МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«Розробка веб-інтерфейсу програмного пакета автоматичного виявлення термінів»**

Виконав(ла): студент(ка) 2 курсу, групи 8.1221

спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)

освітньої програми Комп'ютерні науки
(назва освітньої програми)

Є.В. Ярош
(ініціали та прізвище)

Керівник Доцент кафедри, доцент, кф-мн, Єрмолаєв В.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра Комп'ютерні науки

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки

(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерні науки, доктор техн. наук, професор

_____ Чопоров С.В.

(підпис)

« ____ » _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ(СТУДЕНТЦІ)

Ярошу Єгору Вікторовичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка веб-інтерфейсу програмного пакета автоматичного виявлення термінів.

керівник роботи (проекту) Єрмолаєв Вадим Анатолійович, кн-фм, доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від «____» _____ 2022 року № _____

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд існуючих аналогів
2. Проектування програмної системи
3. Способи реалізації програмної системи

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.		
2.	Збір вихідних даних.		
3.	Обробка методичних та теоретичних		
	джерел.		
4.	Розробка першого розділу.		
5.	Розробка другого розділу.		
6.	Оформлення та нормоконтроль		
	кваліфікаційної роботи.	04.11.2022	
7.	Захист кваліфікаційної роботи.	12.12.2022	

Студент _____ Є.В. Ярош
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.А. Єрмолаєв
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ _____
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка веб-інтерфейсу програмного пакета автоматичного виявлення термінів»: 59 с., 15 рис., 8 табл., 52 джерела.

ВЕБ-ІНТЕРФЕЙС, API HTTP REST, ПРОГРАМНИЙ ПАКЕТ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ТЕРМІНІВ, АВТОМАТИЧНЕ ВИЯВЛЕННЯ ТЕРМІНІВ, ТЕРМІНОЛОГІЧНЕ НАСИЧЕННЯ, ДОВГОВИКОНУВАНІ ЗАВДАННЯ, ВЕЛИКІ ДАНІ.

Об’єкт дослідження – аналізатор текстів.

Мета роботи: спроектувати веб-додаток, який надає можливість отримувати терміни з текстів великого обсягу.

Метод дослідження – аналітичний, порівняльний.

Проект спирається на розроблений Єрмолаєвим В.А. та ін. метод виявлення термінів, який дозволяє обробляти тексти великого розміру. Наявна реалізація доступна лише у вигляді вихідного програмного коду на мові Python і потребує інсталяції та налаштування середовища виконання, що ускладнює її використання та розповсюдження. Тому актуальною є розробка веб-служби, яка зможе надати можливість використання широкому колу користувачів.

Особливістю запропонованого проекту є виконання довготривалих процесів у режимі асинхронного конвеєра.

Розроблений проект описує деталізовані структури даних, протоколи взаємодії веб-служби з клієнтом, сценарії роботи серверної частини.

SUMMARY

Master’s Qualification Thesis «Development of the web interface of automated term extraction software»: 59 pages, 15 figures, 8 tables, 52 references.

WEB INTERFACE, HTTP REST API, AUTOMATED TERM EXTRACTION SOFTWARE, AUTOMATIC TERM EXTRACTION, TERMINOLOGICAL SATURATION, LONG-RUNNING TASKS, BIG DATA

The object of research is a text analyzer.

The goal of the work: to design a web application that provides the ability to obtain terms from large volumes of texts.

The research method is analytical and comparative.

The project is based on the work developed by V. A. Ermolaev et al. a automated term extraction method that allows processing large texts. The current implementation is only available as Python source code and requires a runtime environment to be installed and configured, making it difficult to use and distribute. Therefore, the development of a web service that can provide the possibility of use to a wide range of users is relevant.

A feature of the proposed project is the execution of long-term processes in the asynchronous conveyor mode.

The developed project describes detailed data structures, web service interaction protocols with the client, scenarios of the server part.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПК - персональний комп’ютер

АВТ - Автоматичне виділення термінів

ПП - програмного продукту

СКБД - система керування базами даних

БД - база даних

C-value – міра належності словосполучення до множини термінів.

ВСТУП

Автоматичне виявлення термінів (далі АВТ) можна визначити як автоматизований процес ідентифікації термінології з корпусу спеціалізованих текстів. Незважаючи на численні дослідження, це залишається не менш складним завданням тому що терміни дуже важко визначити. Терміни зазвичай описуються як «лексичні одиниці, що представляють поняття домен», але такі визначення залишають місце для багатьох запитань про фундаментальний характер термінів. Оскільки АВТ має автоматично ідентифікувати терміни зі спеціального тексту, відсутність консенсусу щодо основних характеристик термінів проблематична. Розбіжності охоплюють обидва практичні аспекти, наприклад довжина терміну та частина мови, а також теоретичні міркування про різницю між словами (або словосполучення/фрази) і терміни. Це створює великі труднощі для багатьох аспектів АВТ, від збору даних до методології виявлення та оцінки.

Методів автоматичного виявлення термінів доволі багато але більша їх частина реалізована у вигляді прототипів якими важко користуватися або взагалі є недоступними для звичайних користувачів [1, 2]. Наразі відомо про такі сервіси, як BIOTEX, TerMine, FiveFilters, TaaS, Terminology Extraction, TBXTools, UPM Term Extractor. Вони дають можливість будь кому знаходити терміни в англійських текстах але вони не дають можливість працювати з текстами великих об’ємів.

Метою роботи є проектування WEB-інтерфейсу для програмного пакета автоматичного визначення термінів, який розробили Єрмолаєв В.А., Коса В.В., Добровольский Г.А. та ін. [1, 2, 6]. Перевагою є можливість розділу великих текстів на частини, обробки кожної частини окремо та комбінації результатів в єдиний перелік термінів. Задачами роботи є:

- Створення огляду існуючих аналогів.
- Створення проекту програмної системи.
- Способи реалізації.

Дипломна робота складається з 3 частин кожна з яких присвячена одній із названих вище задач.

1 ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ

1.1 Методи автоматичного виявлення термінів

Збір даних, тобто створення предметних корпусів з термінів потребує часу та зусиль. Якщо використовується ручна анотація термінів, то існує ймовірність відсутності консенсусу щодо протоколу анотації. Це призводить до дефіциту доступних ресурсів. На додачу це означає, що кілька доступних наборів даних складно поєднувати і часто вони охоплюють лише одну мову та домен. Хоча метод ручної анотації часто обходили, починаючи з наявних ресурсів таких як онтології або термінологічні бази даних, спеціалізовані словники чи книжкові покажчики, такі стратегії не матимуть таких же переваг і рідко охоплюють усі терміни у всьому корпусі на відміну від анотації вручну.

У більшості методів автоматичного виявлення термінів використовуються лінгвістична та статистична обробки тексту [2, 3, 4, 5]. Часто, на початку аналізу, проводиться відкидання, так званих, стоп-слів. Стоп-слова - це слова або фрази, які не несуть смислового навантаження. Наприклад, це розділові знаки, текстові символи, прийменники, займенники, короткі слова та ін., тобто будь-які слова, що не несуть додаткового сенсу. Таку оптимізацію доречно проводити як при лінгвістичній так і при статистичній обробках тексту.

Серед лінгвістичних виділяються методи обробки тексту за допомогою n-грам, токенізації, стемінгу, лематизації та POS tagging. Дамо їхні краткі визначення.

N-грами - послідовність з n елементів. Елементами послідовності можуть бути символи, слова, склади слів або звуки.

Токени - це набір символів, які відповідають певним шаблонам. Токенізація - це процес перетворення послідовності символів в послідовність токенів, та визначення їх типів. Наприклад, це розбиття речення на окремі слова або токени: іменники, дієслова, займенники і т. д..

Стемінг - це процес скорочення слова, за допомогою відкидання допоміжних частин: закінчення чи суфіксу. Наслідки такого скорочення можуть бути схожі на виділення кореня слова, але метод стемінгу використовує інші правила. В результаті основа слова може відрізнятися від його морфологічного кореня.

POS tagging - процес обробки тексту, завданням якого є визначення до якої частини мови належить слово та присвоєння йому відповідного тегу, наприклад іменник, дієслово і т.д.. Отримані результати використовуються при подальшому синтаксичному аналізу речень та виділенні словосполучень, що є важливою частиною лінгвістичної обробки. [6]

Лематизація - процес приведення слова до леми, його словникової форми. Проводиться у кілька етапів. Спочатку визначається частина мови до якої належить слово, так званий POS tagging. Після, в залежності від визначеної частини мови, до слова застосовуються відповідні правила стемінгу. Таким чином якість дуже залежить від правильності розпізнання частини мови.

Найскладніший варіант лінгвістичної обробки включає синтаксичний аналіз речень та виділення словосполучень, які можуть бути термінами. Наприклад, застосовуються шаблони словосполучень [1].

Статичний аналіз передбачає процес застосування статистики для оцінки важливості виділеного терміну [13, 7, 8, 9]. Найчастіше використовується оцінка частоти зустрічаємості терміну, term frequency (TF) та її аналоги [10].

Для оцінки важливості терміну у тексті, що є частиною набору документів використовуються TF-IDF, C-value і т.д.. TF-IDF (IDF - inverse

document frequency). Вага (значимість) терміну пропорційна кількості його вживань у тексті, і обернено пропорційна частоті вживання у інших документах набору [11, **12, 13, 14]**.. C-value - метод виділення багатослівних термінів, заохочує словосполучення, що не входять до складу інших, більш довших. Довгі терміни зустрічаються у текстах рідше, ніж короткі, для врахування цього і був запропонований метод C-value [15].

Статистична обробка може проводитись у два етапи: оцінювання виділених термінів та їх подальше ранжування. Ранжуванням є оцінювання отриманих термінів, яке показує ймовірність того, що це дійсно термін. Існують різні способи вимірювання оцінки, які можна поділити на ті, де обчислюються частота появи терміну в тексті та оцінюється його попадання у контексти [15], а також ті, де використовуються фонові або референтні корпуси або проводиться моделювання тем та доменів [16, 17, 18, 19]. Фонові (референтні) корпуси - це підмножини текстів, з якими відбувається порівняння отриманого корпусу.

Таким чином рішення про значущість терміну приймається, в залежності від його оцінки. Більша частина методів спирається на оцінки, отримані за допомогою лише одного з методів розрахунку оцінок. І, в залежності від встановленого порогу відсікання, відрізняє значущі терміни від незначущих. Методи, що використовують оцінки отримані за допомогою декількох методів розрахунку оцінок, застосовують їх лінійні комбінації, обчислюють загальну вагу, голосування, або контрольоване чи напівконтрольоване навчання [20, 3, 21, 22].

Оцінки ефективності найпоширеніших алгоритмів ABT, які базуються на методі C-value, приведені у Таблиці 1.1. [3, 6, 15]. Розглянуті методи застосовувались на наборі даних GENIA, описаному нижче. В огляді [3] використовувалось відкрите програмне забезпечення для ABT, написане на Java – JATE 2.0. В огляді [6] використовувалось програмне забезпечення з відкритим кодом написане на Scala (4S) – ATR4S.

Таблиця 1.1. - Оцінки ефективності найпоширеніших алгоритмів АВТ.

Метод	Метрики	Значущість терміну (+/-)	Повертає лише значимі терміни	Точність (GENIA; середнє)	Тривалість виконання, Інструмент
TTF	TTF	+	-	0.70; 0.35	0.34 JATE
ATF	Середня Частота Термінів	+	-	0.71; 0.33 0.75; 0.32	0.37 ATR4S 0.35 JATE
TTF-IDF	TTF+IDF	+	-	0.82; 0.51	0.35
RIDF	Залишкова IDF	-		0.71; 0.32 . 80; 0.49	0.53 ATR4S 0.37 JATE
C-value	C-value, NC-value	+	-	0.73; 0.53 0.77; 0.56	1.00 ATR4S 1.00 JATE
Weirdness	Дивність	-	-	0.77; 0.47 0.82; 0.48	0.41 ATR4S 1.67 JATE
GlossEx	Лексична когезія терміну (cohesion), Специфічність домену	-		0.70; 0.41	0.42 JATE
Term Extractor	Достовірність Домену, Консенсус Домену, cohesion, Структурна Відповідність	-	+	0.87; 0.46	0.52 JATE
PU-ATR	NC-value, Специфічність домену	-	+	0.78; 0.57	809.21 ATR4S

З Таблиці 1.1 видно, що найбільш надійним методом є C-value [3, 6, 15]. Він показує більшу продуктивність в порівнянні з найшвидшими методами.

1.2 Інструменти для автоматичного виявлення термінів

Виявлення термінів можна проводити різними способами: вручну, самостійно формуючи список термінів документа, або автоматично, за допомогою засобів виявлення термінології. Першій спосіб є більш точним, але й найдовшим. Зважаючи на потреби в обробці великої кількості різної текстової інформації росте попит на інструменти виявлення термінів.

Існують різні інструменти автоматичного виявлення термінів [6]. Деякі працюють в відповідній предметній області та відповідними мовами, наприклад BIOTEX, веб-додаток, який реалізує автоматичне виявлення біомедичних термінів із вільного тексту англійською та французькою мовами, при цьому він залежність від домену [23].

TerMine анотує терміни з коротких текстів англійською мовою, є обмеження за доступом (його треба запросити) та розміром тексту. Використовує C-value та надає інтерфейс SOAP. SOAP (англ. Simple Object Access Protocol) - протокол обміну структурованими повідомленнями в розподілених обчислювальних системах, базується на форматі XML [24, 25].

FiveFilters від MedialabPrado перетворює фрагмент тексту або веб-статтю на список відповідних термінів-іменників, що спрощує аналіз та дозволяє використати прості лінгвістичні інструменти, такі як POS tagger і деякий статистичний аналіз для визначення ваги термінів. Застосовується для виділення ключових тегів. Має обмежений розмір тексту, що подається на вход [26]. Інтерфейсна частина представлена на рис.1.1

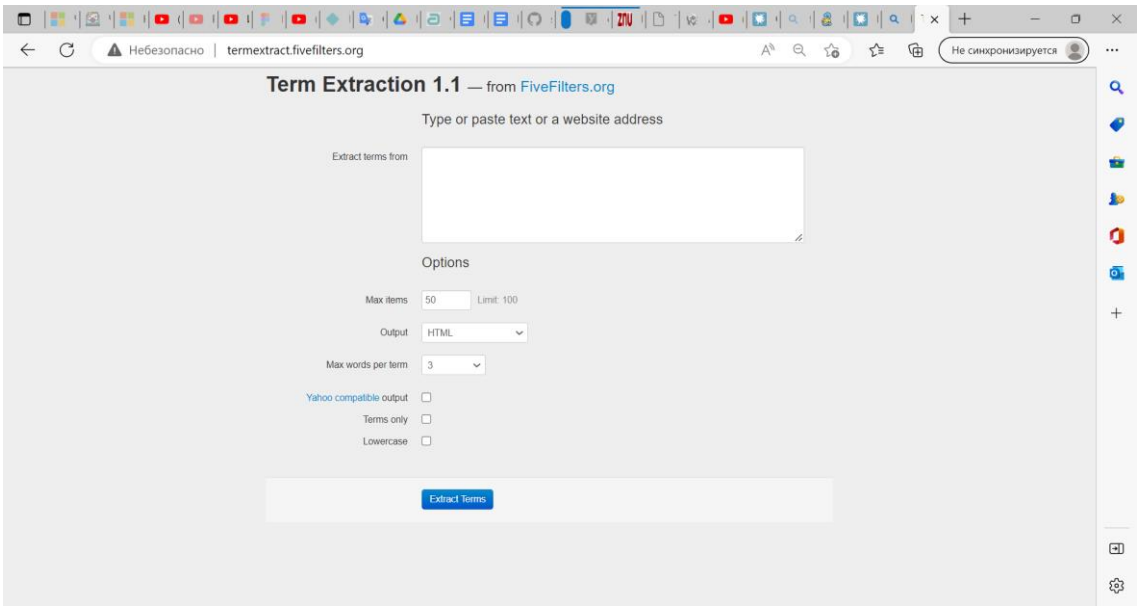

Рисунок 1.1 Інтерфейсна частина FiveFilters

TaaS (TaaS EU Project) дозволяє додати документ, витягнути терміни та зіставити їх із бажаною мовою. Базується на лінгвістичному аналізі для позначення частин мови, використовує морфо-синтаксичні шаблони, застосовує такі статистичні ознаки, як показник частоти. Не надає оцінки значущості терміну [27]. Інтерфейсна частина TaaS представлена на рис.1.2.

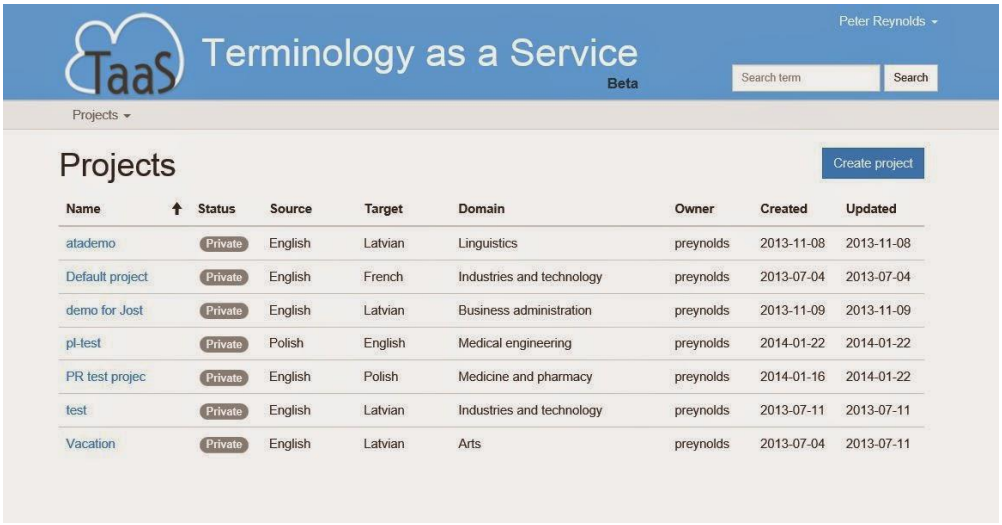

Рисунок1.2 Інтерфейсна частина TaaS

Для початку роботи треба створити новий проект, для чого потрібно ввести назву проекту, вихідну мову, цільову мову та домен. Також можна додати опис, назву продукту, ім’я клієнта та бізнес-підрозділ (рис.1.3).

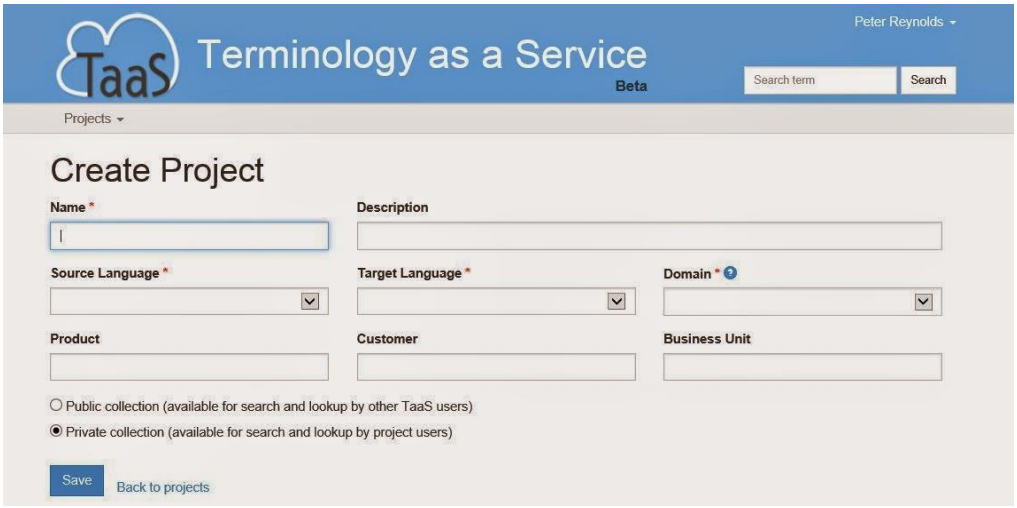

Рисунок 1.3 Create Project в TaaS

Після створення проекту одкривається простір проекту з вкладками, виділеною є вкладка для документів. Можна просто перетягнути документ, який треба додати, або натиснути «Додати документи».

На вкладці «Extraction» є можливість вибрати функцію виявлення термінів (рис.1.4).

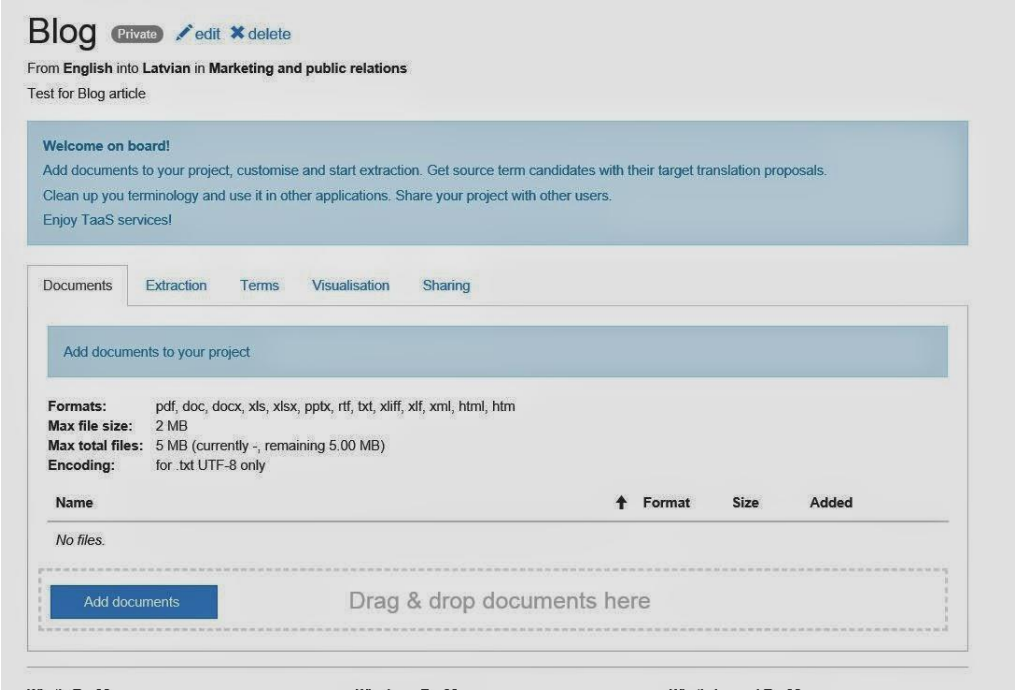

Рисунок 1.4 Проект з вкладками в TaaS

Terminology Extraction - веб-додаток, який здобуває терміни із вставленого тексту. Базується на порівнянні частоти слів у даному документі з їх частотою у мові. Ідея полягає в тому, що слова, які дуже часто зустрічаються в документі, але рідко в мові, ймовірно, і є термінами. Використовує Статистику Пуассона, оцінку максимальної вірогідності та IDF, оцінка терміну повертається у вигляді числового значення. Має обмеження за мовою, працює з: англійською, французькою, італійською [28].

Інтерфейсна частина Terminology Extraction представлена на (рис.1.5)

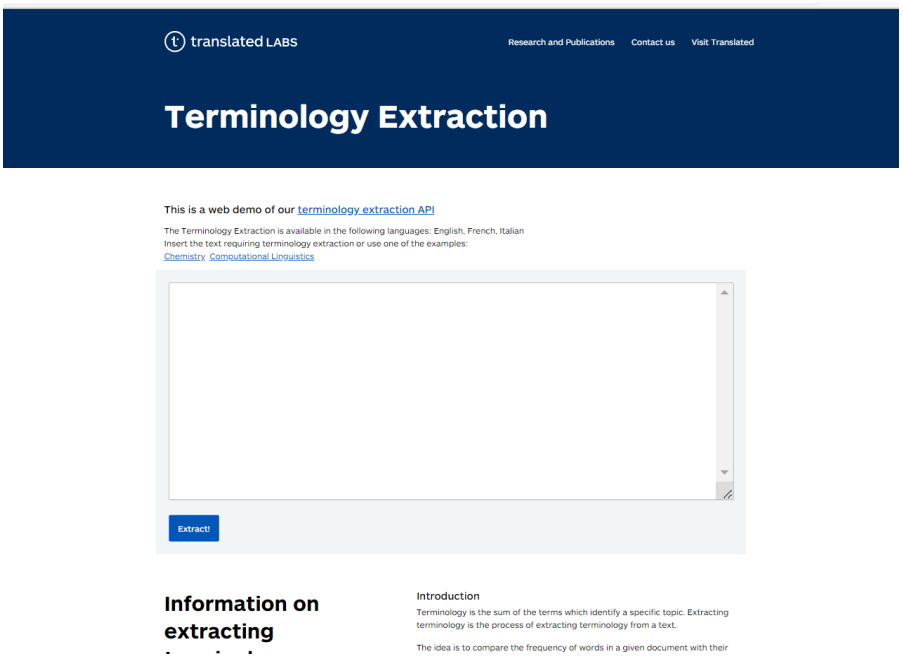

Рисунок 1.5 Інтерфейсна частина TermFinder

TBXTools дозволяє легко та швидко видобувати термінологію та керувати нею. Реалізує статистичні та лінгвістичні методи разом із кількома утилітами для створення та керування термінологічними базами даних. Написаний на Python і використовує набір інструментів для природної мови NLTK (Natural Language Toolkit). Незалежний, багатомовний, використовує мовний корпус, проводить попередню обробку, видаляя n-грами зі стоп-словами [29]

UPM Term Extractor – написаний на Java, застосовується для здобуття термінів та їх відносин з наукових статей. Незалежний, базується на використанні C-value. Має обмеження за розміром текстових даних, не більше 15 M [30, 31].

Лише декілька розглянутих інструментів автоматичного виявлення термінів мають доступний інтерфейс, більшість з них лежать на GitHub, та не має легкого у застосуванні інтерфейсу.

1.3 Набори даних для перевірки методів виявлення термінів

Набір даних - колекція однотипних даних, що застосовується в задачах машинної обробки даних. Розглянемо набори даних для перевірки методів виявлення термінів.

Набір даних ACTER (Annotated Corpora for Term Extraction Research) містить спеціалізовані корпуси трьома мовами (англійською, французькою та голландською) та охоплює чотири сфери (корупція, виїздка (вид верхової їзди), серцева недостатність та енергія вітру) які були анотовані вручну [32]. Для оцінки використовуються стандартні методи оцінювання (точність, повнота, F-міра) і ранжування. Анотації представляються у вигляді неструктурованих списків всіх унікальних анотованих термінів (один термін і його мітка на рядок), при цьому діапазон кожної появи анотованих термінів у їх контексті не вказаний.

Для англійської мови часто використовуються такі два анотовані набори даних: GENIA (розроблений в 2003 р.) і ACL RD-TEC 2.0 (розроблений в 2016 р.) [33, 34].

GENIA – це колекція з 2000 тез з бази даних в галузі біомедицини MEDLINE, крім того «фактори транскрипції в клітинах крові людини». В результаті двома експертами домену було анотовано більш ніж 400 тис. токенів, на виході було отримано 93 293 анотації термінів.

ACL-RD-TEC 2.0 складається з 300 анотованих рефератів, які було взято із довідкового корпусу антології ACL. Анотація була проведена також двома експертами, які обробили 33 тисяч токенів, що призвело до 6818 анотацій термінів.

Розробники корпусу ACL RD-TEC указують три основні переваги перед GENIA:

– Дослідники АВТ матимуть краще розуміння матеріалу

– Корпус ACL RD-TEC охоплює три десятиліть, що дозволяє провести деякі дослідження еволюції термінів

– Анотація більш прозора з інструкціями щодо застосування, є у вільному доступі та можливістю завантажувати анотації обох експертів окремо.

Є й інші приклади, наприклад CRAFT corpus (The Colorado Richly Annotated Full Text Corpus) [35], ще один англійський корпус у біомедичній сфері (99 907 анотацій на 560 тисяч лексем) (2012 р), англійський автомобільний корпус (28 656 анотацій понад 224 159 лексем) (2012-2014 р.), діахронічний англійський корпус з машинобудування (+10 тис. анотацій на 140 тис. слів) (2016 р), французький корпус наук про мову (14 544 унікальних підтверджених термінів знайдених у 397 695 словах) (2014 р.), невеликий німецький корпус про DIY, кулінарію, полювання та шахи, який зосереджений на інтер-анотаторній угоді між спеціалістами (912 анотацій на які принаймні 5 із 7 анотаторів погодилися, понад 3075 слів) (2018 р.). Поки це не є повним списком, він ілюструє важливі та логічні тенденції; створені золоті стандарти які є досить великими (понад 10 тисяч анотацій) або охоплює кілька мов та/або домени.

Хоча це не цілком проблематично, анотаційні вказівки для всіх цих корпусів відрізняються і отже, самі анотації також. Це створює труднощі, оскільки порівняння продуктивності АВТ на кількох корпусах не обов’язково відображатиме відмінності в продуктивності між доменами чи мовами, але також може показувати контраст між різними стилями анотацій. Відмінності можуть бути досить суттєвими. Наприклад у GENIA та ACL RD-TEC, вкладені анотації не допускаються в CRAFT вони дозволені лише за певних умов, а в проект TermITH вони дозволені в більшості випадків. Крім того, важливо зазначити, що анотації проекту TermITH базуються на ручному анотуванні результатів АВТ, а не на ручному анотуванні в необробленому тексті. Останнє зауваження полягає в тому, що деякі корпуси анотовані

кількома термінами або були навіть анотовані відповідно до великих таксономій, тоді як інші не роблять жодних відмінностей крім термінів.

1.4 Web-інтерфейс для довготривалих задач

Дамо визначення web-інтерфейсу. Web-інтерфейс - це сукупність засобів, за допомогою яких користувач взаємодіє з веб-сайтом або веб-додатком через браузер. Однією з основних вимог до розробки web-інтерфейсів є їх однаковий зовнішній вигляд і однакова функціональність при роботі в різних браузерах.

Web-інтерфейс дає можливість універсального віддаленого доступу до служб та пристроїв, у цьому технології практично нема альтернатив. Але водночас, оскільки такий інтерфейс доступний усім, постають серйозні питання забезпечення безпеки, зокрема автентифікація та авторизація користувачів, шифрування переданих даних від сторонніх очей, модерація вмісту тощо.

Веб-додаток - розподілений додаток, в якому за клієнтську частину відповідає браузер, а за серверну – веб-сервер. Браузер найчастіше реалізує так звані тонкі клієнти, це коли логіка додатку виконується сервером, а браузер переважно реалізує зображення інформації, отриманої з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є кросплатформовість, тобто клієнтська частина не залежать від конкретної операційної системи користувача.

Розглянемо архітектуру веб-додатків. Веб-додаток отримує запит від клієнта і виконує його, після цього формує веб-сторінку і відправляє її клієнтові мережею з використанням протоколу HTTP. Веб-додаток може бути клієнтом інших служб, наприклад, бази даних або стороннього веб-додатку, розташованого на іншому сервері.

Для створення веб-додатків використовуються різноманітні серверні технології та мови програмування: ASP, ASP.NET, Java, Groovy, PHP, Python тощо. Клієнтська частина може використовувати: JavaScript, Flash, Java, JavaFX, ActiveX, Silverlight.

Як правило, програмний код виконується синхронно, тобто послідовно: лише одна конкретна операція відбувається на даний момент часу. Якщо деяка функція залежить від результату виконання іншої функції, то вона повинна дочекатися доки потрібна їй функція не завершить свою роботу і не поверне результат. При такому підході виконання програми з точки зору користувача буде призупинено.

Веб-додаток може мати справу з тривалими процесами. Наприклад, робота з базами даних або виконання складних обчислень. Поки здійснюється одна операція, добре було б виконати ще кілька, такий підхід називається асинхронним. Асинхронне програмування збільшує ефективність, тому що дозволяє не блокувати основний потік виконання.

Асинхронність - це процес обробки введення/виводу, що дозволяє продовжити обробку інших завдань, не чекаючи завершення попереднього завдання.

Для більшої інтерактивності й продуктивності, до розробки веб-додатків, розумно використовувати підхід, при якому веб-додаток буде здатен відправляти веб-запити до сервера у фоновому режимі. Така технологія називається Ajax та дозволяє не перезавантажувати сторінки веб-додатку цілком, а лише довантажувати в відповідному місці необхідні дані з сервера, що значно пришвидшує роботу [36, 37].

Але бувають задачі, вирішення яких потребує занадто багато часу, наприклад, годину, дві а той більше. В більшості випадків користувач не може закрити браузер або вимкнути комп’ютер поки задача не буде виконана, що приводить до деяких незручностей. Перш за все втрачається

гнучкість: користувач не може видати завдання, зайнятися своїми справами, а потім отримати результат, не залежачи від місця та обладнання.

Було б добре організувати веб-інтерфейс таким чином, щоб була можливість запустити список завдань, піти у справах, якщо необхідно виключити персональний комп'ютер (ПК), потім повернутися, включити, отримати результат.

1.5 Висновки

У цьому розділі було розглянуто методи автоматичного виявлення термінів, у більшості з яких використовуються спочатку лінгвістична, а потім і статистична обробки тексту. Крім того проводиться попередня обробка та відкидаються слова, що не несуть додаткового сенсу.

Лінгвістичні методи обробки тексту найчастіше використовують n-грами, токенізацію, стемінг, лематизацію та POS tagging. Складний варіант лінгвістичної обробки включає синтаксичний аналіз речень та виділення словосполучень, які можуть виявитися термінами. Наприклад, виділяються шаблони словосполучень. У разі статичного аналізу найчастіше оцінюється частота зустрічаємості терміну та її аналоги.

У Таблиці 1.1 було приведено оцінки ефективності найпоширеніших алгоритмів АВТ.

Було розглянуто різні інструменти автоматичного виявлення термінів BIOTEX, TerMine, FiveFilters, TaaS, Terminology Extraction, TBXTools, UPM Term Extractor. Після розгляду були виявлені такі особливості застосування:

- можливість працювати лише в відповідній предметній області;
- наявність обмеження за мовами;
- наявність обмеження за розміром тексту.

Більшість з розглянутих інструментів англомовні та мають обмеження за розміром тексту. До того ж всі вони працюють лише в режимі реально часу та не надають можливість задавати декілька текстів одночасно.

Також було розглянуто набори даних для перевірки методів виявлення термінів: ACTER, GENIA, ACL RD-TEC, CRAFT corpus.

Були розглянуті різні інструменти автоматичного виявлення термінів, більшість з яких не має легкого у застосуванні інтерфейсу.

Наведені принципи побудови веб-додатків, указані переваги асинхронного підходу. Для більшої інтерактивності й продуктивності, було б добре організувати веб-інтерфейс таким чином, щоб була можливість задати список завдань і через деякий час отримати результат.

2 ПРОЕКТ ПРОГРАМНОЇ СИСТЕМИ

2.1 Постановка задачі

Метою роботи є проектування WEB-інтерфейсу для програмного пакета автоматичного виділення термінів, який реалізує метод, розроблений Єрмолаєвим В.А., Косой В.В., Добровольским Г.А. та ін. [1, 6, 2]. Так як указаний програмний пакет зберігається на GitHub, та не має легкого у застосуванні інтерфейсу.

Виконання задачі автоматичного виявлення термінів в тексті великого розміру може зайняти занадто багато часу, та буде потребувати великої кількості ресурсів. Рішенням даної задачі може стати можливість розділу великих текстів на частини, обробки кожної частини окремо та комбінації результатів в єдиний перелік термінів. Навіть в такому випадку обробка може зайняти багато часу.

Доречно було б організувати роботу з програмою таким чином, щоб була можливість завдати завдання, зайнятись іншими справами, навіть виключити ПК. Через деякий час зайти до програми, переглянути статус обробки або отримати результат. При цьому бажано зберегти гнучкість - незалежність від місця та обладнання користувача.

Зважаючи на це, для реалізації було обрано розробку веб-додатку, в такому випадку за клієнтську частину буде відповідати браузер, а за серверну – веб-сервер. Складність роботи полягає в тому, що отримання термінів є тривалим процесом і має виконуватися асинхронно.

На вхід подаються тексти у вигляді файлів. На виході користувач отримує терміни в порядку зменшення їх важливості.

2.2 Опис методу виявлення термінів

Термін - це слово або словосполучення, яке позначає окреме семантично виважене поняття певної вузької специфічної виробничої галузі, наукового знання чи будь-якої професійної діяльності людини у сучасному світі і вступає в системні зв'язки з іншими словами та словосполученнями даної галузі [38].

Семантичним відбитком набору документів називається набір термінів, здобутих з усіх його документів. Виходячи з цього, набір документів буде повно описувати області знань тоді, коли вона містить піднабор з тим самим набором термінів, що і увесь набір, таке явище називається термінологічним насиченням.

Для опису методу виявлення термінів дамо деякі визначення [**Ошибка! Закладка не определена.**].

Реченням *s* буде називатись мовна одиниця, яка зв’язана із навколишнім текстом змістовно, а не граматично.

Слово вживається у звичайному значенні, позначаючи мінімальну структурно-смислову, вільно відтворювану одиницю мови, що служить для побудови висловлювань (речень) [39]. Слово відрізняється від терміну.

Словосполученням *c* називається слово або скінчена послідовність слів [39].

Терміном τ [40] буде називатись словосполучення, що позначає поняття заданої області знань. Термін може складатися із кількох слів і завжди має самостійне значення.

Стоп-слова [8] – це словосполучення які не несуть смислового навантаження. Для багатьох мов існують переліки загальновідомих стоп-слів, також у текстах вибраної тематики можуть бути специфічні стоп-слова, які варто виявляти окремо.

C-value – міра належності словосполучення до множини термінів.

Термінологічне насичення [21, 41] впорядкованої множини публікацій, буде визначатися вимогою, коли додавання публікацій у кінець переліку майже не змінить перелік термінів.

Процес АВТ складається із лінгвістичної та статистичної обробок [**Ошибка! Закладка не определена.**]. Під час лінгвістичної обробки проводиться розмітка частин мови, лінгвістичний фільтр, виключення стоп-слів.

Під час розмітки частин мови кожному слову вхідного тексту приписується граматична мітка («іменник», «прикметник», «дієслово» і т.д), яка, в подальшому, використовується лінгвістичним фільтром.

За допомогою заданих налаштуваннями шаблонів, лінгвістичний фільтр виявляє словосполучення, які можуть бути термінами. Більшість термінів складаються із іменників та прикметників, зрідка прийменників. В методі використовуються шаблони словосполучень, записані у позначеннях, близьких до регулярних виразів:

іменник+ іменник

((прикметник|іменник)+|((прикметник|іменник)∗(прийменник)?)(прикметник|іменник)∗)іменник.

Під час статистичної обробки для кожного словосполучення *c* обчислюється дійсне число C-value, яке відображає міру того, що словосполучення є терміном. Мірою відмінності наборів термінів є Termhood Difference [41].

2.3 Варіанти використання та сценарії використання

Варіант використання (англ. Use Case) - при розробці програмного продукту (ПП) означає опис поведінки системи, перелік запитів та відповідей на них. Тобто варіант використання описує, «хто» і «що» може зробити з програмним продуктом, що розробляється.

Головне призначення діаграми варіантів використання полягає у формалізації функціональних вимог до ПП на ранній стадії проектування.

Метою роботи є проектування web-інтерфейсу для програмного пакета автоматичного виділення термінів, який зберігається на GitHub, та не має легкого у застосуванні інтерфейсу. Складність роботи полягає в тому, що отримання термінів є довгим процесом і має виконуватися асинхронно.

В якості реалізації було обрано розробку веб-додатку. Такий підхід дозволить організувати роботу з ПП так, щоб користувач мав можливість задати завдання чи список завдань, після зайнятись іншими справами, навіть виключити ПК. А через деякий час зайти до веб-додатку, переглянути статус обробки або отримати результат.

На вхід подаються тексти у вигляді файлів. На виході користувач отримує терміни в порядку зменшення їх важливості. Обробка проводиться серверною частиною.

Через пряму залежність часу виконання від розміру тексту, тексти великого обсягу розділяються на фрагменти. Обробка проводиться для кожного фрагменту окремо, а отримані результати об'єднується в загальний результат, який і отримує користувач (рис.2.1.).

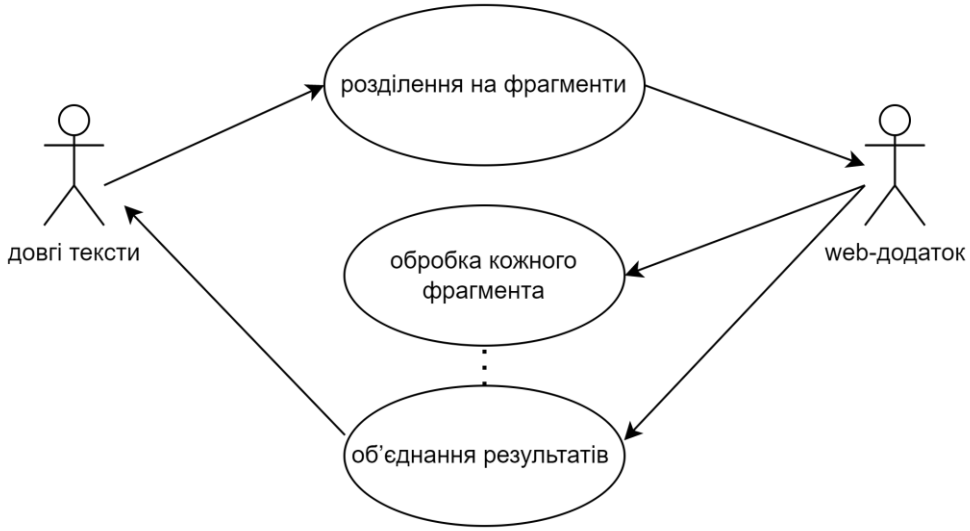

Рисунок 2.1 Процес обробки довгих текстів

Розглянемо сценарії використання веб-додатка, почнемо з клієнтської частини. Так як отримання термінів є доволі довгим процесом, який виконується серверною стороною, користувач повинен мати можливість задати завдання чи список завдань, а через деякий час зайти до веб-додатку, переглянути статус обробки або отримати результат.

Така задача потребує ідентифікації користувачів, для цього йому потрібно виконати такі дії:

- зареєструватися;
- війти в систему;
- по електронній пошті отримати новий пароль;
- за бажанням змінити пароль.

Діаграма варіантів використання дій користувача при ідентифікації представлена на рис.2.2.

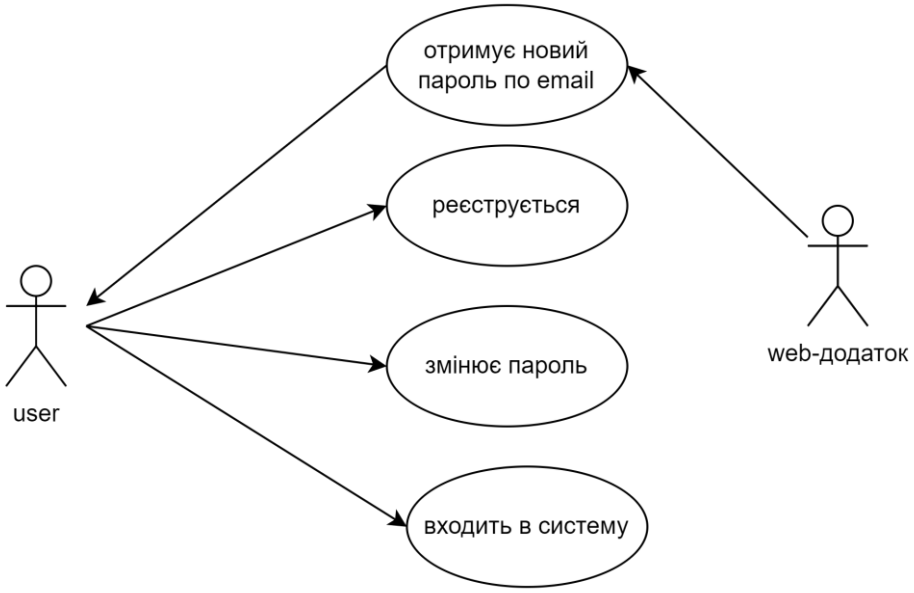

Рисунок 2.2 Діаграма варіантів використання дій користувача при ідентифікації.

Для коректного керування системою потрібен адміністратор, перелічимо його задачі при роботі зі списком користувачів:

– адміністратор активує / деактивує користувача;

– адміністратор бачить список користувачів;

– адміністратор змінює пароль користувача.

Діаграма варіантів використання дій адміністратора при роботі зі списком користувачів представлена на рис.2.3.

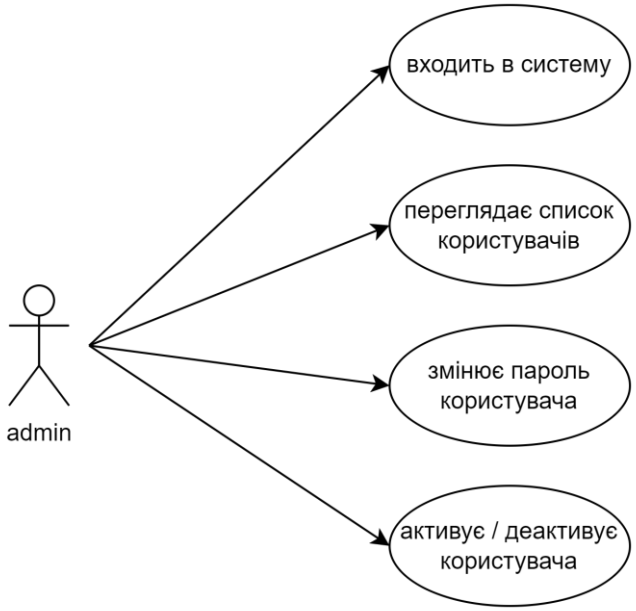

Рисунок 2.3 Діаграма варіантів використання дій адміністратора при роботі зі списком користувачів.

Після входу в систему користувач повинен мати можливість задати завдання чи список завдань, переглянути його, побачити, які з завдань виконані. Для кожного завдання користувач створює конвеєр - упорядкований список операцій, які потрібно виконати. Інакше кажучи задає впорядковану сукупність кроків, які обираються зі списку. Серверна частина виконує кроки з черги, логі та результати виконання зберігаються у файлах.

Існують шаблони конвеєрів, за замовчуванням обирається шаблон лише на виявлення термінів, але можна додавати й інші. Кожен користувач може запускати лише один конвеєр. Тобто він виконує такі дії:

– створює конвеєр;

– подає вхідні дані (це файли і метадані до них, рідше - імена файлів в потрібному порядку);

– запускає конвеєр;

– бачить список своїх конвеєрів;

– видаляє конвеєр (всі пов'язані з ним дані видаляються). Конвеєр можна видалити, якщо він не запущений, тобто не містить кроку, який потрібно виконати.

Діаграма варіантів використання роботи користувача з конвеєрами представлена на рис.2.4.

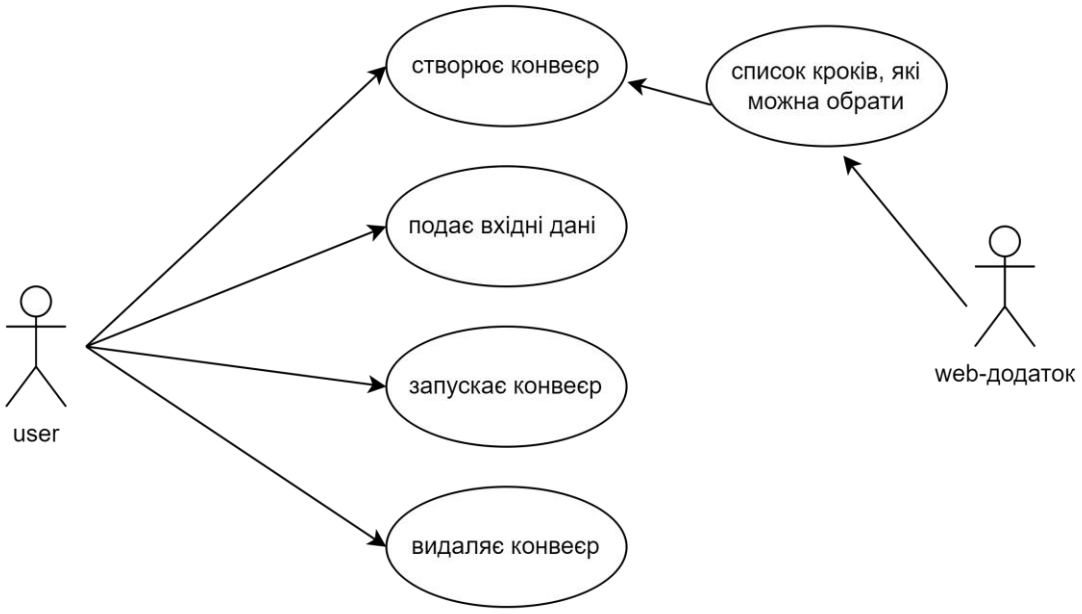

Рисунок 2.4 Діаграма варіантів використання роботи користувача з конвеєром

Після того як буде створено хоча б один конвеєр користувач зможе:

– бачити кроки будь-якого конвеєра та їх статус ("готово", "в черзі", "в процесі", "чекає");

– додати крок до черги, така можливість з’явиться, якщо в черзі для цього користувача немає інших кроків;

– видалити крок з черги;

– обачити статус кожного кроку;

– в будь-який момент переглянути логи будь-якого кроку (якщо вони існують).

Діаграма варіантів використання роботи користувача з кроками, заданими у конвеєрі, представлена на рис.2.5.

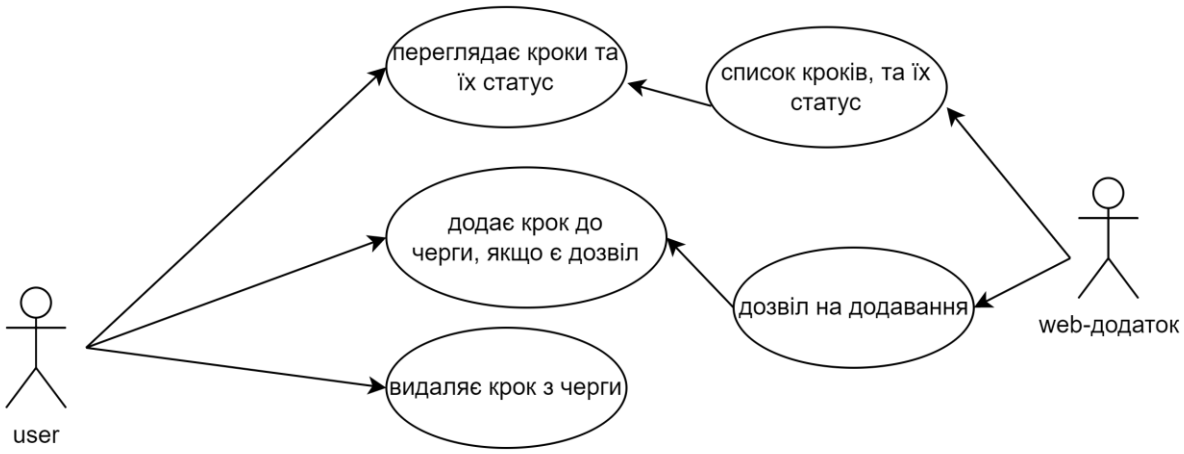

Рисунок 2.5 Діаграма варіантів використання роботи користувача з кроками

Для кожного кроку користувач може виконувати дії, представлені рис.2.6.

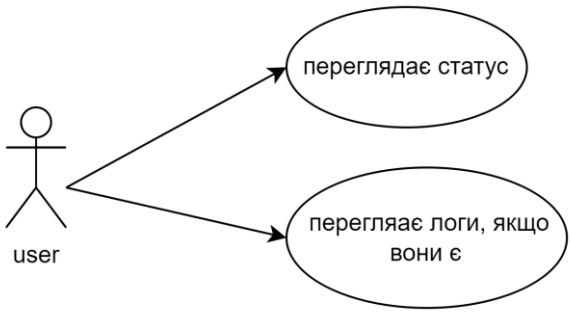

Рисунок 2.6 Діаграма варіантів використання роботи з кроком

Існує спеціальний крок "ручне редагування", для нього відкривається редактор вхідного файлу, і результат зберігається у вихідний файл.

Після завершення будь-якого кроку система надсилає повідомлення користувачеві та позначає крок як завершений. А користувач отримує на пошту повідомлення з посиланням для перегляду результату і логів.

Якщо є можливість продовжити автоматично, запускається наступний крок. Для цього має бути задана така властивість кроку, як «можливість продовжувати автоматично».

Адміністратор при роботі з конвеєрами може:

– бачити список конвеєрів для кожного користувача, є фільтр, який позволяє побачити запущені конвеєри;

– бачити список запущених конвеєрів;

– бачити список запущених завдань.

Відповідна діаграма представлена на рис.2.7.

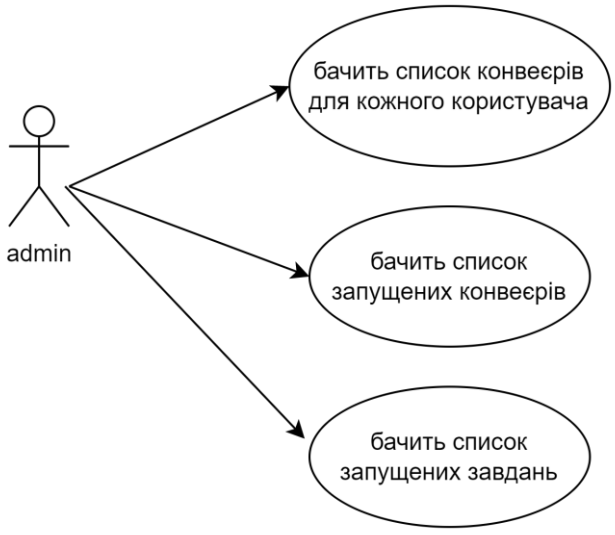

Рисунок 2.7 Діаграма варіантів використання при роботі адміністратора з конвеєрами.

2.4 Структура даних

Розглянемо структуру даних, потрібну для реалізації web-додатку. Спираючись на розглянуті варіанти використання, виділимо такі сутності: Users (Користувач), Admin (Адміністратор), Task (Задача), Stage (Етап), Stage_type (Кроки обробки) (Таблиці 2.1-2.6.).

Таблиця 2.1. - Перелік таблиць

Таблиця	Опис
Users	містить інформацію про користувачів веб-сайту
Admin	містить інформацію про адмінів веб-сайту
Task	містить інформацію про задачі користувача
Stage	містить інформацію про завантажений текст та кроки його обробки
Stage_type	містить інформацію про кроки обробки тексту

Таблиця 2.2 - Таблиця Users

Назва поля	Опис	Тип даних
user_id	id-номер користувача, первинний ключ	Ціле
user_password	пароль	Текст
user_email	адреса електронної пошти	Текст
user_login	логін	Текст
user_name	ім’я користувача	Текст

Таблиця 2.3 - Таблиця Admin

Назва поля	Опис	Тип даних
admin_id	id-номер адміна, первинний ключ	Ціле
admin_user	id-номер користувача, зовнішній ключ	Ціле

Таблиця 2.4 - Таблиця Task

Назва поля	Опис	Тип даних
task_id	id-номер задачі, первинний ключ	Ціле
task_user	id-номер користувача, що поставив задачу, зовнішній ключ	Ціле
task_started_at	Дата, час, коли була поставлена задача	Дата, час
task_finished_at	Дата, час, коли була виконана задача	Дата, час

Таблиця 2.5 - Таблиця Stage_type

Назва поля	Опис	Тип даних
stage_type_id	id-номер кроків обробки, первинний ключ	Ціле
stage_type_ordering	порядок кроку	Ціле
stage_type_options	перелік додаткових параметрів	JSON

Таблиця 2.6 - Таблиця Stage

Назва поля	Опис	Тип даних
stage _id	id-номер етапа, первинний ключ	Ціле
stage_type_id	id-номер кроку, зовнішній ключ	Ціле
stage_task_id	id-номер задачі, зовнішній ключ	Ціле
stage_input	посилання на місце, де зберігається текст	Текст
stage_res	посилання на місце, де зберігається результат	Текст
stage_log	посилання на місце, де зберігаються логи (логи)	Текст
stage_link_options	перелік додаткових параметрів	JSON
stage_status	статус виконання обробки, обирається з переліка: "готово", "в черзі", "в процесі", "чекає"	Текст
stage_started_at	Дата, час, коли обробка почала виконуватись	Дата, час
stage_finished_at	Дата, час, коли обробка була виконана	Дата, час

Зв’язки між сутностями показані на рис.2.8

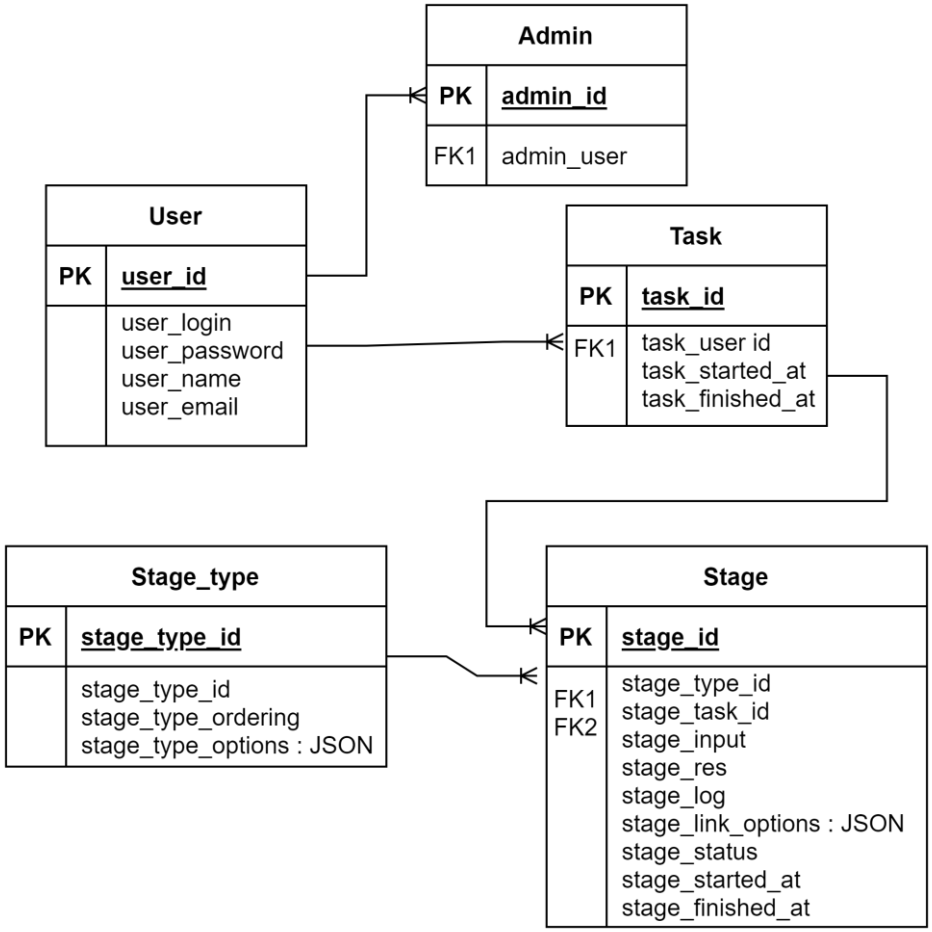

Рисунок 2.8 ER-Діаграма.

2.5 Висновки

В цьому розділі була проведена постановка задачі, указана мета роботи.

Метою роботи є проектування WEB-інтерфейсу для програмного пакета автоматичного виділення термінів. Для реалізації було обрано розробку веб-додатку, в такому випадку за клієнтську частину буде відповідати браузер, а за серверну – веб-сервер. Складність роботи полягає в тому, що отримання термінів є тривалим процесом і має виконуватися асинхронно.

Вхідними даними є тексти у вигляді файлів. На виході користувач отримує терміни в порядку зменшення їх важливості.

Проведено опис методу виявлення термінів. Розглянуто варіанти та сценарії використання, приведено діаграми варіантів.

Спираючись на розглянуті варіанти використання, розроблено структуру даних, потрібну для реалізації web-додатку, виділено сутності: Users (Користувач), Admin (Адміністратор), Task (Задача), Stage (Етап), Stage_type (Кроки обробки).

3 СПОСОБИ РЕАЛІЗАЦІЇ

3.1 Опис програмного пакета який реалізує метод виявлення термінів

Проведемо опис програмного пакету, який реалізує метод виявлення термінів, потрібний для реалізації web-додатку. Призначення файлів програмного пакета [42] указано в таблиці 3.1. В кожному файлі реалізовано необхідні для обробки кроки.

Таблиця 3.1. - Призначення файлів програмного пакета

Файл	Призначення
000_download.sh	завантаження до 20000 базових публікацій, може знадобитися кілька годин
001_tokenizer.sh	виконує токенізацію за допомогою інструментів NLTK
002_rarewords.sh	виявляє рідкісні слова
003_joint_probabilities.sh	оцінює ймовірності збігу токенів
004_stopwords.sh	виявляє стоп-слова
005_reduced_joint_probabilities.sh	оцінює ймовірності збігу токенів після виключення рідкісних слів та стоп-слів
006_SSNMF.sh	створює топ-модель, прблизно за 1-2 години
007_restricted_snowball.sh	завантаження до 20000 релевантних публікацій, може знадобитися кілька годин
008_search_path_count.sh	робить підрахунок кількості шляхів пошуку [43]
009_extend_items_google_scholar.sh	скрипт завантажує кілька сотень публікацій з Google Scholar, тому щоб уникнути бана необхідно використовувати проксі. Адреса проксі є параметром проксі у конфігураційному файлі. ./data/YOUR_DATA_DIRECTORY/config.ini

009_extend_items_google_scholar_resume.sh	іноді потрібно відновити попередню команду
010_download_pdfs.sh	завантажує PDF-файли, які доступні безкоштовно
011_export_xlsx.sh	формує підсумковий звіт з ./docs/data-requirements.txt та файл 011_exported.xlsx - остаточний список публікацій
Можна розширити конвеєр за допомогою кроку ATE (Automatic Term Extraction)	
012_ate_pdf2txt.sh	вилучення простого тексту з PDF-файлів
013_ate_clear_txt.sh	очистити вилучені тексти
014_ate_generate_datasets.sh	об'єднувати вилучені тексти в послідовність наборів даних
015_ate_get_terms.sh	витягувати терміни
016_ate_clear_terms.sh	видалити терміни для сміття (список термінів для сміття є у файлі ./data/YOUR_DATA_DIRECTORY/ate_stopwords.csv)
017_ate_saturation.sh	робить аналіз термінологічної насиченості

За бажанням конвеєр можна розширити кроком ATE (Automatic Term Extraction). Діаграма компонентів пакету ATE представлена на рис.3.1 [**Ошибка! Закладка не определена.**].

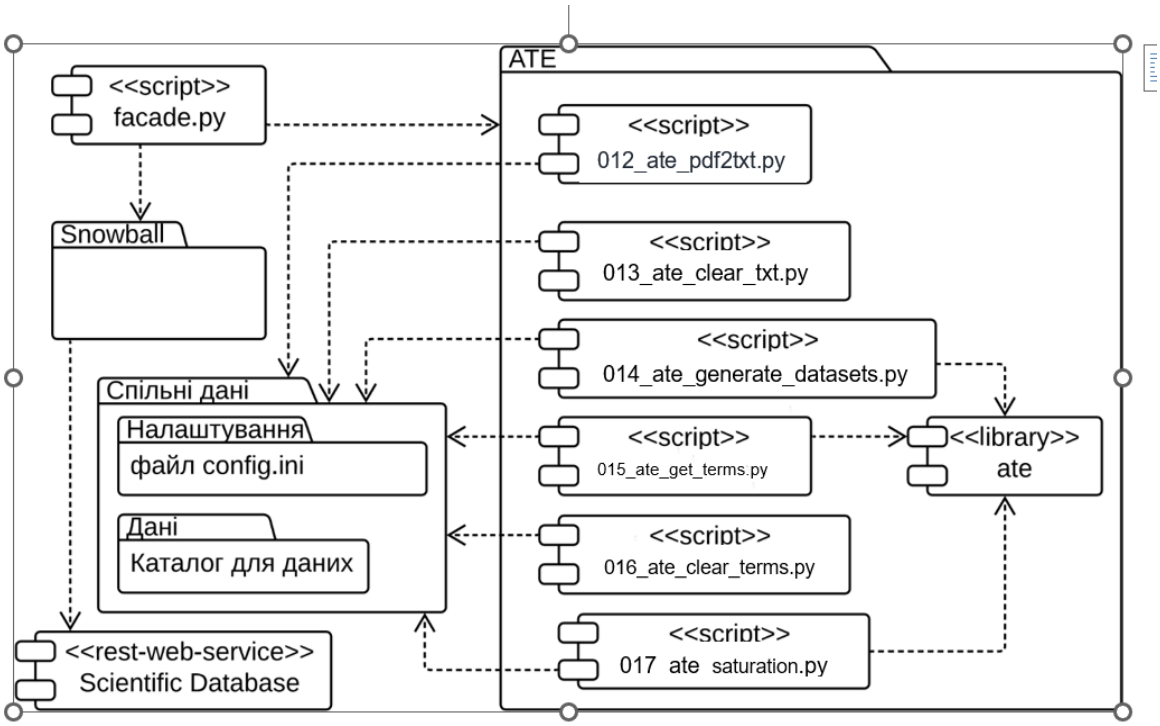

Рисунок 3.1 Діаграма компонентів пакета ATE.

Розглянемо скрипти пакета ATE більш детально.

У 012_ate_pdf2txt.sh виконується виклик функції вилучення простого тексту з PDF-файлів, яка запрограмована в 012_ate_pdf2txt.py. Вхідними параметрами є:

– дані про файл конфігурації (--config);
– місце, куди будуть записуватись txt файли(--txtdir);
– місце, де знаходяться pdf файли (--pdfdir).

У 013_ate_clear_txt.sh виконується виклик функції, яка проводить очистку вилучених текстів, та запрограмована в 013_ate_clear_txt.py. Вхідними параметрами є:

– дані про файл конфігурації (--config);
– місце, де знаходяться необроблені txt файли(--rawtxtdir);
– місце, куди будуть записуватись очищені файли (--cleartxtdir).

У 014_ate_generate_datasets.sh виконується виклик функції, яка об'єднує вилучені тексти в послідовність наборів даних, та запрограмована в 014_ate_generate_datasets.py. Вхідними параметрами є:

– дані про файл конфігурації (--config);

– місце, де будуть знаходитись набори даних (--datasetdir);

– місце, де знаходяться очищені файли (--cleartxtdir);

– потрібно при обробці стратегії (--increment_size=20);

– місце, де знаходиться 011_exported_edited.xlsx - файл з остаточним списком публікацій (--metadatafile);

– стратегія обробки, обирається зі списку(--strategy="citation-per-year-desc").

Список параметру strategy: citation-desc|citation-per-year-desc|spc-desc|time-desc|time-asc|time-bidir|random |partial-citation-desc|partial-citation-per-year-desc|partial-spc-desc|partial-time-desc|partial-time-asc|partial-time-bidir|partial-random

Якщо буде потреба в генерації часткових наборів даних у 014_ate_generate_partial_datasets.sh виконується виклик функції, яка запрограмована в 014_ate_generate_datasets.py з такими параметрами:

– дані про файл конфігурації (--config);

– місце, де будуть знаходитись часткові набори даних (--datasetdir);

– місце, де знаходяться очищені файли (--cleartxtdir);

– потрібно при обробці стратегії (--increment_size=20);

– місце, де знаходиться дані з файлу 011_exported_edited.xlsx у форматі JSON (--metadatafile);

– стратегія обробки, обирається зі списку(--strategy=" partial-spc-desc ").

У 015_ate_get_terms.sh виконується виклик функції, яка витягує терміни, та запрограмована в 015_ate_get_terms.py. Вхідними параметрами є:

– дані про файл конфігурації (--config);

– місце, де знаходяться набори даних, вхідний файл TXT (--in_dir_dataset);

– місце, де будуть знаходитись отримані набори термінів, вихідний файл CSV, що містить терміни (--out_dir_terms);

– місце, де знаходяться файл ate_stopwords.csv зі стоп-словами, одне слово на рядок (--stopwords);

– показує детальну інформацію про виконання (--trace).

Якщо буде потреба в витягуванні термінів з часткових наборів даних у 015_ate_get_terms_partial.sh викликається функція, яка запрограмована в 015_ate_get_terms.py з такими параметрами:

– дані про файл конфігурації (--config);

– місце, де знаходяться часткові набори даних, вхідний файл TXT (--in_dir_dataset);

– місце, де будуть знаходитись отримані набори термінів, вихідний файл CSV, що містить терміни (--out_dir_terms);

– місце, де знаходяться файл ate_stopwords.csv зі стоп-словами, одне слово на рядок (--stopwords);

– показує детальну інформацію про виконання (--trace).

Отримані, після витягування термінів з часткових наборів даних, набори необхідно об’єднати. У 015_ate_merge_terms_partial.sh викликається функція, яка відповідає за злиття та запрограмована в 015_ate_merge_terms_partial.py з такими параметрами:

– дані про файл конфігурації (--config);

– місце, де знаходяться часткові набори даних, вхідний файл TXT (--in_dir_dataset);

– місце, де будуть знаходитись вихідний файл CSV з термінами (--out_dir_terms);

– місце, де знаходяться файл ate_stopwords.csv зі стоп-словами, одне слово на рядок (--stopwords);

– показує детальну інформацію про виконання (--trace).

У 016_ate_clear_terms.sh виконується виклик функції, яка видаляє мусорні терміни, тобто очищує список термінів. Перелік таких термінів є у файлі ate_stopwords.csv, функція запрограмована в 016_ate_clear_terms.py. Вхідними параметрами є:

– дані про файл конфігурації (--config);

– місце, де знаходяться набори термінів, вхідний файл TXT (--in_terms);

– місце, де будуть знаходитись очищені набори термінів, вихідний файл CSV, що містить терміни (--out_terms);

– місце, де знаходяться файл ate_stopwords.csv зі стоп-словами, одне слово на рядок (--stopwords);

– показує детальну інформацію про виконання (--trace).

У 017_ate_saturation.sh виконується виклик функції, яка робить аналіз термінологічної насиченості, функція запрограмована в 017_ate_saturation.py. Вхідними параметрами є:

– дані про файл конфігурації (--config);

– місце, де знаходяться набори термінів, вхідний файл TXT (--in_terms);

– місце, де будуть знаходитись очищені набори термінів, вихідний файл CSV, що містить терміни (--out_terms).

Розглянутий пакет ATE дозволяє проводити виявлення термінів з часткових наборів даних з їх подальшим об’єднанням в єдиний результат.

3.2 Програмне оточення та бібліотеки

У якості мови програмування було обрано мову Python. Через її легкість, популярність та наявність необхідних для розробки бібліотек.

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією [44].

До переваг мови Python відносят:

– логічний синтаксисом, який легко читається та сприймається;

– умовну легкість, підходить для фахівців-початківців;

– наявність великої інтернет-спільноти;

– гнучкість та масштабованість.

Розробка «на python» йде швидше, ніж на більшості інших мов. Python є інтерпретованою мовою програмування. Це означає, що до запуску він є звичайним текстовим файлом. Відповідно програмувати можна майже на всіх платформах.

Завдяки популярності, python містить багато бібліотек, які полегшують процес розробки. Для створення програм з обробкою природної мови використовується пакет Python під назвою NLTK (Natural Language Toolkit Package) [45].

Для створення лаконічних та досить швидких HTTP API-серверів використовувався фреймворк FastAPI [46], який містить вбудовану валідацію, серіалізацію та асинхронність. У FastAPI роботою з web займається Starlette, а за валідацію відповідає Pydantic.

Starlette [47] - новий, швидкий фреймворк, що реалізує підхід ASGI, який базується на асинхронності.

Pydantic [48] забезпечує підказки типу під час виконання та надає зручні для користувача помилки, коли дані некоректні. Задати типи даних можна за допомогою чистого, канонічного python, потім зробити валідацію за допомогою pydantic.

Для роботи з базами даних було обрано SQLAlchemy.

SQLAlchemy [49] - це бібліотека на мові Python для роботи з реляційними СУБД із застосуванням технології ORM. Для синхронізації об'єктів Python та записів реляційної бази даних. Надає можливість описувати структури баз даних та способи взаємодії з ними мовою Python без використання SQL, підтримує транзакції та створення запитів з використанням функцій та виразів Python.

У якості системи керування базами даних було обрано MySQL.

MySQL [50] - одна з найпоширеніших вільних систем керування базами даних. Використовується для створення динамічних вебсторінок, оскільки підтримується різними мовами програмування.

MySQL - компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

Можливості сервера MySQL:

– простота у встановленні та використанні;

– підтримується необмежена кількість користувачів, що одночасно працюють із БД;

– кількість рядків у таблицях може досягати 50 млн;

– висока швидкість виконання команд;

– наявність простої і ефективної системи безпеки.

Для налагодження та тестування розробленого веб-додатку було обрано Postman .

Postman [51] - програма для роботи з API. Це популярний API клієнт, який дозволяє розробляти, тестувати та документувати API.

За допомогою Postman можна надсилати HTTP/s запити до сервісів і отримувати від них відповіді. За допомогою такого підходу можна протестувати бекенд сервіси та переконатися, що вони коректно працюють.

3.3 Протокол роботи web-додатку

Переглянемо перелік дій користувача після входу в систему: користувач задає завдання чи список завдань, переглядає його, та може побачити, які з завдань виконані.

Для кожного завдання користувач створює упорядкований список операцій, які потрібно виконати - конвеєр. Тобто задає впорядковану сукупність кроків, які може обрати зі списку. Серверна частина виконує кроки з черги, результати виконання та логи зберігає у файли.

Кожен користувач може запускати лише один конвеєр, при цьому він виконує такі дії:

– створює конвеєр;

– подає вхідні дані;

– запускає конвеєр;

– переглядає список своїх конвеєрів та їх статус;

– видаляє конвеєр, якій має статус «виконано», при цьому всі пов'язані з ним дані видаляються.

Більш наглядно робота з конвеєром показана на діаграмі послідовності (Sequence diagram) [52] (рис. 3.2.)

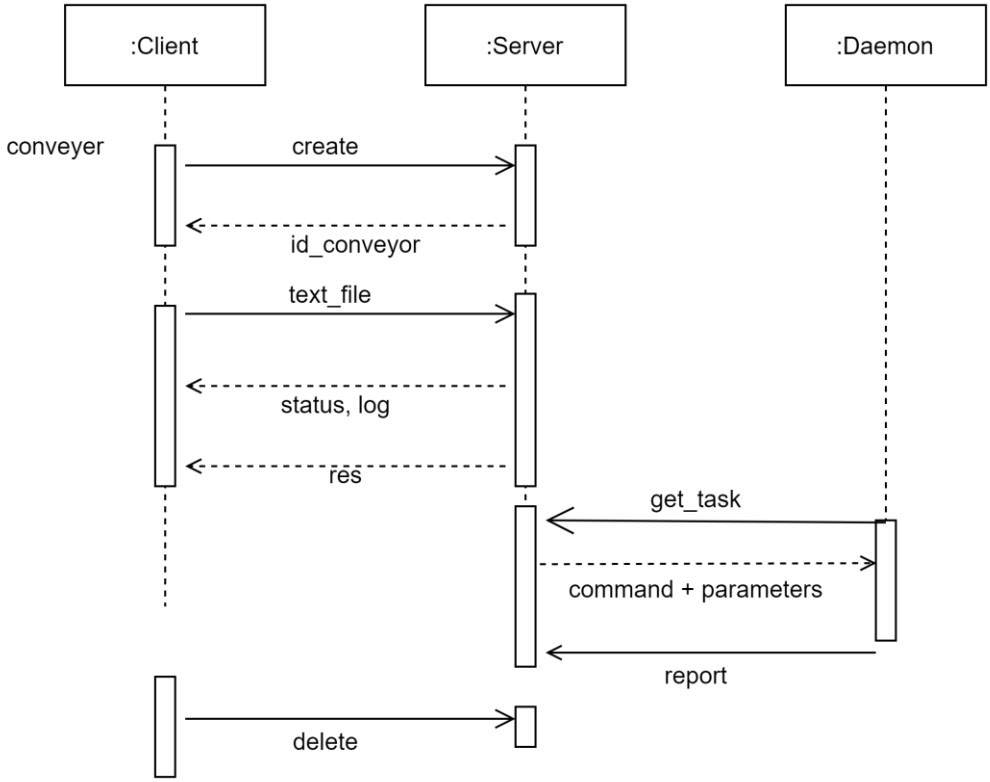

Рисунок 3.2 Діаграма послідовності роботи з конвеєром.

Комунікація між web-сервером і браузером здійснюється за допомогою протоколу. Розглянемо деякі протоколи роботи web-додатку більш детально.

Створення конвеєру:

Post/conveyor/create(id_user, name_conveyor, secret_key)

Серверна частина створює запис у БД, (Таблиці Task, Stage), копіює з таблиці Stage_type в таблицю Stage тип кроку, який потрібно виконати. Та видає ідентифікатор конвеєра id_conveyor.

Після цього користувач може загрузити вхідні дані - створюється директорія із зразка, куди завантажується файл, який закодовано в base64 encoded file:

Post/conveyor/data(id_conveyor, secret_key, text_file)

Після цього в БД статус змінюється на «чекає»:

Post/conveyor/start(id_conveyor, secret_key)

Окремо виконується процес, який, з періодичністю десь 1 хв, перевіряє таблицю Stage, знаходить за статусом задачі, які можна стартувати, виконує, після фінішу ставить статус виконано, а наступному завданню по конвеєру ставить «можна продовжувати».

Користувач може переглянути в список кроків та їх статус

Post/conveyor/status(id_conveyor, secret_key)

На виході повертається статус конвеєра, і, можливо, текстове повідомлення.

Користувач може переглянути список своїх конвеєрів та їх статус:

Post/conveyor/logs(id_conveyor, secret_key, stage _id)

На виході повертається журнал логів.

Після виконання всіх кроків конвеєра користувач отримує файл з результатами:

Post/conveyor/results(id_conveyor, secret_key)

На виході повертається файл с результатами.

Якщо конвеєр має статус «виконано», його можна видалити, при цьому також видаляються всі пов'язані з ним дані на серверы:

Post/conveyor/delete(id_conveyor, secret_key)

3.4 Висновки

Проведено опис програмного пакету, який реалізує метод виявлення термінів, потрібний для реалізації web-додатку. Призначення файлів програмного пакета указано в таблиці 3.1. В кожному файлі реалізовано необхідні для обробки кроки.

Більш детально розглянуто частину програмного пакета, пакет ATE, який дозволяє проводити виявлення термінів з часткових наборів даних з їх

подальшим об’єднанням в єдиний результат. На рис.3.1. представлена діаграма компонентів пакета ATE.

Розглянуті обрані для реалізації програмне оточення та бібліотеки. У якості мови програмування було обрано мову Python. Для обробки природної мови - пакет NLTK. Для створення лаконічних та досить швидких HTTP API-серверів - фреймворк FastAPI. У FastAPI для роботи з web використовується Starlette, а за валідацію відповідає Pydantic. У якості системи керування базами даних було обрано MySQL. Для налагодження та тестування розробленого веб-додатку було обрано Postman.

Було приведено перелік дій користувача після входу в систему. Користувач може задавати завдання чи список завдань, переглядати його, та бачити, які з завдань виконані. Для кожного завдання створюється конвеєр, який містить упорядкований список операцій (береться зі списку), які потрібно виконати. Серверна частина виконує кроки зі списку, а результати виконання та логи зберігає у файли. Діаграма послідовності роботи з конвеєром приведена на Рис. 3.2.

Також більш детально було розглянуто деякі протоколи, які здійснюють комунікацію між браузером і web-сервером.

ВИСНОВКИ

У роботі були розглянуті методи автоматичного виявлення термінів. Більшість з яких, для покращення результатів, спочатку проводять попередню обробку тексту та відкидають слова, що не несуть додаткового сенсу. Потім, на очищеному тексті, застосовують спочатку лінгвістичну, а потім і статистичну обробки тексту.

У лінгвістичних методах найчастіше використовуються n-грами, токенізація, стемінг, лематизація та POS tagging. Складний варіант лінгвістичної обробки включає синтаксичний аналіз речень та виділення словосполучень, які можуть виявитися термінами. Наприклад, виділяються шаблони словосполучень. У разі статичного аналізу найчастіше оцінюється частота зустрічаємості терміну та її аналоги.

Було приведено оцінки ефективності найпоширеніших алгоритмів АВТ у Таблиці 1.1.

Аналізувались різні інструменти АВТ: BIOTEX, TerMine, FiveFilters, TaaS, Terminology Extraction, TBXTools, UPM Term Extractor.

Проведений розгляд показав, що більшість з розглянутих інструментів мають обмеження за розміром тексту та, в основному, орієнтовані на англійську мову. До того ж всі вони працюють лише в режимі реально часу та не надають можливість задавати декілька текстів одночасно. Крім того більшість з них не має легкого у застосуванні інтерфейсу.

Також, для перевірки методів виявлення термінів, було розглянуто набори даних: ACTER, GENIA, ACL RD-TEC, CRAFT corpus.

Було приведено принципи побудови веб-додатків та указані переваги асинхронного підходу. Аби збільшити інтерактивність та продуктивність, веб-інтерфейс треба організувати таким чином, щоб була можливість задати список завдань і через деякий час отримати результат.

В другому розділі була проведена постановка задачі та указана мета роботи. Метою роботи є проектування web-інтерфейсу для програмного пакета автоматичного виділення термінів. Для реалізації було обрано розробку веб-додатку, в якому за клієнтську частину буде відповідати браузер, а за серверну – веб-сервер. Складність роботи полягає в тому, що отримання термінів є тривалим процесом і має виконуватися асинхронно.

В якості вхідних даних задаються тексти у вигляді файлів. А вихідними є терміни в порядку зменшення їх важливості. Здійснено опис методу виявлення термінів. Розглянуто варіанти та сценарії використання, приведено діаграми варіантів. Зважаючи на поставлену задачу, розроблено структуру даних, потрібну для реалізації web-додатку.

В третьому розділі приведено опис програмного пакету, який реалізує метод виявлення термінів, потрібний для реалізації web-додатку (таблиця 3.1.) В кожному файлі реалізовано необхідні для обробки кроки.

Більш детально розглянуто призначення файлів програмного пакета ATE(Automatic Term Extraction), який дозволяє проводити виявлення термінів з часткових наборів даних з їх подальшим об’єднанням в єдиний результат. На рис.3.1. представлена діаграма компонентів пакета ATE.

Розглянуті обрані для реалізації програмне оточення та бібліотеки. Мовою програмування було обрано мову Python. Для обробки природної мови - пакет NLTK. Для створення лаконічних та досить швидких HTTP API-серверів - фреймворк FastAPI. У якості системи керування базами даних - MySQL. Для проведення налагодження та тестування - Postman.

Було приведено дії користувача після входу в систему: користувач може задавати список завдань, для кожного з яких створюється конвеєр, що містить упорядкований перелік операцій (береться зі списку), які потрібно виконати. Сервер виконує кроки з переліку, а результати виконання та логи поміщає у відповідні файли. Діаграма послідовності роботи з конвеєром приведена на Рис. 3.2. Було детально розглянуто деякі протоколи, які

здійснюють комунікацію між браузером і web-сервером у частині, яка відповідає за роботу з конвеєрами..

Таким чином в роботі було спроектовано web-інтерфейс для програмного пакета автоматичного виділення термінів.

ПЕРЕЛІК ДЖЕРЕЛ

1. Добровольський Г. А. : Модель, метод та інформаційна технологія відбору наукових публікацій у процесі підготовки бібліографічного покажчика : дис. доктора філософії : 05.13.06 / Геннадій Анатолійович Добровольський. – Харків, 2021. – 216 с.

2. Kosa V. Cross-evaluation of automated term extraction tools by measuring terminological saturation / V. Kosa, D. Chaves-Fraga, D. Naumenko, E. Yuschenko, C. Badenes-Olmedo, V. Ermolayev, A. Birukou, // Revised selected papers of ICTERI 2017. Cham, Germany: Springer-Verlag, CCIS.– 2018.– vol. 826, pp. 135–163.

3. Zhang Z. A comparative evaluation of term recognition algorithms / Z., Zhang, J., Iria, C., Brewster, F. Ciravegna // In: 6th Int Conf on Language Resources and Evaluation.– 2008.– pp. 2108–2113.

4. Maynard D. Natural Language Processing for the Semantic Web / D., Maynard, K., Bontcheva, I.Augenstein // Morgan & Claypool.– 2017.– doi: 10.2200/S00741ED1V01Y201611WBE015

5.Manning C. D. Foundations of Statistical Natural Language Processing / C. D., Manning, H. Schutze // Bradford Book & MIT Press, Cambridge, MA, London, U.K.– 2000.– 620p.

6 Коса В. В. : Метод експериментального дослідження термінологічного насичення в колекціях документів для здобуття знань: дис. доктора філософії : 122 - Комп’ютерні науки / Вікторія Вікторівна Коса. – Запоріжжя, 2021. – 244 с.

7 Тодоріко О. О. Словниковий пошук за схожістю за допомогою хешів на основі сигнатур / О. О. Тодоріко, Г. А. Добровольський // Вісник Херсонського національного технічного університету.– 2010.– №. 3(39). – с. 467 – 471.

8 Manning C. D. Introduction to Information Retrieval / C. D. Manning, Prabhakar Raghavan, Hinrich Schütze // Cambridge University Press, 2008.– 496 p.

9. Korkontzelos I. Reviewing and evaluating automatic term recognition techniques / I., Korkontzelos, I. P., Klapaftis, S. Manandhar // In: Nordström B., Ranta A. (eds.) Advances in Natural Language Processing. GoTAL 2008. LNCS. Springer, Berlin, Heidelberg .– 2008.– vol. 5221.– pp. 248–259.

10 Medelyan, O., Thesaurus based automatic keyphrase indexing / O. Medelyan, I. H. Witten, // In: Marchionini, G., Nelson, M. L., Marshall, C. C. (eds.) ACM/IEEE Joint Conf on Digital Libraries .– 2006.– pp. 296–297.

11 Zhang, Z. Jate 2.0: Java automatic term extraction with Apache Solr / Z., Zhang, J.,Gao, F. Ciravegna, // Proceedings of the 10th international conference on language resources and evaluation LREC 2016. Portorož, Slovenia: European Language Resources Association.– 2016.– pp. 2262–2269.

12. Dobrovolskyi H. Principal component analysis in topic modelling of short text document collections / H. Dobrovolskyi, N. Keberle // in CEUR Workshop Proceedings.– 2017.– vol. 1851.– pp. 48–54.

13. Тодорико О. А. Оценка сигнатурных алгоритмов поиска по сходству в словаре / О. А. Тодорико, Г. А. Добровольский // Вісник Херсонського національного технічного університета.– 2011.– №. 2(41).– с. 250 – 254.

14. Тодорико О. А. Оцінка якості автоматичної класифікації коротких текстових документів / О. А. Тодорико, Г. А. Добровольский // Вісник Запорізького національного університету : зб. наук. пр. Фізико-математичні науки.– 2010.– №. 2.– с. 131 – 140.

15. Astrakhantsev N. ATR4S: toolkit with state-of-the-art automatic terms recognition methods in scala. URL: https://arxiv.org/abs/1611.07804 (дата звернення: 30.10.22).

16. Astrakhantsev N.: Methods and software for terminology extraction from domain-specific text collection : PhD thesis. Institute for System Programming of Russian Academy of Sciences. URL: 74f43ef0a8bb2fb9ec78bb04b2200ecf.pdf (дата звернення: 30.10.22).

17. Bordea G. Domain-independent term extraction through domain modelling. URL: https://www.researchgate.net/publication/262567630_Domain-independent_term_extraction_through_domain_modelling (дата звернення: 30.10.22).

18 Badenes-Olmedo C. Efficient clustering from distributions over topics. URL: https://www.researchgate.net/publication/347300759_Efficient_Clustering_from_Distributions_over_Topics (дата звернення: 30.10.22).

19. Dobrovolskyi H. Collecting the seminal scientific abstracts with topic modelling, snowball sampling and citation analysis / H. Dobrovolskyi, N. Keberle, // in CEUR Workshop Proceedings.– 2018.– vol. 2105.– pp. 179–192.

20. Park Y. Automatic glossary extraction: beyond terminology identification. URL: https://aclanthology.org/C02-1142.pdf (дата звернення: 30.10.22).

21. Tatarintseva O. Quantifying ontology fitness in OntoElect using saturation- and vote-based metrics / O. Tatarintseva, V., Ermolayev, B., Keller, W.-E. Matzke, // In: Ermolayev, V., et al. (eds.) Revised Selected Papers of ICTERI 2013. CCIS, . Springer, Heidelberg.– 2013.– vol. 412.– pp.136–162.

22. Nokel M. An experimental study of term extraction for real information-retrieval thesauri. / M., Nokel, N. Loukachevitch, // In: 10th Int Conf on Terminology and Artificial Intelligence.– 2013.– pp.69–76.

23. Lossio-Ventura Juan Antonio BIOTEX: A system for Biomedical Terminology Extraction, Ranking, and Validation ISWC / Juan Antonio Lossio-Ventura, Clement Jonquet, Mathieu Roche, Maguelonne Teisseire // International Semantic Web Conference, Oct 2014, Riva del Garda, Italy.– 2014.– pp.157-160.

24. TerMine Web Service URL: http://www.nactem.ac.uk/software/ termine/webservice/ (дата звернення: 30.10.22).

25. TerMine URL:http://www.nactem.ac.uk/software/termine/ (дата звернення: 30.10.22).

26. Term Extraction - FiveFilters.org URL: https://www.fivefilters.org/term-extraction/ (дата звернення: 30.10.22).

27. Using TaaS for Terminology Extraction URL: https://blog.memoq.com/using-taas-for-terminology-extraction (дата звернення: 30.10.22).

28. Terminology Extraction service - API and Demo (translatedlabs.com) URL: https://translatedlabs.com/terminology-extraction (дата звернення: 30.10.22).

29. TBXTools. A Python class for Terminology Extraction and Management URL: https://sourceforge.net/projects/tbxtools/ (дата звернення: 30.10.22).

30. Corcho O. Indexing Your ROS Repository for Documentation Generation http://wiki.ros.org/rosdistro/Tutorials/Indexing%20Your%20ROS%20 Repository%20for%20Documentation%20Generation (дата звернення: 30.10.22).

31. Epnoi URL: https://github.com/ontologylearning-oeg/epnoi-legacy (дата звернення: 30.10.22).

32. ACTER Annotated Corpora for Term Extraction Research URL: https://github.com/AylaRT/ACTER (дата звернення: 30.10.22).

33. Kim, J.-D. GE- 94 NIA corpus - a semantically annotated corpus for bio-textmining. / J.-D., Kim, T.,Ohta, Y., Tateisi, J. Tsujii, // Bioinformatics.– 2003.– 19(1).– pp.180–182.

34. Qasemizadeh B. The ACL RD-TEC 2.0: A Language Resource for Evaluating Term Extraction and Entity Recognition Methods. URL: https://aclanthology.org/L16-1294.pdf (дата звернення: 30.10.22).

35. Bada, M. Concept annotation in the CRAFT corpus. / M., Bada, M. Eckert, D., Evans et al. // BMC Bioinformatics.– 2012.– vol.13.– pp.161–180.

36. Quan Nguyen Mastering Concurrency in Python: Create faster programs using concurrency, asynchronous, multithreading, and parallel programming Paperback / Nguyen Quan, Packt Publishing, 2019.– 446 p. URL: http://onreader.mdl.ru/ MasteringConcurrencyInPython/content/index.html (дата звернення: 30.10.22).

37. Hattingh, C. Using Asyncio in Python: Understanding Python's Asynchronous Programming Features / C. Hattingh, O'Reilly Media, Incorporated, 2020. – 152 p. URL: http://onreader.mdl.ru/UsingAsyncio Python3/content/ index.html (дата звернення: 30.10.22).

38. Лексикон загального та порівняльного літературознавства.– Чернівці: Золоті литаври / голова ред. А. Волков.– 2001. – С. 561. – 634 с.

39. Ю. Ковалів, Літературознавча енциклопедія, К.: ВЦ-Академія, т. 1, 2007с. 427–428.

40. Astrakhantsev N. Atr4s: toolkit with state-of-the-art automatic terms recognition methods in scala / N. Astrakhantsev Language Resources and Evaluation.– 2018.– vol. 52, no. 3.– pp. 853–872.

41. Ermolayev V. Ontologies of time: Review and trends. URL: https://www.researchgate.net/publication/271906680_Ontologies_of_Time_Review_and_Trends (дата звернення: 30.10.22).

42. Snowball URL: https://github.com/gendobr/snowball/tree/openalex (дата звернення: 30.10.22).

43. Main path analysis URL: https://en.wikipedia.org/wiki/ Main_path_analysis (дата звернення: 30.10.22).

44. Quan Nguyen Mastering Concurrency in Python: Create faster programs using concurrency, asynchronous, multithreading, and parallel programming

Paperback: Packt Publishing, 2019. 446 p. URL: http://onreader.mdl.ru/MasteringConcurrencyInPython/content/index.html (дата звернення: 30.10.22).

45. Natural Language Processing With Python's NLTK Package – Real Python. URL: https://realpython.com/nltk-nlp-python/ (дата звернення: 30.10.22).

46. FastAPI (tiangolo.com). URL: https://fastapi.tiangolo.com/ (дата звернення: 30.10.22).

47. Starlette URL: https://www.starlette.io/ (дата звернення: 30.10.22).

48. Pydantic (helpmanual.io) URL: https://pydantic-docs.helpmanual.io/ (дата звернення: 30.10.22).

49. SQLAlchemy Documentation — SQLAlchemy 2.0 Documentation URL: https://docs.sqlalchemy.org/en/20/ (дата звернення: 30.10.22).

50. MySQL: MySQL Documentation URL: https://dev.mysql.com/doc/ (дата звернення: 30.10.22).

51. Postman API Platform | Sign Up for Free URL: https://www.postman.com/ (дата звернення: 30.10.22).

52. Explore the UML sequence diagram - IBM Developer URL: https://developer.ibm.com/articles/the-sequence-diagram/ (дата звернення: 30.10.22).