МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Центр післядипломної освіти
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)

(рівень вищої освіти)

Дослідження комбінованих методів вилучення структурованих даних для мов з вільним порядком слів

Виконала:
студентка 2 курсу групи ІПЗздм-21-2
Орлова Г.О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми Освітньо-наукова

Керівник доц. Кириченко І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. Кафедри _____ З.В. Дудар

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної Інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентки Орлової Ганни Олександрівни
(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження комбінованих методів вилучення структурованих даних для мов з вільним порядком слів

затверджена наказом університету від «03» 04 2023 р. № 83Стз

2. Термін подання студентом роботи до екзаменаційної комісії «01» 05 2023 р.

3. Вихідні дані до роботи розробка методики для вилучення фактів та подій з текстових даних, реалізація програмного інструменту на розробленій методиці, Проектування та створення додатка для вирішення завдання вилучення фактів із резюме.

4. Перелік питань, що потрібно опрацювати в роботі вивчення предметної галузі, аналітичний огляд існуючих методів вилучення та структурування інформації з тексту, розробка методики розв'язання задачі, вибір відповідних інструментів для розробки рішення, розробка бібліотеки на основі запропонованої методики, проектування та створення додатка для вирішення завдання вилучення фактів із резюме.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Огляд предметної галузі обробки природних мов	01.04.2023 - 05.04.2023	виконано
2	Комбінований підхід для отримання структурованих даних з тексту	06.04.2023 - 10.04.2023	виконано
3	Проектування та розробка програми з вилучення інформації з резюме на основі запропонованого підходу	11.04.2023 - 15.04.2023	виконано
4	Тестування програми з вилучення інформації	16.04.2023 - 20.04.2023	виконано
5	Підготовка пояснювальної записки	21.04.2023 - 25.04.2023	виконано
6	Підготовка презентації та доповіді	25.04.2023 - 30.04.2023	виконано
7	Перевірка роботи на плагіат та нормоконтроль	06.05.2023	виконано
8	Захист кваліфікаційної роботи	18.05.2023	виконано

Дата видачі завдання 01 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Кириченко І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 67 с., 27 рис., 2 табл. і 53 джер.

ОБРОБКА ПРИРОДНИХ МОВ, КОНТЕКСТНО-ВІЛЬНІ ГРАМАТИКИ, ГРАМАТИКИ ЗАЛЕЖНОСТЕЙ, ВИТЯГАННЯ ІНФОРМАЦІЇ, ВИТЯГАННЯ ФАКТІВ.

Об'єктом дослідження є поточна проблематика завдання по витягуванню і структуризації інформації з текстових даних. Метою роботи є розробка методики по витяганню фактів і подій з текстів, яка ґрунтується на комбінуванні підходів використання правил контекстно-вільних граматик спільно з аналізом синтаксичних дерев граматик залежностей.

В процесі дослідження проводиться аналіз поточної проблематики завдання і існуючих методів рішення, опис пропонованого підходу і існуючих програмних інструментів для його реалізації.

В результаті дослідження була реалізована програмна бібліотека, що дозволяє розробляти шаблони для витягання фактів з текстів українською мовою. Розроблено веб-застосування для аналізу документів резюме претендентів як рішення окремого випадку завдання. Зроблений висновок, що розроблена бібліотека показала свою успішну застосовність для вирішення цих задач.

В якості подальших кроків планується розширення функціонала для складання шаблонів, використання готових загальнодоступних тезаурусів / словників для можливості використання синонімів, публікація проекту у відкритий доступ, а також розробка банку основних шаблонів.

Сфери застосування: Аналіз різних документів і звітів з відкритих джерел, аналіз описів товарів, різних подій в новинах по певних темах, бренд аналітика.

NATURAL LANGUAGE PROCESSING, CONTEXT FREE GRAMMARS, DEPENDENCY GRAMMARS, INFORMATION EXTRACTION, FACT EXTRACTION.

The object of the study is the current problem of the task of extracting and structuring information from textual data. The purpose of the work is to develop a methodology for extracting facts and events from texts, which is based on combining the approaches of using rules on context-free grammars together with the analysis of syntactic trees of dependency grammars.

In the process of research, an analysis of the current problems of the task and existing solution methods, a description of the proposed approach and existing software tools for its implementation is carried out.

As a result of the research, a software library was implemented that allows you to develop templates for extracting facts from texts in the Ukrainian language. A web application has been developed for the analysis of applicants' resume documents as a solution to a particular case of the task. It is concluded that the developed library has shown its successful applicability for solving these problems.

As further steps, it is planned to expand the functionality for compiling templates, use ready publicly available thesauruses / dictionaries for the possibility of using synonyms, publish the project in open access, as well as develop a bank of basic templates.

Areas of application: Analysis of various documents and reports from open sources, analysis of product descriptions, various events in the news by certain topic, brand analytics.

Умови публікації пояснювальної записки

Я, Орлова Ганна Олександрівна
(прізвище, ім’я, по батькові)
студентка групи ІПЗздм-21-2 здобувач вищої освіти на другому (магістерському) рівні

кафедра програмної інженерії
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему «Дослідження комбінованих методів вилучення структурованих даних для мов з вільним порядком слів», що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

На сьогодні в епоху Великих даних об'єми виробленою людством інформації більші ніж коли-небудь і її кількість росте з кожним днем. Людина вже не в змозі вручну обробляти, аналізувати, витягати знання з неструктурованих даних, переважно текстових, і передавати їх по різних каналах. У зв'язку з цим особливої актуальності набула необхідність перетворення текстів, написаних природною мовою в структуроване представлення для застосування в прикладних завданнях.

Під витяганням інформації мається на увазі пошук в слабо структурованих документах окремих фактів, які являють собою певний інтерес. Сьогодні вже існують рішення, що дозволяють витягати з текстів різні іменовані сутності з прийнятною якістю, проте завдання по витяганню їх відносин, фактів і подій є не дуже пропрацьованим і тому найбільш актуальним.

Основною метою цієї роботи є розробка методики по витягання фактів і подій з текстів, яка ґрунтується на комбінуванні підходів використання правил на контекстно-вільних граматиках спільно з аналізом синтаксичних дерев граматик залежностей. Ця методика є спробою вирішити цю задачу для мов з вільним порядком слів, зокрема для української мови.

1 ОГЛЯД ПРЕДМЕТНОЇ ГАЛУЗІ ОБРОБКИ ПРИРОДНИХ МОВ

Обробка природних мов (Natural Language Processing – NLP) є загальним напрямом досліджень, пов'язаним з вивченням проблем розуміння і аналізу текстів або розмови природною мовою комп'ютером, для вирішення прикладних завдань [1-3]. Дослідники цієї області прагнуть збирати і систематизувати знання про те, як люди розуміють і використовують природну мову, щоб створювати і розробляти методи, інструменти, системи і програми для вирішення задач. Область NLP спирається на ряд дисциплін, таких як: комп'ютерні науки, математика, лінгвістика, штучний інтелект, психологія.

До найбільш поширених завдань в NLP можна віднести машинний переклад, автоматичне реферування тексту, розпізнавання і синтез мови, генерацію тексту [1-2]. Також обробка природних мов використовується для аналізу тексту, такого як класифікація (сентимент-аналіз, фільтрація спаму та ін.), створення пошукових і питально-відповідних систем, включаючи віртуальних співрозмовників, завдання по витяганню інформації та ін.

1.1 Історія становлення обробки природних мов

Прийнято вважати, що історія обробки природної мови розпочалася з 1950-х років, після публікації Аланом Тюрингом статті "Computing Machinery and Intelligence" [4]. У ній він запропонував емпіричний тест, в подальшому названий Тестом Тюрінга, метою якого було визначення здібностей у машини здійснювати дії, які не відрізняються від обдуманих дій людини замість того, щоб розглядати питання чи "Можуть машини думати?". Суть тесту полягала в наступному: суддя спілкується з одним комп’ютером та однією людиною природною мовою. Завдання судді визначити, хто із співрозмовників людина, а хто робот. Завдання комп'ютера

– спілкуватися із співрозмовником так, щоб ввести суддю в оману і змусити думати, що він насправді людина.

Одночасно з цим, почався пошук рішень завдань машинного перекладу. Відправною точкою став Джорджтаунський експеримент, який відбувся 7 січня 1954 року в штаб-квартирі IBM в Нью-Йорку [5]. В ході експерименту був показаний повністю автоматичний переклад більше 60 речень з російської мови на англійську. Система, яка лежала в основі цього перекладу, була досить проста: в її лексиконі було всього 6 граматичних правил і внутрішній словник з 250 лексичних одиниць.

У ці ж роки з'являються перші діалогові програми і системи, які ведуть з людиною діалог природною мовою. Першими віртуальними співрозмовниками, які отримали широкий розголос, були комп'ютерні програми ELIZA [6] і SHRDLU [7], розроблені в 1966 і 1968 роках відповідно.

ELIZA – це програма-співрозмовник, яка пародіювала діалог з психотерапевтом. Уся її робота полягає в перефразовуванні висловлювань співрозмовника. Вона брала значимі слова (дієслова і іменники) з введеної людиною пропозиції і підставляла їх в шаблонну фразу. Наприклад, на висловлювання: "У мене болить голова", вона відповідала: "Чому ви говорите, що у вас болить голова"? (див. рисунок 1.1). На ключові слова "мати", "батько", "син" ELIZA могла попросити детальніше розповісти про сім'ю [6]. В ході діалогу у багатьох людей створювалася ілюзія спілкування з людиною, хоч і короткочасна.

На відміну від ELIZA, програма SHRDLU в якості інтерфейсу приймає від користувача звичайні вирази англійською мовою. Після отримання команди вона переміщала прості об'єкти у свій "світ блоків" – кубики, кулі, конуси різних кольорів і розмірів. Приклади команд : "помісти на", "зніми з" і т. д. SHRDLU мала внутрішню пам'ять, запам'ятовувала історію дій, назви цих об'єктів і могла відповідати на питання в контексті своєї пам'яті [7].

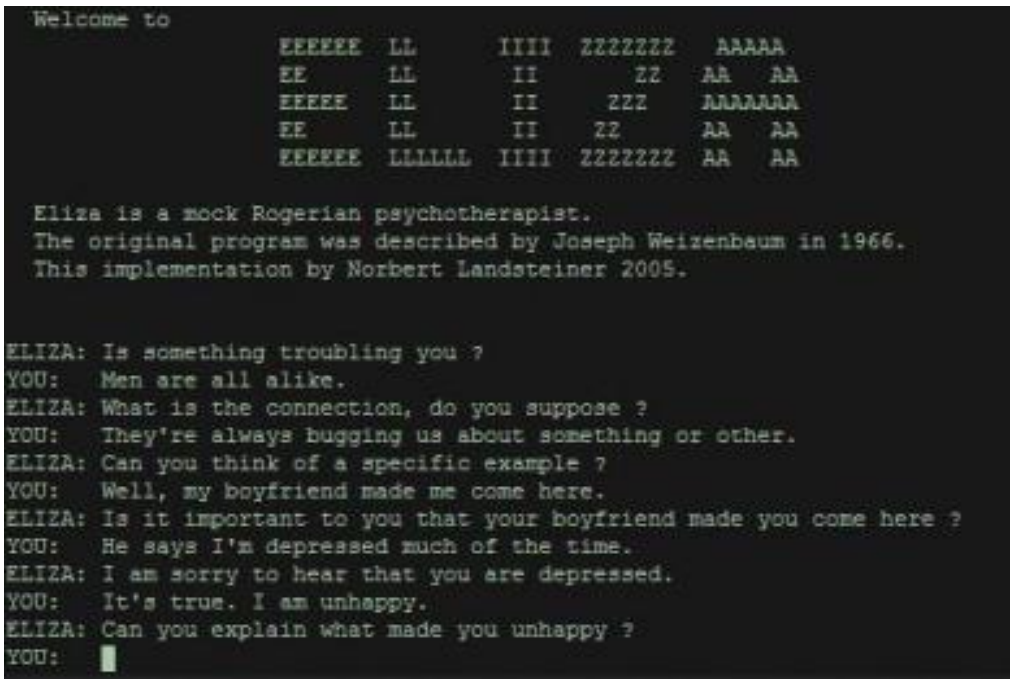


Рисунок 1.1 – Діалог з Eliza

До 1980-х років усі системи по роботі з природними мовами ґрунтувалися на складному наборі складених вручну правил. Проте з кінця 1980-х з введенням алгоритмів машинного навчання вони знайшли своє застосування також в завданнях по обробці мов [8]. Крім того, у той час обчислювальна потужність комп'ютерів безперервно зростала за законом Мура [9].

Одним з перших і важливих етапів аналізу тексту стало розмічування частин мови. Для завдання розмічування частин мови (part-of-speech tagging) успішно були застосовані приховані Марківські моделі (CRF) [10].

Завдяки дослідникам з IBM, які розробляли і удосконалювали усе більш складні статистичні моделі, системи машинного перекладу досягли помітних успіхів [11]. Проте моделі машинного перекладу вимагають наявності паралельних корпусів текстів. Прикладами існуючих паралельних корпусів є стенограми засідань ООН і Європейського Союзу, переклади різних фільмів і серіалів і безліч інших складених корпусів і перекладів. Але все-таки через обмежену опрацьованість самих корпусів систем перекладу по тематиках і розмірах актуальним питанням залишається можливість ефективного навчання на обмежених наборах даних [12].

Багато недавніх досліджень дедалі більше стосуються теми навчання без вчителя [13] чи навчання з підкріпленням [12]. Такі алгоритми спрямовані на навчання за даними, які були розмічені вручну. У випадку навчання з підкріпленням в основному обсязі нерозмічених даних використовується невелика кількість розмічених. Було показано, що такий спосіб навчання може значно покращити точність навчання [12].

З 2010 року навчання ознак спільно з методами глибокого навчання отримало широке застосування в NLP. Рішення на основі глибокого навчання показали найкращі результати у таких завданнях як мовне моделювання, синтаксичний розбір, вилучення іменованих сутностей, машинний переклад тощо [14–17]. Векторне уявлення слів, наприклад Word2vec, fastText, Bert та ін., дозволяє зіставляти кожному слову вектор у багатовимірному просторі на основі контекстів, що зустрічаються. Згідно з припущенням, яке зветься дистрибутивною гіпотезою, слова зі схожим змістом будуть зустрічатися в схожих контекстах, тобто будуть семантично близькими [18]. Вектори слів використовуються під час кластеризації слів на корпусах текстів, оцінки семантичної близькості, аналізі тональності та інших завдань [18].

1.2 Вилучення інформації

Вилучення інформації (Information Extraction – IE) – це ідентифікація, наступна або одночасна класифікація та структурування семантичних класів конкретних згадок, знайдених у неструктурованих джерелах даних, таких як текст природною мовою [1]. Прикладами інформації можуть бути будь-які описи подій, що відбуваються.

До неструктурованих джерел даних можна віднести письмовий та усний текст, зображення, відео та аудіозаписи. Такі дані не структурно неузгоджені, а, найімовірніше, інформація в них закодована таким чином, що комп'ютеру складно

їх інтерпретувати. Передбачається, що в процесі вилучення дані перетворюються на структуровану форму для подальшої можливості їх обробки та аналізу [1].

У процесі вилучення вся інформація в текстах ідентифікується з урахуванням особливостей мовної організації. Незалежно від мови, текст складається з цілого набору виразів та правил, з яких він складається. Це можливо як наслідок принципу композиційності [19], загального поняття лінгвістичної філософії, що є основою багатьох сучасних підходів до мови. Відповідно до цього принципу, значення будь-якого складного мовного виразу є функцією значень його складових частин. Тобто, як правило, типове речення містить ряд його складових: суб'єкт і предикат, що виражаються підметом і присудком. Значення кожного слова у реченні, їх послідовність та морфологічні ознаки дозволяють нам розуміти зміст цього речення [19].

Незважаючи на те, що семантична інформація в тексті не вказується явно з обчислювальної точки зору, її можна вилучити, враховуючи поверхневі закономірності, що відображають її непрозору внутрішню організацію [1]. Система вилучення буде використовувати набір шаблонів, які створюються вручну, або виводяться автоматично, що дозволятиме витягти та подати інформацію з тексту у структурованому вигляді. Більш детальні алгоритми та методи будуть розглянуті пізніше. Використання терміна «вилучення» має на увазі, що необхідна семантична інформація явно присутня в лінгвістичній організації тексту, тобто вона легко доступна в лексичних елементах (словах та групах слів), граматичних конструкціях (фразах, реченнях, часових виразах), прагматичному упорядкуванні та риторичній структурі (абзацах, розділах) вихідного тексту [1].

Крім того, виявлення знань також можливе за допомогою методів збору статистичних даних, які працюють з інформацією, що витягнеться з тексту. У всіх операціях вилучення інформації часто є обов'язковим етапом попередньої обробки, як збирання додаткових ознак. Наприклад, дані, вилучені з поліцейських звітів, можуть використовуватися як вхідні для застосування моделі інтелектуального аналізу, виявлення загальних тенденцій злочинності або для алгоритму міркувань на основі конкретного випадку, який намагатиметься передбачати

місцезнаходження наступного злочину, заснованому на схожих випадках.

1.3 Підзавдання вилучення інформації

Типові завдання з вилучення інформації включають наступне [1–3, 20–34]:

- розпізнавання іменованих сутностей (Named entity recognition – NER). Суть завдання полягає в пошуку згадок іменованих об'єктів за заздалегідь визначеними категоріями, такими як імена людей, організації, розташування, значення дат і часу.
- здатність кореференції (Coreference resolution). Вона полягає в об'єднанні згадки об'єктів із текстових описів у групи. В якості члена ланцюжка може виступати ім'я, субстантив, займенник.
- вилучення відносин (Relationship extraction). Це завдання пов’язане з пошуком спільних згадок іменованих сутностей, визначенням і класифікацією семантичних між ними. Наприклад, сутності «людина» та «організація» можуть бути пов'язані як: «директор компанії», «співробітник» тощо.
- вилучення атрибутів, фактів та подій (Fact / Event Extraction) – пошук та класифікація різних фактів: події, думки, відгуки, контактні дані, новини, оголошення та ін.

Нижче буде розглянуто існуючі підходи до вирішення цих завдань.

1.3.1 Розпізнавання іменованих сутностей

На сьогоднішній день переважну кількість існуючих систем можна розділити на такі що ґрунтуються на граматичних правилах та із застосуванням машинного

навчання, а також гібридні підходи, що поєднують обидва варіанти (див. рисунок 1.2) [20–29]. З одного боку, створені вручну системи пов’язані з використанням граматик, які, як правило, мають високу точність. Такі системи засновані на контекстно вільних граматиках, введених Ноамом Чомським, в яких одні синтаксичні категорії можуть бути описані через інші або через послідовність кінцевих термінів і, як правило, зліва направо [4]. Однак, для написання граматик потрібні великі словники предметної області, наприклад, словники імен та прізвищ, процес складання яких є досить трудомістким і може тривати більше місяця роботи.

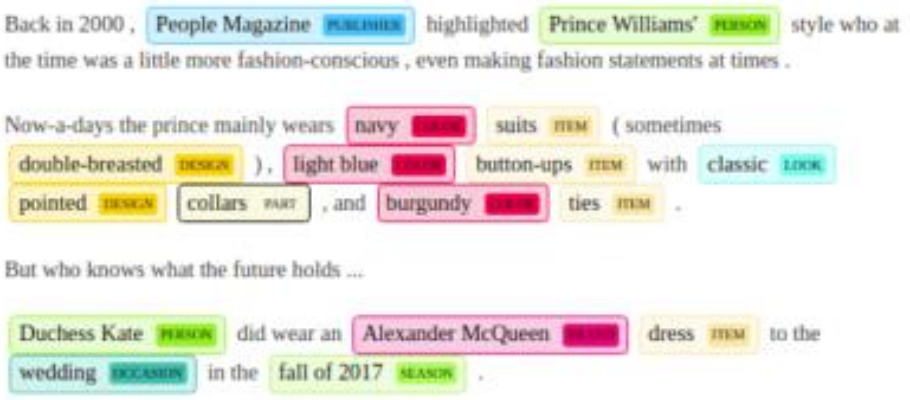


Рисунок 1.2 – Приклад розпізнавання іменованих сутностей [5]

При використанні машинного навчання для вирішення завдання розпізнавання іменованих сутностей досягається найкращий результат, якщо застосовувати архітектуру нейронної мережі, що складається з рекурентних шарів, наприклад, шарів Bi-directional Long Short-Term Memory (Bi-LSTM), призначених для обробки послідовностей [20]. На відміну від звичайної LSTM (Long Short-Term Memory), обробка послідовності Bi-LSTM відбувається як зліва направо, так і в зворотному напрямку. Це корисно, оскільки в цьому випадку враховується не тільки попередній контекст, а й майбутній (див. рисунок 1.3). Крім того, можливе застосування шару CRF, який створює матрицю переходу станів для прогнозування поточного тега кожного слова: чи є цей тег сутністю, якщо так, якого типу [22].

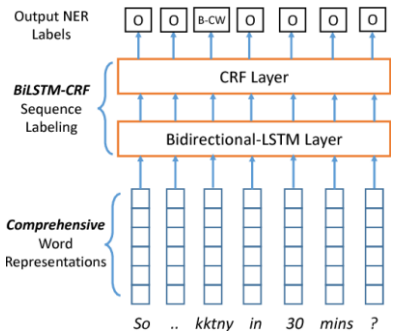


Рисунок 1.3 – Огляд архітектурного підходу для вирішення задачі NER [22]

На вході нейромережі можуть подаватися передбачені векторні слова, отримані за допомогою таких моделей як Word2vec, Glove, FastText та інших. Крім того, окрім векторів слів можна використовувати різні комбінації векторів символів, отриманих за допомогою тих же методів, і теги частин мови, як результат інших моделей, і будь-якої іншої додаткової інформації [22].

Існує досить багато готових рішень для NER, в основі яких схожі архітектури: Stanford NLP [16], spaCy [23], NLTK [24] та інші. Для української мови можна виділити декілька рішень [25–27]. Дослідження показують [20], що на сьогодні навіть найсучасніші системи NER розроблені для однієї предметної області, погано працюють у будь-якій іншій, незалежно від принципу їх роботи. При використанні машинного навчання в локальній предметній області може не виявитися навчальної вибірки та/або часу для її створення, а також необхідної кількості документів. У цьому випадку краще використовувати підхід, заснований на граматичних правилах [29]. У деяких випадках є сенс комбінувати ці підходи [28–29]. Так, наприклад, для розпізнавання імен та організацій використовувати заздалегідь підготовану модель, а для розпізнавання іменованих об'єктів предметної області реалізувати ці правила.

1.3.2 Здатність кореференції

Здатність кореференції є пошуковим завданням, пов'язаним зі знаходженням усіх ланцюжків згадок, які посилаються на ту саму сутність у тексті. З її допомогою

можна вирішити безліч завдань в NLP, таких як: автореферування, системи питання-відповіді, чат-боти та завдання з вилучення інформації (див. рисунок 1.4).

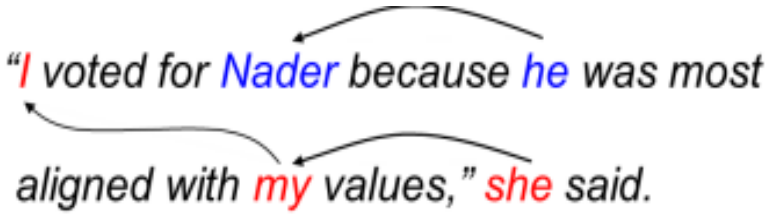


Рисунок 1.4 – Приклад ланцюжків згадок про сутності

Для створення моделі, що вирішує це завдання, застосовуються нейромережі, навчені на досить великій вибірці. Можна виділити такі види моделей [30–31]:

– mention-pair. Дана модель працює за принципом бінарної класифікації, що передбачає кореферентність для кожної знайденої пари сутностей. Проте їх застосування практично складно здійснити. По-перше, може порушуватися властивість транзитивності, тобто зустрічаються такі згадки, що кореферують із двома іншими, але не пов'язані між собою. По-друге, має місце сильна незбалансованість класів, тому що більшість сутностей між собою не є згадками одного й того ж самого.
– mention-ranking. Ця модель використовується для ранжування найімовірніших згадок. Для кожної знайденої згадки попарно оцінюється можливість кореферування з кожною попередньою сутністю або ключовими словами. Вибирається одна найбільш імовірна згадка. Навчання може відбуватися з різних ознак: відстань, облік векторів слів, даних морфології .
– entity-based. Модель виділяє кластери згадок різних сутностей, займенників та субстантивів. Кожна наступна знайдена сутність порівнюється з попередніми знайденими кластерами. З набору виявлених кластерів вибирається той, у якого оцінка ймовірності його приналежності буде максимальною. Якщо найбільш відповідних кластерів не знайдеться, тоді з аналізованого не віднесеного елемента створюється новий кластер.

Така модель показує точніший результат дозволу кореференції [6]. По-перше, її результати не порушують умови транзитивності; по-друге, вони менш суперечливі, оскільки порівнюються сутність із кластером, що включають інформацію від усіх елементів кластера.

Крім підходів, заснованих на машинному навчанні, існують підходи до правил і евристик. Однак їх використання є досить загальним рішенням і не прив'язується до конкретних предметних областей.

1.3.3 Вилучення відносин

Завдання щодо вилучення відносин (Relation Extraction) полягає в об'єднанні сутностей певного типу, наприклад, люди, організації, локації, у певні заздалегідь визначені семантичні категорії: «одружений», «живе», «працює в» та інші [32–33]. На рисунку 1.5 наведено приклади таких відносин.

Клас Належність	Приклади	Тип
Персональні Організаційні Предметні Просторові	мати, одружений директор, оф. представник володіти, робити	PERS -> PERS PERS -> ORG (PERS|ORG) -> OBJ
Близькість Напрям	поряд з к півдню від	LOC -> LOC LOC -> LOC

Рисунок 1.5 – Приклади спільних відносин

Для вирішення цієї задачі може застосовуватися метод навчання з учителем (Supervised learning), що представляє класифікатор, який приймає на вхід текстові фрагменти, кілька іменованих сутностей та їх типи, що зустрічаються у фрагменті

[33]. На виході визначається їх наявність та тип, або відсутність відношення. В результаті навчальна вибірка повинна представляти набір, що включає фрагмент тексту, пару сутностей, їх типи і тип зв'язку. Приклади, у яких відсутні зв'язки, є негативними для навчальної вибірки. Крім цих ознак можуть використовуватися більш нетривіальні: наявність певних синтаксичних конструкцій, дистанція між словами, шлях у синтаксичному дереві, вікно тексту до першої сутності, після другої та між ними тощо.

Крім того, вирішити це завдання можна шляхом застосування підходу з мінімальним залученням вчителя (Distant Supervision) [32]. Передбачається, що навчальна вибірка в цьому випадку або відсутня, або розширена. Для розширення навчальної вибірки використовується готова база знань (Freebase, DBpedia), що містить інформацію про те, як пов'язані ті чи інші сутності. Далі сутності витягуються з тексту, та пов'язуються з екземплярами з бази знань і фільтруються ті пропозиції чи фрагменти, у яких спільно згадуються ці пари. Так генерується навчальна вибірка. За відсутності зв'язку в базі, екземпляри позначаються як негативні приклади. Таким чином можна збирати вибірку та використовувати її для подальшого навчання класифікатора.

Метод Lightly Supervised полягає в поповненні навчальної вибірки з наявних даних за рахунок виявлення синтаксичних шаблонів, які охоплюють відносини, що зустрічаються, і їх подальшого узагальнення. Наприклад, якщо пара сутностей спільно згадується в контексті виду

«Дієслово > сутність1 > прийменник > сутність2»,

тоді будується припущення, що наявність такого шаблону в іншому реченні свідчить про зв'язок цього ж типу [32].

1.3.4 Вилучення атрибутів, фактів та подій

Завдання щодо вилучення фактів становлять найбільший інтерес при аналізі текстових даних і є кінцевими по відношенню до вищезгаданих [1]. На сьогоднішній день не існує універсального підходу до їх загального вирішення.

Натомість розглядаються окремі завдання в залежності від предметної галузі. Прикладами можуть бути: аналіз спортивних подій із новин (команди що беруть участь у матчі, кінцевий рахунок, переможець); аналіз політичних та економічних подій: злиття та поглинання (покупець та купований, сума угоди), зміна посад (співробітник, стара посада, нова посада), відставки та призначення, оголошення/прес-релізи людей та компаній; аналіз різних документів: судові рішення (позивач, відповідач, предмет позову, позовні вимоги, рішення); аналіз резюме (бажана заробітна плата, очікувана посада, досвід роботи, навички). Крім того, існує потреба щодо вилучення різних атрибутів, таких як: властивості товарів з їх опису (виробник, модель, різні характеристики), адреси (область, місто, район, станція метро, вулиця, будинок, приміщення, квартира/офіс), контактні дані організацій (юридична назва, телефон, email, адреса).

Вирішення подібних завдань можна умовно розділити на 2 способи [34]:

– розмітка даних та тренування моделі машинного навчання для отримання сутностей та відносин аналогічним чином, як було розглянуто у попередніх 3-х підпараграфах. Такий підхід здатний дати хороший результат (точність та повноту) тільки за наявності відповідного якісно розміченого корпусу. Іншою особливістю є те, що при виникненні помилки неможливо вручну виправити модель, а лише перенавчити її спільно з пошуком, виправленням чи поповненням корпусу необхідними прикладами.
– підхід з використанням різних евристик та правил, заснованих на формальних граматиках, зокрема контекстно-вільних. Застосовуються готові інструменти, що дозволяють складати подібні правила. В цьому

випадку не потрібна розмітка корпусу, проте потрібен час на складання граматик та тематичних словників для формування цих правил. Такий підхід дає високу точність, але відносно невисоку повноту, яка залежить від якості опрацьованості правил і словників.

1.4 Парсери на основі контекстно-вільних граматик

Існує декілька реалізацій парсерів контекстно-вільних граматик для української мови. Найпопулярнішою бібліотекою, яка працює з контекстно-вільними граматиками, є NLTK [24]. Для українського середовища найпоширеніших і некомерційних парсерів [35, 36] можна назвати Tomita парсер і Yargy. Tomita, яка розроблялась протягом багатьох років, складається з кількох десятків тисяч рядків коду. Бібліотека реалізована мовою C++. Tomita доступна у вигляді бінарного файлу, проте у відкритому доступі немає банку готових граматик. У Tomita-парсері використовується власна мова для опису граматик (див. рисунок 1.6).

```
#encoding "utf-8"
Born -> Verb<kwtype=born>;
City -> Noun<kwtype=city>;
Person -> AnyWord<gram="имя">;
S -> Person interp(BornFact.Person) Born "в" City interp(BornFact.Place);
```

Рисунок 1.6 – Приклад синтаксису граматик для Tomita-парсера

Парсер Yargy, навпаки, проект з відкритим вихідним кодом, опублікованим на Github, доступний у вигляді бібліотеки Python. У зв'язку з цим усі граматики та словники повинні описуватися мовою програмування Python. Yargy використовує морфологічний аналізатор Pymorphy, який також є бібліотекою з відкритим кодом. Більше того, він має кілька розроблених готових наборів правил для отримання

таких сутностей, як імена, дати, гроші, адреси, які повністю доступні на Github [39].

На рисунку 1.7 наведено приклад отримання дат за допомогою Yargy.

```
MONTH_NAME = dictionary(MONTHS)
MONTH = and_(
    gte(1),
    lte(12)
)
DAY = and_(
    gte(1),
    lte(31)
)
YEAR = and_(
    gte(1900),
    lte(2100)
)
DATE = or_(
    rule(DAY, MONTH_NAME, YEAR),
    rule(YEAR, '-', MONTH, '-', DAY),
    rule(YEAR, 'р', '.')
).named('DATE')
parser = Parser(DATE)
text = '''2023 р.
18 липня 2023
2023-01-02
'''
for line in text.splitlines():
    match = parser.match(line)
    display(match.tree.as_dot)
```

Рисунок 1.7 – Приклад отримання дат за допомогою Yargy [36]

Слід зазначити, що ці парсери не показують структуру даного речення, а просто витягують шаблони, що задовольняють факту, який ми шукаємо.

1.5 Синтаксичні парсери

Синтаксичним розбором є завдання зіставлення речення та його синтаксичної структури у вигляді дерева складових або дерева залежностей (див. рисунок 1.8). Такі дерева корисні безпосередньо в програмах, таких як перевірка граматики в системах обробки тексту.

Наприклад, речення, які можуть бути проаналізовані, найімовірніше можуть мати граматичні помилки, або важко читатися. Як правило, дерева служать

важливою проміжною стадією представлення для семантичного аналізу і, таким чином, відіграють важливу роль у таких додатках, як питально-відповідні системи і задачі з вилучення інформації [7]. Оскільки існує два основних підходи до подання синтаксичної структури будь-якого речення, вони також використовуються у різних інструментах його аналізу.

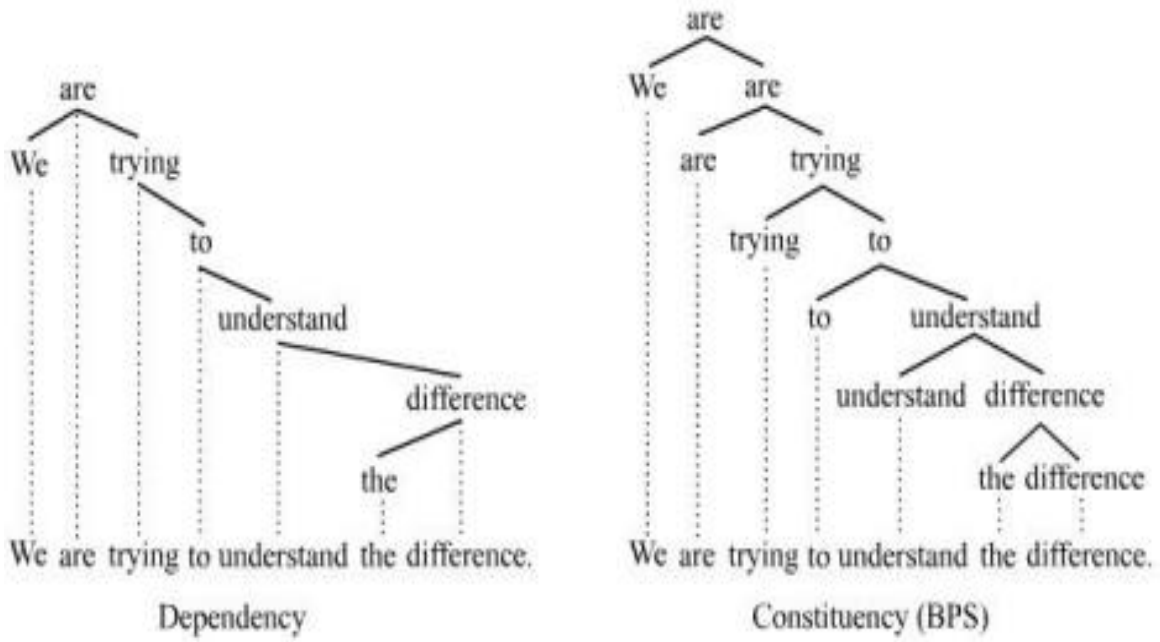


Рисунок 1.8 – Наочне порівняння дерева залежностей (ліворуч) та дерева складових (праворуч) [37]

Граматика залежностей представляє значно більший інтерес для мов з вільним порядком слів, зокрема для української мови, оскільки в рамках цієї моделі слова представлені у вигляді ієрархії компонентів, між якими встановлені залежності. Завдяки цим залежностям є можливість визначати, як пов'язані ті чи інші сутності та згадки. Розглянемо моделі побудови дерев залежностей.

Аналізатор граматик залежностей виділяє граматичну структуру речення, встановлюючи зв'язок між «головними» словами та словами, які є залежними по відношенню до них. Результатом його є синтаксичне дерево, приклад якого показаний на рисунку 1.9.

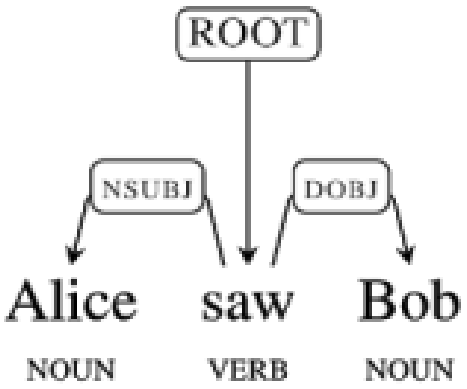


Рисунок 1.9 – Приклад дерева залежності

На відміну від парсерів дерев складових, парсери дерев залежності розробляються з використанням методів машинного та глибокого навчання. Серед найпопулярніших інструментів можна виділити SyntaxNet [40], MaltParser [38] та UDPipe [41]. Щоб навчити таку модель та перевірити її правильність для кожної мови, необхідно мати великий анотований синтаксичний корпус.

1.5.1 Синтаксично анотовані корпуси

Процес створення синтаксичних парсерів починається з інструкції корпусу з допомогою спеціальних програм (наприклад, Brat annotation tool [42]). В цілому будь-який лінгвістичний корпус є сукупністю текстів, які зазвичай структуровані за темами, датами, авторами і т. д. В області комп'ютерної лінгвістики слово «корпус» зазвичай має на увазі сукупність текстів, які мають морфологічну, синтаксичну, семантичну або будь-яку іншу розмітку. Наприклад, синтаксична розмітка зазвичай включає такі ознаки, як частина мови, лема, регістр, стать і теги залежностей для кожного слова. Альтернативна назва для текстового корпусу, що розбирається, яка широко використовується – це дерева синтаксичного розбору (Treebank) [43]. Розробка таких корпусів – дуже складне завдання, яке потребує

великої кількості часу та роботи лінгвістів. В результаті такі корпуси можуть бути використані для безлічі завдань та застосунків.

Одним із найпопулярніших анотованих корпусів для української мови є OpenCorpora [44]. По суті цей проект спрямований на розвиток корпусу зусиллями спільноти. Основна мета проекту – створити повністю анотований корпус, що включає морфологічну, синтаксичну та семантичну розмітку, який був би загальнодоступним для скачування будь-якою особою та для різних цілей згідно ліцензії CC-BY-SA. Іншим популярним корпусом для української мови, що має синтаксичну інструкцію, вважається SyntaxNet. Це набір дерев синтаксичного розбору, створених на основі Національного корпусу української мови, що містять понад 52 000 речень та близько 770 000 слів. Корпус забезпечений морфологічною та синтаксичною анотацією у формі дерева залежностей для кожного речення. Крім того, SyntaxNet містить анотації й інших типів, в першу чергу, лексичну функціональну анотацію в термінах лексичних функцій, як це визначено в моделі «Сенс-текст», яка розглядає формальну граматику української мови [46].

Під час розробки анотованих корпусів лінгвісти зазвичай використовують певну кількість тегів для маркування частин мови, функцій та відношень. На даний момент кожен корпус має власні відношення та вказівки по розмітці, проте було б зручніше користуватися єдиними та універсальними правилами для цієї мети. Одним із таких проектів, який спрямований на створення узгодженої структури для анотації дерева залежностей різними мовами, вважається Universal Dependencies [43]. Цей проект дозволив розробити дерева синтаксичного аналізу значного розміру та різним ступенем якості більш ніж для 30 мов, які надані в єдиному узгодженому форматі. Формат називається CoNLL-U, який історично отримав назву на Конференції з вивчення природної мови (Conference on Natural Language Learning – CoNLL) (див. рисунок 1.10), яка спеціалізується на аналізі мовних залежностей [43].

Як видно з рисунку 1.10, існує три поля токена, пов'язані з тегами частини мови: UPOSTAG (універсальний тег для частини мови), XPOSTAG (тег частини мови, що є локальним по відношенню до кожної мови та корпусу, що згодом

конвертується в XPOSTAG), FEATS (Список морфологічних особливостей, що уточнює універсальний тег частини мови) [43].

ID	FORM	LEMMA		UPOSTAG	XPOSTAG	FEATS	HEAD	DEPREL	DEPS
1	Trešdienas	trešdiena	'Wednesday'	NOUN	ncfsg4		2	nmod	2:nmod:gen
2	vakarā	vakars	'evening'	NOUN	ncmsl1		7	obl	7:obl:loc
3	79	79	'79'	NUM	xn		4	nummod	4:nummod
4	gadu	gads	'year'	NOUN	ncmpg1		5	nmod	5:nmod:gen
5	vecumā	vecums	'age'	NOUN	ncmsl1		7	obl	7:obl:loc
6	mūžībā	mūžība	'eternity'	NOUN	ncfsl4		7	obl	7:obl:loc
7	aizgājis	aiziet	'leave'	VERB	vmnpdmsnasnpn		0	root	0:root
8	tautā	tauta	'nation'	NOUN	ncfsl4		9	obl	9:obl:loc
9	mīlētais	mīlēt	'love'	VERB	vmnpdmsnpsypn		10	amod	10:amod
10	dzejnieks	dzejnieks	'poet'	NOUN	ncmsn1		11	nmod	11:nmod
11	Imants	Imants	'Imants'	PROPN	npmsn1		7	nsubj	7:nsubj
12	Ziedonis	Ziedonis	'Ziedonis'	PROPN	npmsn2		11	flat:name	11:flat:name
13	.	.	'.'	PUNCT	zs		7	punct	7:punct

Figure 1: FrameNet annotation in WebAnno on top of a UD tree (Table 1). Only head nodes are selected while annotating frame elements (FE). The FE spans can be acquired automatically by traversing the respective subtrees: [*trešdienas vakarā*]$_{\text{Time}}$, [*tautā*]$_{\text{Experiencer}}$, [*tautā mīlētais dzejnieks Imants Ziedonis*]$_{\text{Protagonist}}$. Multi-word lexical units (LU) are indicated by generic LU tags: *mūžībā aizgājis*$_{\text{Death}}$ versus *mīlētais*$_{\text{Experiencer-focused-emotion}}$.

Рисунок 1.10 – Приклад результату синтаксичного аналізу, представленого у форматі CoNLL-U

Поля HEAD та DEPREL використовуються для кодування дерева залежностей над словами. Поле HEAD містить посилання на ID залежного до поточного токену слову. Значення DEPREL відображає тип залежності або специфічний підтип такого відношення для кожної конкретної мови. Як і у випадку з морфологією, синтаксична анотація надається тільки для слів, а токени, які не є словами, мають підкреслення в полях HEAD та DEPREL [43].

1.5.2 Моделі синтаксичного аналізу

Розмічені синтаксичні корпуси дозволяють навчати моделі глибокого навчання для їх використання у синтаксичному розборі речень. Модель UDPipe (аналізатор універсальних залежностей) є багатофункціональною моделлю для

токенізації, тегування, лематизації та аналізу залежностей файлів CoNLL-U. В результаті використання цього інструменту ми можемо ввести текст і отримати висновок готового файлу CoNLL-U [41].

Для навчання нейронних мереж надаються анотовані дані у форматі CoNLL-U. Архітектура нейронних мереж (див. рисунок 1.11), що використовується для цього завдання, була докладно описана Ченом та Меннінгом [46].

Вхідний шар має кілька вузлів, які репрезентують кожне слово в дереві речення. Як було згадано Чжаном і Нівром [47] і Ченом і Меннінгом [46], кожен токен може бути представлений у вигляді вектора, що включає теги частини мови, морфологічні теги і мітки зв'язків. Частина мовних міток і міток дуг ініціалізуються випадковим чином і встановлюються в процесі навчання. Вхідний шар має функцію активації Softmax і передається у приховані шари.

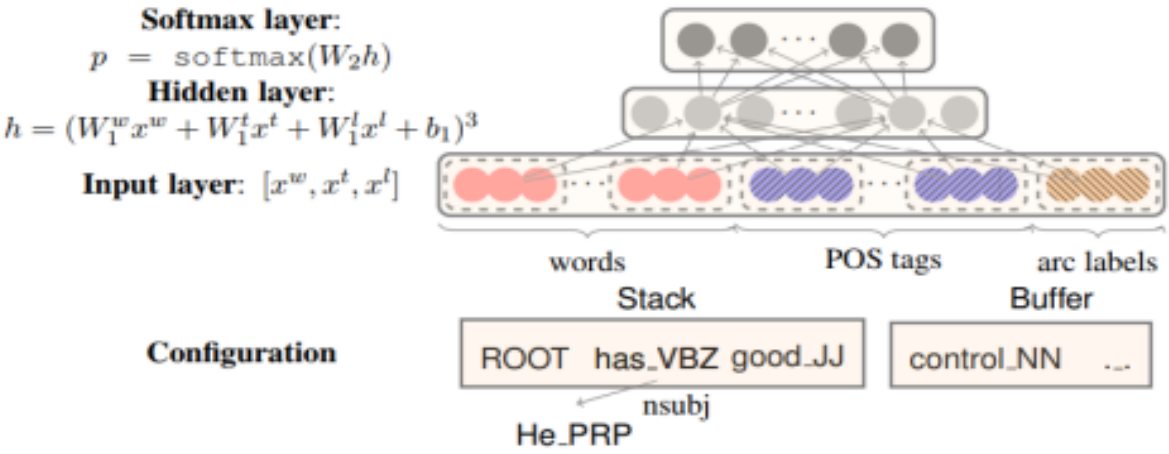


Рисунок 1.11 – Архітектура нейронної мережі для синтаксичного аналізу [46]

Крім того, у процесі навчання використовується векторне уявлення слів, які раніше навчалися у наборі даних Вікіпедії за допомогою скіп-грам моделі з негативною вибіркою. Мережа навчається з використанням стохастичного градієнтного спуску з розміром партії 10. Перехресна ентропія мінімізується за допомогою L2-регуляризації.

2 КОМБІНОВАНИЙ ПІДХІД ДЛЯ ОТРИМАННЯ СТРУКТУРОВАНИХ ДАНИХ З ТЕКСТУ

У цій роботі пропонується підхід до вирішення завдання щодо вилучення фактів для української мови, а також для інших мов з вільним порядком слів. На рисунку 2.1 представлена діаграма з покроковим описом розробленої методики вилучення фактів з тексту.

Попередня обробка тексту	Побудова синтаксичного дерева	Застосування правил КВГ	Злиття результатів	Парсинг дерева
Токенізація на слова та речення Морфологічний аналіз Лемматізація	Застосування моделі UDPipe 2.0, навченою на корпусі SyntaxNet	Написання і застосування правил по витягуванню окремих сутностей і атрибутів з використанням бібліотеки Yargy	Парсинг отриманої структури (ConLL - U) в структуру даних tree Додавання в елементи отриманого дерева витягнуті за допомогою КВГ дані	Написання шаблонів для опису фактів Парсинг синтаксичного дерева по написаних шаблонах

Рисунок 2.1 – Методика пропонованого підходу

Ідея методики полягає у комбінуванні підходу щодо вилучення інформації, заснованого на злитті результатів, отриманих за допомогою контекстно-вільних граматик з результатами виділення шаблонів при синтаксичному розборі тексту. Далі будуть детально розглянуті кроки підходу, а також описана програмна бібліотека, розроблена на його основі.

Основною перевагою граматик залежностей є уніфікованість результатів по відношенню до мов з вільним порядком слів, наприклад, української мови. Тому було б дуже зручно використовувати граматики залежностей для отримання інформації з таких мов. Крім того, неможливо покрити всі синтаксичні шаблони самописними правилами на контекстно-вільних граматиках. Незважаючи на цей недолік, два найпопулярніші парсери для української мови (Tomita та Yargy) використовують контекстно-вільні граматики.

2.1 Вибір парсеру контекстно-вільних граматик

Серед бібліотек, що розглядаються в попередньому розділі, необхідно здійснити вибір, виходячи з вимог і критеріїв, що висуваються:

– функціональні можливості;
– наявність докладної документації та прикладів готових правил;
– доступність інтерфейсу для зручної роботи основною мовою програмування – Python;
– продуктивність.

Найбільш відповідним рішенням, з урахуванням основних вимог, є програмна бібліотека Yargy, реалізована мовою Python. У порівнянні з Tomita парсером, Yargy має схожі функціональні можливості, однак як морфологічний аналізатор він використовує Pymorphy2 [48]. Даний аналізатор на кожну словоформу на вході видає кілька можливих результатів морфологічної інформації та лем, тоді як аналізатор Tomita-парсер вибирає єдиний варіант на основі контексту. Ще однією відмінністю є те, що Tomita-парсер працює через консольний інтерфейс, Yargy – це бібліотека на Python. Крім того, для написання правил Tomita використовує свою мову і Protobuf-файли, а в Yargy граматики і словники описуються як змінні в Python. Головною перевагою Yargy є наявність банку готових граматик, який розроблений для вилучення імен, організацій, локацій, адрес і підтримує додавання власних граматик.

Необхідно відзначити, що бібліотека Yargy поступається за продуктивністю Tomita-парсеру. Це з тим, що він реалізований на Python, тоді як Tomita – мовою C++ і, найімовірніше, недостатньо добре оптимізований. З іншого боку, відносно невелика продуктивність частково компенсується використанням інтерпретатора PyPy, а також розпаралелювання на окремі процеси або навіть комп'ютери за допомогою асинхронних черг для розподілу завдань [49], наприклад, з використанням Redis. На рисунку 2.2 наведено приклад вилучення топонімів, що

починаються з прикметників та закінчуються словами «коаліція» або «область».

```
from yargy import Parser, rule, and_
from yargy.predicates import gram, is_capitalized, dictionary


GEO = rule(
    and_(
        gram('ADJF'),      # так відмічається прикметник, інші відмітки
                           # http://pemorphy2.readthedocs.io/en/Latest/user/
        is_capitalized()
    ),
    gram('ADJF').optional().repeatable(),
    dictionary({
        'коаліція',
        'область'
    })
)


parser = Parser(GEO)
text = '''
Антигітлерівській коаліції належить історична заслуга ...
Одеська область є найбільшою за площею ...
Найменша за чисельністю людей є Чернівецька область ...
'''
for match in parser.findall(text):
    print([_.value for _ in match.tokens])
```

```
['Антигітлерівській', 'коаліції']
['Одеська', 'область']
['Чернівецька', 'область']
```

Рисунок 2.2 – Приклад вилучення топонімів, що починаються прикметниками та закінчуються словами «коаліція» або «область»

Правила Yargy можуть складатися з інших правил і предикатів. Предикат – це функція, яка приймає на вхід токен та повертає True або False. Правила і предикати можуть логічно комбінуватися з допомогою функцій – логічних

операторів and_, or_ і not_ (див. рисунок 2.3).

eq(value)	a==b
caseless(value)	a.lower()==b.lower()
in_(value)	a in b
in_caseless(value)	a.lower() in b
gte(value)	a>=b
lte(value)	a<=b
length_eq(value)	len(a)==b
normalized(value)	Нормальна форма слова == value
dictionary(value)	Нормальна форма слова in value
gram(value)	value є серед грамів слів
type(value)	Тип токену дорівнює value
tag(value)	Тег токену дорівнює value
custom(function[,types])	function як предикат
true	Завжди повертає True
is_lower	str.islower
is_upper	str.isupper
is_title	str.istitle
is_capitalized	Слово написано з великої літери
is_single	Слово в однині

Рисунок 2.3 – Предикати, доступні в бібліотеці Yargy [36]

Таким чином, бібліотека Yargy буде використовуватися на початковому етапі вилучення інформації, зокрема для вилучення іменованих сутностей, значень, дат, маркувань, навичок та ін., що складаються з кількох слів, що йдуть одне за одним. Надалі результат вилучення сутностей буде комбінуватися з синтаксичними деревами.

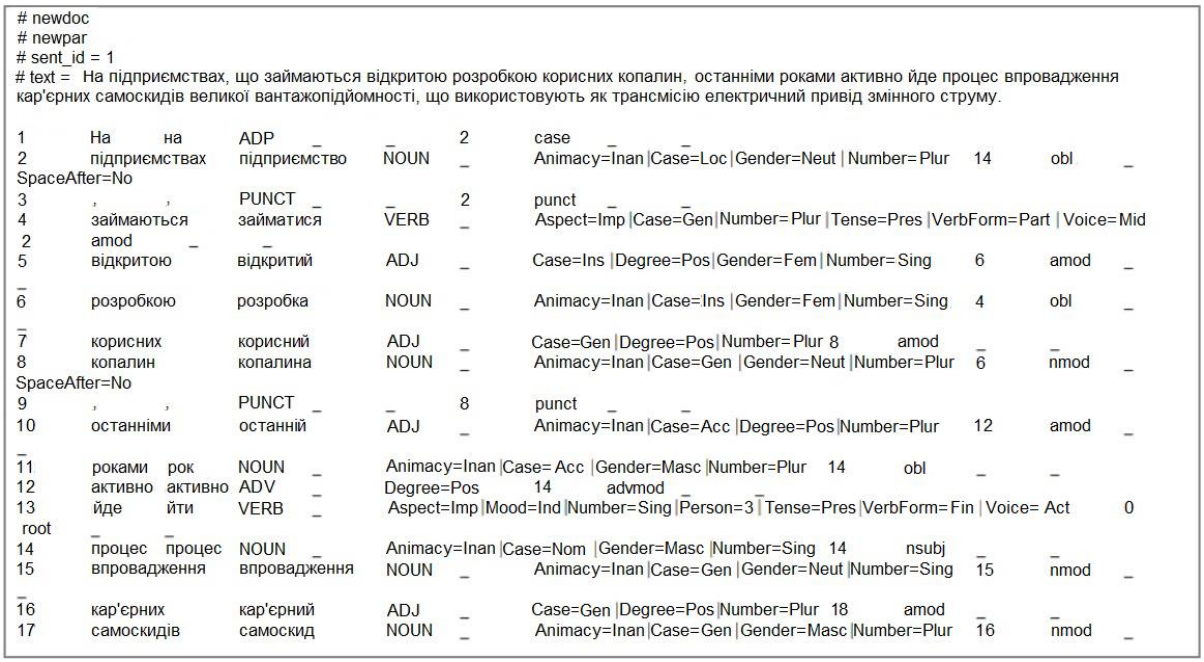
2.2 Вибір моделі синтаксичного аналізу

Серед безлічі існуючих рішень щодо синтаксичного розбору пропозицій також є і з відкритим вихідним кодом, серед яких можна виділити моделі SyntaxNet і UDPipe. Згідно з результатами змагання "Multilingual Parsing from Raw Text to Universal Dependencies" в рамках конференції Conference on Natural Language Learning 2018 (CoNLL 2018), найкращі результати для української мови за метрикою MLAS (Morphology-Aware Labeled Attachment Score) показано на рисунку 2.4 [50]. Дана модель являє собою цілий Pipeline, що включає послідовність етапів токенізації, лематизації, морфологічного аналізу, і синтаксичному розбору пропозиції, заснованого на граматика залежностей (див. рисунок 2.5).

Щоб уникнути втрати якості результатів на етапі синтаксичного аналізу не слід замінювати поточний аналізатор на аналізатор Pymorphy або будь-який інший сторонній, тому що поточна модель навчалася саме на ньому. Етапи лематизації та морфологічного аналізу будуть проводитися паралельно з тими самими етапами синтаксичного аналізу на контекстно вільних граматиках.

	Treebank	MLAS	Best system	Avg	StDev
1.	pl_lfg	86.93	UDPipe Future	73.73	± 7.29
2.	ru_syntagrus	86.76	UDPipe Future	71.63	± 9.36
3.	cs_pdt	85.10	UDPipe Future	**73.61**	± 6.32
4.	cs_fictree	84.23	ICS PAS	69.91	± 7.77
5.	ca_ancora	84.07	UDPipe Future	**74.62**	± 7.69
6.	es_ancora	83.93	Stanford	74.61	± 7.43
7.	it_isdt	83.89	Stanford	**77.14**	± 8.89
8.	fi_pud	83.78	Stanford	62.38	±14.83
9.	no_bokmaal	83.68	UDPipe Future	**70.75**	± 8.92
10.	cs_cac	83.42	UDPipe Future	**71.39**	± 6.89
11.	bg_btb	83.12	UDPipe Future	**73.18**	± 7.15
12.	fr_sequoia	82.55	Stanford	70.42	± 9.04
13.	sl_ssj	82.38	Stanford	62.41	± 9.18
14.	no_nynorsk	81.86	UDPipe Future	**68.62**	± 9.45
15.	ko_kaist	81.29	HIT-SCIR	**70.18**	± 9.36
16.	ko_gsd	80.85	HIT-SCIR	63.73	±16.02
17.	fi_tdt	80.84	Stanford	**65.27**	± 9.22
18.	fa_seraji	80.83	UDPipe Future	**71.23**	± 7.77
19.	pl_sz	80.77	Stanford	64.80	± 8.49
20.	fro_srcmf	80.28	UDPipe Future	**65.19**	±16.58
21.	la_ittb	79.84	ICS PAS	**67.77**	± 8.37
22.	ti_ftb	79.65	TurkuNLP	66.11	± 8.86
23.	sv_talbanken	79.32	Stanford	**68.05**	± 8.49
24.	ro_rrt	78.68	TurkuNLP	67.43	± 7.24
25.	el_gdt	78.66	Stanford	64.29	± 8.28
26.	fr_gsd	78.44	Stanford	**69.33**	± 8.59
27.	hi_hdtb	78.30	UDPipe Future	68.48	± 5.88
28.	sr_set	77.73	UDPipe Future	67.33	± 5.96
29.	da_ddt	77.31	Stanford	65.00	± 6.89
30.	et_edt	76.97	TurkuNLP	63.59	± 8.34
31.	nl_alpino	76.52	Stanford	62.82	± 9.81
32.	en_ewt	76.33	Stanford	**66.84**	± 5.86
33.	pt_bosque	75.94	Stanford	66.22	± 6.76
34.	cs_pud	75.81	UDPipe Future	60.47	±11.36
35.	af_afribooms	75.67	UDPipe Future	**63.76**	± 7.06
36.	sk_snk	75.01	Stanford	56.82	± 8.32
37.	en_pud	74.86	Stanford	**63.05**	± 7.89
38.	nl_lassysmall	74.11	Stanford	61.95	± 9.12
39.	hr_set	73.44	Stanford	60.08	± 7.07
40.	en_gum	73.24	ICS PAS	**61.72**	± 7.69
41.	ja_gsd	72.62	HIT-SCIR	59.52	± 6.20
42.	uk_iu	72.27	UDPipe Future	55.45	± 8.08
43.	en_lines	72.25	ICS PAS	**62.35**	± 8.04
44.	eu_bdt	71.73	UDPipe Future	58.49	± 8.62
45.	gl_ctg	70.92	Stanford	57.92	±14.10
46.	ar_padt	68.54	Stanford	53.28	± 6.12
47.	it_postwita	68.50	Stanford	51.72	± 8.80
48.	id_gsd	68.36	Stanford	**61.03**	± 6.49
49.	lv_lvtb	67.89	Stanford	53.31	± 7.96
50.	hu_szeged	67.13	UDPipe Future	53.08	± 8.01
51.	zh_gsd	66.62	HIT-SCIR	50.42	± 5.87
52.	sv_lines	66.58	Stanford	**57.40**	± 7.43
53.	fr_spoken	64.67	HIT-SCIR	53.17	± 5.61
54.	he_htb	63.38	Stanford	45.22	± 4.94
55.	ca_proiel	63.31	Stanford	**50.28**	± 6.69
56.	ru_taiga	61.59	ICS PAS	37.16	± 7.53
57.	gl_treegal	60.63	UDPipe Future	**47.35**	± 5.93
58.	grc_proiel	60.27	Stanford	**47.62**	±11.82
59.	la_proiel	59.36	Stanford	**47.79**	± 6.90
60.	de_gsd	58.04	TurkuNLP	39.13	±10.35
61.	ur_udtb	57.98	TurkuNLP	**49.64**	± 4.21
62.	no_nynorsklia	57.51	ICS PAS	37.08	± 7.78
63.	sme_giella	57.47	TurkuNLP	**38.29**	±12.37
64.	got_proiel	56.45	UDPipe Future	**46.18**	± 5.36
65.	tr_imst	55.73	Stanford	45.26	± 6.15
66.	grc_perseus	54.98	HIT-SCIR	35.65	±12.31
67.	sv_pud	51.74	TurkuNLP	**39.41**	± 7.78
68.	la_perseus	49.77	ICS PAS	28.67	± 8.06
69.	vi_vtb	47.61	HIT-SCIR	**32.45**	± 7.28
70.	sl_sst	45.93	ICS PAS	**33.12**	± 5.33
71.	ga_idt	45.79	TurkuNLP	**33.70**	± 5.18
72.	ug_udt	45.78	UDPipe Future	**35.08**	± 5.96
73.	br_keb	13.91	Uppsala	1.52	± 3.34
74.	hy_armtdp	13.36	CUNI x-ling	**5.94**	± 2.92
75.	ja_modern	11.82	Uppsala	**6.45**	± 2.59
76.	hsb_ufal	9.09	LATTICE	4.66	± 2.37
77.	kk_ktb	8.93	CUNI x-ling	**5.04**	± 2.34
78.	kmr_mg	7.98	IBM NY	4.01	± 1.96
79.	th_pud	6.29	CUNI x-ling	0.42	± 1.27
80.	pcm_nsc	5.30	KParse	**3.00**	± 1.30
81.	bxr_bdt	2.98	AntNLP	1.33	± 0.72
82.	fo_oft	1.07	CUNI x-ling	0.37	± 0.21

Рисунок 2.4 – Ранжування найкращих результатів синтаксичних парсерів (метрика MLAS) [50]



```
# newdoc
# newpar
# sent_id = 1
# text = На підприємствах, що займаються відкритою розробкою корисних копалин, останніми роками активно йде процес впровадження кар'єрних самоскидів великої вантажопідйомності, що використовують як трансмісію електричний привід змінного струму.

1	На	на	ADP	_	_	2	case	_	_
2	підприємствах	підприємство	NOUN	_	Animacy=Inan|Case=Loc|Gender=Neut|Number=Plur	14	obl	_
SpaceAfter=No
3	,	,	PUNCT	_	_	2	punct	_	_
4	займаються	займатися	VERB	_	Aspect=Imp|Case=Gen|Number=Plur|Tense=Pres|VerbForm=Part|Voice=Mid
2	amod	_	_
5	відкритою	відкритий	ADJ	_	Case=Ins|Degree=Pos|Gender=Fem|Number=Sing	6	amod	_
_
6	розробкою	розробка	NOUN	_	Animacy=Inan|Case=Ins|Gender=Fem|Number=Sing	4	obl	_
_
7	корисних	корисний	ADJ	_	Case=Gen|Degree=Pos|Number=Plur	8	amod	_	_
8	копалин	копалина	NOUN	_	Animacy=Inan|Case=Gen|Gender=Neut|Number=Plur	6	nmod	_
SpaceAfter=No
9	,	,	PUNCT	_	_	8	punct	_	_
10	останніми	останній	ADJ	_	Animacy=Inan|Case=Acc|Degree=Pos|Number=Plur	12	amod	_
_
11	роками	рок	NOUN	_	Animacy=Inan|Case=Acc|Gender=Masc|Number=Plur	14	obl	_	_
12	активно	активно	ADV	_	Degree=Pos	14	advmod	_	_
13	йде	йти	VERB	_	Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act	0
root	_	_
14	процес	процес	NOUN	_	Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing	14	nsubj	_	_
15	впровадження	впровадження	NOUN	_	Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing	15	nmod	_
_
16	кар'єрних	кар'єрний	ADJ	_	Case=Gen|Degree=Pos|Number=Plur	18	amod	_	_
17	самоскидів	самоскид	NOUN	_	Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur	16	nmod	_
```

Рисунок 2.5 – Приклад синтаксичного аналізу речення

2.3 Комбінування підходу по розбору на основі контекстно-вільних граматик та граматик залежності

На основі переваг та недоліків контекстно-вільних граматик та граматик залежності, їх результати можливо комбінувати для вирішення завдань із вилучення інформації. На першому етапі правила контекстно-вільних граматик використовуються для отримання сутностей. Далі за допомогою UDPipe будується синтаксичне дерево речення. Потім токени з першого кроку зіставляються з елементами отриманого дерева. Інформація про отримані на першому етапі сутності додається до елементів синтаксичного дерева. У результаті ми отримуємо дерево, деякі елементи якого позначені як сутності, і таким чином з'являється можливість описувати шаблони виділення складніших фактів. Насамперед для аналізу даних необхідно отримати синтаксичне подання кожного речення, що розглядається. Так, отримані у форматі CoNNL-U дані перетворюються на структуру даних «дерево», що містять інформацію у вигляді словника у кожного токена-елемента дерева про його морфологічні та синтаксичні ознаки. Наприклад,

аналізуючи таке речення: «Джо Байден обрав нового представника США до ООН» і розглянувши токен «Байден», отримаємо словник, поданий на рисунку 2.6. Далі до кожного токену необхідно додати поле spans, що показує пару індексів символів початку та закінчення входження цього токена у тексті. Це необхідно для коректного зіставлення токенів, отриманих за допомогою UDPipe з токенами, отриманими Yargy, тому що на їх основі працюють різні токенізатори.

```
OrderedDict([('id', 2),
             ('form', 'Байден'),
             ('lemma', 'Байден'),
             ('upostag', 'PROPN'),
             ('xpostag', None),
             ('feats',
                OrderedDict([('Animacy', 'Anim'),
                             ('Case', 'Nom'),
                             ('Gender', 'Masc'),
                             ('Number', 'Sing')])),
             ('head', 1),
             ('deprel', 'appos'),
             ('deps', None),
             ('misc', None)])
```

Рисунок 2.6 – Приклад інформації токена «Байден»

Наступним кроком є вилучення іменованих сутностей за допомогою бібліотеки Yargy. Внаслідок аналізу даного речення аналізатор знаходить іменовану сутність (див. рисунок 2.7).

Джо Байден обрав нового представника США до ООН

Рисунок 2.7 – Приклад вилучення імені за допомогою Yargy

Після зіставлення токенів вилучених сутностей з токенами синтаксичного дерева можна писати правила й шаблони, наприклад для вилучення зв'язків «суб'єкт-предикат», в яких суб'єктом виступає іменована сутність. Використовуючи зв'язок типу «nsubj» для зв'язку з предикатом токеном і тип токена «VERB» можна витягувати найпростіші фрази. Приклад показаний на рисунку 2.8.

```
{ 'subject' : 'Джо Байден', 'predicate': 'обрав'}
```

Рисунок 2.8 – Приклад видобутого відношення «суб'єкт-предикат»

Розширимо шаблон до суб'єкт-предикат-об'єкт. Задамо шаблон опису зв'язку з типом «obj» між токенами предикату і об'єкта. Після цього отримаємо факт, який показаний на рисунку 2.9.

```
{ 'subject' : 'Джо Байден', 'predicate': 'обрав', 'object': 'представника'}
```

Рисунок 2.9 – Приклад видобутого відношення «суб'єкт-предикат-об'єкт»

Даний підхід не обмежується вилученням пар і трійок виду «суб'єкт-предикат-об'єкт», він може описуватися складнішими різноманітними шаблонами і застосовуватися для випадків отримання більш цікавих фактів. Слід зазначити, що, по-перше, для побудови таких шаблонів потрібна доступна візуалізація дерев для ефективної роботи і, по-друге, конструктор для зручного написання, читання і редагування таких шаблонів. Для вирішення цих завдань було розроблено бібліотеку.

2.4 Опис розробленої програмної бібліотеки

Розроблена бібліотека пропонує наступний функціонал:

– візуалізація синтаксичних дерев;
– можливість опису синтаксичних шаблонів;
– вилучення фактів за шаблонами.

Метод візуалізації дерев реалізовано за допомогою бібліотеки matplotlib. Необхідно відзначити, що кожне синтаксичне дерево має деякі особливості, такі як:

– граф є односпрямованим;
– біля основи знаходиться лише один кореневий елемент;
– кожен елемент може мати лише один батьківський і кілька дочірніх елементів.

У зв'язку з цим для візуалізації такого спрямованого графа потрібно

обчислювати координати кожного елемента, попередньо обчисливши кількість елементів, що припадають на кожен рівень і загальну глибину дерева. Приклад візуалізації представлений на рисунку 2.10.

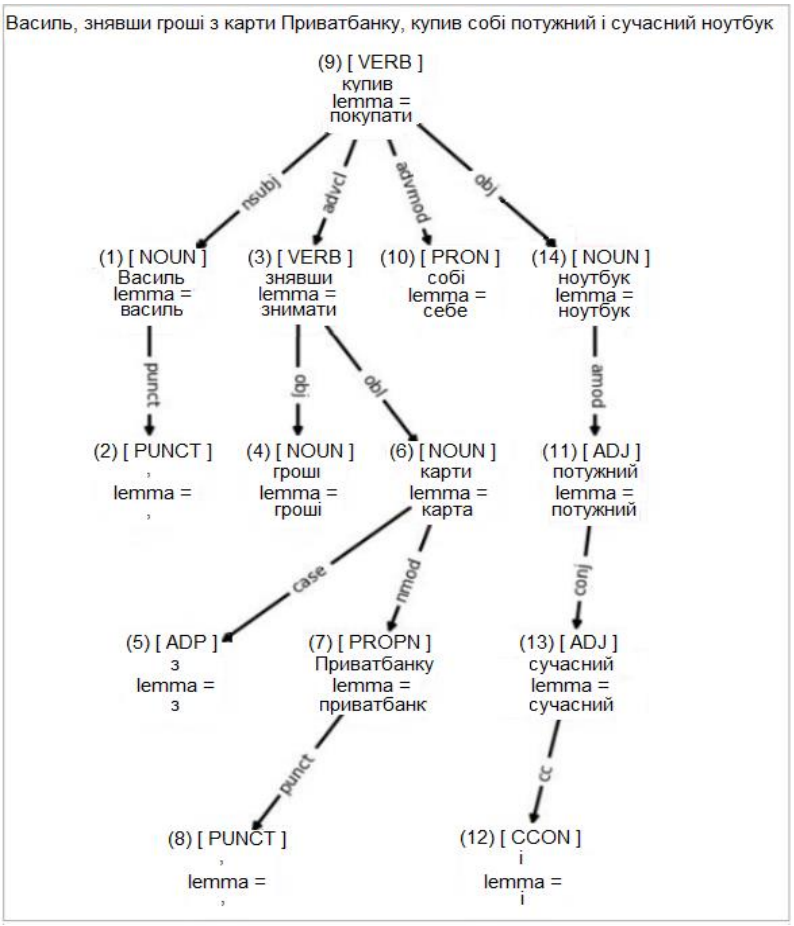


Рисунок 2.10 – Візуалізація синтаксичного дерева

Для розробки шаблонів із вилучення фактів було створено конструктори мовою програмування Python. Їх можна умовно поділити на предикати, тобто функції, що перевіряють кожен токен на відповідність певному правилу. Наприклад, функція «pos(…)» перевіряє частину мови токена на відповідність значенню аргументу. «pos_in(…)» як аргумент приймає масив доступних частин мови. Використання функцій «lem» та «lem_in» перевіряють на відповідність лему слова; regex – згадка регулярного виразу у словоформі. Функції rel і rel_in перевіряють на відповідність типу зв'язку у дочірнього елемента. Специфікація доступних на даний момент предикатів представлена в таблиці 2.1.

Таблиця 2.1 – Специфікація предикатів для побудови шаблонів

Предикати	
pos(value)	Part of Speech == value
pos_in(value)	Part of Speech in value
lem(value)	Лема == value
lem_in(…)	Лема in value
regex(value)	Регулярний вираз value спрацьовує у словоформі
rel(value)	Тип зв’язку == value
rel_in(value)	Тип зв’язку in value
ent(value)	Парсер КВГ знайшов сутність із типом value

Для логічного комбінування правил розроблені конструктори (див. таблицю 2.2): child(…) – конструктор для опису дочірнього елемента, який приймає як аргумент інші конструктори чи правила. _and, _or, _not – логічні оператори для угрупування правил та конструкторів.

Таблиця 2.2 – Специфікація конструкторів для побудови шаблонів

Конструктори	
child(…)	Вказівка на дочірній елемент
_and(…)	Логічне «і»
_or(…)	Логічне «або»
_not(…)	Логічне заперечення

На рисунку 2.11 представлений приклад використання правила для вилучення фактів за шаблоном, що містять іменник, дієслово та число з відсотком.

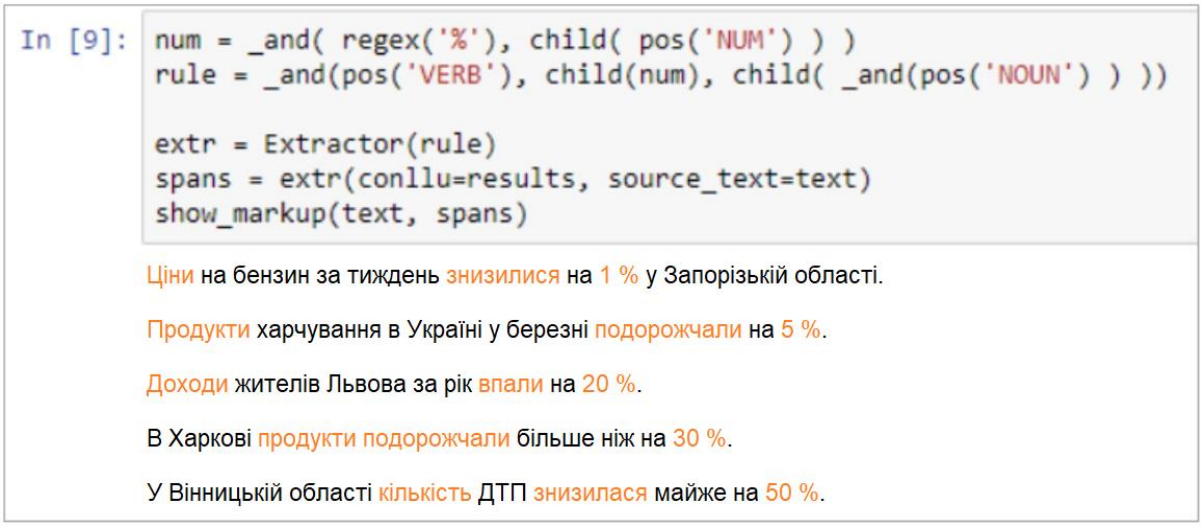


Рисунок 2.11 – Приклад написання шаблонів у синтаксичній структурі

Як видно, такий шаблон відмінно працює на прикладах зміни цін, кількостей ДТП, доходів.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМИ З ВИЛУЧЕННЯ ІНФОРМАЦІЇ З РЕЗЮМЕ НА ОСНОВІ ЗАПРОПОНОВАНОГО ПІДХОДУ

З метою перевірки та демонстрації працездатності запропонованого підходу та розробленої на його основі бібліотеки, було створено застосунок, метою якого є аналіз даних з документів резюме та вилучення наступної інформації: контактні дані (ПІБ, дата народження, телефон, адреса електронної пошти, очікувана посада, ЗП), опис досвіду роботи (організація, посада, період роботи), різні навички, компетенції та додаткова інформація (дозвіл на роботу, готовність до відряджень, військовий квиток).

3.1 Проектування програми та вибір засобів розробки

На рисунку 3.1 представлена діаграма життєвого циклу із завантаження, аналізу та збереження даних кандидата в базу.

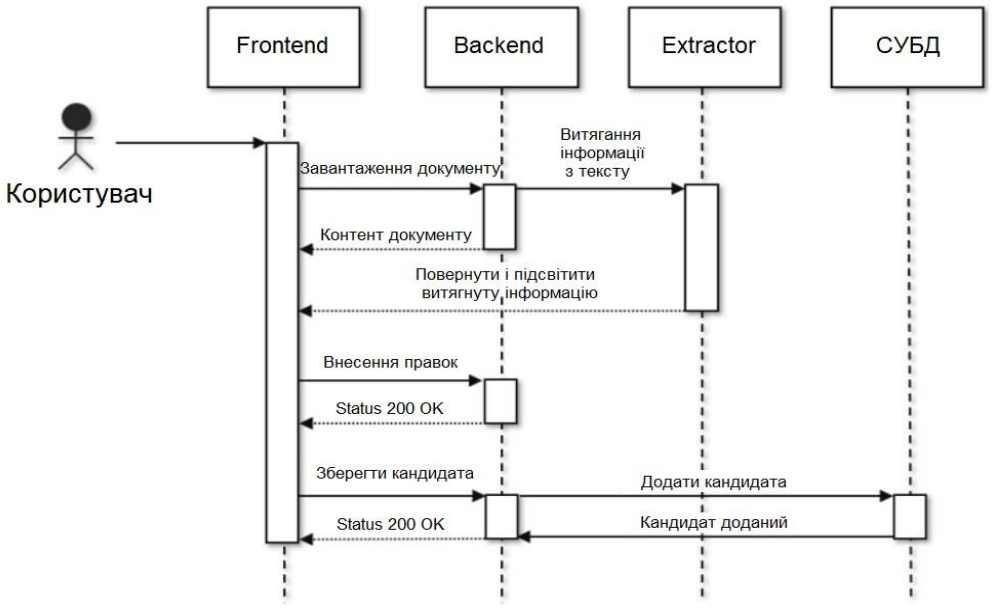


Рисунок 3.1 – Діаграма послідовності програми вилучення інформації з резюме

Як видно з представленої діаграми, користувач взаємодіє з додатком через частину Front-end. Back-end частина, у свою чергу, реалізує основну логіку та операції з СУБД, виклики компонента Extractor та віддає кінцеву відповідь користувачу засобами REST API. Компонент Extractor є окремим процесом, який виконує вилучення інформації з тексту за допомогою розроблених правил-шаблонів, отриманих з використанням реалізованої бібліотеки. Взаємодія Back-end із Extractor здійснюється через асинхронні черги. Ґрунтуючись на поставлених завданнях, а також особливостях роботи створеної бібліотеки, для розробки даного застосунка був обраний наступний стек технологій:

– для реалізації Front-end частини – Vue.js спільно з фреймворком для компонентів Vuetify, розробленого відповідно до специфікації Material Design;
– розробка Back-end здійснювалася за допомогою мови Python спільно з фреймворком Flask. Як головну перевагу Flask можна виділити гнучкість і зручність використання в відносно невеликих проектах. Для об'єктно-реляційного відображення застосовувалася бібліотека SQLAlchemy;
– як СУБД було обрано PostgreSQL. PostgreSQL є найбільш розвиненою об'єктно-реляційною системою на сьогоднішній день та гідною альтернативою комерційним базам даних.

Таким чином, цей застосунок є відносно нескладним з архітектурної точки зору та виконує лише одну основну функцію: вилучення структурованої інформації з резюме для заповнення бази даних.

3.2 Збір даних для написання граматик

Для розробки граматик, що забезпечують достатню повноту під час вилучення інформації, було здійснено збір даних резюме претендентів з відкритих

джерел. Як джерело використовувався один із відомих порталів для пошуку роботи, а також виконавців.

Вилучення набору резюме проводилося шляхом розробки парсера даних на основі бібліотеки Selenium та ChomeDriver як браузера.

На рисунку 3.2 представлено графічну візуалізацію схеми зі збору даних.

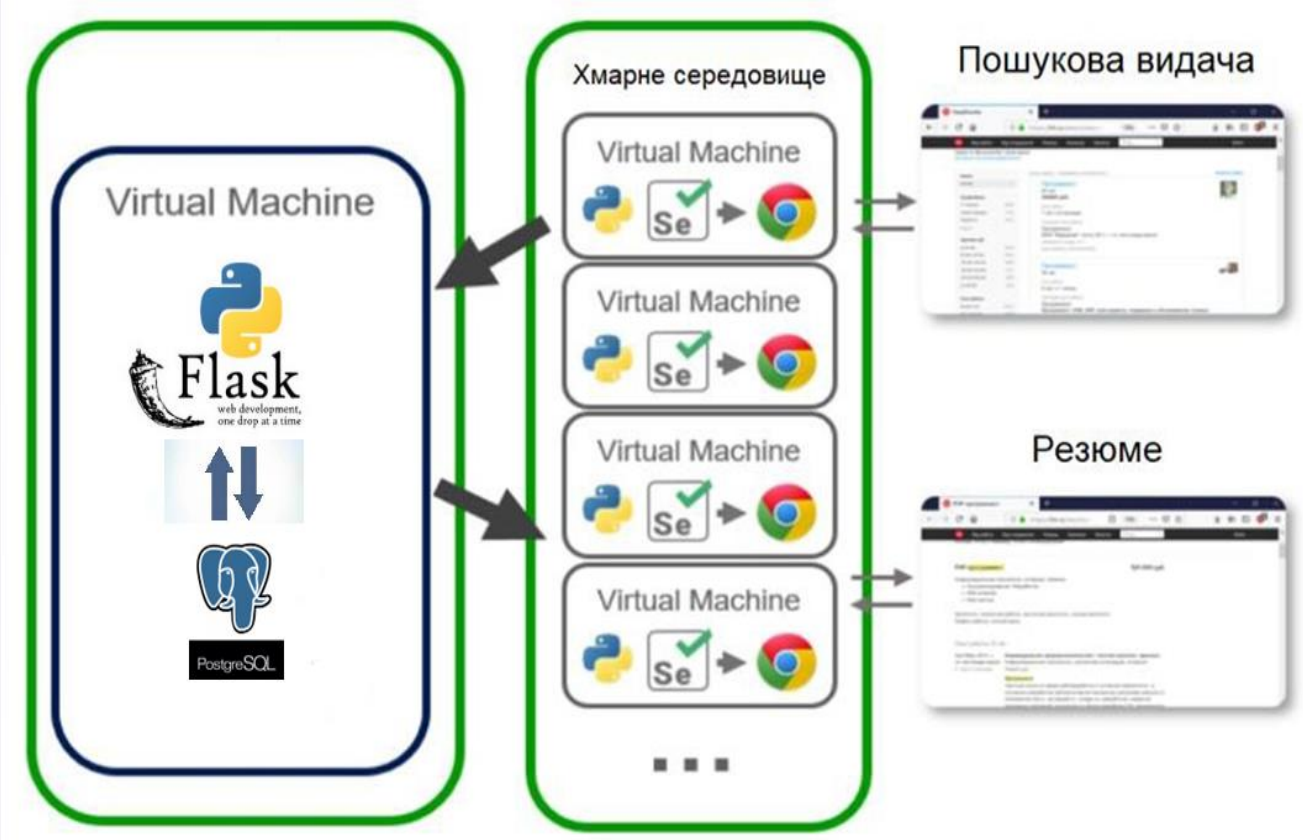


Рисунок 3.2 – Схема збору даних резюме претендентів

Для складання словників пошуковий робот знаходив посаду, що нас цікавить, після чого розбирав пошукову видачу. Результати пошукової видачі – посилання на конкретні профілі-резюме претендентів, які передавалися з мікросервісу черги, це було реалізовано з використанням Python і Flask. Інші роботи зверталися до черги за набором посилань на конкретні профілі, після чого відвідували їх, розбирали HTML код сторінки та передавали результати назад у мікросервіс. Після цього розібрані дані розміщувалися в сховищі на основі PostgreSQL. Пошук здійснювався за запитами, що відповідають популярним на сайтах пошуку роботи посадам та компетенціям:

- менеджер з продажу;
- Data Scientist;
- аналітик Big Data;
- фахівець SAP;
- керівник проектів;
- Autodesk;
- Системний адміністратор;
- інженер-геолог;
- інженер з промислової безпеки;
- науковий співробітник;
- інженер проекту;
- водопостачання;
- архітектор;
- журналіст;
- НДІ;
- промислова безпека;
- токар;
- будівельне проектування;
- охорона праці;
- інженер-електрик.

В результаті було зібрано 2482707 анкет претендентів. Після цього етапу для написання граматик було складено словники ключових навичок та назв посад. На рисунках 3.3 та 3.4 представлені найбільш популярні ключові навички та бажані позиції, виходячи з опису досвіду роботи претендентів. Слід зазначити, що пошукові запити для збору даних містили не репрезентативну вибірку користувачів, лише тих, компетенції яких цікавили компанію-замовника.

```
In [31]: counter.most_common(40)

Out[31]: [('користувач пк', 13594),
          ('робота в команді', 11100),
          ('autocad', 10709),
          ('управління проектами', 10704),
          ('ведення переговорів', 10551),
          ('організаторські навички', 10510),
          ('adobe photoshop', 8494),
          ('ділове листування', 7499),
          ('управління персоналом', 6959),
          ('письменна мова', 6856),
          ('керівництво колективом', 6038),
          ('посвідчення водія категорії b', 5959),
          ('ms powerpoint', 5848),
          ('англійська мова', 5740),
          ('ділове спілкування', 5216),
          ('ms outlook', 5021),
          ('ms office', 4840),
          ('ms excel', 4488),
          ('Укладення договорів', 4397),
          ('ms word', 3943),
          ('навчання персоналу', 3841),
          ('робота з великим обсягом інформації', 3397),
          ('проектна документація", 3357),
          ('archicad', 3135),
          ('coreldraw', 3083),
          ('Охорона праці та техніка безпеки', 2799),
          ('Internet', 2793),
          ('Проведення презентацій', 2760),
```

Рисунок 3.3 – Найбільш популярні ключові навички / компетенції

```
In [34]: counter.most_common(40)

Out[34]: [('архітектор', 6498),
          ('інженер', 5829),
          ('системний адміністратор', 5596),
          ('менеджер з продажу', 5488),
          ('керівник проекту', 4764),
          ('головний інженер', 3943),
          ('токар', 3743),
          ('керівник проектів', 3438),
          ('журналіст', 2810),
          ('архітектор-дизайнер', 2101),
          ('токар-універсал', 2087),
          ('продавець-консультант', 2073),
          ('інженер пто', 1997),
          ('менеджер по роботі з клієнтами', 1932),
          ('провідний архітектор', 1873),
          ('інженер-конструктор', 1748),
          ('кореспондент', 1733),
          ('директор', 1618),
          ('провідний інженер', 1508),
          ('інженер-проектувальник', 1504),
          ('дизайнер', 1451),
          ('головний інженер проекту', 1434),
          ('технічний директор', 1337),
          ('менеджер', 1330),
          ('начальник ділянки', 1249),
          ('менеджер проектів', 1222),
          ('генеральний директор', 1157),
          ('бухгалтер', 1088),
```

Рисунок 3.4 – Найбільш популярні посади виходячи з описів досвіду роботи

На основі отриманих словників складалися правила, що дають змогу отримувати посади, а також навички та компетенції з резюме. Більшість граматик реалізована безпосередньо за допомогою бібліотеки Yargy, однак для вилучення фактів з попередніх місць роботи претендентів, відповідних зв'язкам «Організація-посада-період», вилучаються дані з використанням запропонованого комбінованого підходу.

3.3 Опис програми

Розроблений додаток є сторінкою, через яку можна як завантажити файл у форматі doc/docx/pdf, так і безпосередньо скопіювати текст резюме у вкладку «Вихідний текст». Під час завантаження файлу поле заповнюється автоматично. Розроблені шаблони та правила дозволяють отримувати такі атрибути з анкет:

- ПІБ, контактні дані, бажана посада, ЗП;
- досвід роботи (організація, посада, період роботи);
- навички (технології та особисті якості);
- знання мов;
- додаткова інформація (дозвіл на роботу, готовність до відряджень, військовий квиток тощо).

Після невеликого очікування роботи алгоритму у вкладці «Вихідний текст» з'явиться той самий текст, але з підсвіченими областями, у яких знайшлися згадки (див. рисунок 3.5). Поля праворуч також заповнюються автоматично. Однак для того щоб користувач міг доповнити інформацію або виправити недоліки алгоритму, можна редагувати вміст полів. Після редагування користувач програми натискає кнопку «Зберегти кандидата». Потім його дані завантажуються на сервер у базу даних для подальшого аналізу та інших цілей.

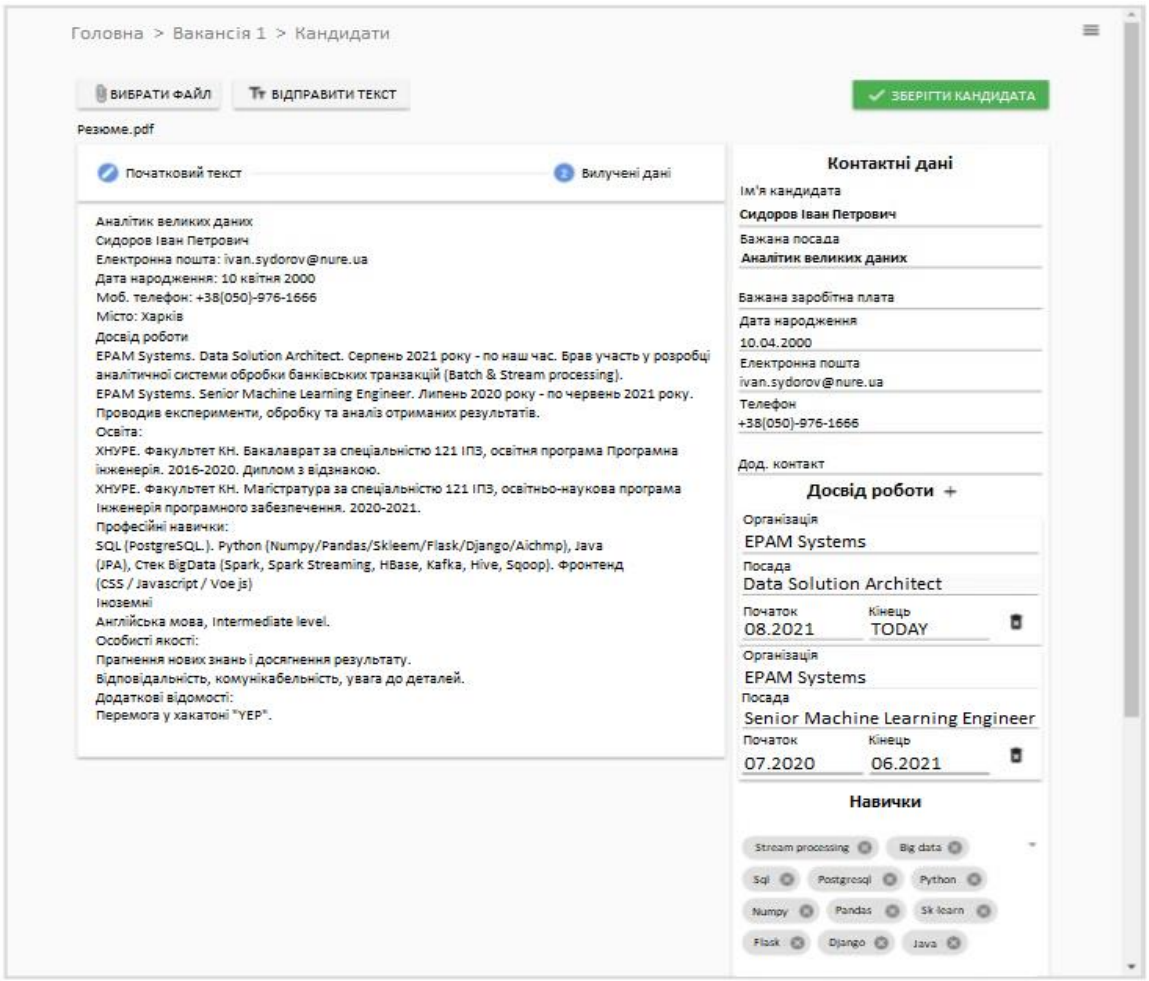


Рисунок 3.5 – Сторінка з вилученим резюме

У розробці кінцевої програми застосовувалися такі інструменти та технології: Python, Flask, PostgreSQL, Vue.js, Selenium. Для складання правил, словників та їх перевірки коректно використовувалися дані з 2 482 707 анкет претендентів. В результаті за допомогою розробленого додатка можна витягувати та заповнювати базу даних всією основною інформацією: контактні дані, досвід роботи, компетенції тощо.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи магістра було запропоновано комбінований підхід для отримання інформації з тексту. Даний підхід полягає у використанні контекстно-вільних граматик спільно з синтаксичними шаблонами, які видобувають з дерев на основі граматик залежності. На основі запропонованого підходу розроблено програмну бібліотеку, що включає власні шаблони для вилучення підграфів з дерев залежностей з урахуванням морфологічних, синтаксичних ознак та результатів вилучення за допомогою контекстно-вільних граматик. У роботі показано, що розроблена бібліотека успішно застосовується на прикладі завдання щодо вилучення інформації з резюме претендентів. Крім того, було створено веб-застосунок для аналізу документів та текстів резюме. Створено шаблони для отримання всієї необхідної інформації, що включає контактні дані, опис досвіду роботи, різні навички, компетенції, досвід роботи з технологіями.

В якості подальших кроків планується розширення функціоналу для складання шаблонів, застосування готових загальнодоступних тезаурусів/словників для можливості використання синонімів, публікація проекту у відкритий доступ, а також розробка банку основних шаблонів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Moens, M.-F. Information Extraction: Algorithms and Prospects in a Retrieval Context. – Netherlands: Springer. – 2009. – 255 p.

2. Lewis Tunstall, Leandro von Werra. Natural Language Processing with Transformers. Revised Edition. – O'Reilly Media, 2022. – 406 p.

3. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition // Prentice Hall series in Artificial Intelligence. – 2009.

4. Chowdhury G.G. Natural Language Processing // Annual review of information science and technology. – 2003. – Vol. 37. – №. 1. – P. 51-89.

5. Reese, Richard M. Natural Language Processing with Java (Community Experience Distilled). – Packt Publishing, 2015. – 262 p.

6. Wortzel A. ELIZA REDUX: A Mutable Iteration // Leonardo. – 2007. – Vol. 40. – №. 1. – P. 31-36.

7. Winograd T. Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. – Massachusetts. Inst. Of Tech. – Cambridge. – 1971. – AI Technical Report № 84.

8. Thuraisingham B. A primer for Understanding and Applying Data Mining // It Professional. – 2000. – Vol. 2. – №. 1. – P. 28-31.

9. Waldrop M. M. The chips are down for Moore's law // Nature News. – 2016. – Vol. 530. – №. 7589. – P. 144.

10. Sha F., Pereira F. Shallow Parsing with Conditional Random Fields // Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology Vol. 1. – Association for Computational Linguistics. – 2003. – P. 134-141.

11. Collins M., Koehn P., Kučerová I. Clause Restructuring for Statistical Machine Translation // Proceedings of the 43rd annual meeting on association for computational

linguistics. – Association for Computational Linguistics. – 2005. – P. 531-540.

12. Kaelbling L. P., Littman M. L., Moore A. W. Reinforcement Learning: A Survey // Journal of artificial intelligence research. – 1996. – Vol. 4. – P. 237-285.

13. Hofmann T. Unsupervised Learning by Probabilistic Latent Semantic Analysis // Machine learning. – 2001. – Vol. 42. – №. 1-2. – P. 177-196.

14. Goldberg Y. A Primer on Neural Network Models for Natural Language Processing // Journal of Artificial Intelligence Research. – 2016. – Vol. 57. – P. 345-420.

15. Jozefowicz R. et al. Exploring the Limits of Language Modeling // arXiv preprint arXiv:1602.02410. – 2016.

16. Manning C. et al. The Stanford CoreNLP Natural Language Processing Toolkit // Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. – 2014. – C. 55-60.

17. Vinyals O. et al. Grammar as a Foreign Language // Advances in neural information processing systems. – 2015. – C. 2773-2781.

18. Cui P. et al. A Survey on Network Embedding // IEEE Transactions on Knowledge and Data Engineering. – 2018.

19. Szabó Z. Compositionality. – Metaphysics Research Lab: Stanford University. – First published Thu Apr 8, 2004. – Substantive revision May 24, 2017.

20. Lin B. Y. et al. Multi-channel Bilstm-crf Model for Emerging Named Entity Recognition in Social Media // Proceedings of the 3rd Workshop on Noisy Usergenerated Text. – 2017. – P. 160-165.

21. Allahyari M. et al. A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques // arXiv preprint arXiv:1707.02919. – 2017.

22. Akbik A., Blythe D., Vollgraf R. Contextual String Embeddings for Sequence Labeling // Proceedings of the 27th International Conference on Computational Linguistics. – 2018. – P. 1638-1649.

23. Hershcovich D. et al. Syntactic Interchangeability in Word Embedding Models // arXiv preprint arXiv:1904.00669. – 2019.

24. Hardeniya N. et al. Natural Language Processing: Python and NLTK. – Packt Publishing Ltd. – 2016.

25. Hope D. A Graph-Based Soft Clustering Algorithm Applied to Word Sense Induction / D. Hope, B. Keller // Computational Linguistics and Intelligent Text Processing: 14th International Conference, 2013. – pp. 368-381.

26. Thushan Ganegedara. Natural Language Processing with TensorFlow: Teach Language to Machines using Python's Deep Learning Library. 1st Edition, Kindle Edition. – Packt Publishing, 2018. – 474 p.

27. Delip Rao, Brian McMahan. Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning 1st Edition. – O'Reilly Media, 2019. – 256 p.

28. Steven Bird, Ewan Klein, Edward Loper. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit 1st Edition. – O'Reilly Media, 2009. – 502 p.

29. Jiang R., Banchs R. E., Li H. Evaluating and Combining Name Entity Recognition Systems // Proceedings of the Sixth Named Entity Workshop. – 2016. – P. 21-27.

30. Ng V. Machine Learning for Entity Coreference Resolution: A Retrospective Look at Two Decades of Research // Thirty-First AAAI Conference on Artificial Intelligence. – 2017. – P. 4877-4884.

31. Ng V. Entity Coreference Resolution // IEEE Intelligent Informatics Bulletin. – 2016. – Vol. 17. – №. 1. – P. 7-13.

32. Emani C. K., Cullot N., Nicolle C. Understandable Big Data: a Survey // Computer science review. – 2015. – Vol. 17. – P. 70-81.

33. Alex Thomas. Natural Language Processing with Spark NLP: Learning to Understand Text at Scale. 1st Edition. – O'Reilly Media, 2020. – 364 p.

34. Hogenboom F. et al. A Survey of Event Extraction Methods from Text for Decision Support Systems // Decision Support Systems. – 2016. – Vol. 85. – P. 12- 22.

35. Cristian M. A Survey of Stemming Algorithms in Information Retrieval / Cristian M., Antonio A., Imbert R. Ramírez J. // Information research, Vol. 19, No. 1, 2014. – pp. 605-625.

36. Ojokoh B. A Review of Question Answering Systems / B. Ojokoh // Journal of Web Engineering 17, 2019. – pp. 717-758.37. Yoav Goldberg. Neural Network Methods for Natural Language Processing (Synthesis Lectures on Human Language Technologies, 37). – Morgan & Claypool Publishers, 2017. – 310 p.

38. Andor D. et al. Globally Normalized Transition-based Neural Networks // arXiv preprint arXiv:1603.06042. – 2016.

39. Gomez-Perez J. M., Denaux R., Garcia-Silva A. A Practical Guide to Hybrid Natural Language Processing. Combining Neural Models and Knowledge Graphs for NLP. Springer. 2020. – 281 p.

40. Poibeau, Thierry; Kosseim, Leila (2001). "Proper Name Extraction from Non-Journalistic Texts". Language and Computers. 37 (1): 144–157.

41. Straka M., Straková J., Hajic J. Prague at EPE 2017: The UDPipe System // EPE 2017. – 2017. – P. 65.

42. Stenetorp P. et al. BRAT: a Web-based Tool for NLP-assisted Text Annotation // Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics. – Association for Computational Linguistics, 2012. – P. 102-107.

43. Nivre J. et al. Universal Dependencies v1: A Multilingual Treebank Collection // LREC. – 2016.

44. Denis Rothman. Transformers for Natural Language Processing: Build, Train, and Fine-tune Deep Neural Network Architectures for NLP with Python, PyTorch, TensorFlow, BERT, and GPT-3, 2nd Edition. – Packt Publishing, 2022. – 564 p.

45. Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, Harshit Surana. Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems 1st Edition. – O'Reilly Media, 2020. – 454 p.

46. Chen D., Manning C. A Fast and Accurate Dependency Parser using Neural Networks // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) – Association for Computational Linguistics. – Doha, Qatar. – 2014. – P. 740–750.

47. Zhang Y., Nivre J. Transition-based Dependency Parsing with Rich Non-local

Features // Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers – Vol. 2, Stroudsburg, PA, USA, – 2011 – P. 188–193.

48. Lim S.K., Muis A.O., Lu W., Ong C.H. MalwareTextDb: A Database for Annotated Malware Articles. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017. Vol. 1, pp. 1557– 1567.

49. Helen M. Meng, Po-Chui Luk, Kui Xu, Fuliang Weng. GLR Parsing with Multiple Grammars for Natural Language Queries // ACM Transactions on Asian Language Information Processing 1(2). – June 2002. – P. 123-144.

50. Zeman D. et al. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies // Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. – 2018. – P. 1-21.

51. Sharonova, N., Kyrychenko, I., Gruzdo, I., Tereshchenko, G. Generalized Semantic Analysis Algorithm of Natural Language Texts for Various Functional Style Types // CEUR Workshop Proceedings, 2022, 3171, pp. 16–26.

52. Smelyakov K., Chupryna A., Karachevtsev D., Kulemza D., Samoilenko Y., Patlan O. Effectiveness of Preprocessing Algorithms for Natural Language Processing Applications // 2020 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 6-9 Oct. 2020, Kharkiv, Ukraine. – pp. 1-5.

53. Chetverikov, G., Puzik, O., Tyshchenko, O. Analysis of the Problem of Homonyms in the Hyperchains Construction for Lexical Units of Natural Language // International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2018, 1, pp. 356–359, 8526663 doi: 10.1109/STC-CSIT.2018.8526663

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

51. Sharonova, N., Kyrychenko, I., Gruzdo, I., Tereshchenko, G. Generalized Semantic Analysis Algorithm of Natural Language Texts for Various Functional Style Types // CEUR Workshop Proceedings, 2022, 3171, pp. 16–26.

52. Smelyakov K., Chupryna A., Karachevtsev D., Kulemza D., Samoilenko Y., Patlan O. Effectiveness of Preprocessing Algorithms for Natural Language Processing Applications // 2020 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 6-9 Oct. 2020, Kharkiv, Ukraine. – pp. 1-5.

53. Chetverikov, G., Puzik, O., Tyshchenko, O. Analysis of the Problem of Homonyms in the Hyperchains Construction for Lexical Units of Natural Language // International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2018, 1, pp. 356–359, 8526663 doi: 10.1109/STC-CSIT.2018.8526663