знання базових алгоритмів комп'ютерного зіру(Canny edge detection, contour curvature, segmentation, boundary tracing, erosion)

· Лінійна алгебра

· Теорія ймовірностей

· Алгоритми та структури даних

· Розуміння алгоритмів машинного навчання

Власні проекти

В рамках курсової роботи було розроблено систему запису на вибіркові дисципліни за допомогою технології Laravel та HTML.

https://github.com/maxflexx/kursova_php

Реалізував алгоритм, який допомагає знаходити схожі картинки(однакові, картинки з додатковими ефектами, однакові об'єкти з інших кутів), використовуючи ssim/

https://github.com/maxflexx/image-similarity

Рис. 2. Приклад надісланого резюме

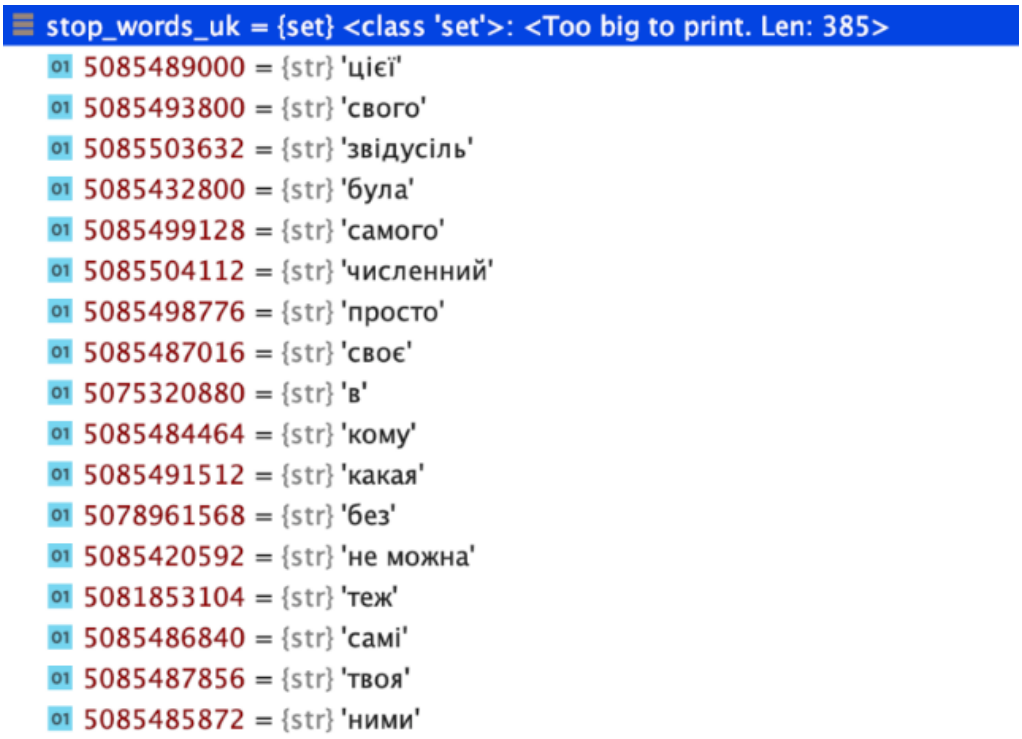

Рис. 3. Приклад «стоп-слів»

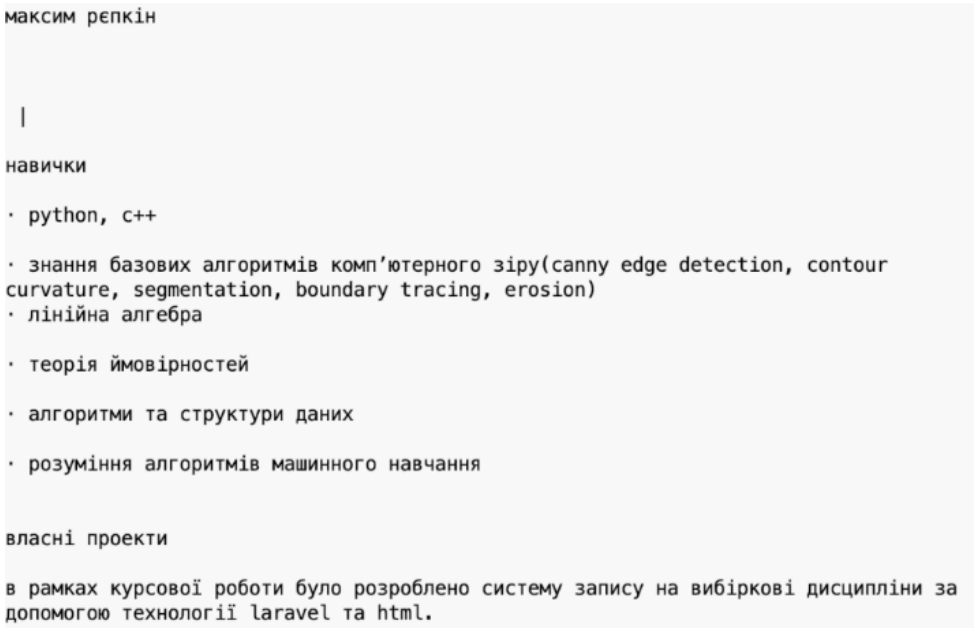

Рис. 4. Резюме після першого етапу

Етап 2. На цьому етапі здійснюється пошук по термінах, які вже наявні в онтології. Під час цього процесу виділяються терміни таких типів.

1. Компанії, в яких кандидат працював раніше. База IT компаній була взята з сайту dou.ua і нараховує 7400 позицій.
2. Університет, в якому навчався кандидат. База університетів була взята з сайту vstup.info.
3. Навички, якими володіє кандидат. Цю базу було створено вручну, враховуючи всі популярні мови програмування, фреймворки та інші технології.

Після виконання зазначених кроків резюме має вигляд (рис. 5).

Етап 3. На цьому етапі здійснюється пошук усіх можливих термінів. Для виконання цього завдання здійснюється лематизація та POS-tagging. Можливі терміни мають задовольняти такі умови.

1. Бути іменником, прикметником або дієсловом.
2. Якщо це термін з кількох слів, то між словами не повинно бути розділових знаків та пропусків рядків.

Результат виконання цього етапу виглядає так (рис. 6).

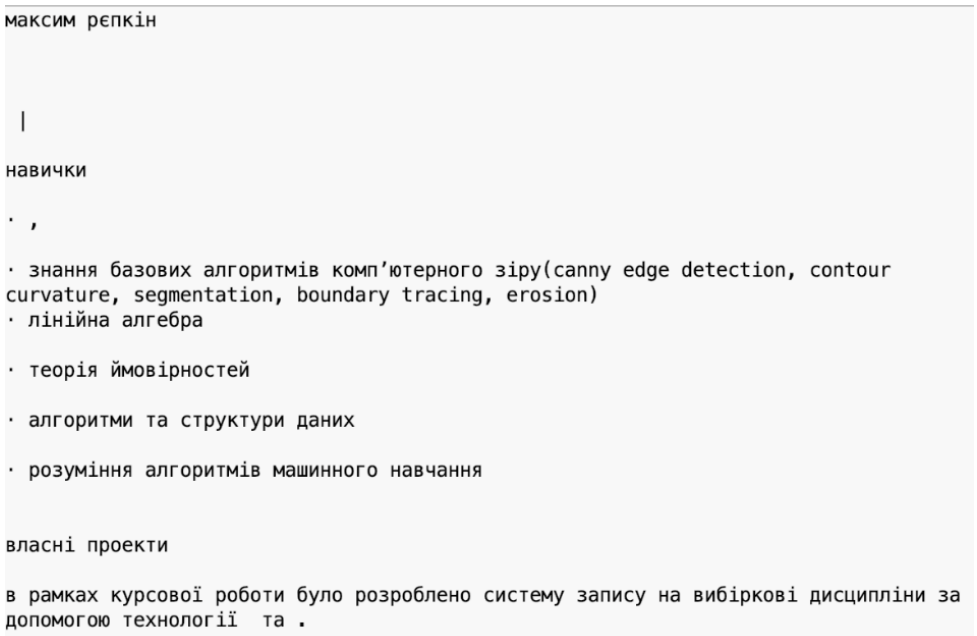

Рис. 5. Резюме після другого етапу

```
00 = {str} 'максим'
01 = {str} 'навичка'
02 = {str} 'знання базовий алгоритм компютерний зір'
03 = {str} 'лінійний алгебра'
04 = {str} 'теорія ймовірність'
05 = {str} 'алгоритм структура даний'
06 = {str} 'розуміння алгоритм машинний навчання'
07 = {str} 'власний проект'
08 = {str} 'рамка курсовий робота розробити система запис вибірковий дисципліна'
09 = {str} 'допомога технологія'
```

Рис. 6. Результат третього етапу

Всі можливі терміни зберігаються в онтології в класі “CandidateTerms”, щоб адміністратор мав змогу продивитись ці терміни та прийняти рішення щодо того, чи дійсно вони є термінами, які необхідно додати.

Швидкість аналізу резюме складає приблизно 1MB / секунда. Це достатньо швидко, зважаючи на те, що загалом розмір резюме не перевищує 5MB.

Класифікаційна система здатна аналізувати дві мови: українську і англійську. Для англійської всі кроки не відрізняються, проте використовується бібліотека “nltk”, а не “pullenti”. Система побудована таким чином, що додавання нових мов у аналізатор буде відбуватись без переписування або редагування вже наявного коду, що робить її легко розширюваною.

Можливості розширення

Є декілька шляхів розширення представленої системи:

1. Розширення онтології, щоб робити більш широку класифікацію.
2. Удосконалення системи аналізу природньої мови. Додавання нових мов та оптимізація здійснених операцій, щоб підвищити швидкість аналізу.
3. Додавання можливості горизонтального розширення, що дозволить запускати кластери машин, які будуть аналізувати резюме та додавати їх до онтології.
4. Додавання можливості робити висновки щодо університетів, наприклад, “Випускники університету N більше розуміються на технологій K”.
5. Додавання можливості робити висновки щодо компаній. Так можна скласти нову онтологію: Компанія-Технологія.

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 з цього аналізу можна зробити висновок, що онтологія має великі переваги над нейронними мережами в контексті розглянутої задачі, бо нам не потрібно заново тренувати модель кожного разу, коли змінюються дані. На великих об’ємах даних це є критичним.

Література

1. Glybovets A., Glybovets M., Polyakov M. Intelligent networks. NaUKMA. Dnipropetrovsk. 2014. 462 p.
2. Жежерун О.П., Рєпкін М.С. Класифікаційна система з підбору персоналу. Наукові записки НаУКМА. 2019.
3. Шинкарук В. І. Філософський енциклопедичний словник. Київ: Інститут філософії імені Григорія Сковороди: Абрис. 2002. 742 с.
4. Лапшин В. А. Онтологии в информационных системах. Современный подход. Москва. 2009.
5. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies [Електронний ресурс].

6. Lovins Julie Beth. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*. 1968. T. 11.
7. Manning Christopher D., Raghavan Prabhakar, Schütze Hinrich. Introduction to Information Retreival. Cambridge University Press.
8. DeRose Steven J. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 1988.

References

1. Glybovets A., Glybovets M., Polyakov M. Intelligent networks. NaUKMA. Dnipropetrovsk. 2014. 462 p.
2. Zhezherun O., Repkin M. Classification system for personnel selection. *Scientific Notes of NaUKMA*. 2019.
3. Shynkaruk V. Philosophical encyclopedic dictionary. Kyiv: Grigory Skovoroda Institute of Philosophy: Abris. 2002. 742 p.
4. Lapshin V. Ontologies in information systems. Modern approach. Moscow. 2009.
5. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies [Електронний ресурс].
6. Lovins Julie Beth. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*. 1968. T. 11.
7. Manning Christopher D., Raghavan Prabhakar, Schütze Hinrich. Introduction to Information Retreival. Cambridge University Press.
8. DeRose Steven J. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 1988.

Одержано 18.10.2020

Про авторів:

Жежерун Олександр Петрович,
кандидат фізико-математичних наук,
завідувач кафедри мультимедійних систем факультету інформатики.
Кількість наукових публікацій в українських виданнях – більше 80.
Індекс Гірша – 4.

Рєпкін Максим Сергійович,
студент 1 року навчання магістерської програми «Інженерія програмного забезпечення» факультету інформатики.
Кількість наукових публікацій в українських виданнях – 1.

Місце роботи авторів:

Національний університет
«Києво-Могилянська академія»
04655, Київ, вул. Г. Сковороди, 2.

E-mail: zhezherun@ukma.edu.ua,
ryepkin.maksym98@gmail.com