ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка методики обробки запитів користувачів з використанням технологій NLP»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми «Інженерія програмного забезпечення»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Тимур ГОРЯЧЕВ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-61

Тимур ГОРЯЧЕВ

Керівник: Олег ІЛЬЇН
д.т.н., професор

Рецензент:
науковий ступінь, вчене звання
Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Горячеву Тимуру Вікторовичу

1. Тема кваліфікаційної роботи: Розробка методики обробки запитів користувачів з використанням технологій NLP

керівник кваліфікаційної роботи Олег ІЛЬЇН д.т.н., професор,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023 р. №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, матеріали науково-дослідної та переддипломної практики, моделі та методи обробки інформації.

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

1. Аналіз сучасного стану питання обробки запитів користувачів.
2. Проектування системи автоматизованої обробки запитів користувачів .
3. Розробка моделей та структур даних.
4. Проведення моделювання та аналіз результатів.

5. Перелік графічного матеріалу: *презентація*

1. Сучасні рішення обробки текстів природньою мовою.
2. Компоненти системи автоматизованої обробки запитів користувачів.
3. Бізнес процеси обробки запитів користувачів.
4. Блок-схеми категоризації запитів та формування відповідей.
5. Структура бази даних.
6. Результати моделювання.

6. Дата видачі завдання «19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10-05.11.23	
2	Дослідження питання категоризації електронної пошти	06.11-12.11.23	
3	Дослідження програмних розробок та інструментарію NLP	13.11-19.11.23	
4	Дослідження методів та моделей NLP	20.11-26.11.23	
5	Проектування системи автоматизованої обробки запитів користувачів	27.11-03.12.23	
6	Розробка моделей та проведення моделювання	04.12-10.12.23	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.23	
8	Розробка демонстраційних матеріалів	21.12-29.12.23	

Здобувач вищої освіти _____ Тимур ГОРЯЧЕВ
(підпис)

Керівник
кваліфікаційної роботи _____ Олег ІЛЬЇН
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 83 стор., 2 табл., 18 рис., 70 джерел.

Мета роботи – підвищення ефективності процесу автоматизованої обробки текстових запитів користувачів з використання технологій Natural Language Processing (NLP).

Об'єкт дослідження – процес обробки запитів користувачів електронною поштою.

Предмет дослідження – методи та засоби обробки текстових запитів користувачів природньою мовою.

Короткий зміст роботи: у роботі проведено дослідження сучасного стану питання категоризації електронних листів, проаналізовано методи та моделі технології обробки текстів природньою мовою та можливості їх використання для автоматизованої класифікації запитів користувачів електронною поштою. Розглянуто сучасні програмні засоби та інструменти обробки текстів природньою мовою, спроектовано систему автоматизованої обробки запитів користувачів, розроблено базу даних, інтерфейс користувача та розроблено моделі, структури даних, алгоритмічне та програмне забезпечення сервісів автоматизованої системи. Проведено моделювання та аналіз отриманих результатів.

КЛЮЧОВІ СЛОВА: ТЕХНОЛОГІЇ NLP, ЕЛЕКТРОННА ПОШТА, МОДЕЛЬ, БАЗА ДАНИХ, АВТОМАТИЗОВАНА СИСТЕМА.

ABSTRACT

Text part of the master's qualification work: 86 pages, 18 pictures, 2 table, 70 sources.

The purpose of the work – increasing the efficiency of the users text requests automated processing using Natural Language Processing (NLP) technologies.

Object of research – processing of user requests by e-mail

Subject of research – methods and means of users text requests processing in natural language.

Summary of the work: Paper presents conducted study of modern methods and means for processing natural language texts, the analysis showed that there are different approaches, including methods of machine mining, deep mining and classical methods of processing natural texts. Author has proposed methodology for processing user requests in natural language received by email, which includes processing incoming email, categorizing user requests, generating template responses to requests or redirecting them to designated responsible employees. Author has designed a mathematical model of query categorization, which being based on statistical aspects takes into account the weight of certain key phrases, that makes it suitable for a wide range of input texts, algorithms, that allow categorizing the text of user requests using NLP technology using designed model and system architecture for automatic processing of e-mail incoming letters. Selected technologies for designed system include programming language Python, libraries spaCy, NLTK, imaplib, smtplib and transformers language models, SQL Server 2019 and Management Studio 19 environment for database management, ASP.NET web services technology for the user interface and Microsoft Visual Studio 2022 as development environment. Conducted simulations and analysis of the obtained results showed a sufficient level of accuracy of the developed model for categorizing user requests.

KEYWORDS: NLP TECHNOLOGIES, EMAIL, MODEL, DATABASE, AUTOMATED SYSTEM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI	–	Artificial Intelligence
ANN	–	Artificial Neural Network
API	–	Application Programming Interface
ASR	–	Automated Speech Recognition
BERT	–	Bidirectional Encoder Representations from Transformers
BPMN	–	Business Process Model and Notation
GPT	–	Generative Pre-trained Transformer
GRU	–	Gated Recurrent Units
HTML	–	HyperText Markup Language
IMAP	–	Internet Message Access Protocol
KNN	–	K-Nearest Neighbor
LSTM	–	Long short-term memory
MLP	–	Multilayer Perceptron
NBC	–	Naive Bayes Classifier
NER	–	Named entity recognition
NLP	–	Natural Language Processing
RF	–	Random forest
RNN	–	Recurrent Neural Network
SMTP	–	Simple Mail Transfer Protocol
SQL	–	Structured Query Language
SVM	–	Support Vector Machines
XML	–	Extensible Markup Language
БД	–	База даних
ЗВО	–	Заклад вищої освіти
СКБД	–	Система керування базами даних

ВСТУП

Питання обробки електронної пошти гостро стоїть у багатьох організаціях та підприємствах. Приймальні комісії вищих навчальних закладів не є виключенням. Співробітники приймальних комісій змушені обробляти велику кількість звернень, особливо у період прийому документів та проведенню вступних іспитів. Специфіка більшості таких запитів або у наданні шаблонних відповідей (як графік роботи, перелік документів, загальна інформація про спеціальності, тощо), або у необхідності залучення спеціалістів з вирішення певних питань, що можуть надати більш детальну та вузько специфічну інформацію. При цьому робота співробітника прийомної комісії, що до обробки запитів у листах, у переважної більшості випадків є суто механічною: знайти шаблон відповіді та відправити письмо, або переслати його до відповідальної з цього питання особі. На ці дії витрачається велика кількість робочого часу, який можливо було присвятити виконанню робочих задач.

В сучасних соціальних умовах в Україні, таких як пандемія короновірусу та активні військові дії, велика кількість людей не мають постійного доступу до звичайного телефонного або мобільного зв’язку, або інтернету. В таких умовах використання електроної пошти для надсилання запиту з метою отримання певної інформації є оптимальнім засобом комунікації, оскільки не потребує миттєвої відповіді та дозволяє користувачу отримати відповідь на запит у будь-який зручний час. Крім того, обсяг електронних листів до приймальної комісії ЗВО (закладу вищої освіти) критично зростає у період роботи приймальної комісії поприйому документів абітурієнтів.

Текст запитів користувачів, що надходять електроною поштою, формується людьми та звичайною людською мовою. Аналіз природньої мови краще за все виконують, звичайно, самі люди, які є носями мови, знають мову запиту, або за допомогою перекладача. Сучасні технології с певною мірою точності здатні аналізувати тексти природньою мовою, провідними технологіями для цього вважаються технології обробки природньої мови (Natural Language Processing, або

NLP), що дозволяють обробляти тексти, визначати їх суттєві елементи, перекладати, конспектувати, тощо.

Таким чином, актуальною стоїть задача вивчення сучасних підходів NLP, апаратних та програмних засобів, які використовуються для класифікації запитів користувачів та розробки системи автоматизованої обробки запитів користувачів, що надходять електронною поштою.

Об`єкт дослідження – процес обробки запитів користувачів електронною поштою.

Предмет дослідження – методи та засоби обробки текстових запитів користувачів природньою мовою.

Мета роботи – підвищення ефективності процесу автоматизованої обробки текстових запитів користувачів з використання технологій Natural Language Processing (NLP).

Методи дослідження – методи комп'ютерної лінгвістики, методи та технології NLP, методи теорії систем та теорії інформації, методи проектування та розробки програмного забезпечення, технології баз даних, апарат математичної статистики.

Практична значущість результатів полягає в використанні розробленої методики та автоматизованої системи для класифікації запитів електроною поштою з метою підвищення ефективності та точності відповідей на вхідні листи та ефективного використання робочого часу співробітниками приймальної комісії ЗВО.

Для досягнення мети вирішувалися наступні завдання.

1. Аналіз сучасних методів та засобів обробки природньої мови.
2. Проектування архітектури системи автоматизованої обробки вхідних листів електронної скриньки, обрання технологій та засобів реалізації компонентів системи.
3. Розробка математичної моделі категоризації запитів
4. Розробка алгоритмів категоризації тексту запитів користувачів із використанням NLP та формування тексту відповідей.

5. Проведення моделювання роботи розроблених засобів та оцінка ефективності обробки запитів користувачів природньою мовою.

Апробація результатів та публікації Результати роботи обговорювалися на IV Науково-практичній конференції «Проблеми комп'ютерної інженерії» (м.Київ), Науково-практичному форумі «ТАК» (м.Луцьк).

Матеріали роботи надруковані в таких виданнях:

1. Ільїн О.Ю., Горячев Т.В. Система автоматизованої обробки запитів до приймальної комісії ВНЗ // IV Науково-практична конференція «Проблеми комп'ютерної інженерії». Збірник тез. – Київ: ДУТ, 2023,с 175-177.

2. Ільїн О.Ю., Горячев Т.В. Модель категоризації запитів користувачів з використанням технологій NLP // Науково-практичний форум «ТАК»: телекомунікації, автоматика, комп'ютерно-інтегровані технології: зб. доповідей Всеукр. наук.-практ. конф. молодих вчених, 5-6 грудня 2023 р. / ДВНЗ «ДонНТУ; відп. ред. Г.В. Ступак. – Луцьк: ДВНЗ «ДонНТУ», 2023, с.154-156.

1. Ільїн О.Ю., Золотухіна О.А., Горячев Т.В. Система автоматизованої категоризації запитів електронною поштою з використанням технологій NLP // Наукові записки Державного університету телекомунікації. Подано до друку.

Структура роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ОБРОБКИ ЗАПИТІВ КОРИСТУВАЧІВ

1.1. Сучасний стан питання класифікації електронної пошти

1.1.1. Огляд досліджень з класифікації вмісту електронних листів

Управління величезною кількістю електронних листів, отриманих від користувачів, є дуже складною проблемою, яку потрібно вирішувати ефективним способом. У галузі обробки електронної пошти проводилося багато досліджень та випробовувалося багато різних підходів.

Категоризація електронної пошти полягає в розподілі електронних листів на різні категорії відповідно до певних умов, процес категоризації можна визначити як «процес перегляду необроблених електронних листів і прийняття рішення, що з ними робити». Було розглянуто ряд досліджень, присвячених кластеризації та класифікацію вмісту електронної пошти, а також ряд питань, вирішення яких стає можливим за використання класифікації електроних листів, наприклад:

- визначення, чи створено електронний лист автоматично або особисто;
- категоризація електронних листів контактних центрів;
- класифікація електронних листів для автоматизованого обслуговування;
- аналіз соціальних мереж за допомогою електронної пошти;
- виявлення спаму або фітингу, тощо.

Існує також багато методів, які застосовуються для класифікації електронної пошти:

- підходи на основі інтелектуального аналізу,
- контрольовані алгоритми навчання,
- техніка спільного навчання,
- спільне навчання з єдиним природним набором ознак;
- підходи на основі регресії, тощо.

У роботі Mujtaba G. та Shuib L. [1] проведено огляд питання застосування класифікації електронної пошти біля 100 дослідженнях з Web of Science та бази Scopus, що проводилися за п’ятьма аспектами:

- сфери застосування класифікації електронної пошти;
- набори даних, що використовуються в областях додатків;
- набори ознак, що використовуються в кожній області програми;
- алгоритми класифікації електронної пошти;
- показники продуктивності.

Відповідно до результатів роботи, питання категоризації та класифікація електронної пошти використовується у 15 областях застосування, які для спрощення було поділено на п’ять доменів: спам, фішинг, спам та фішинг, категоризація за кількома папками та інші сфери. У якості наборів даних У сфері категоризації писем найчастіше використовуються PU набори (Positive-Unlabeled) - тип набору даних, в якому існують лише позитивні та невизначені приклади та немає негативних.прикладів, та набір даних електронної пошти Enro, який дослідники та фахівці з обробки природньої мови NLP використовують для різноманітних досліджень, таких як вивчення патернів комунікації, розробка алгоритмів обробки природної мови та вивчення патернів обману. Найбільш використовуваними ознаками для категоризації електроних листів є ознаки з заголовку листа, основної частини листа, що вважаються основними, та ознаки на основі термів та фраз, як текстових шаблонів, а також лексичні (слова, інформація чи задача для дії, тощо) та нелексичні особливості (використання жирного шрифту, заглавних букв, характеристики вкладень, контекстна інформація, тощо). Методи класифікації, що використовуються для класифікації листів, також поділяються на 5 категорій: машинне навчання з учителем, машинне навчання без вчителя, напівконтрольоване машинне навчання, навчання на основі контенту та статистичне навчання. До показників продуктивності, що найчастіше використовуються при мультикласової класифікації електроних листів є

- узгодженість міток класів даних із мітками класифікаторів, якщо вони розраховані за сумами потекстових рішень (точність);

- ефективність класифікатора для ідентифікації міток класів, якщо вони розраховані за сумами потекстових рішень (відгук);
- відношення між позитивними мітками класів та тими, які надані класифікатором на основі сум потекстових рішень (f-міра);
- середня покласова ефективність класифікатора(середня точність);
- середня покласова помилка класифікації (коефіцієнт помилки).

Автори Alsmadi I. та Alhami, I. у своїй роботі показують, що найкращим алгоритмом для кластеризації та класифікації електронної пошти є NGram – послідовність із N суміжних слів або літер із певного джерела тексту [2]. Використовуючи текстову колекцію наборів електронних листів, яки підходять до алгоритму NGram, і алгоритм, що найкраще підходить для двомовного тексту, вони провели експеримент на основі електронних листів англійською та арабською мовами. Головною перепоною було те, що сервери електронної пошти або програми можуть містити різні типи попередньо визначених папок, окрім стандартних: поштова скринька, надіслані або кошик та дозволяють користувачам додавати нові папки для певних цілей .

У дослідженні класифікації електронної пошти Katakis, I., Tsoumakas, G. та Vlahavas I. [3] автори стверджують, що машинне навчання та інтелектуальний аналіз даних можуть бути набагато кращими за інші звичайні рішення для автоматизації завдань керування електронною поштою, обговорюють особливість вмісту електронної пошти та яку особливу обробку він вимагає. Вони розробили програму для класифікації електронної пошти на основі кількох методів, таких як опорно-векторне кластерування (Support Vector Machines, SVM) та наївний баєсів класифікатор (Naive Bayes Classifier, NBC).

Aery M. та Chakravarthy S. [4] представляють систему для групування та підсумовування електронних повідомлень, що враховує тему та вміст повідомлень електронної пошти. Системи класифікує електронні листи на основі дій користувачів та створює підсумки кожного вхідного повідомлення за допомогою підходу неконтрольованого навчання. Вони стверджують, що система може вирішити проблеми перевантаження електронної пошти, труднощів у

визначенні пріоритетів листів та пошуку зархівованих повідомлень на сервері електронної пошти.

Іншою концепцією автоматизованого керування електронною поштою від Kushmerick N. та Lau T. [5] є програми, які забезпечують підтримку високого рівня для структурованих дій в електронній комерції. Автори визначають формальну діяльність як автомати зі скінченним станом, які відповідають статусу процесу, а переходи представляють повідомлення, що пересилаються між учасниками та пропонують кілька алгоритмів машинного навчання без контролю.

Schuff D., Turetken O. та ін [6] впроваджують ефективні засоби керування електронною поштою, які розглядають повідомлення як корисну інформацію. Цей інструмент допомогає заощадити людські ресурси за рахунок відносно дешевої додаткової потужності апаратного забезпечення та пропускної здатності мережі. Крім того, їх додаток забезпечує автоматичну фільтрацію, кластеризацію, розглядає велику кількість електронних листів не як джерело перевантаження інформацією, а як інструмент управління знаннями.

Дослідження Chailiornkaew. L, Lirexawanlirasut T. та McAleer M. [7] пропонують класифікувати електронну пошту на чотири категорії: продажі, доставка, виставлення рахунків і транспортування. Для системи класифікації застосовуються два параметри: кількість слів у базі даних порівняно зі зразками електронних листів і прийнятний відсоток для класифікації електронних листів.

Підхід до багатокатегорійної класифікації електронної пошти з використанням семантичних ознак і методу динамічної реконструкції ієрархії категорій, за якого користувач реорганізує всі повідомлення електронної пошти за категоріями запропоновано у роботі Park S. Та An D. [8].

Sakurai S. та Suyama A. [9] запропонували метод вилучення ключових понять з електронних листів та їх статистичну інформацію, яка була застосована до трьох типів завдань аналізу: аналізу продукту, аналізу вмісту та аналізу адреси, у яких набуті словники відношення понять дали високі коефіцієнти точності в класифікації.

Aloui A. та Neji M. [10] розробили мультиагентну систему EQASTO (система відповідей на запитання електронної пошти, яка використовує методи інтелектуального аналізу тексту та онтології), що полегшує обробку електронних листів за допомогою комбінації методів аналізу тексту та онтологічних методів для класифікації семантично, система отримує електронні листи, генерує та автоматично надсилає відповіді учням.

Sharaff A. та Nagwani N. K. [11] використовують текстову подібність між атрибутами електронної пошти за допомогою прихованого розподілу Діріхле для визначення категорійних термінів. Abu-Nimeh S., Nappa D. та ін. [12] порівняли точність прогнозування численних алгоритмів машинного навчання, таки як: дерева класифікації та регресії, опорно-векторне кластерування (SVM), випадковий ліс (Random forest, RF), нейронні мережі та логістична регресія. Almomani A., Gupta B. та ін. [13] розглянули методи фільтрації фішингової електронної пошти та представили типи фішингових атак, класифікації фішингової електронної пошти та методи оцінки.

У галузі класифікації електронної пошти окрему нішу займає галузь класифікації з використанням NLP, машинного навчання та глибокого машинного навчання [14, 15]. Серед досліджень автори використовують методи навчання під наглядом [16], де використовувалися наївний баєсів класифікатор (NBC) [17], штучні нейронні мережі (Artificial Neural Network, ANN), k-найближчі сусіди (K-Nearest Neighbor, KNN), логістична регресія, класифікатор C4.5, багатошаровий перцептрон (Multilayer Perceptron, MLP), алгоритм машинного навчання AdaBoost (adaptive boosting) та методи випадкового лісу (RF) [18].

1.1.2. Сучасні програмні рішення для класифікації електронної пошти

Scalifi Ai Email Classification Suite. Потужна платформа керування електронною поштою, дозволяє виконувати автоматичну обробку, зберігаючі час та інші ресурси. При використанні Scalifi Ai процес пересилання електронної

пошти автоматизований та мінімізує навантаження на співробітників та мінімізує кількість людських помилок [12].

Програма предоставляє користувачам наступні функціональні можливості:

– керування доступом користувачів,

– автоматизовані інформаційні панелі аналізу даних,

– автоматична класифікація та позначення тегами електронних листів за допомогою штучного інтелекту;

– підтримка кількох постачальників послуг електронної пошти;

– налаштування за допомогою візуальних інструментів із функціями перетягування, без кодування;.

– забезпечення чіткої ідентифікації секретних даних, захист критично важливих даних та належне збереження.

Механізми, що реалізовані у программі засобами NLP:

– багатомовна класифікація – листи класифікуються на основі мови автора та перекладаються на мову користувача;

– класифікація електронних листів на основі імен відправника за технологією Named Entity Recognition;

– класифікаця та маркування на основі намірів або кількох намірів (замовлення, бізнес-вимоги тощо);

– класифікація та маркування на основі настроїв клієнтів (скарги, або вдячність);

– класифікація спаму для видалення небажаних електронних листів;

– класифікація на основі визначених користувачами правил (тематики або жанри).

Программа складається з трьох головних модулів:

– Data Ingestion – містить всі необхідні попередньо створені конектори для прийому даних із баз даних, сховищ даних, API (Application Programming Interface) та файлових систем;

– Model Training – вирішує питання керування інфраструктурою під час навчання моделей, виключаючи проблеми з конфігурацією та масштабованосттю.

– Data Analytics - допомогає аналізувати дані за допомогою настроюваних запитів у поєднанні з пакетною обробкою даних із навчених моделей.

Вбудований модуль Data Analytics надає цінну та практичну інформацію про різні аспекти спілкування електронною поштою в організації:

– обсяги та шаблони електронної пошти, що дозволяє визначати закономірності обігу, періоди пікової активності, тенденції у комунікації;

– час відповіді на електронни листи, що дозволяє оцінювати продуктивність роботи окремих співробітників чи команд;

– аналіз трафіку електронної пошти;

– найпопулярнішіх відправників та одержувачів, що дозволяє розробляти цільові комунікаційні стратегії;

– ефективність та продуктивність електронної пошти.

– аналіз вмісту електронної пошти, дозволяє визначити тенденції, важливі обговорення та сфери уваги;

– безпека та відповідність електронної пошти, що дозволяє відстежувати метадані електронної пошти, вкладення та поведінку користувачів, щоб запобігти порушенням безпеки.

Інтерфейс програми наведено на рисунку 1.1.

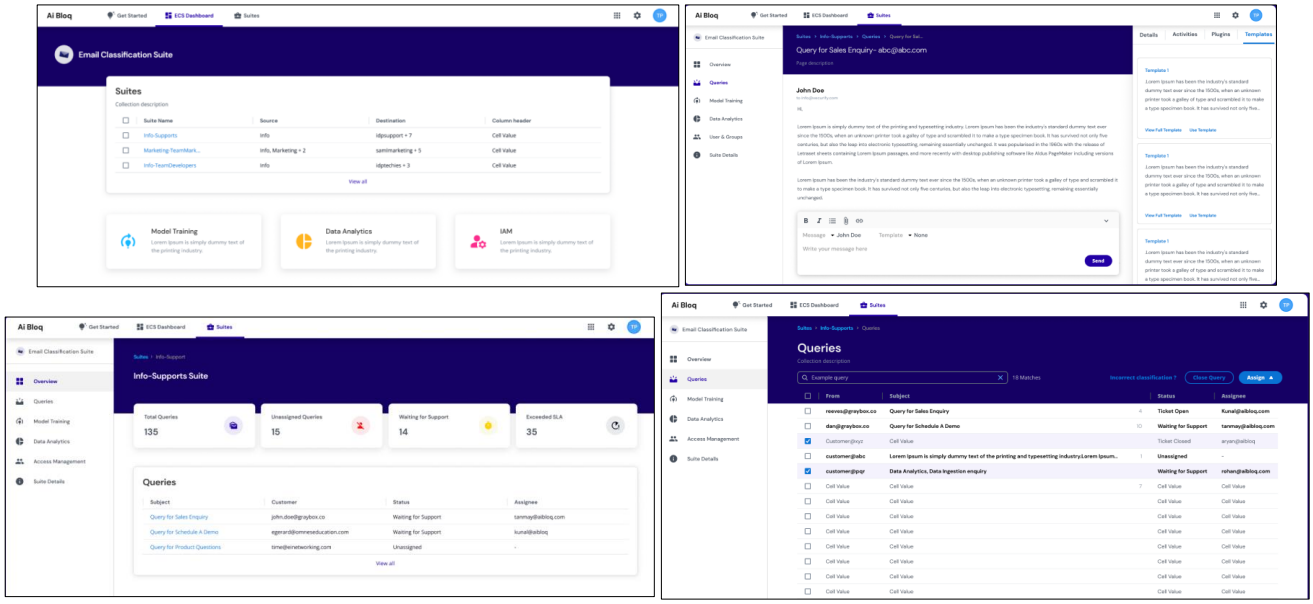


Рис. 1.1. Інтерфейс програми Scalifi Ai

Front. Масштабна платформа для командної роботи з клієнтами. Платформа дозволяє налавати послуги підтримки продажів та керування обліковими записами клієнтів. Front значно спрощує процес спілкування з клієнтами співпрацею команди в режимі реального часу, підвищуючи ефективність служби підтримки та забезпечує автоматизовану обробку електронної пошти [20]. Використання Front забезпечує централізаці. повідомлень по багатьох каналах, направлення та пере направлення сповіщень до потрібної особи, надавати статистику операцій із клієнтами у графічному виді. Front не інтегрується безпосередньо у поштвий обліковий запис, а централізує роботу Gmail та Outlook пошти у єдиній теці «Вхідні».

Функціонал платформи Front дозволяє :

– інтегрувати облікові записи Gmail, Outlook, SMS і соціальних мереж.

– розділяти поштові скриньки на спільні та індивідуальні;

– працювати над електроними листами індивідуально чи колективно;

– спільно працювати над чернетками;

– делегувати розмови та синхронізувати календар;

– автоматизовувати роботу с поштою запитами;

– відстеження переписки та ланцюги листів;

– виконувати аналіз клієнтів та співробітників;

– створювати розклад змін да автоматично призначати адреси найменьш навантаженим співробітникам;

– доступ до інтерфейсу прикладного програмування API;

– проводити інтеграцію з CRM (Customer Relationship Management) системами та платформами обміну повідомленнями.

Інтерфейс програми наведено на рисунку 1.2.

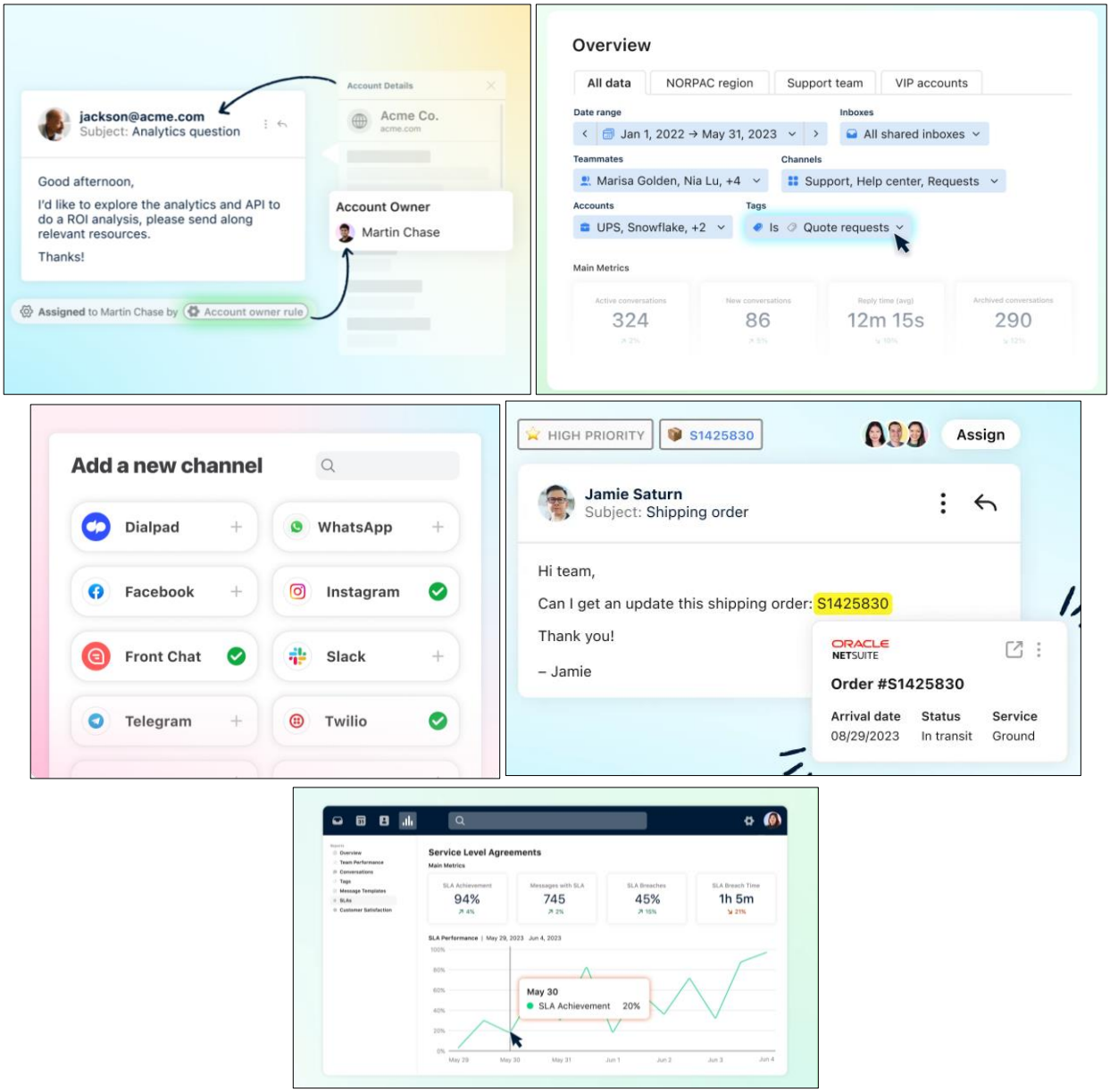


Рис. 1.2. Інтерфейс програми Front

AI Builder Microsoft Power Platform – набір інструментів, який дає можливість користувачам Microsoft Power Platform обробляти інформацію з використанням моделей штучного інтелекту, таких як прогнозування, обробка форм, виявлення об'єктів, класифікація категорій, а також вилучення об'єктів через досить простий і зрозумілий інтерфейс. Дає змогу використовувати аналітику для автоматизації процесів і отримання аналітичних даних у Power Apps AND Power Automate [21].

Платформа підтримує дві категорії моделей:

– готові моделі – готові екземпляри моделей налаштовані заздалегідь, що дозволяють обробку даних за допомогою підготовлених сценаріїв AI, включаючи вилучення ключових фраз, визначення мови, аналіз тональності, розпізнавання тексту, зчитування візитних карток, обробку квитанцій та інших сценаріїв;

– користувальницькі моделі – можна навчити відповідно до бізнес-потреб, надавши необхідні дані для їх навчання.

Відмінною особливістю цього продукту є тісна інтеграція з усіма продуктами лінійки Microsoft Power Platform, а також простий режим налаштування екземплярів AI для впровадження у бізнес-процеси.

Один із зісобів використання інструментарію – це категоризація листів. Power Automate «прослуховує» нові електронні листи, обробляє їх текст, надсилає до AI Builder для обробки та зберігає визначені категорії в цільовій системі. Модель класифікації дозволяє ідентифікувати текстові записи за допомогою тегів, які будуть використовуватися для таких речей, як:

– аналіз настроїв;

– виявлення спаму;

– маршрутизація запитів клієнтів;

– інші потреби бізнесу.

Інтерфейс програми наведено на рисунку 1.3.

Mail-cat від ParallelChain — це рішення для управління взаємовідносинами з клієнтами на основі штучного інтелекту, сучасний інтелектуальний інструмент-помічник, який допомагає користувачам із завданням класифікації електронної пошти, дозволяє конвертувати дизайни електронних листів в сучасні, адаптивні та оптимізовані електронні листи HTML. Mail-cat автоматично фільтрує, категоризує, об’єднує вхідні заявки та запускає відповіді, постійно оновлює свій класифікатор відповідно до того, чи приймає рекомендація користувач чи ні. Для електронного листа без міток він рекомендує три найкращі категорії, до яких цей електронний лист найімовірніше належить [22].

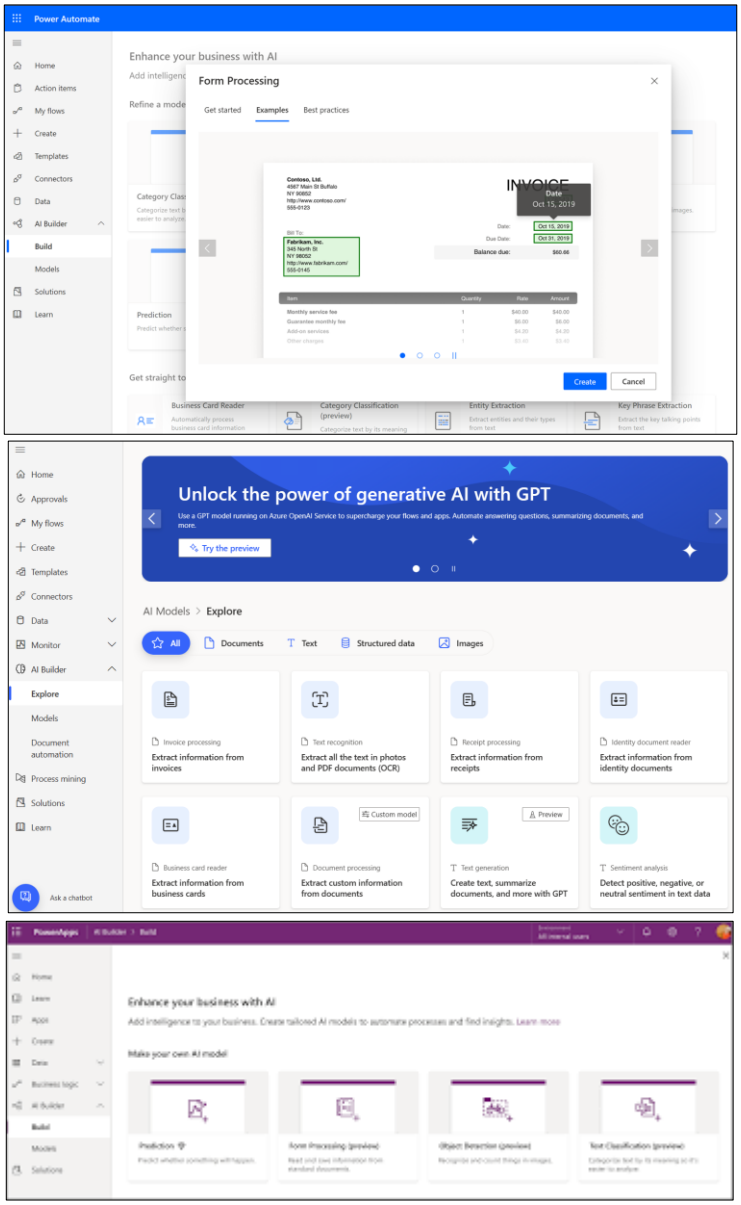


Рис. 1.3. Інтерфейс програми AI Builder

Особливості функціоналу Mail-cat:

– автоматизована категоризація – вхіні листи автоматично сортуються та розподіляються за такими категоріями, як терміновість, відповідальні відділи та типи клієнтів.

– використання NLP для розуміння змісту запитів і підвищення точності категоризації, полегшуючи автоматичні відповіді, відповідні контексту;

– інтелектуальна ескалація підтримки – листи з високим пріоритетом можна передати для негайного розгляду на основі терміновості, складності або цінності для клієнта;

– тригери відповіді на основі правил – конкретні відповіді чи дії запускаються відповідно до визначених компанією правил, які можуть ґрунтуватися на таких умовах, як тип листа, пріоритет, статус клієнта або певні ключові слова у вмісті листа.

– злиття або конкатенація листів – пов’язані або дубльовані листи ідентифікуються та об’єднуються в єдину групу, щоб усунути фрагментарне спілкування та неефективність процесу підтримки.

– інтеграція з поштовими системами, можливість розгортання у Outlook, Gmail, Hotmail та інших.

– локальна обробка та зберігання даних – листи можуть містити конфіденційні дані компанії та користувачів, розгортання MailCat локально гарантує, що вся обробка та зберігання даних відбуваються у локальній інфраструктурі.

Інтерфейс програми наведено на рисунку 1.4.

1.2. Програмні розробки у галузі обробки природньомовних текстів

1.2.1. Поняття та сфера використання NLP

Natural Language Processing (NLP), або обробка природної мови – це область галузі інформатики та штучного інтелекту (Artificial Intelligence, AI), яка приділяє увагу наданню можливості комп’ютерам інтерпретувати, розуміти, реагувати на людську мову та генерувати її. Мета NLP полягає у разробці алгоритмів та моделей, які здатні забезпечити безперешкодну та ефективну взаємодію між людьми та комп’ютерами за допомогою природної мови, не накладаючи умови та вимоги спеціального синтаксису чи команд при зверненні до комп’ютера [23].

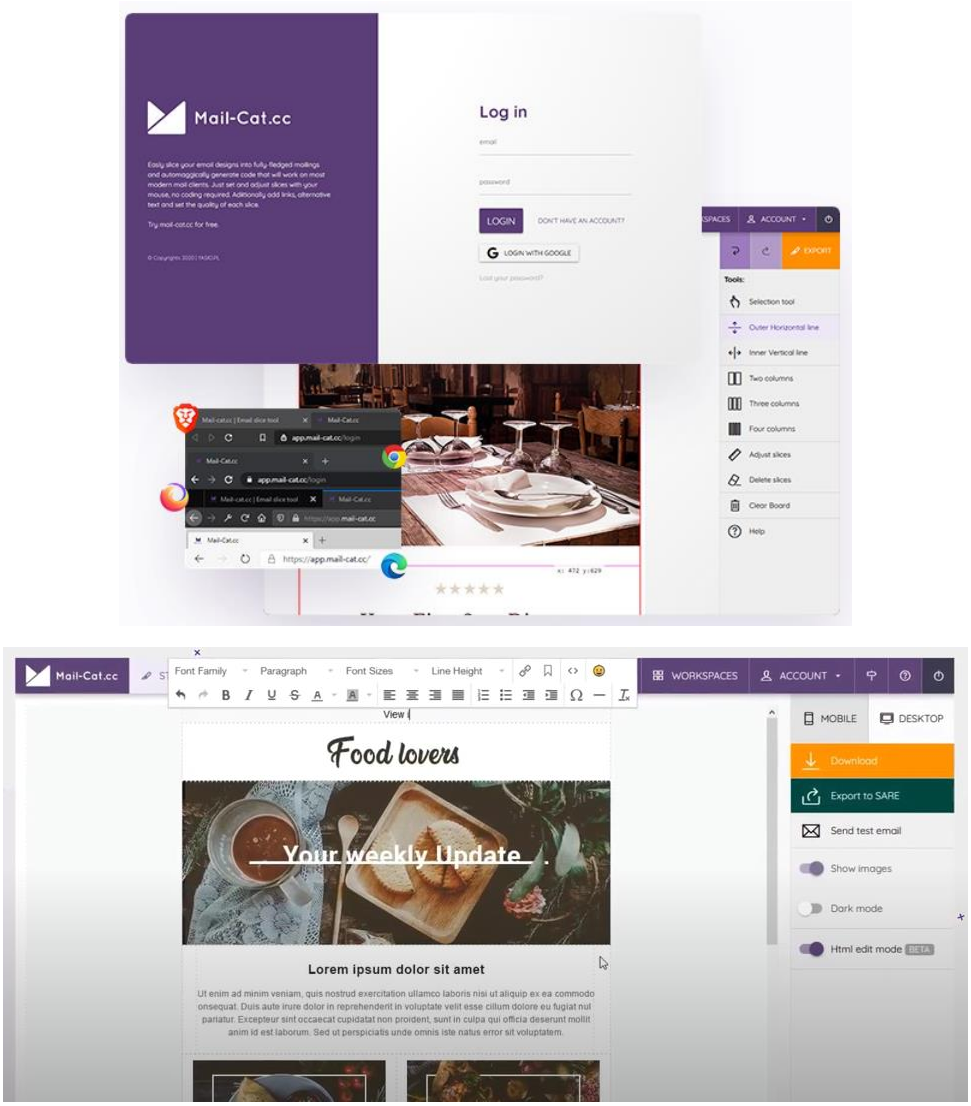


Рис. 1.4. Інтерфейс програми AI Builder

Історія розвитку NLP простежується до середини 20 століття, хоча її база лежить у лінгвістиці, інформатиці та штучному інтелекті. Першим етапом була пропозиція Алана Тюрінга щодо тесту Тюрінга в 1950-х роках – вимірювання здатності машини демонструвати людський інтелект, включаючи розуміння мови. У той час почалися спроби виконання машинного перекладу, що вважається зародженням галузі NLP.

У 60-70х роках 20го сторіччя з’являються ранні системи природної мови засновані на правилах, такі як ELIZA та SHRDLU, що намагалися імітувати її розуміння. Система ELIZA імітувала психотерапевта, використовуючи заздалегідь визначені правила, щоб реагувати на вхідні дані користувача, а система SHRDLU

демонструвала більш складне розуміння мови, але була обмежена певною областю планування, відомою як «світ блоків».

У 80-х роках 20го сторіччя почався перехід до статистичних методів із впровадженням алгоритмів машинного навчання та розробкою великомасштабних корпусів, таких як Корпус Брауна. У 90-х роках підходи машинного навчання використовувалися все частіше – Всесвітня павутина надавала безпрецедентну кількість текстових даних для досліджень і застосування.

На початку 21го сторіччя значно зростає інтерес до пошуку інформації, завдяки появі ефективних пошукових систем. В цей період ще більші набори даних стають доступними, що сприяє створенню більш надійних і точних мовних моделей.

У 10-х роках 21го сторіччя відбувається значний прогрес у технологіях глибокого навчання, таких як рекурентні нейронні мережі (Recurrent neural network, RNN) і мережі довготривалої короткочасної пам'яті (Long short-term memory, LSTM), які значно розширили коло задач для NLP. Представлення архітектури Transformers призвело до появи таких мовних моделей, як GPT (Generative Pre-trained Transformer) і BERT (Bidirectional Encoder Representations from Transformers), які вважаються зараз найпросунішими.

Обробка природної мови застосовується у багатьох сферах взаємодії людини з комп'ютером. Розглянемо основні з них.

1. Аналіз настроїв (Sentiment analysis), або аналіз думок, передбачає визначення настроїв або емоційного тону, що стоїть за фрагментом тексту. Методи NLP дозволяють ідентифікувати та класифікувати почуття як позитивні, негативні чи нейтральні. Аналіз настроїв має багато застосувань, наприклад моніторинг соціальних мереж, управління репутацією бренду, аналіз відгуків клієнтів і дослідження ринку.

2. Машинний переклад (Machine translation) – процес автоматичного перекладу тексту з однієї мови на іншу за допомогою алгоритмів NLP. Машинний переклад став можливим завдяки багатьом дослідженням і розробкам у рамках

NLP, і має зараз широкий спектр застосувань від служб перекладу в реальному часі, таких як Google Translate, до керування багатомовним контентом для веб-сайтів і підприємств.

3. Фільтрація електронної пошти (Email filteration) - найпростіше початкове застосувань NLP онлайн, у повідомленнях знаходилися слова чи фрази, які сигналізують про спам-повідомлення. Одне з найпоширеніших нових застосувань NLP – це класифікація електронної пошти. Наприклад, Gmail. розпізнає, чи належать електронні листи до однієї з трьох категорій (основні, соціальні або рекламні), на основі їх вмісту.

4. Інформаційний пошук (Information Retrieval) – Важлива підгалузь NLP – пошук інформації, яка спрямована на отримання релевантної інформації з великого набору даних. Її застосування є повсюдним, починаючи від пошукових систем і закінчуючи академічними дослідженнями, де швидкий і точний пошук інформації має вирішальне значення. У цьому напрямку розробляються системи відповідей, щоб надавати конкретні відповіді на запитання, поставлені природною мовою, якы зазвичай реалізуються в роботах обслуговування клієнтів та освітньому програмному забезпеченні.

5. Резюмування тексту (Text summarization) передбачає створення стислих анотацій довгих фрагментів тексту, зберігаючи важливу інформацію та зберігаючи зв’язність. Техніки NLP можна використовувати для виділення ключових фраз, речень або абзаців і побудови інформативного резюме. Це особливо корисно в таких сферах, як збирання новин, керування документами та пошук інформації.

6. Розпізнавання іменованих сутностей (Named entity recognition, NER) – процес ідентифікації та класифікації іменованих сутностей у тексті, таких як люди, організації, місця, дати, тощо. Має численні програми, включаючи вилучення інформації, аналіз даних і організацію вмісту. Наприклад, Розпізнавання сутностей можна використовувати для заповнення графів знань, увімкнення семантичного пошуку або фільтрації та категоризації статей новин.

7. Чат-боти та розмовні AI (Chatbots and Conversational AI) використовують NLP для розуміння та генерування людських відповідей природною мовою. Ці системи можуть бути розроблені для конкретних завдань, таких як підтримка клієнтів, або більш загальних цілей, як особисті помічники (наприклад, голосові системи Siri, Google Assistant або Amazon Alexa), що використовуються для багатьох завдань, від пошуку в Інтернеті до домашньої автоматизації, Розмовний штучний інтелект, шляхом автоматизації та забезпечення персоналізованої та ефективної взаємодії, привніс багато змін до різних галузей, як то обслуговування клієнтів, охорона здоров'я, фінанси, електронна комерція та інші.

8. Моделювання мови (Language Modeling). Однією з основоположних підгалузей NLP є моделювання мови. Моделювання природньої мови передбачає розробку статистичних або нейронних моделей, спрямованих на передбачення послідовності слів у певному тексті. Такі моделі є ключовими в програмах, таких як передбачення тексту, функції автозаповнення на клавіатурах та служби машинного перекладу.

Також алгоритми NLP застосовуються в охороні здоров'я для допомоги в інтерпретації складних медичних записів, що допомагає лікарям приймати більш обґрунтовані рішення при діагностиці пацієнтів та у процесі їх лікування. У юридичних секторах ця технологія використовується для ретельної перевірки контрактів, що традиційно вимагає людського досвіду та значних витрат часу. Кожна з цих сфер застосування має унікальну складність і може перетинатися з іншими, але разом вони пропонують комплексне уявлення про можливості та застосування обробки природної мови.

1.2.2. Сучасні системи обробки природньої мови

Популярність технологій NLP та їх активний розвиток призвели до виникнення великої кількості готових систем та наборів інструментарію для використання їх у будь-якій сфері діяльності людини. Розглянемо найбільш поширені та відомі з них.

Apache OpenNLP [24] – це набір інструментів для обробки тексту природною мовою, , що використовує машинне навчання. Проект з відкритим вихідним кодом, є частиною Apache Software Foundation. Бібліотека OpenNLP підтримує найбільш поширені завдання NLP, такі як: токенізація, сегментація речень, тегування частин мови (POS), виділення іменованих сутностей, синтаксичний аналіз та вирішення кореференції, що зазвичай потрібні для вирішення більш просунутих задач обробки тексту. Також OpenNLP включає максимальну ентропію та машинне навчання на основі перцептрона. Метою проекту OpenNLP є надання інструментарію для вищезгаданих завдань, великої кількості готових моделей для різноманітних мов, а також анотованих текстових ресурсів, з яких ці моделі походять. Бібліотека Apache OpenNLP містить кілька компонентів, які дозволяють створити повний конвеєр обробки природної мови. Ці компоненти включають наступні:

- визначення мови (Langdetect);
- сегментування речень (Sentdetect);
- токенізація (Tokenizer);
- тегування частин мови (Postagger);
- лематизація (Lemmatizer);
- чанкінг (Chunker);
- парсінг (Parser);
- категоризація документів (Document Categorizer);
- виділення іменованих сутностей (Name Finder);
- вирішування кореференції (Coreference Resolution).

Компоненти містять частини, які дозволяють виконувати необхідну задачу обробки природної мови, навчання моделі та її оцінювання. Apache OpenNLP забезпечує API для інтеграції кожного із цих засобів у програми, написані мовою Java. Крім того, для зручності експериментів і навчання передбачено інтерфейс командного рядка (CLI). Проект OpenNLP також активно розвивається і підтримується спільнотами розробників, що дозволяє забезпечити надійність і актуальність функціональності.

Microsoft Text Analytics API [25] – набір веб-сервісів текстової аналітики з пакету мовних служб Microsoft Azure Cognitive Service, що надає можливості розширеної обробки та аналізу неструктурованого тексту для таких завдань, як аналіз настрою, вилучення ключових фраз або визначення мови. Text Analytics API складається з наступних модулів [25]:

- визначення мови (Detect Language);
- розпізнавання сутностей (Entities);
- вилучення ключових фраз (Key Phrases);
- аналіз настрою (Sentiment).

Для використання Text Analytics API не потрібні навчальні дані, модулі можуть бути використані API REST або з клієнтською бібліотекою для .NET, Python, Node.js, Go або Ruby. Головною відмінністю Microsoft Azure Cognitive Service є використання контейнерів для інтелектуальної обробки даних [26], що забезпечує гнучкість у виборі розташування для розгортання та розміщення служб. Контейнери ізольовані один від одного і від базової операційної системи, при цьому вони витрачають менше ресурсів, аніж віртуальна машина. Також контейнери забезпечують високу пропускну здатність та низьку затримку, не обмежують кількість транзакцій за секунду, їх можна збільшувати та зменшувати для задоволення попиту, надавши необхідні апаратні ресурси.

OpenTextIDOL Unstructured Data Analytics [27] – це розширена платформа пошуку, виявлення знань і аналітики на базі платформи Autonomy IDOL (Intelligent Data Operating Layer) для обробки та аналізу великих обсягів даних від компанії Autonomy [28]. OpenText IDOL використовує штучний інтелект та машинне навчання, забезпечує уніфіковану аналітику тексту, мовлення та відео з підтримкою понад 1000 форматів даних, забезпечує доступ до 150 сховищ даних (наприклад, Documentum, Dropbox тощо), а також індексує дані без переміщення та збоїв. Він включає в себе ряд технологій, таких як:

- автоматичне розпізнавання мовлення (Automated Speech Recognition, ASR) для перетворення аудіо-вмісту в текст;

− аналіз тональності для визначення емоційної окраски тексту (позитивної, негативної, нейтральної);

− категоризація та класифікація для організації та структурування інформації;

− пошук семантичного контенту для більш точного та ефективного пошуку та вилучення інформації з тексту;

− обробка зображень і відео, може аналізувати і вилучати інформацію з мультимедійних файлів;

− інтеграція з іншими системами, підтримує взаємодію з різними додатками та платформами.

Платформу створено на основі технологій машинного навчання та глибоких нейронних мереж [27]. При пошуку інформації та виявленні знань, використовуючи набір даних природною мовою, IDOL здійснює формування відповідей на запитання, використовуючи попередній досвід спілкування з користувачем. Це дозволяє будувати прості та контекстуальні діалоги між користувачем та системою. IDOL не спирається на глибоке знання граматичної структури мови, а набуває розуміння через контекст вживання слів. Це особливо корисно при аналізі розмовної чи неформальної мови, яка не дотримується лінгвістичних правил NLP-систем.

Lexalytics Intelligence Platform [29] – інтегрована платформа текстової аналітики та обробки природньої мови від компанії Lexalytics, призначена для обробки великих обсягів текстових даних з метою вилучення корисної інформації та отримання розуміння контексту. У 2014 році Lexalytics придбала Semantria з метою розширити базу клієнтів з багатомовною підтримкою, після чого було розроблено Semantria API [30] інтерфейс програмування додатків, що надається компанією Lexalytics, призначений для роботи з їх текстовим аналізом та аналітикою, та використовує SaaS та хмарні технології.. Продукт Semantria є сервісом аналізу тексту SaaS, що пропонується у вигляді плагіна на основі API та Excel.

Основними компонентами текстового аналізу платформи Lexalytics є наступні:

- оцінка настрою (Sentiment);
- токенізація та тегування частей мови (Tokenization and POS Tagging);
- категоризація документів (Categorization);
- визначення сутностей(Entity Recognition);
- вилучення теми (Themes);
- лексичні ланцюжки (Lexical Chaining);
- визначення мови(Language Detection);
- коротке викладення тексту(Summarization);
- зв’язки між об’єктами (Relationships).

MonkeyLean [31] – платформа надає сервіс автоматизованого створення, навчання та інтеграції користувацьких та попередньо навчених моделей, виконувати аналіз тексту та створення власних класифікаторів без необхідності глибокого розуміння програмування та алгоритмів машинного навчання для таких завдань як класифікація тексту, аналіз думок та вилучення іменованих сутностей. MonkeyLearn може бути використаний для різних завдань аналізу тексту, таких як обробка звернень клієнтів, соціальних медіа моніторингу, аналізу відгуків тощо. Платформа надає візуальний інтерфейс для створення та навчання моделей, що робить її доступною для людей без глибоких знань машинного навчання.

MonkeyLearn дозволяє користувачам використовувати REST API для отримання доступу до платформи безпосередньо або через клієнтську бібліотеку та забезпечує підтримку мов програмування Python, Ruby, Javascript [32]. В якості основних форматів файлів, що використовуються для обробки та навчання, використовуються формати CSV та Excel.

Попереднє навчання моделі кожного модуля (крім модуля Preprocessing) потребує завантаження даних для навчання безпосередньо в платформу або ж використовуючи хмарні сервіси (наприклад, Google Drive), нерозмічені дані потребують ручної анотації. Наприклад, для навчання модуля NER потрібно

спочатку встановити список сутностей, а потім анотувати дані через платформу. Після навчання моделі виводяться графіки з результатами навчання, використовуючи поширені метрики (наприклад, точність (precision) і повнота (recall)). Після навчання користувач може подавати масиви даних до платформи, отримуючи результат NLP-обробки.

Основні компоненти платформи включають наступні:

- попередня обробка (Preprocessing);
- аналіз настрою (Sentiment Analysis);
- вилучення теми та слів тематики (Topic Labeling);
- вилучення іменованих сутностей (NER);
- вилучення властивостей суб'єктів (Feature Extraction);
- визначення намірів суб'єкта (Intent Detection);
- витягувати важливих слів (Keyword Extraction).

Neticle Text Analysis API [33] – набір інструментів корпоративної текстової аналітики від компанії Neticle. дозволяє класифікувати документи, визначати емоційний тон документи, отримувати іменовані сутності на основі ключових слів (словників), детальну візуалізацію теми та маршрутизацію електронної пошти. Крім того автоматичне розпізнавання тем і розпізнавання емоцій можна вирішити за допомогою API, у хмарі чи локально. Інструментарій дозволяє користувачеві доповнювати та додавати словники ключових слів, а також додавати синоніми до них. Neticle Text Analysis оцінює вхідні тексти, використовуючи оцінку «індекс думки»,значенням від -3 до +3, залежно від релевантності ключових слів.

Основними компонентами текстового аналізу платформи Neticle є наступні:

- визначення мови (Language Detection);
- попередня обробка, стемминг, токенізація, видалення стоп-слів (Preprocessing);
- визначення класу думки (Sentiment analysis);
- визначення теми (Topic recognition);

- вилучення іменованих сутностей (NER);
- визначення атрибутів об’єкту (Attribute recognition);
- визначення статі особи за його ім’ям (Gender recognition);
- визначення класу думки для обраного об’єкту (Entity oriented sentiment analysis).

Google Cloud Natural Language API [34] – сервіс обробки природньої мови від Google Cloud використовує API REST. Текстові масиви можуть бути надіслані через запит або завантажені через Google Cloud Storage. Особливістю є інтеграція, можливість використання результатів аналіза в інших сервісах Google Cloud.

API надає різні можливості для аналізу тексту, має кілька методів для виконання аналізу та анотації такі як:

- аналіз настроїв (Sentiment analysis);
- аналіз сутностей (Entity analysis);
- аналіз настрою сутностей (Entity sentiment analysis);
- синтаксичний аналіз (Syntactic analysis);
- класифікація тексту (Content classification).

Основні компоненти сервісу включають наступні:

- визначення мови (Language Detection);
- вилучення речень (Sentence Extraction);
- розділення речень на токени (Tokenization);
- синтаксично-морфологічний аналіз (Syntatic Analysis);
- визначення тематики (Content classification);
- визначення емоцій (Sentiment analysis);
- визначення іменованих сутностей (Entity analysis);
- визначення емоційного стану до сутностей (Entity sentiment analysis).

Saga Natural Language Understanding (NLU) [35] – фреймворк надає масштабовану структуру для створення та підтримки моделей природної мови для взаємодії з користувачем і розуміння документів для 60 мов, дозволяє користувачам створювати та підтримувати гнучкі та масштабовані моделі для

обробки текстових масивів. Saga NLU поєднує безліч методів моделювання мови та машинного навчання в єдине, зручне для користувача семантичне середовище, що дозволяє обробляти різні варіанти використання природної мови. Архітектура фреймворку побудована на Apache Spark та Spark MLlib.

Основними компонентами текстового аналізу, що Saga NLU є наступні:

- визначення мови тексту (Language identefication);
- визначення меж речень (Sentence detection);
- токенізація, лематизація, стемминг тексту (Tokenize);
- вилучення синтаксично-морфологічної інформації (Syntactic morphological analysis);
- виявлення груп іменників та дієслів (Chunker);
- визнаення загального емоційного тону (Sentiment Analysis);
- визначення тематики документа (Categorize content);
- визначає нові теми, якщо теми відсутні у модулі Categorizing content (Clusterize content);
- вилучення іменованих сутностей (Extract entities);
- визначення намірів суб'єкта(Intent detection);
- визначення класу думки щодо сутності (Extract entities with sentiment);\
- визначення зв’язків між словами (Relationship extraction);
- визначення фактів (Fact extraction);
- пошук подібностей між документами (Similarity detection);
- авто реферування документу(Summarization).

Розглянуті системи надають наглядну інформацію, що до структури та компонентів, що є найпоширенішими у сфері обробки текстів природньою мовою.

1.2.3. Спеціалізовані засоби та інструменти обробки природньої мови

В галузі обробки природної мови розроблено безліч спеціалізованих засобів та інструментів, які допомагають розробникам та дослідникам у створенні додатків та систем, здатних аналізувати та розуміти природну мову [36].

Наведемо декілька основних категорій програмних інструментів, що використовуються в галузі NLP.

1. Бібліотеки машинного навчання та глибокого навчання:

− NLTK (Natural Language Toolkit) – бібліотека мовою Python, складається багатьох бібліотек та однією з найпопулярніших платформ Python для обробки та аналізу тексту природньою мовою, пропонує простий вступ до програмування додатків обробки мови, саме їй віддають перевагу як початківці так і досвідчені розробники [37];

− SpaCy – бібліотека мовою Python з відкритим вихідним кодом для обробки тексту з акцентом на високу продуктивність та простоту використання, зарекомендувала себе як бібліотека для використання у виробництві, спрощуючи розробку програм, які зосереджені на обробці значних обсягів тексту за короткий проміжок часу [38];

− Genism – це спеціальна бібліотека Python, що розроблено для індексації документів, моделювання тем і аналізу схожості документів, містить реалізацію популярних алгоритмів Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) і Word2Vec, працює з використанням ресурсів Corpora, алгоритми в Genism залежать від пам'яті, що стосується розміру Corpus – це означає, що віна здатна обробити вхідні дані, які перевищують доступну оперативну пам’ять у системі;

− NumPy – бібліотека мовою Python, надає можливість обробки багатовимірних масивів та матриці, має великий набір математичних функцій, що є корисними для обчислень у глибокому навчанні [39];

− PyNLPl (Pineapple) – бібліотека Python, яка складається з кількох модулів Python, розроблених спеціально для завдань NLP, найбільшою особливістю PyNLPl є його комплексна бібліотека для розробки формату лінгвістичної анотації (FoLiA) XML.

2. Фреймворки глибокого навчання:

− TensorFlow та PyTorch – популярні фреймворки для розробки моделей глибокого навчання, які можуть бути використані для обробки природної мови [40,41];

− Transformers (Hugging Face) – бібліотека, що надає моделі та інструменти для роботи з трансформерами, які показали видатні результати NLP [42].

3. Засоби для обробки тексту:

− Tokenizers (Hugging Face) – інструменти для поділу тексту на токени, що є важливим етапом обробки тексту для подальшого аналізу [43];

− Beautiful Soup та lxml – бібліотеки на мові Python для парсингу HTML та XML, що може бути корисним при обробці тексту з веб-сторінок [44]

− WildNLP — платформа для тестування стабільності моделі в природному середовищі, де відбуваються пошкодження тексту, наприклад помилки клавіатури або орфографічні помилки.

4. Моделі обробки природної мови:

− BERT (Bidirectional Encoder Representations from Transformers) – модель, передбачена на великих обсягах тексту, що показала видатні результати широкому спектрі NLP-задач [45];

− GPT (Generative Pre-trained Transformer) – сімейство моделей, створених OpenAI, передбачених на величезних обсягах тексту для генерації людиноподібного тексту [46].

− XLNet (eXtreme Language Model) – авторегресійна мовна модель, націлена на обробку контекстних залежностей, використовує для цього рекурентну архітектуру та функцію Cross-Layer Attention для [47].

5. Інструменти для аналізу сентименту:

− VADER (Valence Aware Dictionary and sEntiment Reasoner) – бібліотека для аналізу сентименту у тексті[48];

− TextBlob – інструмент Python для обробки текстових даних, включаючи визначення сентименту [49], зосереджен на тому, щоб зробити загальні функції обробки тексту доступними через прості у використанні інтерфейси – об’єкти в TextBlob можна використовувати як інструкції Python, які можуть надавати функціональність NLP для створення програм аналізу тексту.

6. Інструменти для отримання інформації:

− Stanford NER – інструмент для отримання та класифікації іменованих сутностей з тексту[50];

− spaCy – окрім попередньої обробки, spaCy також надає функціональність для вилучення інформації з тексту [38].

7. Інструменти для роботи з діалоговими системами:

− Rasa – відкрита платформа для створення чат-ботів та діалогових систем з використанням машинного навчання [51].

8. Хмарні сервіси аналізу тексту:

− Amazon Comprehend – хмарний сервіс від Amazon для аналізу тексту, що надає функціональність, таку як вилучення ключових фраз, аналіз настрою та визначення іменованих сутностей [52].

− IBM Watson Natural Language Understanding – хмарний сервіс від IBM для аналізу тексту з функціональністю, такий як категоризація контенту, аналіз сентименту та вилучення сутностей [53].

− Microsoft Azure Text Analytics – хмарний сервіс від Microsoft для аналізу тексту, включаючи функції аналізу сентименту, вилучення ключових фраз та іменованих сутностей [54].

Сучасні засоби обробки природньої мови дозволяють виконувати широкий спектр задач від найпростішого розбиття тексту на речення до глибоко аналізу тексту з розпізнаванням тематики та емоцій [30]. Найбільша частина засобів є безкоштовними та доступними для користування, що надає багато можливостей для використання їх відповідно до вирішення конкретних завдань.

1.2.4. Методи та моделі обробки природньої мови

Наразі існує багато підходів до обробки природньої мови, від застарілих вже підходів на основі правил, до популярних у теперішній час моделей глибоко навчання. Наведемо основні з них відповідно до етапів розвитку [23].

1. Підходи на основі правил. На початку NLP підходи, засновані на правилах, були нормою. Це включало використання регулярних виразів для зіставлення шаблонів тексту, синтаксичних дерев для аналізу речень у

структурованих форматах, автоматів із кінцевим станом для таких завдань, як морфологічний аналіз, граматики на основі функцій для детальнішого синтаксичного аналізу та логіки першого порядку для представлення семантики мови для машини висновок.

2. Статистичні моделі. У міру розвитку галузі наприкінці 90-х і початку 2000-х відбувся зсув у бік статистичних моделей. Такі алгоритми, як Naive Bayes, стали популярними для класифікації тексту та фільтрації спаму. Приховані моделі Маркова зазвичай використовувалися для розпізнавання мовлення та позначення частин мови. Дерева рішень знайшли місце в лінгвістичному вивченні правил, а алгоритм TF-IDF отримав широке поширення для пошуку документів та інформації. Методи статистичного машинного перекладу також почали поступово відмовлятися від алгоритмів перекладу на основі правил.

3. Алгоритми машинного навчання. Наступна хвиля прогресу NLP прийшла з широким впровадженням технік машинного навчання. Машини опорних векторів SVM широко використовувалися в задачах категоризації тексту, тоді як випадкові ліси використовувалися для різноманітних завдань класифікації та регресії. Кластеризація K-means стала корисною для кластеризації документів і тематичного моделювання. Крім того, Reinforcement Learning знайшов застосування в діалогових системах та інших інтерактивних програмах NLP.

4. Моделі глибокого навчання [40-42]. Обробка мови значно змінилася з появою алгоритмів глибокого навчання. Рекурентні нейронні мережі RNN та їх більш просунуті версії, такі як одиниці довгострокової короткочасної пам’яті LSTM і керовані рекурентні блоки (Gated Recurrent Units, GRU), стали основними алгоритмами для вирішення проблем прогнозування послідовності. Механізми уваги помітно покращили продуктивність систем машинного перекладу, а моделі BERT та GPT встановили нові стандарти продуктивності в ряді завдань NLP.

5. Трансформери [42]. В останні роки трансформаторна архітектура стала домінувати в сфері NLP. Ця архітектура базується на глибоких нейроних мережах та є основою для більшості найсучасніших моделей. Варіанти та наступники трансформера, такі як T5 (Text-To-Text Transfer Transformer) і GPT-3,

продовжують розширювати межі можливостей NLP. Модель BERT також отримала різні спеціалізовані варіанти трансформеної архітектури, такі як RoBERTa та DistillBERT. Також продовжуються дослідження та розробки з метою підвищення ефективності цих моделей.

Мовні моделі трансформерів зазвичай мультиязичні, але й існують зменшені версії для окремих або декількох мов. Для обробки текстів українською мовою використовують наступні спеціалізовані мовні моделі трансформерів:

− BERT (bert-base-ukrainian-cased) – модель BERT від групи дослідників з мови Ukrainian NLP Community;

− Electra (EMBEDDIA/electra-base-ukrainian) – модель Electra від команди EMBEDDIA;

− Roberta (youscan/roberta-base-ukrainian) – модель RoBERTa від групи дослідників з мови Ukrainian NLP Community.

Окрім спеціалізованих моделей використовують наступні мультимовні моделі з підтримкою української мови:

− LaBSE (Language-agnostic BERT Sentence Embedding);

− T5 – модель Text-To-Text Transformer (T5) від Google Research;

− XLM-RoBERTa;

− XLNet;

− DistilBERT – скорочена версія BERT, що підтримує декілька мов, у тому числі і українську;

− •Multilingual BERT – багатозадачна багатомовна модель BERT з підтримкою української мови.

Розглянемо основні методи обробки текстів природньою мовою.

Попередня лінгвістична обробка (Linguistic Preprocessing) є першим базовим кроком у конвеєрі обробки природної мови, який готує необроблений текст для подальшого аналізу та розуміння. Він включає розбиття та уточнення тексту на основні компоненти, гарантуючи, що дані чисті та структуровані. Цей процес є вирішальним для наступних етапів NLP, оскільки він безпосередньо впливає на

точність і ефективність моделей і алгоритмів, застосованих пізніше. Наведемо кілька ключових методів попередньої лінгвістичної обробки:

2) токенізація (Tokenization) – процес перетворення тексту в його складові слова або підслова, широко відомі як токени;

3) виділення коренів і лемматизація (Stemming and Lemmatization) – скорочення слів до їх основи або кореневої форми, визначення основи є грубим евристичним процесом, тоді як лемматизація розглядає морфологічний аналіз слів;

4) позначення частин мови (Part-of-Speech Tagging) – позначення кожного слова в реченні відповідною частиною мови (наприклад, іменник, дієслов, прикметник тощо);

5) видалення стоп-слів (Stop Word Removal) – видалення загальновживаних слів (наприклад, «і», «є»), які можуть не додати значного значення під час аналізу тексту;

6) сегментація речень (Sentence Segmentation) – поділ тексту на окремі речення, що особливо корисно для завдань, які працюють на основі кожного речення, як аналіз настроїв або переклад.

Синтаксичний розбір (Syntactical Parsing) працює зі структурними аспектами мови, щоб розшифрувати розташування та зв’язок слів у реченнях. Незважаючи на те, що самі слова вже несуть значення, те, як вони організовані та пов’язані одне з одним у реченнях, дозволяє глибше зрозуміти передане повідомлення. Синтаксичний розбір, по суті, є процесом аналізу речень для визначення їхньої граматичної структури. Це розуміння має ключове значення для багатьох завдань NLP, оскільки допомагає розрізняти нюанси та тонкощі людської мови. Наведемо основні прийоми, пов’язані з синтаксисом і аналізом:

1) розбір залежностей (Dependency Parsing) – визначення граматичних зв’язків між словами в реченні для формування дерева залежностей, метод фіксує залежності між словами, вказуючи, значення яких слів залежить від інших;

2) синтаксичний аналіз групи (Constituency Parsing) – розбиття речень на підфрази або «складові», часто представлені у вигляді дерева, підхід фокусується

на ієрархічній структурі речень, групуючи слова у вкладені складові на основі синтаксичних правил;

3) чанкинг (Chunking) – виділення іменних груп (noun phrases, NP) або груп слів, пов'язаних із іменником фрагментів (чанків) із тексту на основі синтаксичних патернів, що уявляють собою послідовності слів, об'єднані певними граматичними відносинами, процес чанкінгу містить застосування граматичних шаблонів або правил для виділення чанків на основі частин мови (Parts of Speech, POS) та інших лінгвістичних ознак;

4) граматичні та продуційні правила (Grammar and Production Rules): набір правил, які визначають дійсні структури речень у мові. Ці правила керують процесом розбору, гарантуючи, що похідні структури є граматично правильними;

5) дерева синтаксичного аналізу (Parse Trees) – візуальне представлення синтаксичної структури речень, можуть бути використані для зображення відносин як залежності, так і виборчого округу;

6) вирішення неоднозначності (Ambiguity Resolution) – вирішення ситуацій, коли речення можна проаналізувати різними способами через неоднозначне формулювання чи структуру, ефективні методи аналізу спрямовані на вибір найбільш ймовірної інтерпретації на основі контексту.

Розуміння синтаксису та використання ефективних методів синтаксичного аналізу є важливими для таких завдань, як машинний переклад, відповіді на запитання та резюмування тексту, де розуміння структурних нюансів мови може значно підвищити якість результатів.

Семантичний аналіз працює у сфері значення в мові, намагаючись зрозуміти основні поняття та зв’язки, що передаються словами та реченнями. Тоді як синтаксис зосереджується на структурі мови, семантика займається змістом і нюансами значення. У сфері обробки природної мови семантичний аналіз відіграє ключову роль у подоланні розриву між людською мовою та машинним розумінням, дозволяючи системам інтерпретувати текст у спосіб, який більше узгоджується з людським розумінням. Наведемо основні прийоми та концепції, пов’язані із семантичним аналізом:

1) розпізнавання іменованих об’єктів (Named Entity Recognition) – ідентифікація та класифікація іменованих об’єктів, таких як особи, організації та місця розташування в тексті, має вирішальне значення для таких завдань, як вилучення інформації та відповіді на запитання;

2) розпізнавання значення слова (Word Sense Disambiguation) – визначення значення слова на основі його контексту, допомагає зрозуміти слова, які мають кілька значень, забезпечуючи правильне тлумачення в певному контексті;

3) позначення семантичних ролей (Semantic Role Labeling)– визначення семантичних ролей слів у реченні, таких як підмет, об’єкт або присудок, забезпечує глибше розуміння дій і сутностей, описаних у реченні;

4) онтології та графіки знань (Ontologies and Knowledge Graphs) – структуровані представлення знань, що фіксують зв’язки між сутностями та поняттями, відіграють важливу роль у таких завданнях, як семантичний пошук і міркування.

5) вилучення зв’язків (Relationship Extraction) – визначення зв’язків між іменованими об’єктами в тексті, наприклад, хто де працює або хто з ким пов’язаний.

6) розв’язання співвідношень (Coreference Resolution) – визначення, коли різні слова чи фрази в тексті стосуються однієї і тієї ж сутності, як визнання того, що «головний герой» та «він» у уривку стосуються однієї особи.

Семантичний аналіз є основою для безлічі передових програм NLP, від чат-ботів і систем рекомендацій до семантичних пошукових систем. Розуміючи значення слів і речень, системи NLP можуть природніше й ефективніше взаємодіяти з користувачами.

Для системи атоматизованої обробки запитів користувачів у якості моделі для обробки запитів природньою мовою було обрано моделі трансформерів, як найбільш сучасних та розвинутих моделей NLP.

2 ПРОЕКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОЇ ОБРОБКИ ЗАПИТІВ КОРИСТУВАЧІВ

Процес класифікації електронної пошти поділяється на три етапи: попередня обробка, навчання та класифікація. При роботі автоматизованої системи категоризації листів спочатку виконується збір набір даних електронної пошти, потім очищення набору даних. Завдання очищення даних загальновідоме як попередня обробка даних. На етапі попередньої обробки електронний лист перетворюється на набори ключових слів та фраз. Рівень попередньої обробки також усуває непотрібні слова або стоп-слова, щоб зменшити кількість даних, які потрібно перевірити для їх розташування. Також на етапі попередньої обробки виконуються різні лінгвістичні, семантичні розбори та синтаксичний аналіз, тощо. На рівні навчання розробляються набори функцій, тобто опису ознаки, які представляють вимірювання певного аспекту. Після виділення ознак для класифікації обираються найбільш показникові ознаки, з метою підвищення продуктивності класифікатора, ефективності та точності прийнятих рішень. Класифікатор створюється та зберігається для класифікації майбутніх вхідних електронних листів.

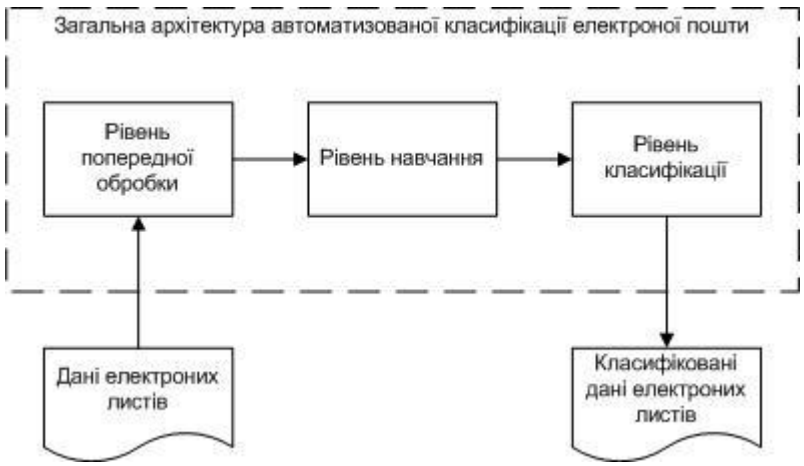


Рис.2.1. Загальна архітектура автоматизованої класифікації електронних листів

Попередня обробка даних електроних листів надає похідні дані для подальшого навчання системи та класифікації. Оскільки електрони листи, що є запитами від користувачів, у загальному випадку створюються та надсилаються саме користувачами, основним джерелом даних для його класифікації є саме текст листа, що написано природньою мовою.

2.1. Бізнес процес обробки запитів електронною поштою

До переваг обробки запитів людиною можна віднести: високу ступень відповідності сформованої відповіді на запит, за рахунок здатності людини враховувати контекст та індивідуальні особливості кожного запиту, розпізнати емоції та вирази та реагувати на невизначені ситуації, де правила чи шаблони не завжди застосовні.

До недоліків обробки запитів особисто відносяться наступні: часові обмеження періоду обробки запитів – лише у робочі часи; як слідство, можливість відсутності відповідей у критичний для запитувача період часу; широкий спектр сфер питань, в яких робітники приймальної комісії не можуть надати кваліфікованої відповіді; брак часу для аналізу великих обсягів даних та формування статистичної звітності по результатах обробки запитів.

Розробимо бізнес модель обробки запитів, що надходять електроною поштою до приймальної комісії ЗВО. Для опису операцій (потоку робіт) процесу обробки запитів використаємо технологію Swimlane (технологія плавальних доріжок) та нотацію для моделей та бізнес-процесів BPMN (Business Process Model and Notation) [55]. Нотація використовується для опису процесів на нижньому рівні, а діаграма кожного процесу уявляє собою умовний алгоритм виконання цього процесу. На діаграмі визначаються виконавці процесу, події, матеріальні та документальні потоки які його супроводжують.

Розглянемо бізнес-процес як модель існуючого стану процесу. У якості виконавців процесу обробки запитів електроною поштою виступають

співробітник приймальної комісії, він відіграє активну роль. Також до учасників можно віднести користувачів, що надсилають запити, поштовий сервер та відповідальних осіб, що відповідають на певні запити, але їх активна діяльність не відноситься до процесу, що розглядається. На рисунку 2.2 наведено схему бізнес процесу.

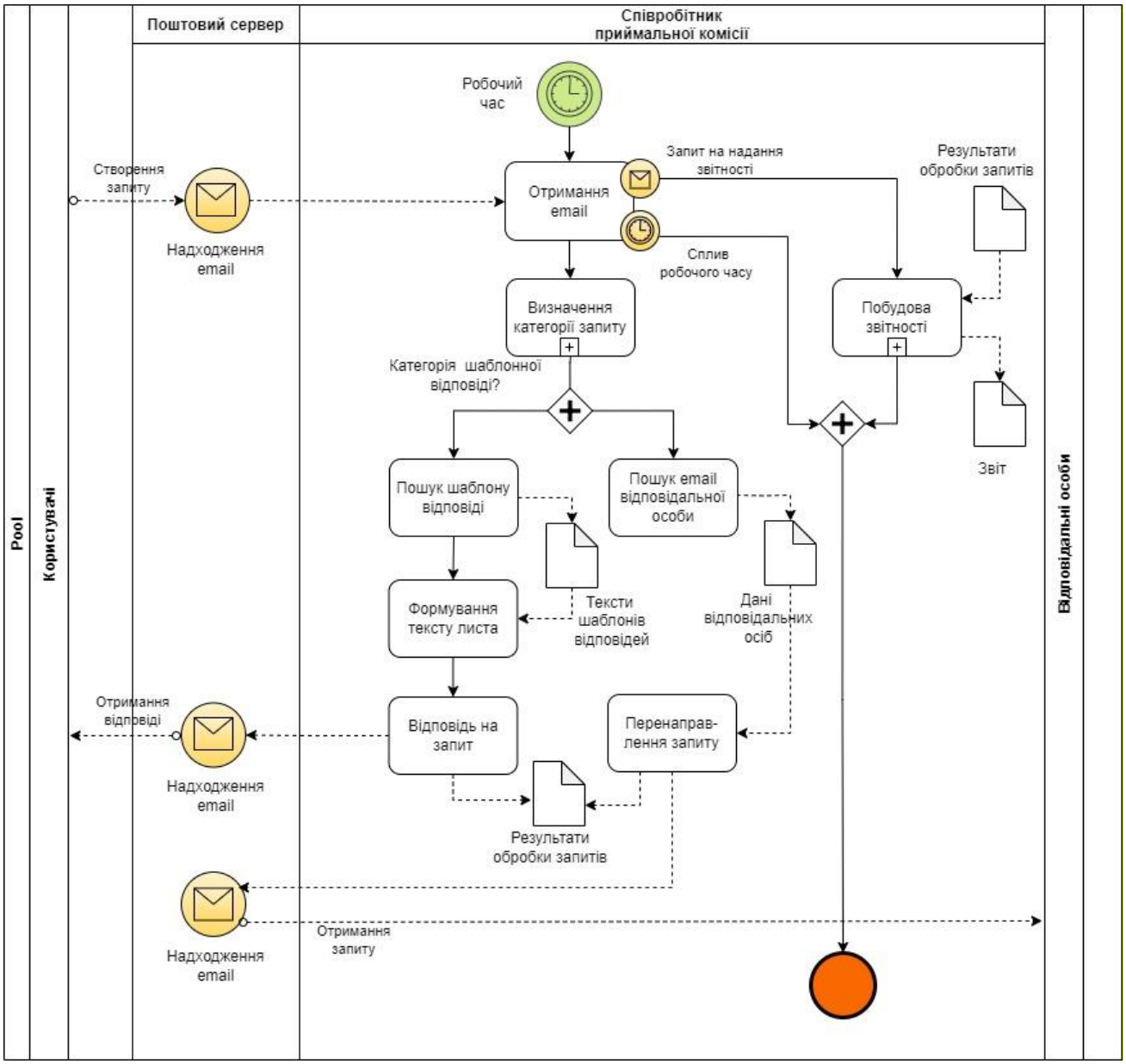


Рис. 2.2. Бізнес процес обробки запитів

Бізнес-процес починається з події – початок робочого часу, подія коли виконавець починає працювати над обробкою запитів. Після початку робочого

часу виконавець отримує (відкриває) електронний лист, перечитує його та визначає або якою повинна бути відповідь, якщо лист відноситься до запитів, що потребують однієї з заздалегідь відомих шаблонних відповідей, або до запиту, де необхідно долучити кваліфіковану по даному питанню особу.

У випадку шаблонної відповіді співробітник визначає тип відповіді, шукає шаблон у певних сховищах створює лист-відповідь, формує текст відповіді та відправляє лист до поштового серверу, звідки лист буде направлено користувачу, що надіслав запит. У випадку запиту, що потребує долучити кваліфіковану особу, відповідальну за певні категорії запитів, співробітник приймальної комісії перенаправляє лист відповідальній особі, попередньо визначивши з наявного переліку цю особу та її контактні дані – адресу електронної скриньки.

По завершенню обробки запиту з одного листа, співробітник отримує (відкриває) наступний лист та обробляє його аналогічним шляхом. Процес закінчується по виникненню події – закінчення робочого часу, або може бути перервано на інший тип робіт з запитами – оформлення звітності за обробленими електронними листами.

Для усунення наведених недоліків пропонується використання для обробки запитів сучасних інформаційних технологій шляхом впровадження системи автоматизованої обробки запитів до приймальної комісії ЗВО, що надходять електронною поштою. За рахунок використання автоматизації етапів отримання, категоризації, пере направлення листів, формування шаблонних відповідей та надсилання їх буде збережено робочий час співробітника приймальної комісії, скорочено час між запитом та відповіддю, а також, при організації роботи системи у режимі служб, цілодобова обробка запитів без обмежень на робочий час працівника.

Розглянемо оптимізований бізнес процес, що наведено на риунку 2.3.

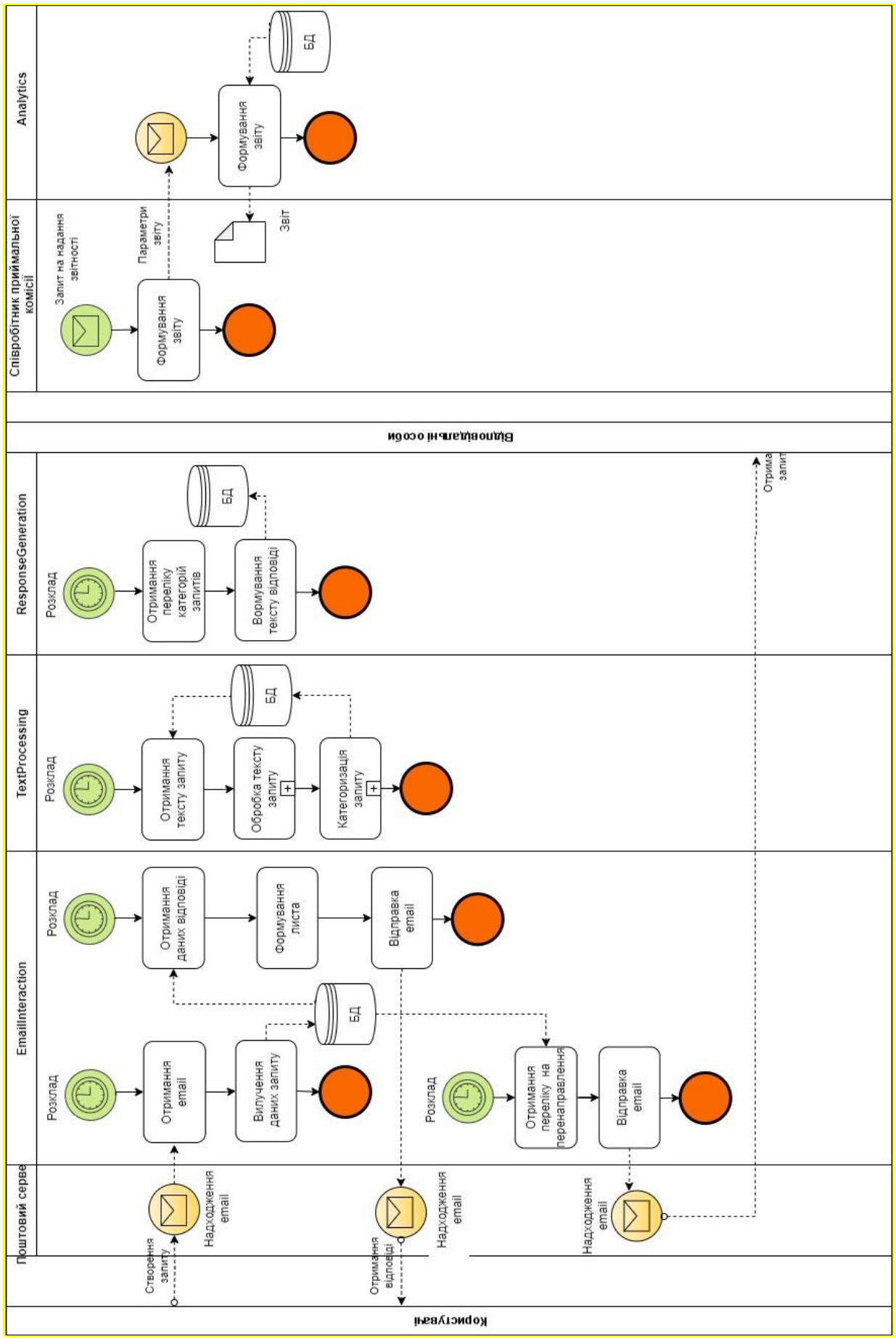


Рис. 2.3. Оптимізований бізнес процес обробки запитів

Служби роботи з електронною поштою, обробки тексту запиту та формування відповідей працюють за розкладом.

Служба роботи з електронною поштою працює у трьох режимах: отримання та вилучення даних з листів, надсилання листів-відповідей та пере направлення листів до відповідальних осіб. Служба працює з базою даних, вносячи та отримуючи дані з неї.

Служба обробки текстової інформації також працює за розкладом, але незалежно від інших служб. Служба отримує текстові дані для обробки з бази даних, обробляє їх, класифікує запити та розміщує результати категоризації також у базі даних.

Служба формування відповідей працює за розкладом та незалежно від інших служб. Вона отримує з бази даних інформацію про категорії запитів, формує тексти відповідей та заносить результати формування відповідей у базу даних. За цими результатами служба взаємодії з електронною поштою формує листи-відповіді та надсилає їх, або перенаправляє листи відповідальним особам.

Таким чином, співробітник майже повністю виключається з процесу обробки запитів користувачів та лише користується інтерфейсом доступу до аналітичної частини для отримання статистичних даних та формування звітності.

2.2. Архітектура системи автоматизованої обробки запитів

Система автоматизованої обробки запитів користувачів, отриманих електронною поштою призначена для оптимізації процесу категоризації запитів та формування релевантних відповідей на них. Беручі до уваги вимоги до системи та результати оптимізації бізнес процесу обробки запитів співробітниками приймальної комісії було розроблено архітектуру системи, яку наведено на рисунку 2.4.

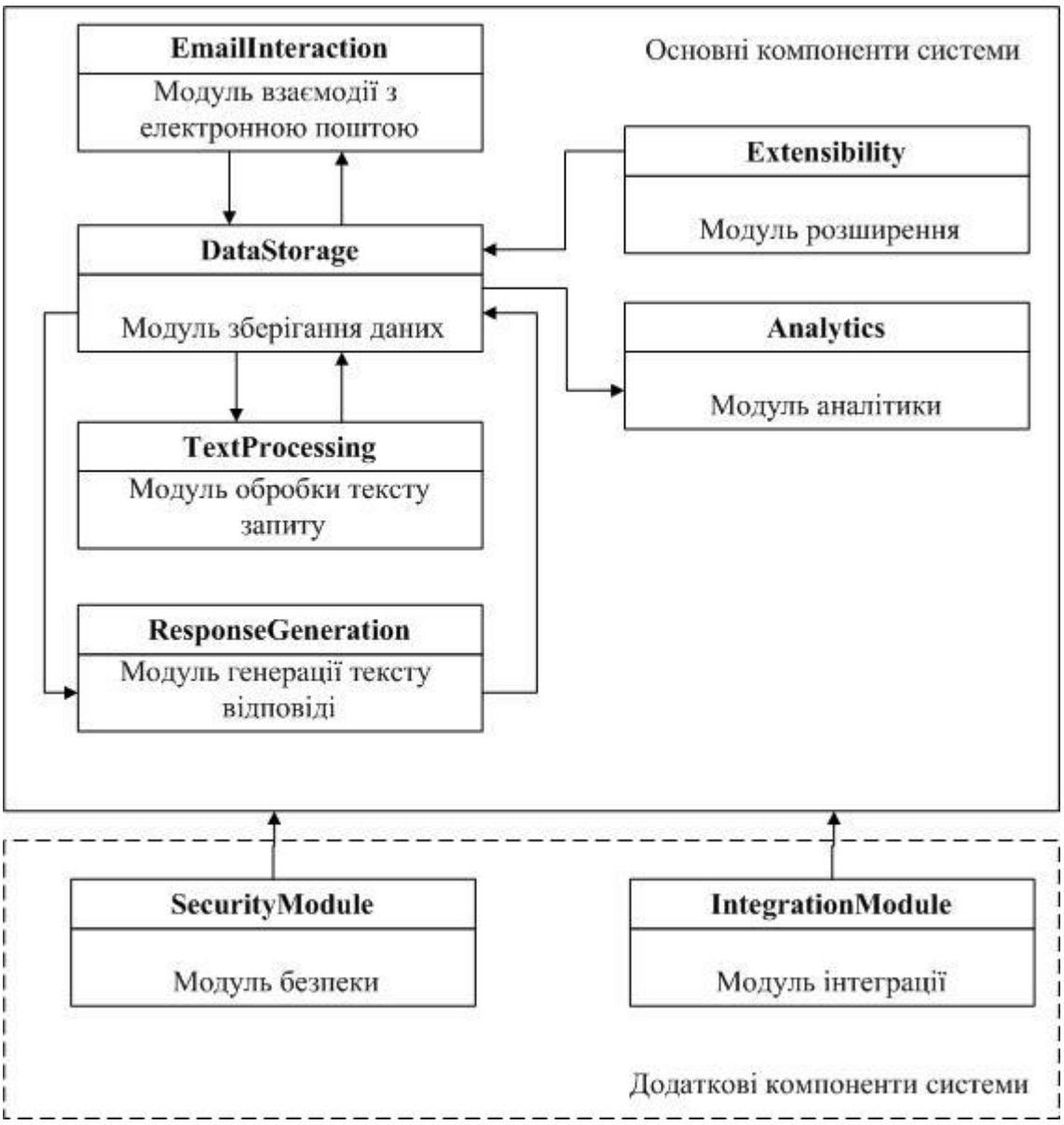


Рис. 2.4. Архітектура системи автоматизованої обробки запитів користувачів

Розглянемо елементи розробленої архітектури.

Модуль взаємодії з електронною поштою EmailInteraction. Виконує з'єднання з поштовим сервером (IMAP, SMTP), отримання.вхідних листів, вилучення тексту, відправка листа-відповіді, перенаправлення листів.

Модуль обробки тексту запиту TextProcessing. Виконує попередню обробку тексту листів, оцінку за ключовими словами та фразами, та визначає категорії до яких відноситься запит.

Модуль генерації тексту відповіді ResponseGeneration. Відповідає за вилучення даних для відповіді та формування тексту листа-відповіді

Для реалізації вищезазначених модулів слід розглядати такі методи та підходи, що забезпечать функціонування їх як служб, у фоновому режимі та за розкладом з метою усунення часових обмежень обробки запитів та скорочення часу періоду від отримання запиту до відсилання відповіді [56].

Модуль зберігання даних DataStorage. Відповідає за зберігання даних системи, зберігання ключових слів та фраз, шаблонних відповідей, інформації, що до категоризації тексту, іншої інформації, що необхідна для побудови відповідей на запити. Для реалізації модуля необхідно розглядати системи керування базами даних та технології доступу до даних.

Модуль розширення Extensibility. Відповідає за надання інтерфейсу для зручне керування даними системи (додання, корегування, видалення).

Модуль аналітики Analytics. Відповідає за надання інтерфейсу для формування та отримання статистики обробки запитів та візуалізацію статистичних даних.

Найкращім рішенням для реалізації інтерфейсних модулів виглядає надання користувачам web-интерфейсу, що забезпечує кросплатформенність доступу та легкість розгортання на різних серверах, або навіть хмарних платформах [57].

МодульSecurity. Відповідає за забезпечення безпечного функціонування системи та цілісності даних.

Модуль інтеграції Integration. Відповідає за інтеграцію з завнішними для системи сервісами (чати, оповіщення, нагадування, тощо).

2.3. Технології реалізації системи обробки запитів

Для реалізації системи обробки запитів користувачів, необхідно проаналізувати та обрати зручні та доступні технічні засоби. До ключових технологій необхідних для розробки проекту можна віднести:

– мова та технологія програмування сервісів взаємодії з електронною поштою, обробки текстів запитів та формування текстів відповідей;

– тип бази даних та СКБД (Система керування базою даних);

– фреймворк та бібліотеки для надання веб-інтерфейсу користувачам системи.

2.3.1. База даних та СКБД

База даних (БД) – це набір даних, рызного типу, що пов'язані між собою спільною ознакою або властивістю. Бази даних поділяють на два основних види, выдповідно до опису структур даних: реляційні та нереляційні. Для управління базами даних використовуються спеціальні комплекси мовних та програмних засобів – системи керування базами даних (СКБД). Основні функції, що покладаються на СКБД – це створення, керування та спільнео використання БД багатьма користувачами [58].

Реляційні СКБД керуються жорсткою структуризацією та типізацією відомостей про дані при побудові БД. Основною структурною одиницею у у реляційних СКБД є таблиця – модель вертикальних стовпців із унікальними іменами та горизонтальних строк із даними [59]. Для оперування даними реляційні БД використовують структуровану мову запитів (Structured Query Language, SQL), якапередбачає чітке визначення структури даних.

Нереляційні БД дозволяють будувати динамічну структуру даних [60], для їх розробки використовуються різні моделі: документо-орієнтована, модель на основі пар «ключ-значення», графова модель, стовпчикова модель, модель часових рядів та багатомодельні БД. Головною перевагою нереляційних БД є можливість оперувати даними, не формуючи їхню структуру заздалегідь.

Для системи, що розробляється, було вирішено використовувати реляційну БД та центральний сервер з БД, так як система не потребує розподіленої БД для обробки даних та доступу до даних модулів системи.

Для реляційних SQL баз даних використовують насупні СКБД.

Oracle Database (Oracle RDBMS) – об'єктно-реляційна система СКБД від компанії Oracle, уявляє собою один з найбільш надійних і широко використовуваних СКБД, що дозволяє керування фізичною структурою БД не впливаючи на доступ до логічних структур зберігання [61]. До переваг даної СКБД можна віднести багатоплатформену підтримку, підтримку об’єктно-орієнтованого підходу, легку масштабованість та ефективну систему хешування. До недоліків відносять великій об’єм займаного місця на сервері, складність розгортання, витратна при комерційному використанні. Крім того, система потребує кваліфікованих спеціалістів для розгортання та є вимогливою до апаратного забезпечення.

PostgreSQL – відкрита система СКБД, базується на використанні та розширенні SQL, відрізняється перевіреною архітектурою, високою надійністі, має широкий набір функцій, забезпечує цілісність даних та надає можливості до розширення. PostgreSQL має багато доступних розширень та інноваційних рішень, наприклад, геопросторовий розширювач бази даних PostGIS [62]. Перевагами використання таких систем є об’єктно-орієнтованість, підтримка різних платформ розгортання, велика кількість додатків системи. До недоліків можна віднести складність налаштування роботи з системою, може працювати повільніше навіть на простих операціях у порівнянні з іншими СКБД.

SQL Server – система керування реляційними БД від компанії Microsoft. СКБД була розроблена та використовується як конкурент до Oracle Database і MySQL, підтримує стандартну мову SQL та Transact SQL (T-SQL). Основним інструментом для роботи із SQL Server є середовище SQL Server Management Studio (SSMS) [63]. До переваг системи можна віднести зручність роботи з базою даних, високу надійність та можливість відновлення даних, багато інструментарію налаштування. До недоліків відносять обмежену підтримку платформ розгортання, вимогливість до ресурсів системи, комерційні версії продукту також дорогі.

Для керування реляційною базою даних системи було обрано СКБД SQL Server 2019, як найбільш зручну та надійну при роботі з даними.

2.3.2. Мова програмування сервісу взаємодії з електроною поштою

Для створення служби електронної пошти можна використовувати майже будь-яку мову програмування. Однак вибір мов програмування залежить від багатьох факторів. Не можна сказати, що певні мови програмування не перевершують інші з точки зору побудови служби електронної пошти, але все залежить від того, наскільки мова може відповідати вимогам.

Будь-яка мова, яка працює на веб-сервері, може бути інструментом розробки функціоналу взаємодії з електронною поштою. Однак, при розробці служби мають бути враховані наступні файтори:

− база даних для зберігання інформації про електрону пошту має бути сумісною з мовою служби електронної пошти;

− більш доречною є розробка служби є яка буде сумісна з різними платформами;

− служба електронної пошти складається з двох частин: сервера та клієнта, тож мова програмування повинна мати можливість працювати як на сервері, так і на клієнті;

− в загальному випадку служба повинна мати інтерфейс, найпопулярнішим наразі є саме веб-інтерфейси або інтерфейси на основі браузера, оскільки вважаються найбільш дружніми;

− різноманітні плагіни та бібліотеки значно полегшують процес розробки служби, тож доречним є обрання мови, що має пакетні бібліотеки та сумісні плагіни.

Розглянемо деякі мови програмування та їх порівняльні можливості для розробки модулю взаємодії з електроною поштою модулю EmailInteraction.

Для розробки великої та комплексної програми електронної пошти гарним вибором є мова Java, оскільки вона проста у використанні та пропонує велику бібліотеку вбудованих класів, попередньо налаштованих функцій та інших утиліт, які не тільки скорочують час на розробку та спроможні швидко виконувати велику кількість завдань, таких як пошук, зберігання, завантаження, масштабування, автентифікація, тощо. Окрім того, програми електронної пошти

потребують багато пам’яті, а Java ефективніша за багато інших мов у питаггі керування пам’яттю.

Убагатьох аспектах, мова Python схожа на Java, однак, незважаючі на швидкі зріст Python все ще не настільки багатий на функції, як Java. Перевагою Python є те, що мова легша для вивчання та реалізації, тому Python вважається кращим вибором для новачків та недосвітчених розробників Java. Python пропонує велику кількість бібліотек для машинного навчання та розробки компонент штучного інтелекту.

Мова C++ та Java також мають багато спільного, мова також має стандартну, хоча й меньшу, бібліотеку. C++ є сильною та надійною мовою, підходить для реалізації деяких специфічних функцій, але на відміну від Java або Python, її не можна вважати мовою дійсно загального призначення.

2.3.3. Мова програмування сервісу обробки запитів

Моделі обробки природної мови можуть бути застосовані з використанням різних мов програмування, але деякі з них надають зручні бібліотеки та фреймворки, які значно спрощують роботу з NLP. Розглянемо деякі мови програмування та їх порівняльні можливості для розробки модулю обробки запитів користувачів TextProcessing.

Python (TextBlob, SpaCy, NLTK, Genism, PyNLPl, Transformers, scikit-learn). Мова загального призначення, що має багато застосувань, зокрема машинне навчання, дослідження даних і серверне програмування. Завдяки широкому спектру бібліотек і інструментів, спеціально розроблених для аналізу тексту, Python став популярною мовою для дослідників, розробників і спеціалістів з обробки даних, які працюють з текстовими даними. Простий синтаксис Python забезпечує читабельність, пришвидшує код і спрощує вивчення мови. Незалежно від досвіду програмування, Python є вдалим вибором як для початківців, так і для досвідчених інженерів машинного навчання та науковців із обробки даних через його адаптивність і простоту.У Python є широкий вибір перевірених, готових бібліотек з відкритим кодом, які прискорюють процес розробки. У той час як

Keras, Caffe та TensorFlow забезпечують глибоке навчання, Scikit-learn пропонує основні методи машинного навчання, такі як класифікація або регресія. Python є найкращою мовою програмування для обробки природної мови завдяки своїй простій структурі та утилітам обробки тексту, таким як NTLK і SpaCy [64].

1. Java (Apache OpenNLP, Stanford NLP, LingPipe). Функціональна об’єктно-орієнтована мова програмування з простим синтаксисом і простим налагодженням. Java стала переважною мовою для створення мобільних додатків, в основному покладаючись на AI. Завдяки швидшому виконанню та меншому часу виконання, ніж Python, Java чудово справляється з додатками реального світу, що робить її ідеальною мовою AI для задач машинного навчання, що потребують часу. Java забезпечує просту масштабованість масштабних або складних програм AI. Кілька бібліотек машинного навчання підтримують Java, включаючи Massive Online Analysis, програму інтелектуального аналізу даних з відкритим кодом, і Weka, яка використовується для методів машинного навчання, прогнозного моделювання та інших речей. Оскільки розробники широко використовують Java, її легко інтегрувати з такими популярними технологіями обробки великих даних, як Apache Hive, Apache Hadoop і Apache Spark.Технологія віртуальної машини Java також дозволяє програмістам створювати налаштовані інструменти швидко та послідовно писати та запускати код на всіх підтримуваних платформах [65].

2. JavaScript (Natural, Compromise, NLP.js). Високорівнева, слабо типізована, динамічна, об'єктно-орієнтована прототипна мова програмування, що є реалізацією стандарту ECMAScript. Поряд з HTML та CSS є однією з трьох основних технологічних мов розробки веб-сторынок. Мова використовуэться для обробки природньої мови, оперуючи такими бібліотеками як: NLP.js – бібліотека NLP для створення ботів із виділенням сутностей, аналізом настроїв, автоматичним визначенням мови та підтримкою 40 мов; Natural – бібліотека загальних функцій природної мови для Node.js, що підтримує токенізацію, визначення коренів, класифікацію, фонетику, tf-idf, WordNet, подібність рядків та

інші; Compromise.cool – легка бібліотека, яка проста у використанні та дозволяє використовувати NLP у браузері. [66].

3. Scala (Stanford CoreNLP). Мова програмування загального призначення, що підтримує функціональне та об’єктно-орієнтоване програмування. Вихідний код Scala було розроблено для роботи на віртуальній машині Java, тому стеки Java і Scala взаємозаміні. Сумісність Scala з Java робить мову корисною для розробки Android додатків з використанням AI. Бібліотека Apache Spark MLlib & ML пропонує алгоритми кластеризації, класифікації та контрольованого навчання, а також модулі, які можна використовувати для створення конвеєрів даних. Бібліотека BigDL та Apache PredictionIO пропонують функціонал, що полегшує створення та розгортання алгоритмів машинного навчання [67].

4. R (tm, quanteda, openNLP). Добре відома мова для застосування у статистичному навчанні, також широко використовується в NLP та корисна для інтенсивних аналітичних обчислень при навчанні. R підходить для дослідження і аналізу масивних даних у NLP [65].

5. Ruby (Ruby Linguistics, Text, Stanford CoreNLP Ruby). Популярна мова програмування високого рівня, відома своєю простотою, гнучкістю та потужними можливостями. Він часто використовується для веб-розробки, сценаріїв і аналізу даних і відрізняється лаконічним синтаксисом. Бібліотеки TensorFlow, Scikit-learn, Keras, що ряд інструментів і алгоритмів, які можна використовувати для створення та навчання моделей машинного навчання можна легко інтегрувати в додатки Ruby. Бібліотека Treat є комплексною бібліотекою NLP, яка надає різні інструменти для обробки тексту, вилучення інформації та генерування тексту мовою Ruby.

6. C# (OpenNLP Sharp, SharpNLP). Мова C# також має кілька рішень та інструментів NLP, які можна використовувати для виконання таких завдань, як токенізація, тегування частини мови, аналіз настроїв, розпізнавання сутностей тощо: Stanford.NLP – оболонка .NET для бібліотеки Stanford CoreNLP, яка надає широкий спектр функцій NLP, OpenNLP, SharpNL – реалізації .NET бібліотеки Apache OpenNLP, NLTKSharp – порт .NET бібліотеки Natural Language Toolkit

(NLTK) для Python, LingPipe: це бібліотека на основі Java, але її можна використовувати в C# за допомогою віртуальної машини Java IKVM.NET.

Для розробки сервісів модулю взаємодії з електроною поштою модулю EmailInteraction та модулю обробки запитів користувачів TextProcessing було обрано мову Python, як найпоширеніший інструмент програмування для розробки програм аналізу тексту завдяки великій кількості доступних користувальницьких бібліотек, які зосереджені на забезпеченні функцій обробки природної мови. Будь то попередня обробка тексту, аналіз настроїв чи виявлення прихованих тем, Python надає необхідні інструменти та техніки для ефективного вирішення різноманітних завдань NLP. Окрім того, великою перевагою Python є його простота та можливість взаємодії взаємодії з іншими мовами програмування, наявність великої кількості різноманітної документацію та потужної підтримки спільноти.

2.3.4. Засоби програмування користувацького інтерфейсу

Сучасні інформаційні системи, що використовуються для вирішення задач у багатьох сферах, найчастіше розробляються за клієнт-серверною технологією. Клієнт-сервер – це технологія створення програмних рішень, в яких виконання задач з обробки інформації розподіляється між програмою-сервером та програмою-клієнтом [68].

Для розробки системи було обрано багаторівневу архітектуру «клієнт-сервер», структуру якої наведено на рисунку 2.5, структура якої може бути описана концептуальними логічними рівнями – відокремленими функціональними областями, що виконують різні задачі в рамках роботи однієї системи:

− рівень представлень – для введення даних і отримання інформації у зручному та зрозумілому для користувача графічному виді, інтерфейс системи;

− рівень обчислень – алгоритми та правила обробки даних системою, відповідно до яких здійснюється інформаційний обмін між рівнем представлень і рівнем даних;

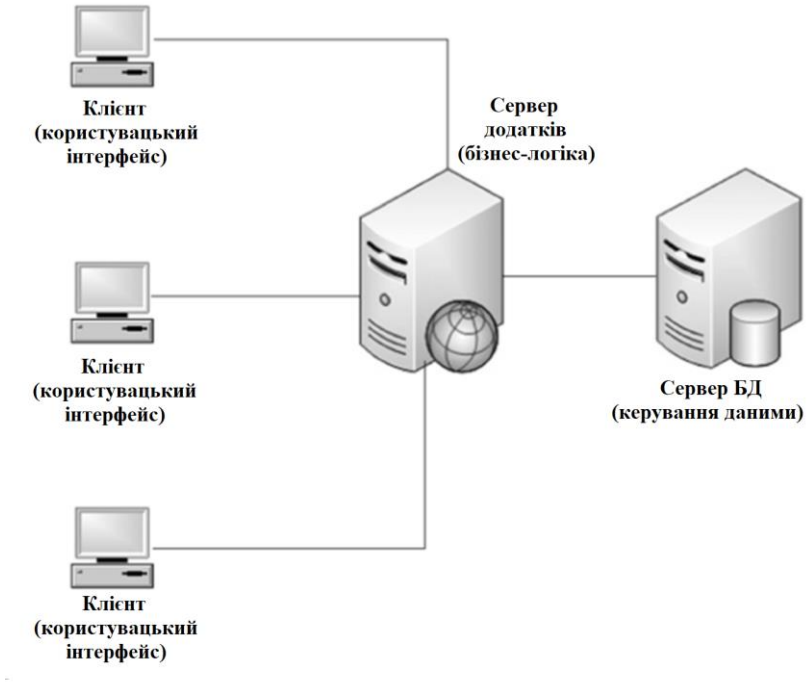


Рис. 2.5. Бацаторівнева архітектура «клієнт-сервер»

– рівень даних – забезпечує зберігання та надання програмними компонентами системи.

До переваг використання архітектури клієнт-сервер можна віднести наступні [69]:

– серверна частина виконує більшу частина роботи та мінімізує навантаження на клієнтську частину;

– зберігання даних на сервері забезпечує кращій захист від різноманітних загроз, ніж на клієнтському пристрої;

– надання кожному клієнту свого рівня доступу до інформаційної системи;

– клієнтська кросплатформність, можливість користування ресурсами сервера незалежно від клієнтсбкої платформи;

– низьке навантаження на мережу за рахунок передачі клієнтом серверу лише команд на виконання.

Однак, окрім переваг, архітектура також має й недоліки:

– при непрацездатності серверу уся система також непрацездатна;

– висока вартість серверного обладнання та його обслуговування, необхідність додаткового персоналу для обслуговування серверного обладнання;

– необхідність забезпечення стабільної роботи канал зв'язку до сервера.

Для розробки програмної структури системи було обрано інтегроване середовище розробки Visual Studio 2019, яке дозволяє розробляти програмні продукти різних видів, а також веб-додатки, використовуючи різні мови програмування. Для створення проекту було обрано кросплатформене програмне забезпечення каркаса веб-додтків з відкритим вихідним кодом ASP.NET Core та мову програмування C#. Структуру розробленого рішення RequestProcessing наведено на рисунку 2.6.

Для роботи з розробленою базою даних було використано технологію Entity Framework об'єктно-реляційного відображення даних БД, яка дозволяє автоматично пов'язати программні класи мови C# з таблицями у БД та підтримує роботу з СКБД MS SQL Server 2019. Для розробки веб-сторінок було обрано технологію Razor Pages – сторінки, що складаються з двох частин: розмітки веб-сторінки cshtml та класу-контроллера, що дозволяє створювати сторінки, які можуть опрацьовувати запити.. Перевага обрання такого рішення можливості використання контролера як моделі сторінки одночасно, що повністю відповідає ідеології об’єктно-орієнтованого програмування – у одному об'єкті інкапсулюються дані та методи роботи з ними [70].

Рішення побудовано за багатошаровою архітектурою, складається з 4х проектів, що відповідають певному шару:

− RequestProcessingCore – шар даних, відповідає за взаємодію з БД;

− RequestProcessingInfrastructure – інфраструктурні компоненти;

− RequestProcessingApplication –бізнес-логіка системи, відповідає за бізнес-процеси системи;

− RequestProcessingWeb – презентаційний шар, відповідає за взаємодію з користувачами системи.

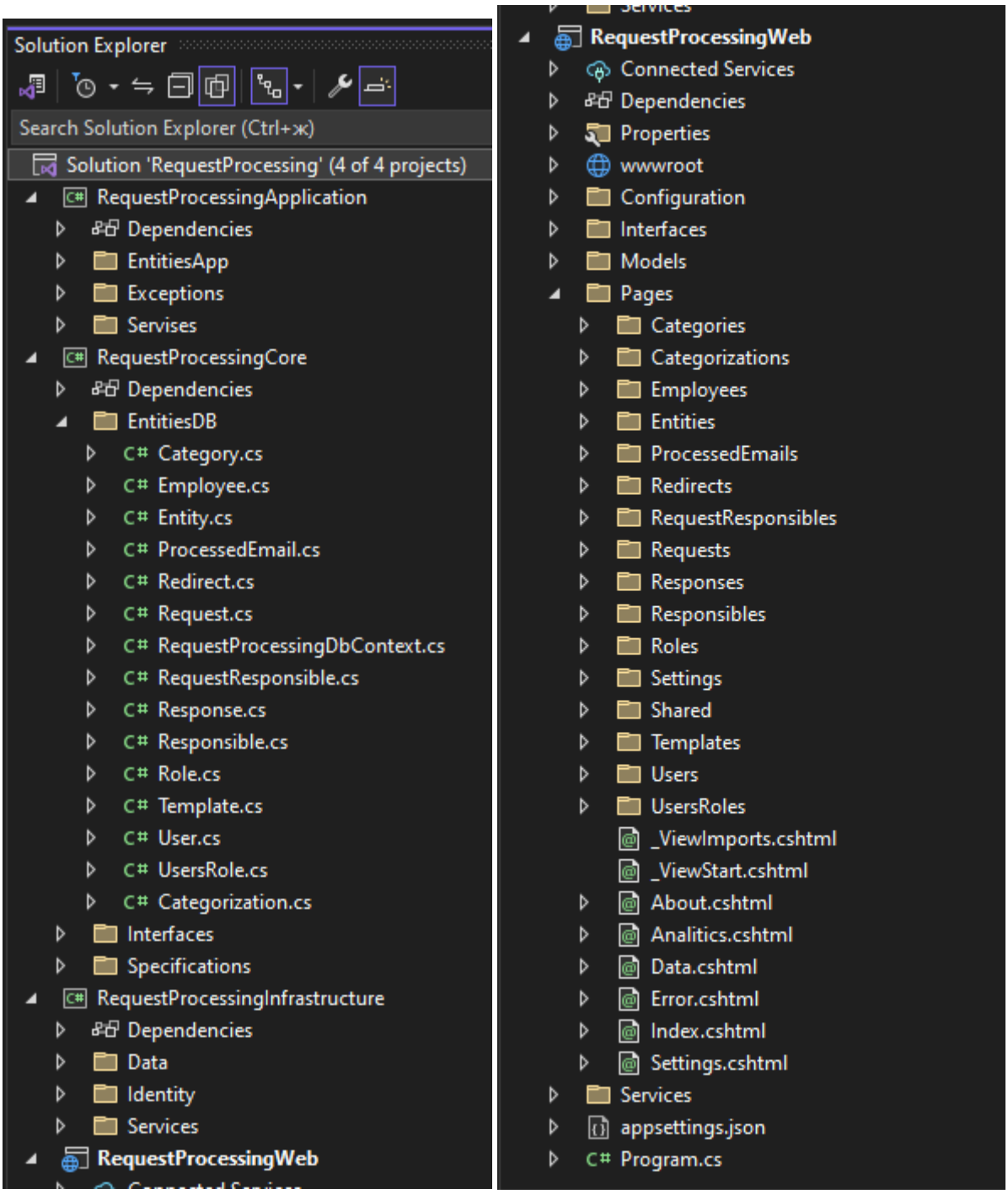


Рис. 2.6. Структура програмного проекту системи RequestProcessing

3 РОЗРОБКА МОДЕЛЕЙ, АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ ТА ПРОВЕДЕННЯ МОДЕЛЮВАННЯ

3.1. Модель електроного листа та запиту

Повідомлення електронної пошти структурно складається з двох частини: заголовок і тіло. Частина заголовка складається з набору полів, таких як «From», «To», «Copies», «Subject» та «Date». Різні постачальники послуг електронної пошти мають різні набори полів для відображення частини заголовка, проте поля «From», «To», «Subject» є загальними (див. рис.3.1).

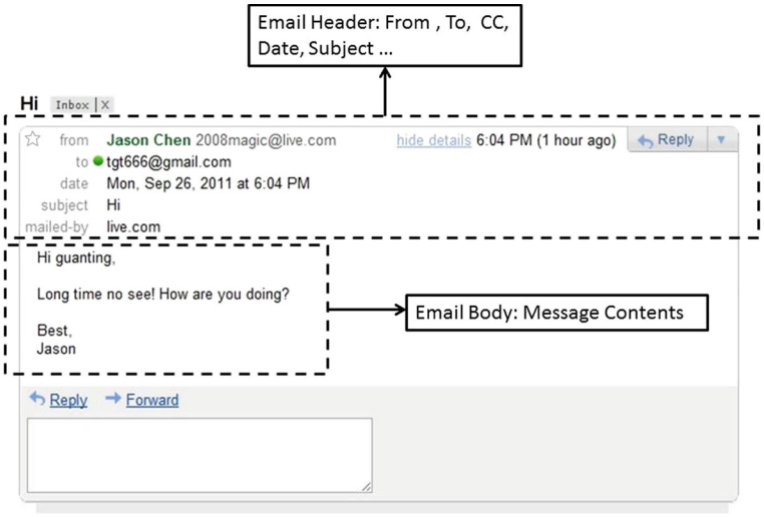


Рис. 3.1. Структура електроного листа

Зазвичай основна частина складається з неструктурованого тексту, іноді разом із графічними елементами, URL-посиланнями, HTML-розміткою та вкладеннями. Представлення даних електроного листа повинно відповідати методам, що будуть використовуватися для його обробки. Відповідне представлення краще служить на етапі аналізу даних, а також допомагає покращити результати завдання вилучення інформації. Представлення даних є

частиною загального етапу обробки даних – обробки «необроблених» (непрочитаних) електронних листів.

Використовуючи підхід на базі властивостей будемо уявляти електроний лист певним набором властивостей. Використаємо модель векторного простору, та представимо електронний лист у вигляді вектора, де вимірами вектору буде множина властивостей, отриманих із електронного листа.

Таки чином, модель електронного листа E_i з m властивостями буде описуватися вектором:

$$E_i = [w_{i1}, w_{i2}, \dots, w_{im}], \tag{3.1}$$

де E_i – електронний лист, $i = \overline{1,n}$;

w_{ij} – значення функції, $j = \overline{1,m}$;

n – кількість листів;

m – кількість властивостей.

У якості властивостей оберемо наступні:

1) відправник (електрона адреса), текстове значення з поля «From», що відповідає шаблону;

2) отримувач (електрона адреса), текстове значення з поля «To», що відповідає шаблону;

3) копія (перелік еклектроних адрес), текстове значення з поля «Copy», що відповідає шаблону;

4) дата, значення з поля «Date» у форматі дати-часу;

5) тема, текстове значення з поля «Subject»;

6) тіло, текстовий вміст тіла листа;

7) вкладення, бінарне значення (0 або 1);

8) посилання, бінарне значення (0 або 1);

Обробка таких данних як вкладення до листа, або посилання у тілі листа не розглядаються в данній роботі, але їх обробка є перспективою розвитку

атоматизованої системи, що розробляється. Тому, у якості властивості листа, будемо визначати двійковим значенням наявність чи відсутність таких данних.

Розроблена модель електроного листа буде використано для першого етапу обробки листів – обробка «необроблених» (непрочитаних) – збереження інформації, про отриманий лист у БД системи.

Для категоризації запитів користувачів, що надходять електронною поштою, будемо використовувати текстову інформацію, що міститься у електронному листі. Інформативними текстовими частинами електроного листа є тектовий вміст його теми – поле «Subject» та текстовий вміст основної частини – поля «Body». Використовуючі для категоризації запитів технології NLP, що дозволяють визнавати ключові фрази з фрагментів тексту, мусимо формувати змістовні фрагменти тексту з полів «Subject» та «Body». Оскільки кожен електронний лист може містити запит, що не є запитом до однієї категорії, доцільним буде проводити категоризацію за окремими реченнями, як певної мовної одиниці, що найчастіше містить одну завершену думку, фразу. Розбиття на речення надасть менше розподілення шуканих ключових фраз, ніж у повному тексті запиту. Приймемо за змістовну одиницю одне речення, тоді моделлю запиту користувача будемо вважати множину речень з полів «Subject» та «Body».

Таким чином, модель запиту користувача, що надходить електроним листом E_i буде описуватися множиною речень S_i:

$$\mathrm{S_i} = \{\mathrm{s_{is1}}, \mathrm{s_{is2}}, \dots, \mathrm{s_{isj}}, \dots \mathrm{s_{ism}}, \mathrm{s_{ib1}}, \mathrm{s_{ib2}}, \dots, \mathrm{s_{ibk}}, \dots, \mathrm{s_{ibp}}\}, \quad (3.2)$$

де $\mathrm{S_i}$ – запит користувача у електронному листі E_i (3.1), $i = \overline{1,n}$;

s_{isj} – речення з поля «Subject», $j = \overline{1,m}$;

s_{ibk} – речення з поля «Body», $k = \overline{1,p}$;

n – кількість листів;

m – кількість речень у полі «Subject»;

p – кількість речень у полі «Body».

Розроблена модель запиту користувача буде використано для другого етапу обробки листів – категоризація запиту користувача.

3.2. Модель категоризації запитів користувачів

Після попередньої обробки тексту електронних листів, що містять запити користувачів природньою мовою необхідно визначити їх відповідність до певних категорій. Признаками віднесення запиту до категорій є наявність в запитах певних ключових слів та/або ключових фраз. Для цього пропонується формування певної кількості категорій, обрання ключових слів та фраз та віднесення їх між собою. Оскільки метою автоматизації процесу класифікації запитів є оптимізація процесу формування відповідей на них, запропоновано розбиття категорій на дві группи – категорії запитів, що потребують шаблонної відповіді (запити, що до надання стандартної інформації, розкладів, переліків, тощо) та категорії запитів, відповіді на які потребують залучення фахівців у певній сфері питань та володіючих певною специфічною інформацією. Приклади переліку категорій та ключових слів або фраз наведено на рисунку 3.2. Таким чином введемо поняття множини категорій (3.3) та множини ключових фраз (3.4).

Множина категорій C, де N – кількість категорій:

$$\mathrm{C} = \{\mathrm{c}_1, \mathrm{c}_2 \dots \mathrm{c}_\mathrm{N}\}. \tag{3.3}$$

Множина ключових фраз L, де M – кількість ключових фраз:

$$L = \{l_1, l_2 \dots l_M\}. \tag{3.4}$$

Ключові фрази можуть відповідати різним категоріям, але мати різний рівень відношення до кожної з них. Тому, пропонується ввести числову характеристику рівня відповідності ключової фрази до категорії, ступені її впливу

на віднесення до категорії – ваговий коефіцієнт, що розподілено на інтервалі [0;1], де значення 1 показує, що якщо фраза присутня, запит точно відноситься до цієї категорії, а 0.5 – з вірогідністю 50%.

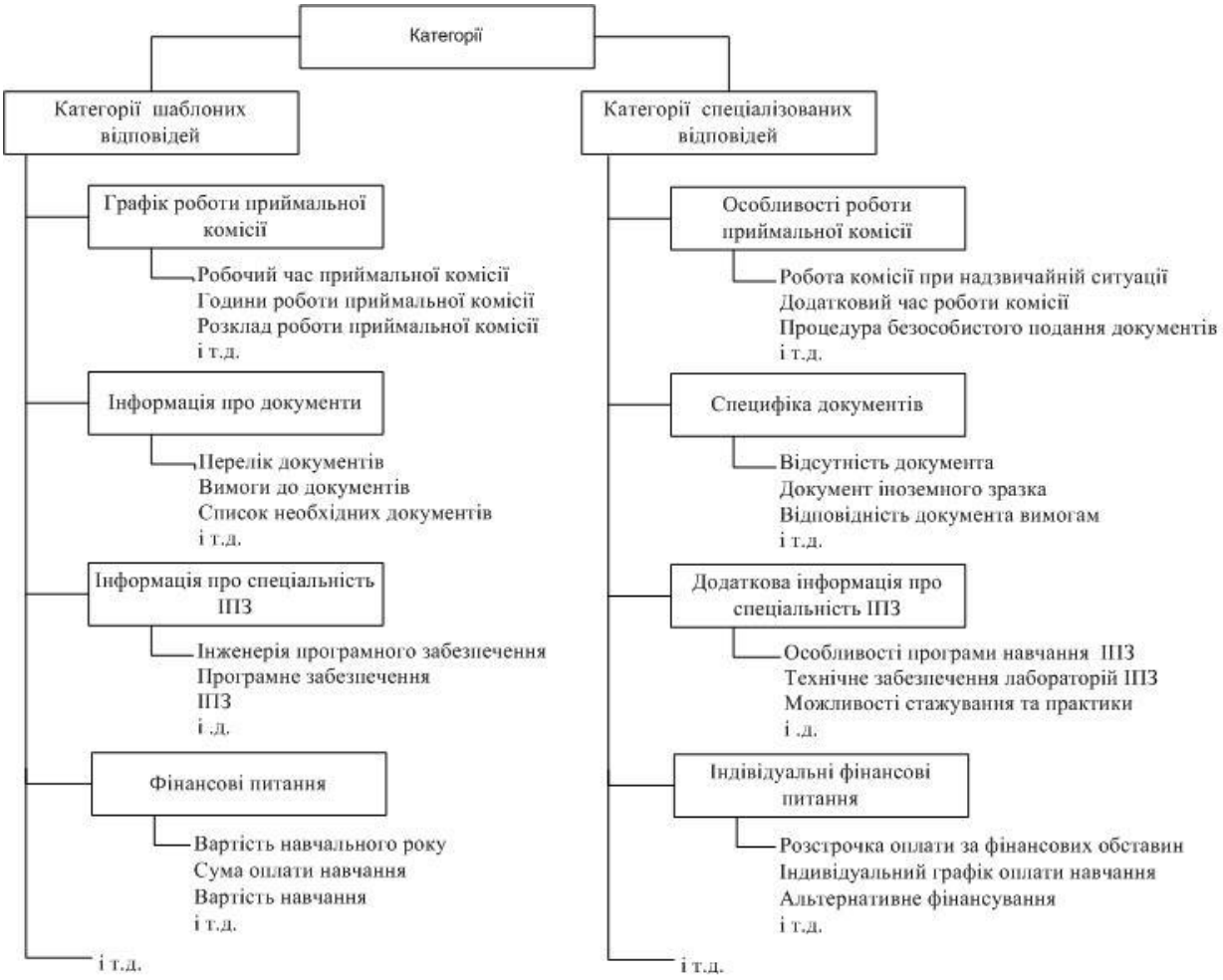


Рис. 3.2. Приклад формування категорій та ключових фраз для категоризації запитів

Введемо поняття ваговий коефіцієнт ключової фрази у категорії та опишемо множену W відносин категорії та ключових фраз з певним ваговим коефіцієнтом w_{ij} як:

$$W = \{(c_i, l_j, w_{ij}) | c_i \in C, l_j \in L\}. \quad (3.5)$$

Множену речень одного запиту S_i (3.2) опишемо як S, де P – кількість речень*:*

$$S = \{s_1, s_2 \dots s_P\}. \quad (3.6)$$

Використовуючі моделі NLP можливо отримати коефіцієнти схожості попередньо опрацьованих текстів речень до ключових фраз. Опишемо їх множиною Sim_k, де sim_{kij} – коефіцієнт схожості ключових фраз з *k* реченням:

$$Sim_k = \{sim_{kij} | \forall (s_k, c_i, l_j)\ c_i \in C, l_j \in L\}. \quad (3.7)$$

Важливо визначити не тільки перелік категорій, до яких відноситься запит, а й оцінити ступінь належності запиту до певної категорії. Для цього пропонується ввести поняття ступені належності речення до ключових фраз у категоріях, шляхом множення коефіцієнту схожості та вагового коефіцієнту. Таким чином отримуємо множену R_k ступенів належності речення *k* – го до лем у категоріях:

$$R_k = \{ w_{ij} \cdot sim_{kij} |\ w_{ij} \in W, sim_{kij}\ \in Sim\}. \quad (3.8)$$

Обираючи максимальні значення ступенів належності у кожній категорії – менші значення не будуть мати впливу, бо належать то той самої категорії – отримаємо вектор максимальних значень ступені належності речення *k* до категорії *i:*

$$M_k = \{max_i(r_{kij})| \in, r_{kij}\ \in R_k\}. \quad (3.9)$$

Обираючи номери категорій, максимальні значення яких перевищують певне порогове значення, отримаємо перелік категорій до яких відносимо речення

k, Вектор номерів категорій SC_k, де *limitValue* – порогове значення віднесення до категорії:

$$SC_k = \{ i \mid m_i > limitValue \}. \quad (3.10)$$

Маючі вектори категорій, до яких відноситься кожне речення запиту окремо, маємо можливість отримати вектор номерів категорій для усього запиту QC, об'єднуючі вектори категорій речень:

$$QC = \bigcup_{k=1}^{P} M_k. \quad (3.11)$$

Таким чином, було розроблено модель категоризації запитів користувачів, з урахуванням багатьох категорій та вагових коефіцієнтів ключових фраз у категоріях.

3.3. Модель предметної області

Дані запитів користувачів до та після обробки, результати категоризації та опрацювання електронних листів треба зберігати для подальшого їх аналізу. Для зберігання даних у системах використовуються БД, для побудови яких необхідно розробити концептуальну модель структури предметної області. На основі концептуальної моделі розробляється реляційна модель БД, визначаються таблиці та відношення між ними, визначаються типи даних для зберігання інформації, тощо. На основі реляційної моделі створюється фізична БД, використовуючи СКБД. Для реалізації розробленої реляційної моделі було використано СКБД Microsoft SQL Server 2019 та утиліту SQL Server Management Studio 19 (SSMS). Найменування таблиць та атрибутів, обрання властивостей та типів даних було виконано відповідно до вимог обраної СКБД. Структуру розробленої БД наведено на рисунку 3.3.

Рис. 3.3. Структура бази даних системи

База даних складається з наступних таблиць:

- Requests –зберігає інформацію про запити, що отримано електронною поштою (адреса відправника, адреса отримувача,адреси надсилання копій, час отримання листа,текст теми, текст вмісту, ознака наявності вкладень, ознака наявності посилань, векторне уявлення запиту, статус обробки, дата та час обробки);
- Responds – зберігає інформацію про листи з відповідями у текстових шаблонахих (ідентифікатор листа-запиту, зміст відповіді, статус обробки, дата та час обробки);
- Redirects – зберігає інформацію про листи, що требі перенаправити відповідальній особі (ідентифікатор листа-запиту, адреса пере направлення, статус обробки, дата та час обробки запису).
- Employees – зберігає інформацію про працівників, які призначені, або можуть бути призначені відповідальними особами за певними категоріями запитів (ПІБ співробітника, посада, адреса електронної скринькі, номер телефону, інша контактна інформація);
- Responsible – зберігає інформацію про відповідальних осіб, відношення категорій до назначених їм відповідальних осіб (ідентифікатор категорії, ідентифікатор співробітника, ознака активності запису);
- Categories – зберігає інформацію про категорії за якими категоризуються запити користувачів (назва категорії, опис категорії, ознака активності запису, ідентифікатор користувача, що модифікував запис, дата та час модифікації запису);
- Categorization – зберігає інформацію про результати категоризації, відношення запитів до певних категорій (ідентифікатор запиту, ідентифікатор категорії, до якої віднесено запит, статус обробки, дата та час обробки запису);
- Entities – зберігає інформацію про ключові фрази та їх вагу у відношенні до певної категорії (ідентифікатор категорії, текст ключової фрази, вага у категорії, ознка активності запису, ідентифікатор користувача, що модифікував запис, дата та час модифікації запису);

- Templates – зберігає інформацію що до шаблонів відповідей на запити до певних категорій, відношення шаблонів до категорій (ідентифікатор категорії, текст шаблону-відповіді, ознака активності запису, ідентифікатор користувача, що модифікував запис, дата та час модифікації запису);
- Users – зберігає інформацію, про користувачів системи (ідентифікатор користувача, логін, пароль, ім’я користувача, ознака активності запису);
- Roles – зберігає інформацію, що до ролей для розподілу доступу (ідентифікатор ролі, назва ролі, перелік дозволень);
- UsersRoles – зберігає інформацію, що до ролей користувачів для розподілу доступу до даних системи (ідентифікатор користувача, ідентифікатор ролі);
- ProcessedEmails – зберігає інформацію, що до електроних скриньок, листи з яких обробляються системою, та доступу до них (одреса електронної скриньки, логін, пароль, поштовий сервер, ознака активності запису);
- Tasks – зберігає інформацію, що до сервісів обробки листів та розкладу їх роботи (ідентифікатор задачі, ім’я сервісу, дата та час початку роботи за розкладом, інтервал запуску, ознака активності задачі).

3.4. Алгоритмічне забезпечення системи

Модуль обробки тексту запиту TextProcessing. Виконує попередню обробку тексту листів, оцінку за ключовими словами та фразами, та визначає категорії до яких відноситься запит. Узагальнену блок-схему категоризації запиту наведено на рисунку 3.4.

Записи у таблицях Requests, Responds та Redirects мають поле Status, значення 0 якого позначає, що запис не оброблено, а значення 1 – оброблено. Таким чином контролюється обробка усіх записів, та навіть у разі виникнення помилки під час опрацювання – після відновлення роботи системи запис буде оброблено.

Рис. 3.4.Узагальнена блок-схема категоризації запиту

Оглядяються усі записи з таблиці Requests, що мають статус 0 (не оброблено) та вилучається текст запиту. Для тексту кожного запиту виконується

поділ на речення, кожне печення попередньо обробляється (переведення усіх символів у нижній регістр, токенізація, лематизація, фільтрація стоп-слів і т.д. виконується засобами NLP мови Python), обчислюється вектор ознак (засобами NLP мови Python), ступінь належності до категорії та перелік категорій до яких носимо речення. Після обробки усіх речень категорії речень об’єднуються у один перелік – перелік категорій запиту. До таблиці категоризації (Categorization) додаються записи з індексом запиту та індексами категорій зі статусом 0 (не оброблено). У подальшому вони обробляються модулем генерації відповідей ResponseGeneration, а після обробки, формування тексту відповіді на запит, або визначення, що лист треба пере направити, модуль ResponseGeneration встановлює статус запиту 1 (оброблено).

Основні функції реалізації сервісу категоризації запитів наведено у додаткуВ.1.

Модуль формування тексту відповіді ResponseGeneration. Відповідає за вилучення даних для відповіді та формування тексту листа-відповіді. Узагальнену блок-схему формування модулем відповідей наведено на рисунку 3.5.

Обробляються усі записи – результати категоризації запиту з таблиці Categorization, що мають статус 0 (не оброблені). Записи групуються за ідентифікатором запиту з метою послідовної обробки категорій одного запиту та завершення роботи з ним. Для кожного запиту визначається перелік категорій. Категорії переглядаються і для кожної категорії, якщо категорія відноситься до категорії з шаблоном відповіді (у таблиці Template є запис для цієї категорії), до тексту відповіді додається відповідний шаблон.

Якщо категорія відноситься до категорій, де відповідь повинна надавати призначена особа (у таблиці відповідальних осіб Responsible є встановлена для цієї категорії особа), інформація про запит та адреса електронної скриньки відповідальної особи заносяться у таблицю Redirects зі статусом 0 (не оброблено) для подальшого перенаправлення запиту. Після обробки запису з таблиці Categorization для нього встановлюється статус 1 (категорію запиту оброблено) та розглядається наступна категорія.

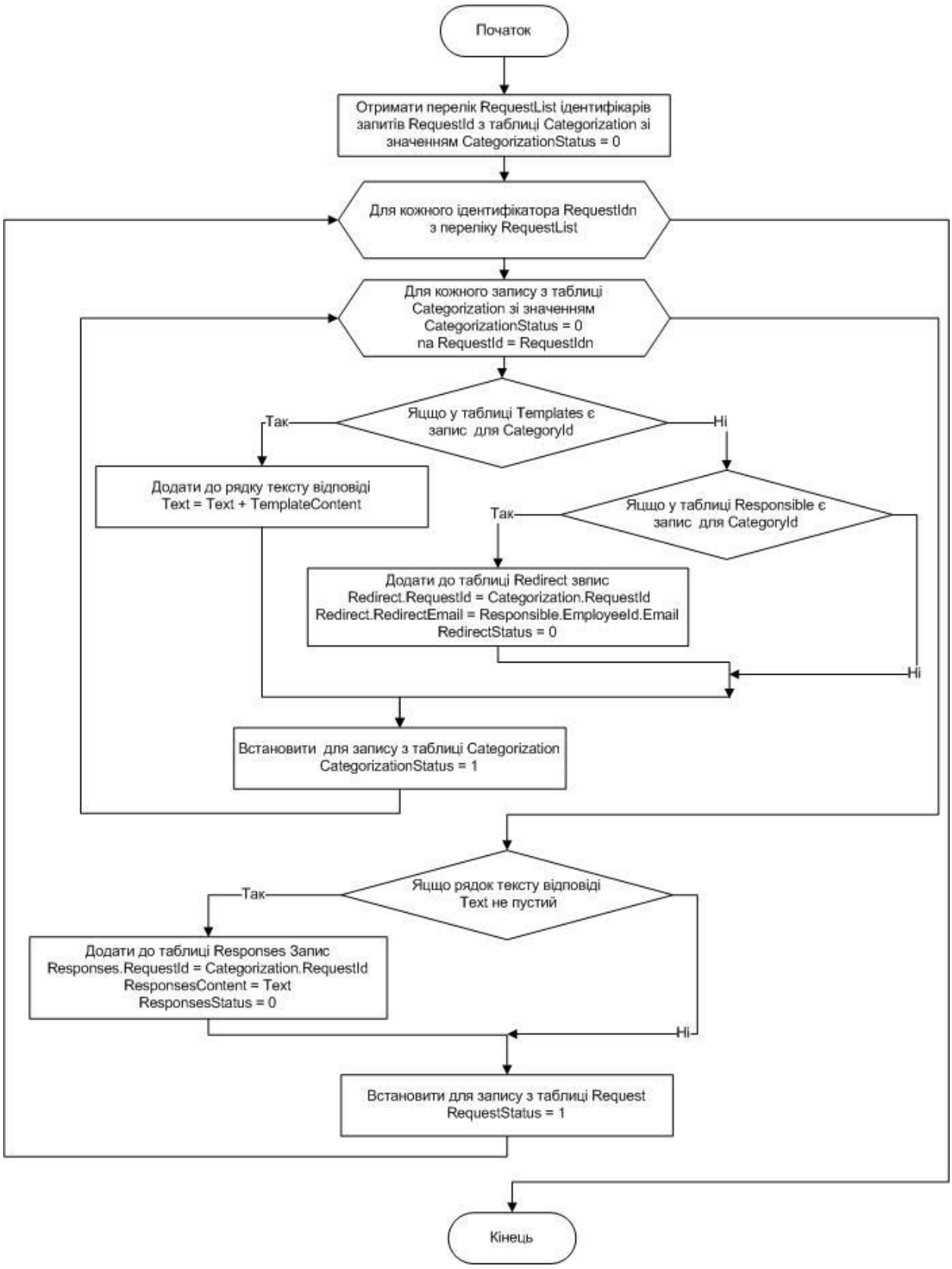


Рис. 3.5. Узагальнена блок-схема формування відповідей

Після огляду усіх категорій запиту, запису у таблиці запитів Requests встановлюється статус 1 (запит оброблено).

Основні функції реалізації сервісу формування відповідей наведено у додатку В.2.

Отримання електроних листів, відправка відповідей та перенаправлення запитів виконуються модулем взаємодії з електронною поштою EmailInteraction.

Сервіс працює у трьох режимах:

- обробка вхідних листів, формування та збереження моделі електронного листа;
- перенаправлення листів;
- відправка листа-відповіді.

На рисунку 3.6 наведено узагальнену блок-схему функціонування сервісу у режимі обробки вхідних листів.

Для роботи з вхідними сповіщеннями електронної пошти у мові Python використовувалася бібліотека imaplib, що надає базовий інтерфейс доступу за стандартним протоколом для отримання доступу до електронної пошти на віддаленому сервері IMAP (Internet Message Access Protocol). При роботі з вхідними сповіщеннями використовувалися наступні основні інструменти:

- mail = imaplib.IMAP4_SSL(«servername») – створення об'єкта IMAP4;
- mail.login(«login», «password») – аутентифікація на поштовому сервері;
- mail.select(«INBOX») – обрання папки з вхідними сповіщеннями;
- status, messages = mail.search(None, «(UNSEEN)») – отримання переліку непрочитаних сповіщень;
- _, msg_data = mail.fetch(message_id, «(RFC822)») – отримання даних конкретного сповіщення, параметр (RFC822) вказує на формат, що потрібна повністю уся інформація;
- mail.logout() – закриття з’єднання з сервером.

Рис. 3.6. Узагальнена блок-схема функціонування сервісу взаємодії з електронною поштою у режимі обробки вхідних листів

На рисунку 3.7 наведено узагальнену блок-схему функціонування сервісу у режимі перенаправлення листів.

Для роботи з відправкою та перенаправленням сповіщень електронною поштою у мові Python використовувалася бібліотека smtplib, що надає базовий інтерфейс доступу за стандартним протоколом для відправлення електронної пошти SMTP (Simple Mail Transfer Protocol). При роботі з відправкою сповіщень використовувалися наступні основні інструменти:

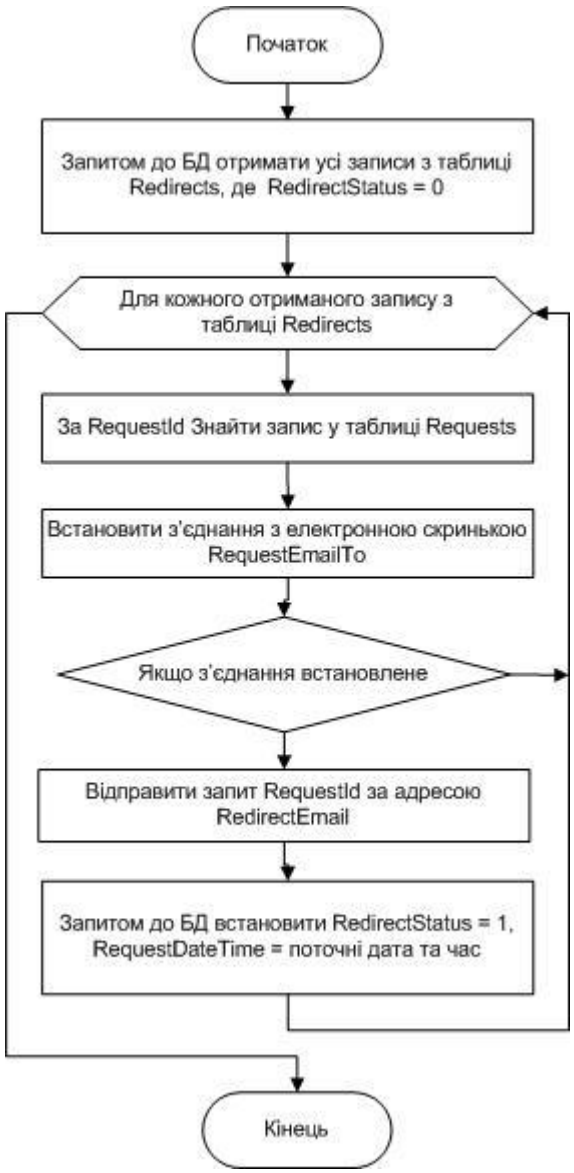


Рис. 3.7. Узагальнена блок-схема функціонування сервісу взаємодії з електронною поштою у режимі перенаправлення листів

− server = smtplib.SMTP(«serveraddress», 587) – створення об'єкта SMTP;

− server.login(«login», «password») – аутентифікація на поштовому сервері;

− server.sendmail(«senderemail», «recipientemail», «message») – відправлення електронного листа;

− server.quit() – завершення з'єднання з сервером.

На рисунку 3.8 наведено узагальнену блок-схему функціонування сервісу у режимі відправки листа-відповіді.

Рис. 3.8. Узагальнена блок-схема функціонування сервісу взаємодії з електронною поштою у режимі відправки відповіді на запит

Основні функції реалізації сервісу взаємодії з електроною поштою наведено у додатку В.3 – В.5.

3.5. Проведення моделювання та аналіз отриманих результатів

Моделювання процесу категоризації запитів користувачів виконувалося за допомогою трьох багатомовних преднавчаних моделей: BERT, XLNet та XLMRoberta, що використовують різні моделі токенайзерів для розбиття тексту на токени:

− модель BERT використовує токенізацію WordPiece, що розбиває текст на слова та підслова;

− модель XLNet використовує токенізацію SentencePiece;

− модель XLMRoberta використовує токенізацію BPE (Byte Pair Encoding).

Моделювання проводилося з наступними значеннями у моделі категоризації.

Множини речень S (3.6):

$$S_1 = \{s_1, s_2 \dots s_{10}\},$$
$$S_2 = \{s_1, s_2 \dots s_{30}\},$$
$$S_3 = \{s_1, s_2 \dots s_{100}\}.$$

Множина категорій C (3.3):

$$C = \{c_1, c_2, c_3\}.$$

Множина ключових фраз L (3.4):

$$L = \{l_1, l_2, l_3\}.$$

Множина вагових коефіцієнтів ключових фраз W (3.5):

$$W = \{(w_{11}, C_1, L_1), (w_{22}, C_2, L_2), (w_{33}, C_3, L_3)\}$$

$$= \{(0.95, C_1, L_1), (0.5, C_2, L_2), (0.1, C_3, L_3)\}.$$

Обрані вагові коефіцієнти відповідно до своїх категорій показують, що:

- *(0.95,C_1L_1)* – вірогідність належності до категорії 95%;
- *(0.5,C_2L_2)* – вірогідність належності до категорії 50%;
- *(0,1,C_3L_3)* – вірогідність належності до категорії 10%.

Значення w_{33} є несуттєво малим для використання фрази C_3, L_3 у якості ключової для категоризації, фраза L_3не буде зберігатися у БД автоматизованої системи у якості ключової до категорії C_3, однак значення додано до моделювання для отримання результатів на більшому діапазоні можливих значень вагових коефіцієнтів.

Обрані ключові фрази мають різний ступень відповідності до груп запитів, а саме:

- C_1L_1 – повністю відповідає запиту;
- C_2L_2 – частково відповідає запиту;
- C_3L_3– абсолютно не відповідає запиту.

У процессі моделювання кожною моделлю окремо було токенізовано відповідними токенізаторами набори речень та ключові фрази та отримано їх токенізовані уявлення у вигляді числових індексів, що відповідають токенам, виявленим токенізатором.

Для отримання значень схожості токенезованих уявлень використовувалися наступні показники, які обраховувалися :

- косинусна схожість – методом cosine_similarity() бібліотеки scikit-learn;
- коефіцієнт кореляції – методом np.corrcoef() бібліотеки numpy;
- евклідова відстань – методом euclidean() бібліотеки scipy.

Отримані середні значення для моделей та множин речень наведено у таблиці 3.1.

Таблиця 3.1.

Коефіцієнти схожості *Sim* (3.7)

Кількість запитів	Мовні моделі	Косінусна схожість			Коефіцієнт кореляції			Евклідова відстань		
		C_1L_1	C_2L_2	C_3L_3	C_1L_1	C_2L_2	C_3L_3	C_1L_1	C_2L_2	C_3L_3
N = 10	BERT	0.900	0.710	0.450	0.900	0.710	0.450	4.76	8.05	11.33
	XLNet	0.990	0.980	0.870	0.990	0.980	0.870	12.68	50.98	45.60
	XLMRoberta	0.997	0.990	0.952	0.997	0.990	0.952	1.328	2.57	2.24
N = 30	BERT	0.908	0.690	0.447	0.908	0.690	0.447	4.77	8.43	11.01
	XLNet	0.980	0.977	0.865	0.980	0.977	0.865	11.86	46.82	44.02
	XLMRoberta	0.995	0.989	0.950	0.995	0.989	0.950	1.201	2.40	2.03
N = 100	BERT	0.920	0.675	0.437	0.920	0.675	0.437	5.68	9.24	10.50
	XLNet	0.979	0.976	0.860	0.979	0.976	0.860	11.48	46.04	43.15
	XLMRoberta	0.995	0.985	0.947	0.995	0.985	0.947	1.201	2.31	2.02

За отриманими результатами можно зробити наступні висновки:

− косінусна схожість та коефіцієнт кореляції надають однакові результати – метрики вимірюють різні показники подібності подібность між двома векторами та залежність від значеннями, що вказує на специфічну побудову токенизованих даних та є підставою для подальших досліджень;

− евклідова відстань не обмежена значеннями, як косінусна схожість [-1;1], тож для використання у розробленній моделі не підходить, або потребує додаткових оцінок та обробок.

Для розрахунку ступені належності запитів до категорій використано отримані значення косинусної схожісті та задані вагові коефіцієнти, отримані результати наведено у таблиці 3.2. Для визначення належності для категорій було обрано два порогових значення 2 значення *limitValue*: 0,45 та 0,5. У таблиці 2, ступені належності, що перевищують обрані порогові значення видылено сірим фоном комірок.

Таблиця 3.2.

Ступені належності R (3.8)

Кількість запитів	Мовні моделі	limitValue = 0,45			limitValue = 0,5		
		C_1L_1	C_2L_2	C_3L_3	C_1L_1	C_2L_2	C_3L_3
N = 10	BERT	0,86	0,36	0,05	0,86	0,36	0,05
	XLNet	0,94	0,49	0,09	0,94	0,49	0,09
	XLMRoberta	0,95	0,50	0,10	0,95	0,50	0,10
N = 30	BERT	0,86	0,35	0,04	0,86	0,35	0,04
	XLNet	0,93	0,49	0,09	0,93	0,49	0,09
	XLMRoberta	0,95	0,49	0,10	0,95	0,49	0,10
N = 100	BERT	0,87	0,34	0,04	0,87	0,34	0,04
	XLNet	0,93	0,49	0,09	0,93	0,49	0,09
	XLMRoberta	0,95	0,49	0,09	0,95	0,49	0,09

За результатами моделюваня можна зробити висновки, що модель BERT найбільш адекватно відображає результат категоризації, а доволі високе значення для абсолютно невідповідної категорії може бути скореговано розробленою моделлю за рахунок налаштування ваг ключових фраз та порогового значення *limitValue*.

ВИСНОВКИ

В результаті виконання магістерської роботи було розроблено методику автоматизованиї обробки запитів користувачів електронною поштою з використанням технологій обробки природньої мови.

При виконанні роботи було виконано наступні задачі.

1.Виконано огляд та проведено аналіз сучасних методів та засобів обробки природньої мови. Аналіз показав, що існують різноманітні підходи, включаючи методи машинного навчання, глибокого навчання та класичні методи обробки природньої мови. Визначено переваги та недоліки кожного підходу з точки зору застосування в системі автоматизованої обробки запитів користувачів електронною поштою.

2. Розроблено архітектуру системи автоматизованої обробки вхідних листів електронної скриньки. Визначено ключові компоненти системи та вибрані технології для їх реалізації: мова програмування Python, бібліотеки spaCy, NLTK та моделі трансформери для обробки природньої мови, pyodbc для роботи з базою даних MS SQL, imaplib, smtplib для роботи з електроною поштою, SQL Server 2019 та інтегроване середовище Management Studio 19 для керування базою даних, технологія веб-сервісів ASP.NET для користувацького інтерфейсу з базою даних, засоби середовища розробки Microsoft Visual Studio 2022. Обрані технології забезпечують швидку та достатньо точну обробку тексту природньою мовою, а також ефективне керування базою даних.

3. Розроблено математичну модель категоризації запитів, що враховує вагу певних ключових фраз при оцінюванні належності речень запиту до певної категорії. Модель базується на статистичних аспектах, що робить її придатною для широкого спектру вхідних текстів.

4. Розроблено алгоритми, що дозволяють категоризувати текст запитів користувачів з використанням NLP-технології та розробленої моделі, а також формування тексту відповідей з набору шаблонів та перенаправлення запитів на електрону пошту відповідної відповідальної особи.

5. Проведено моделювання роботи розроблених засобів та алгоритмів. Оцінено ефективність використання автоматизованої системи обробки запитів користувачів та використання для цього технологій NLP.

ПЕРЕЛІК ПОСИЛАНЬ

1. G. Mujtaba, L. Shuib, R. G. Raj, N. Majeed and M. A. Al-Garadi, "Email Classification Research Trends: Review and Open Issues," in IEEE Access, vol. 5, pp. 9044-9064, 2017, doi: 10.1109/ACCESS.2017.2702187.

2. Alsmadi I. and Alhami, I. 2015. Clustering and classification of email contents. Journal of King Saud University-Computer and Information Sciences, 27(1), 46–57.

3. Katakis, Ioannis & Tsoumakas, Grigorios & Vlahavas, I.. (2006). E-mail Mining: Emerging Techniques for E-Mail Management. 10.4018/978-1-59904-228-2.ch010.

4. Aery, Manu & Chakravarthy, Sharma. (2004). eMailSift: mining-based approaches to email classification. 580-581. 10.1145/1008992.1009130.

5. Kushmerick, Nicholas and Tessa A. Lau. Automated email activity management: an unsupervised learning approach. Proceedings of the 10th international conference on Intelligent user interfaces (2005): 67-74.

6. Schuff, David & Turetken, Ozgur & D'Arcy, John & Croson, David. (2007). Managing E-Mail Overload: Solutions and Future Challenges. Computer. 40. 31 - 36. 10.1109/MC.2007.65.

7. Chailiornkaew, li., lirexawanlirasut, T. &amli; McAleer, M. (2017). You’ve got email: A workflow management extraction system. Journal of Reviews on Global Economics, 6, 342-349.

8. Park S. and An D. U. 2010. Automatic e-mail classification using dynamic category hierarchy and semantic features. IETE Technical Review, 27(6), 478–492.

9. Sakurai and A. Suyama. 2005. An e-mail analysis method based on text mining techniques. Appl. Soft Comput. 6, 1 (November, 2005), 62–71. https://doi.org/10.1016/j.asoc.2004.10.007.

10. Aloui, Awatef & Neji, Mahmoud. (2010). Automatic Classification and Response of E-mails. International Journal for Digital Society. 1. 10.20533/ijds.2040.2570.2010.0001.

11. Sharaff A. and Nagwani N. K. 2019. Identifying Categorical Terms Based on Latent Dirichlet Allocation for Email Categorization. In Emerging Technologies in Data Mining and Information Security (pp. 431–437). Singapore: Springer.

12. Abu-Nimeh, Saeed & Nappa, Dario & Wang, Xinlei & Nair, Suku. (2007). A comparison of machine learning techniques for phishing detection. ACM International Conference Proceeding Series. 269. 60-69. 10.1145/1299015.1299021.

13. A. Almomani, B. B. Gupta, S. Atawneh, A. Meulenberg and E. Almomani, "A Survey of Phishing Email Filtering Techniques" in IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2070-2090, Fourth Quarter 2013, doi: 10.1109/SURV.2013.030713.00020.

14. Young, Tom, Devamanyu Hazarika, Soujanya Poria and E. Cambria. “Recent Trends in Deep Learning Based Natural Language Processing.” IEEE Comput. Intell. Mag. 13 (2017): 55-75.

15. Baharudin, Baharum & Lee, Lam Hong & Khan, Khairullah & Khan, Aurangzeb. (2010). A Review of Machine Learning Algorithms for Text-Documents Classification. Journal of Advances in Information Technology. 1. 10.4304/jait.1.1.4-20.

16. J. Clark, I. Koprinska and J. Poon A neural network based approach to automated e-mail classification. Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003), Halifax, NS, Canada, 2003, pp. 702-705, doi: 10.1109/WI.2003.1241300.

17. Mitchell T. Machine leagning. Chapter 3 Generative and discriminative classifiers: Naive Bayes and logistic regression / McGraw-Hill Science/Engeneering/Math? 1997 – 431p.

18. Blanzieri, Enrico & Bryl, Anton. (2008). A Survey of Learning-Based Techniques of Email Spam Filtering. Artificial Intelligence Review. 29. 10.1007/s10462-009-9109-6.

19. Email Classification Suite. [Електронний ресурс] / Scalifi Ai [Веб-сайт]. – Режим доступу: https://www.scalifiai.com/email-classification-suite.

20. The efficiency of a help desk with the familiarity of email. [Електронний ресурс] / Front [Веб-сайт]. – Режим доступу: https://front.com/.

21. Tag incoming emails with AI Builder. [Електронний ресурс] / Microsoft [Веб-сайт]. – Режим доступу: https://powerautomate-microsoft-com.translate.goog/en-us/blog/tag-incoming-emails-with-ai-builder/?_x_tr_sl=en&_x_tr_tl=uk&_x_tr_hl=uk&_x_tr_pto=wapp.

22. Easily convert your designs . [Електронний ресурс] / Mail-Cat.cc [Веб-сайт]. – Режим доступу:into compliant HTML emails https://mail-cat.cc/

23. Kulkarni A., Shivananda A., Natural Language Processing Recipes. Apress, 2019. — 253 p. [Електронний ресурс] / Ebooksworld [Веб-сайт]. – Режим доступу: https://dl.ebooksworld.ir/motoman/Natural.Language.Processing.Recipes. www.EBooksWorld.ir.pdf.

24. Apache OpenNLP Developer Documentation [Електронний ресурс] / OpenNLP [Веб-сайт]. – Режим доступу: https://opennlp.apache.org/docs/1.9.1 /manual/opennlp.html.

25. Cognitive Services. Text Analytics API (v2.0) [Електронний ресурс] / Microsoft [Веб-сайт]. – Режим доступу: https://westus.dev.cognitive .microsoft.com/docs/services/TextAnalytics.V2.0/operations/56f30ceeeda5650db055a3c7.

26. Cloud Skills Challenge. What is Azure AI Language? [Електронний ресурс] / Microsoft [Веб-сайт]. – Режим доступу: https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/overview.

27. IDOL Unstructured Data Analytics [Електронний ресурс] / OpenText [Веб-сайт]. – Режим доступу: https://www.microfocus.com/en-us/products/information-data-analytics-idol/overview.

28. IDOL Data Sheet [Електронний ресурс] / OpenText [Веб-сайт]. – Режим доступу: https://www.microfocus.com/media/data-sheet/idol_ds.pdf.

29. Transform Complex Text Documents into Data, Insights, & Value [Електронний ресурс] / Lexalytics [Веб-сайт]. – Режим доступу:: https://www.lexalytics.com.

30. Semantria API documentation [Електронний ресурс] / Lexalytics [Веб-сайт]. – Режим доступу: https://semantria-docs.lexalytics.com/docs.

31. No-Code Text Analytics [Електронний ресурс] / MonkeyLearn [Веб-сайт]. – Режим доступу: https://monkeylearn.com.

32. MonkeyLearn Api Documentation [Електронний ресурс] / MonkeyLearn [Веб-сайт]. – Режим доступу: https://monkeylearn.com/api/v3/

33. NeticleText Analysis API [Електронний ресурс] / Neticle [Веб-сайт]. – Режим доступу: https://neticle.com/textanalysisapi/en/.

34. Natural Language API Basics [Електронний ресурс] / GoogleCloud [Веб-сайт]. – Режим доступу: https://cloud.google.com/natural-language/docs/basics.

35. Saga Natural Language Understanding (NLU) Framework [Електронний ресурс] / Accenture [Веб-сайт]. – Режим доступу: https://www.accenture.com/us-en/services/applied-intelligence/saga-natural-language-understanding.

36. Natural Language Processing: A Guide to NLP Use Cases, Approaches, and Tools [Електронний ресурс] / AlexSoft [Веб-сайт]. – Режим доступу: https://www.altexsoft.com/blog/natural-language-processing/.

37. Documentation. Natural Language Toolkit [Електронний ресурс] / NLTK [Веб-сайт]. – Режим доступу: https://www.nltk.org/].

38. Industrial-Strength Natural Language Processing [Електронний ресурс] / spaCy [Веб-сайт]. – Режим доступу: https://spacy.io/.

39. The fundamental package for scientific computing with Python [Електронний ресурс] NumPy [Веб-сайт]. – Режим доступу: https://numpy.org/.

40. Hugging FaceTransformers [Електронний ресурс] / Hugging Face [Веб-сайт]. – Режим доступу: https://huggingface.co/docs/transformers/index.

41. Create production-grade machine learning models with TensorFlow [Електронний ресурс] / TensorFlow [Веб-сайт]. – Режим доступу: https://www.tensorflow.org/.

42. Pytorch. New announcements [Електронний ресурс] / Pytorch [Веб-сайт]. – Режим доступу: https://pytorch.org/.

43. Hugging Face.Tokenizers [Електронний ресурс] / Hugging Face [Веб-сайт]. – Режим доступу: https://huggingface.co/docs/tokenizers/index.

44. 20 Beautiful Soup: Build a Web Scraper With Python [Електронний ресурс] / Pytorch [Веб-сайт]. – Режим доступу: https://realpython.com/beautiful-soup-web-scraper-python/].

45. Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing [Електронний ресурс] / Google AI Blog [Веб-сайт]. – Режим доступу: https://web.archive.org/web/20210113211449/https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html.

46. GPT model [Електронний ресурс] / OpenAI [Веб-сайт]. – Режим доступу: https://platform.openai.com/docs/models.

47. Yang Z., Dai Z., Yang Y., Carbonell J., Salakhutdinov R., Quoc V. Le XLNet: Generalized Autoregressive Pretraining for Language Understanding. Advances in Neural Information Processing Systems 32 (NeurIPS 2019).

48. VaderSentiment's documentation [Електронний ресурс] / VaderSentiment [Веб-сайт]. – Режим доступу: https://vadersentiment.readthedocs.io/en/latest/.

49. Simplified Text Processing [Електронний ресурс] / TextBlob [Веб-сайт]. – Режим доступу: https://textblob.readthedocs.io/en/dev/.

50. Stanford Named Entity Recognizer (NER) [Електронний ресурс] / The standart Natural Processing Group [Веб-сайт]. – Режим доступу: https://nlp.stanford.edu/software/CRF-NER.shtml.

51. A New Generation of AI Assistants [Електронний ресурс] / Rasa [Веб-сайт]. – Режим доступу: https://rasa.com/.

52. Amazon Comprehend [Електронний ресурс] / AWS [Веб-сайт]. – Режим доступу: https://aws.amazon.com/comprehend/?nc1=h_ls.

53. IBM Watson Natural Language Understanding [Електронний ресурс] / IBM [Веб-сайт]. – Режим доступу: https://www.ibm.com/products/natural-language-understanding.

54. Tutorial: Text Analytics with Azure AI services [Електронний ресурс] / Microsoft [Веб-сайт]. – Режим доступу: https://learn.microsoft.com/en-us/azure/synapse-analytics/machine-learning/tutorial-text-analytics-use-mmlspark.

55. BPMN 2.0 – Модель і нотація бізнес процесів [Електронний ресурс] / BPM Offensive Berlin [Веб-сайт]. – Режим доступу: http://www.bpmb.de/images/BPMN2_0_Poster_UA.pdf.

56. Microsoft Build. Розробка додатків служби Windows [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/ru-ru/dotnet/framework/windows-services/

57. Microsoft Build. Загальні архітектури веб-додатків [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures.

58. Ярцев В.П. Організація баз даних та знань: навчальний посібник. / В.П.Ярцев. – К. ДУТ 2018.-214с

59. David M. K. Теория и практика построения баз данных / M. Kroenke David., 2005. 8. Що таке NoSQL [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://aws.amazon.com/ru/nosql/

60. What is NoSQL? [Електронний ресурс] – Режим доступу: https://aws.amazon.com/nosql/?nc1=h_ls.

61. Kyte T. Expert Oracle Database Architecture [Електронний ресурс] – Режим доступу: https://javidhasanov.files.wordpress.com/2012/01/expert-oracle-database-architecture.pdf.

62. What is PostgreSQL? [Електронний ресурс]. – Режим доступу до ресурсу: https://www.postgresql.org/about/.

63. Petkovic D. Microsoft SQL Server 2019. A Beginer’s Guide [Електронний ресурс] – Режим доступу: https://dl1.newoutlook.it/book/2020/03/Microsoft-SQL-Server-2019-A-Beginners-Guide.pdf

64. Bird S., Klein E., Loper E. Natural Language Processing with Python [Електронний ресурс]. – Режим доступу: https://www.researchgate.net /publication/220691633_Natural_Language_Processing_with_Python.

65. Top 5 languages for natural language processing [Електронний ресурс]. – Режим доступу: https://botpenguin.com/top-5-languages-for-natural-language-processing/

66. NLP Libraries for Node.js and JavaScript [Електронний ресурс] – Режим доступу: https://dev.to/devashishmamgain/nlp-libraries-for-node-js-and-javascript-1ja4

67. 9 Best Programming Languages for AI [2023 Project Guide] [Електронний ресурс]. – Режим доступу:: https://www.springboard.com/blog/data-science/best-programming-language-for-ai/.

68. Вахнюк, С.В. Технологія створення програмних та інтелектуальних систем [Текст] : навчальний посібник / С. В. Вахнюк. – Суми : ДВНЗ «УАБС НБУ», 2011. – 254с.

69. Клієнт-серверна архітектура. QATestLab training center [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/

70. ASP.NET documentation [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА
«РОЗРОБКА МЕТОДИКИ ОБРОБКИ ЗАПИТІВ КОРИСТУВАЧІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ NLP»

Виконав: студент 6 курсу, групи ПДМ – 61 Горячев Тимур Вікторович

Керівник: д.т.н., проф., професор кафедри ІПЗ Ільїн Олег Юрійович

Київ - 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення ефективності процесу автоматизованої обробки текстових запитів користувачів з використанням технологій Natural Language Processing (NLP).

Об'єкт дослідження: процес обробки текстових запитів користувачів.

Предмет дослідження: методи та засоби обробки текстових запитів користувачів природньою мовою.

СУЧАСНІ РІШЕННЯ ОБРОБКИ ПРИРОДНЬОЇ МОВИ

3

Системи обробки природньої мови

1. Google Cloud Natural Language API
2. Saga Natural Language Understanding (NLU)
3. Neticle Text Analysis API
4. Lexalytics Intelligence Platform
5. MonkeyLean
6. Apache OpenNLP
7. Microsoft Text Analytics API
8. OpenTextIDOL Unstructured Data Analytics

Мови програмування та бібліотеки NLP

1. Python (TextBlob, SpaCy, NLTK, Genism, PyNLPl, Transformers, scikit-learn)
2. Java (Apache OpenNLP, Stanford NLP, LingPipe)
3. JavaScript (Natural, Compromise, NLP.js)
4. Ruby (Ruby Linguistics, Text, Stanford CoreNLP Ruby)
5. C# (OpenNLP Sharp, SharpNLP)
6. R (tm, quanteda, openNLP)
7. Go (Go NLP, spaGO)
8. Scala (Stanford CoreNLP)

Основні компоненти систем

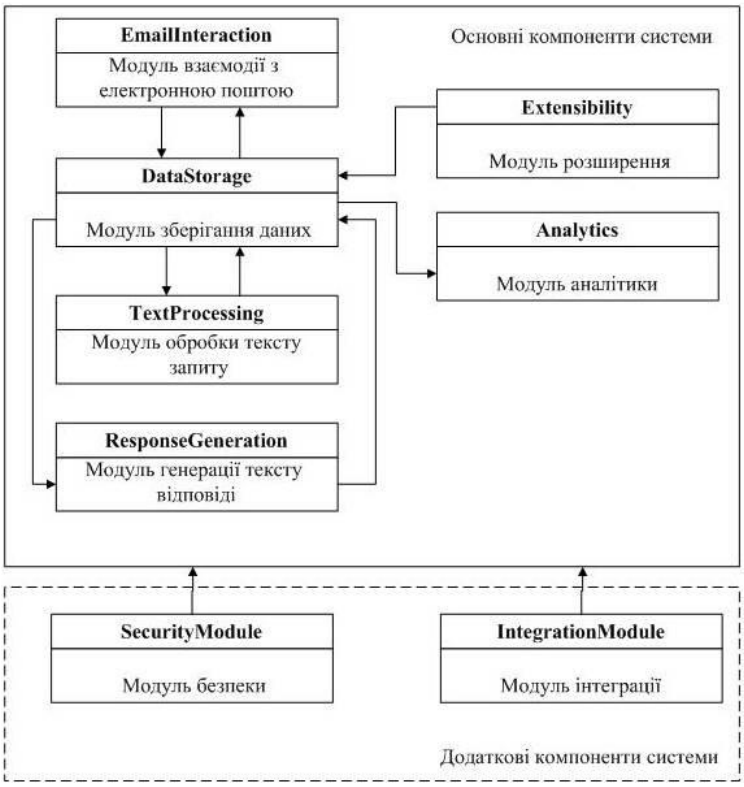
1. Визначення мови
2. Сегментування речень
3. Токенізація
4. Тегування частин мови
5. Лематизація
6. Чанкінг
7. Парсінг
8. Розпізнавання сутностей
9. Визначення емоцій
10. Категоризація документів
11. Вилучення фактів
12. Реферування

Бібліотеки підтримки української мови

1. UKR-BERT
2. RobertaUkrainian
3. ElectraUkrainian
4. BERT, DistilBERT (багатомовна)
5. XLM-RoBERTa (багатомовна)
6. XLNet (багатомовна)
7. MultiFiT (багатомовна)
8. T5 (багатомовна)
9. LaBSE (багатомовна)

КОМПОНЕНТИ СИСТЕМИ АВТОМАТИЗОВАНОЇ ОБРОБКИ ЗАПИТІВ КОРИСТУВАЧІВ

4



EmailInteraction
з'єднання з поштовим сервером (IMAP, POP3), отримання вхідних листів, вилучення тексту, відправка листа-відповіді, перенаправлення листів

TextProcessing
попередня обробка тексту, оцінка за ключовими словами та фразами, категорізація тексту

DataStorage
зберігання ключових слів та фраз, шаблонних відповідей, інформації, що до категоризації тексту, інша інформація

ResponseGeneration
отримання даних для відповіді, формування тексту листа-відповіді

Extensibility
керування даними (додання, корегування, видалення)

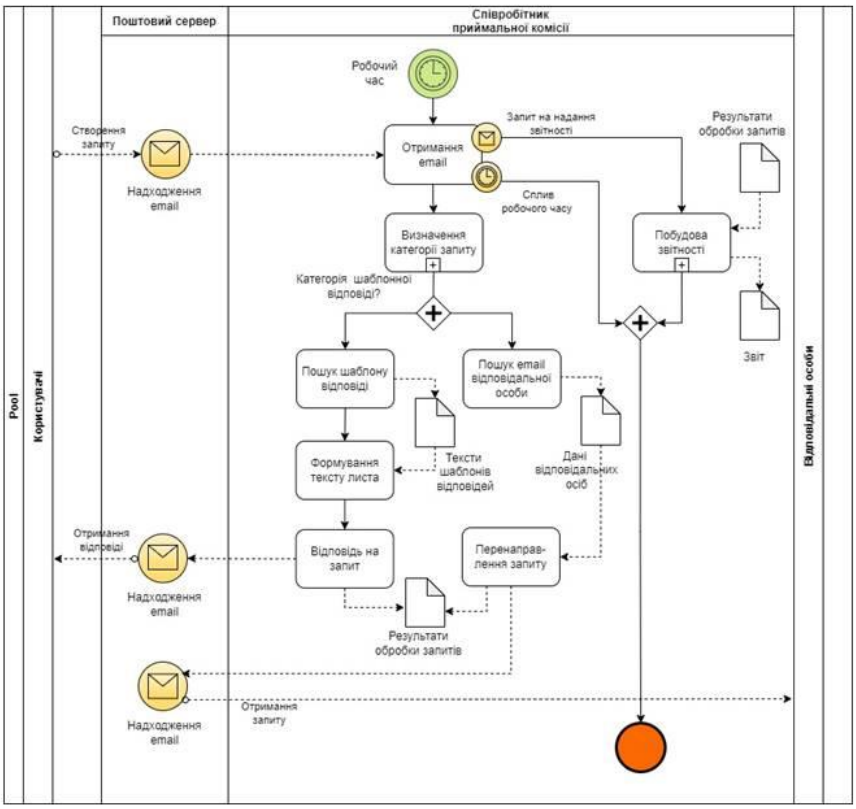
Analytics
формування статистики обробки запитів, ізуалізація статистичних даних

Security
забезпечення безпечного функціонування системи та цілісності даних

Integration
інтеграція з іншими сервісами (чати, оповіщення, нагадування, тощо)

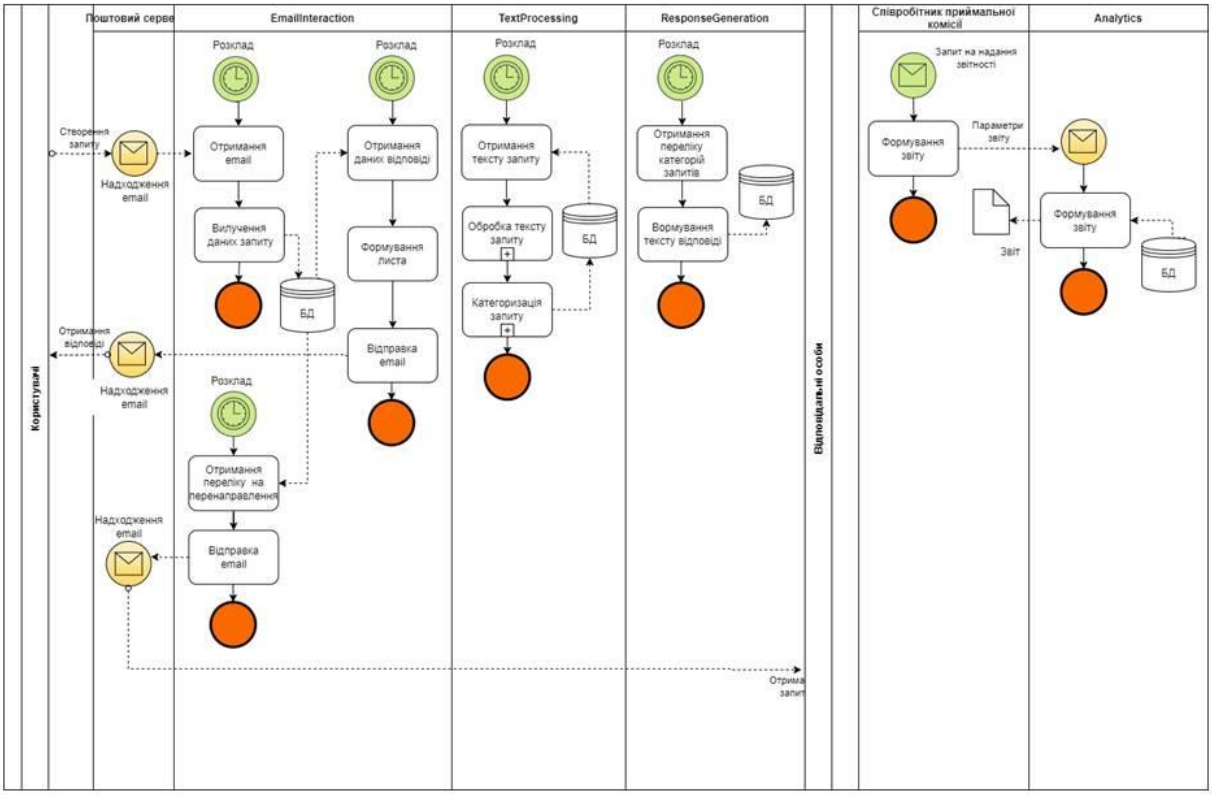
БІЗНЕС ПРОЦЕС ОБРОБКИ ЗАПИТІВ

5



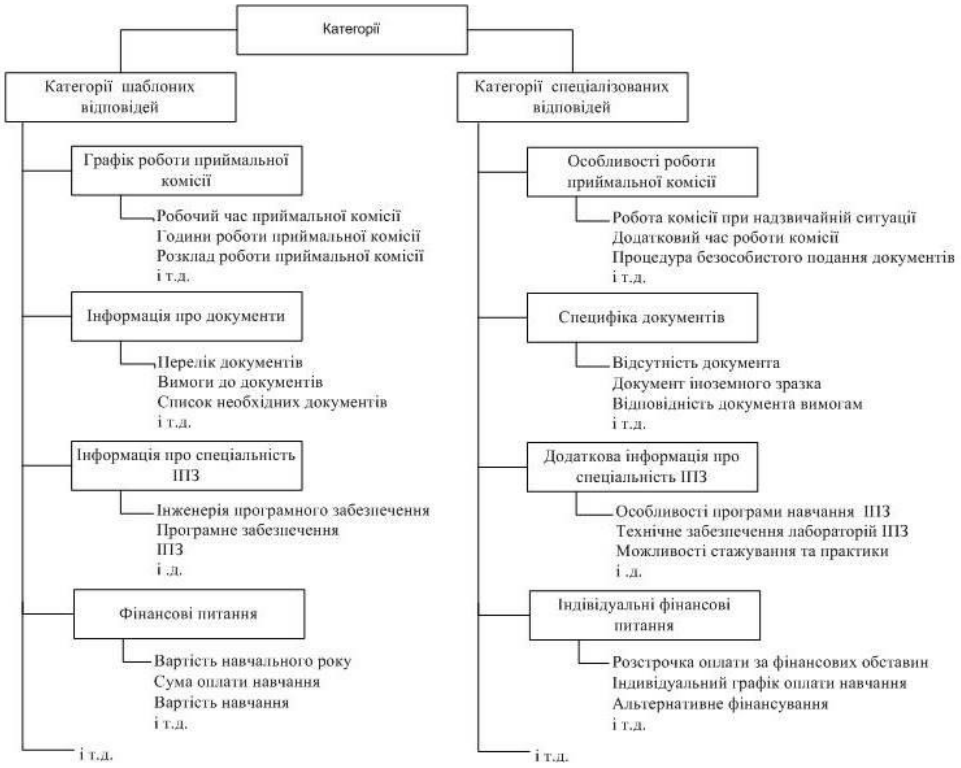
ОПТИМІЗОВАНИЙ БІЗНЕС ПРОЦЕС ОБРОБКИ ЗАПИТІВ

6



ПРИКЛАД КАТЕГОРІЙ ТА КЛЮЧОВИХ ФРАЗ

7



МАТЕМАТИЧНА МОДЕЛЬ КАТЕГОРИЗАЦІЇ ЗАПИТУ

8

Множина категорій

$$C = \{c_1, c_2 \dots c_N\},\ N - \text{кількість категорій} \quad (1)$$

Множина ключових фраз для пошуку схожості

$$L = \{l_1, l_2 \dots l_M\},\ M - \text{кількість фраз} \quad (2)$$

Вагові коефіцієнти фраз у категоріях

$$W = \{(c_i, l_j, w_{ij}) | c_i \in C, l_j \in L, w_{ij} - \text{ваговий коефіцієнт}\}, \quad (3)$$

Множина речень у запиті

$$S = \{s_1, s_2 \dots s_P\},\ P - \text{кількість речень} \quad (4)$$

Коефіцієнти схожості k – го речення до фраз у категоріях

$$Sim_k = \{sim_{kij} | \forall (s_k, c_i, l_j)\ c_i \in C, l_j \in L, S_k \in P, sim_{kij} - \text{коефіцієнт схожісті}\} \quad (5)$$

Ступень належності k – го речення до фраз у категоріях

$$R_k = \{ w_{ij} \cdot sim_{kij} |\ w_{ij} \in W, sim_{kij} \in Sim\} \quad (6)$$

Вектор максимальних значень ступені належності речення k до категорії i

$$M_k = \{ (\max_i (r_{kij}))| \in, r_{kij} \in R_k\} \quad (7)$$

Вектор номерів категорій, до яких належить речення

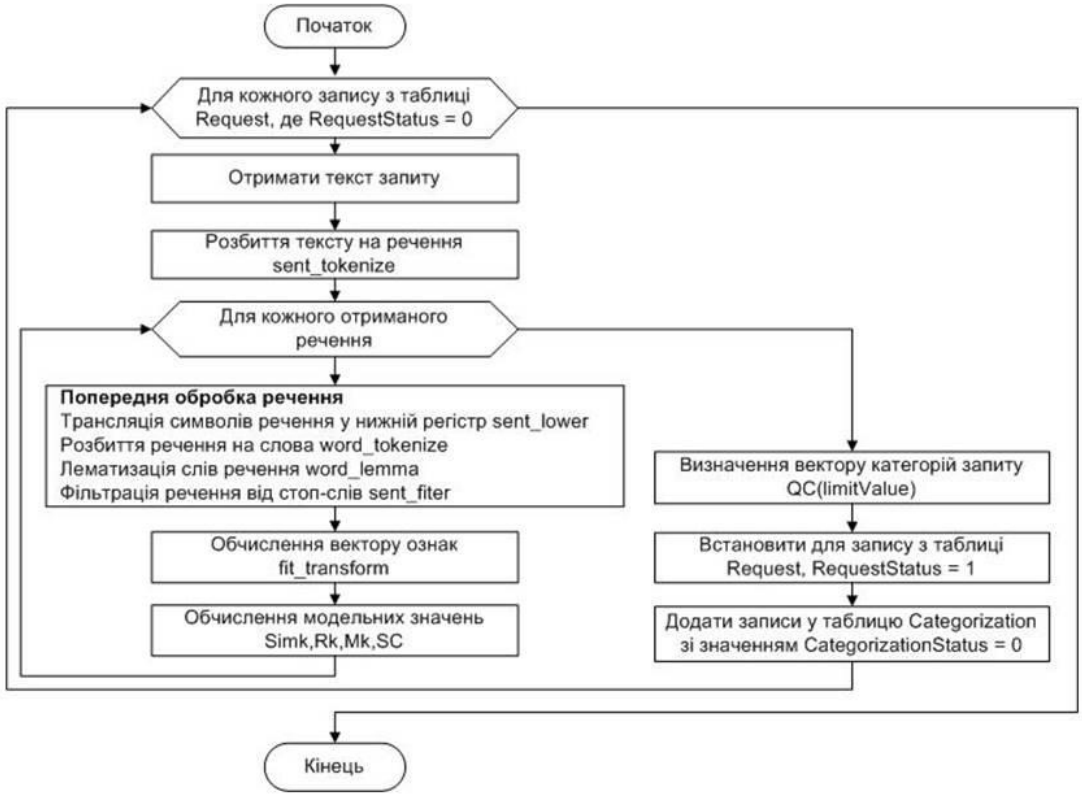
$$SC_k = \{ i \mid m_i > limitValue\},\ limitValue - \text{порогове значення} \quad (8)$$

Вектор номерів категорій, до яких належить запит

$$QC = \cup_{k=1}^{P} Mk \quad (9)$$

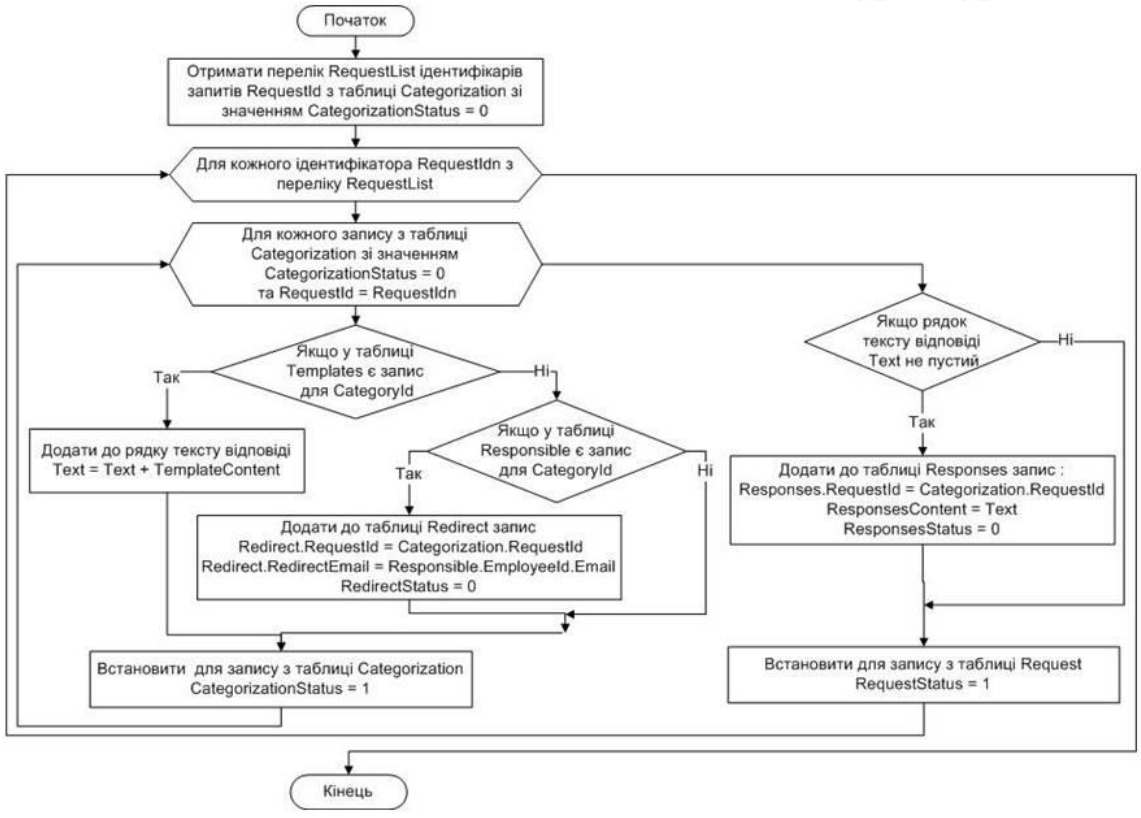
УЗАГАЛЬНЕНА БЛОК-СХЕМА КАТЕГОРИЗАЦІЇ ЗАПИТУ

9

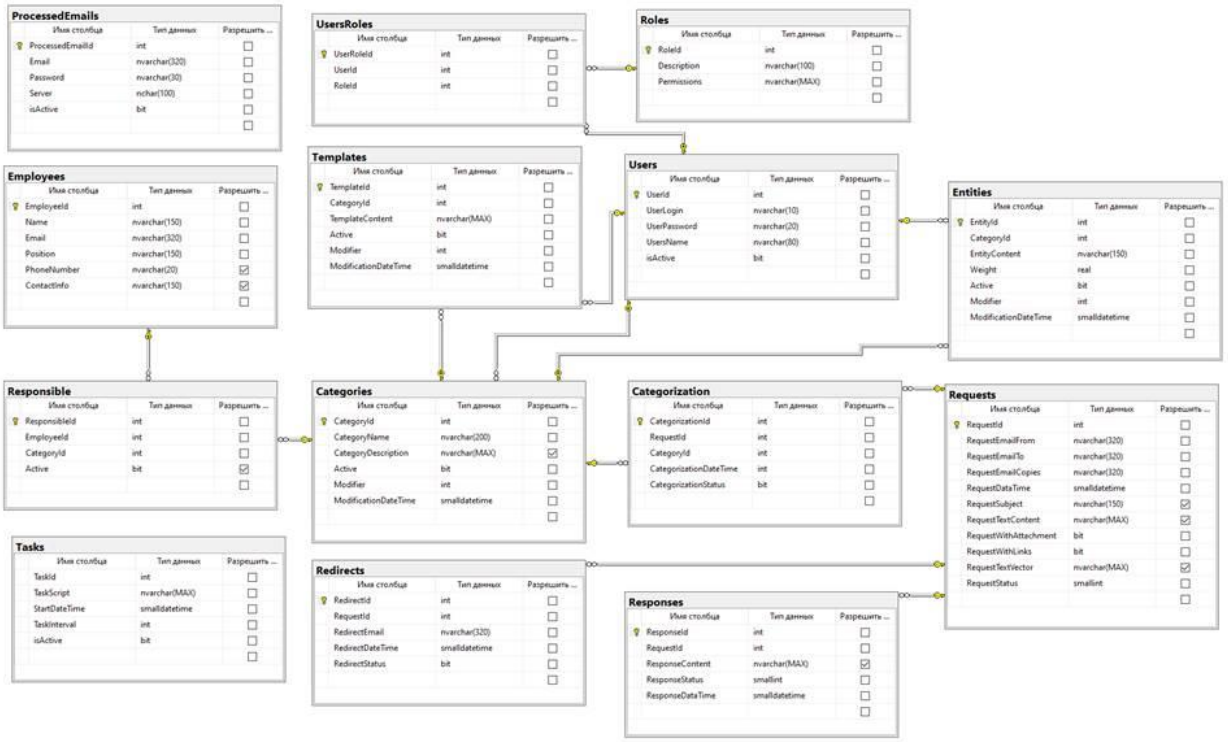


УЗАГАЛЬНЕНА БЛОК-СХЕМА ФОРМУВАННЯ ВІДПОВІДЕЙ

10



СТРУКТУРА БАЗИ ДАНИХ



РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ

C_1L_1 – повністю відповідає запиту $w_{11}C_1L_1 = 0.95$
C_2L_2 – частково відповідає, $w_{22}C_2L_2 = 0.5$
C_3L_3 – абсолютно не відповідає $w_{33}C_3L_3 = 0.1$

Таблиця 1 – Коефіцієнти схожості (Sim)

Кількість запитів	Мовні моделі	Косінусна схожість			Коефіцієнт кореляції			Евклідова відстань		
		C_1L_1	C_2L_2	C_3L_3	C_1L_1	C_2L_2	C_3L_3	C_1L_1	C_2L_2	C_3L_3
N = 10	BERT	0.900	0.710	0.450	0.900	0.710	0.450	4.76	8.05	11.33
	XLNet	0.990	0.980	0.870	0.990	0.980	0.870	12.68	50.98	45.60
	XLMRoberta	0.997	0.990	0.952	0.997	0.990	0.952	1.328	2.57	2.24
N = 30	BERT	0.908	0.690	0.447	0.908	0.690	0.447	4.77	8.43	11.01
	XLNet	0.980	0.977	0.865	0.980	0.977	0.865	11.86	46.82	44.02
	XLMRoberta	0.995	0.989	0.950	0.995	0.989	0.950	1.201	2.40	2.03
N = 100	BERT	0.920	0.675	0.437	0.920	0.675	0.437	5.68	9.24	10.50
	XLNet	0.979	0.976	0.860	0.979	0.976	0.860	11.48	46.04	43.15
	XLMRoberta	0.995	0.985	0.947	0.995	0.985	0.947	1.201	2.31	2.02

Таблиця 2 – Ступені належності (R)

Кількість запитів	Мовні моделі	limitValue = 0,45			limitValue = 0,5		
		C_1L_1	C_2L_2	C_3L_3	C_1L_1	C_2L_2	C_3L_3
N = 10	BERT	0,86	0,36	0,05	0,86	0,36	0,05
	XLNet	0,94	0,49	0,09	0,94	0,49	0,09
	XLMRoberta	0,95	0,50	0,10	0,95	0,50	0,10
N = 30	BERT	0,86	0,35	0,04	0,86	0,35	0,04
	XLNet	0,93	0,49	0,09	0,93	0,49	0,09
	XLMRoberta	0,95	0,49	0,10	0,95	0,49	0,10
N = 100	BERT	0,87	0,34	0,04	0,87	0,34	0,04
	XLNet	0,93	0,49	0,09	0,93	0,49	0,09
	XLMRoberta	0,95	0,49	0,09	0,95	0,49	0,09

ВИСНОВКИ

13

1.Виконано огляд та проведено аналіз сучасних методів та засобів обробки природньої мови. Аналіз показав, що існують різноманітні підходи, включаючи методи машинного навчання, глибокого навчання та класичні методи обробки природньої мови. Визначено переваги та недоліки кожного підходу з точки зору застосування в системі автоматизованої обробки запитів користувачів електронною поштою.

2. Розроблено архітектуру системи автоматизованої обробки вхідних листів електронної скриньки. Визначено ключові компоненти системи та вибрані технології для їх реалізації: мова програмування Python, бібліотеки spaCy, NLTK та моделі трансформери для обробки природньої мови, pyodbc для роботи з базою даних MS SQL, imaplib, smtplib для роботи з електроною поштою, SQL Server 2019 та інтегроване середовище Management Studio 19 для керування базою даних, технологія веб-сервісів ASP.NET для користувацького інтерфейсу з базою даних, засоби середовища розробки Microsoft Visual Studio 2022. Обрані технології забезпечують швидку та достатньо точну обробку тексту природньою мовою, а також ефективне керування базою даних.

3. Розроблено математичну модель категоризації запитів, що враховує вагу певних ключових фраз при оцінюванні належності речень запиту до певної категорії. Модель базується на статистичних аспектах, що робить її придатною для широкого спектру вхідних текстів.

4. Розроблено алгоритми, що дозволяють категоризувати текст запитів користувачів з використанням NLP-технології та розробленої моделі, а також формування тексту відповідей з набору шаблонів та перенаправлення запитів на електрону пошту відповідної відповідальної особи.

5. Проведено моделювання роботи розроблених засобів та алгоритмів. Оцінено ефективність використання автоматизованої системи обробки запитів користувачів та використання для цього технологій NLP.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

14

Тези доповідей:

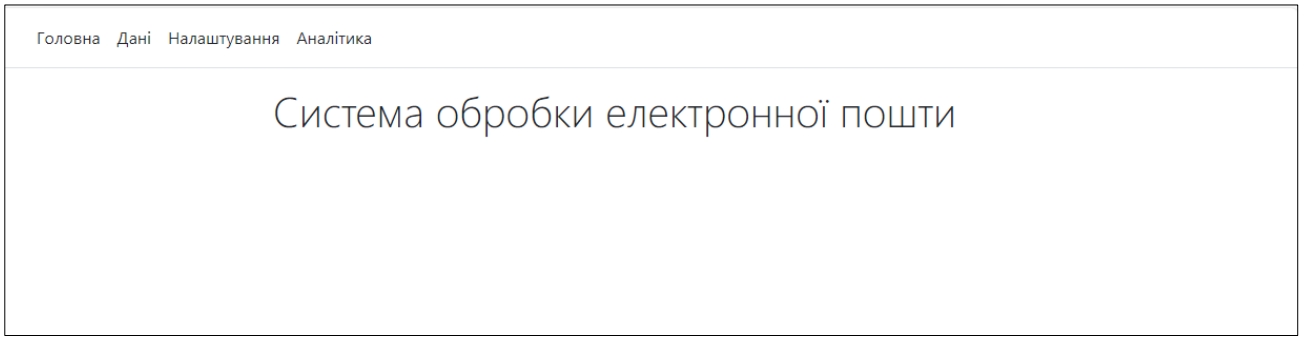
1. Ільїн О.Ю., Горячев Т.В. Система автоматизованої обробки запитів до приймальної комісії ВНЗ // IV Науково-практична конференція «Проблеми комп'ютерної інженерії». Збірник тез. – Київ: ДУТ, 2023,с 175-177.
2. Ільїн О.Ю., Горячев Т.В. Модель категоризації запитів користувачів з використанням технологій NLP // Науково-практичний форум «ТАК»: телекомунікації, автоматика, комп'ютерно-інтегровані технології: зб. доповідей Всеукр. наук.-практ. конф. молодих вчених, 5-6 грудня 2023 р. / ДВНЗ «ДонНТУ; відп. ред. Г.В. Ступак. – Луцьк: ДВНЗ «ДонНТУ», 2023, с.154-156.

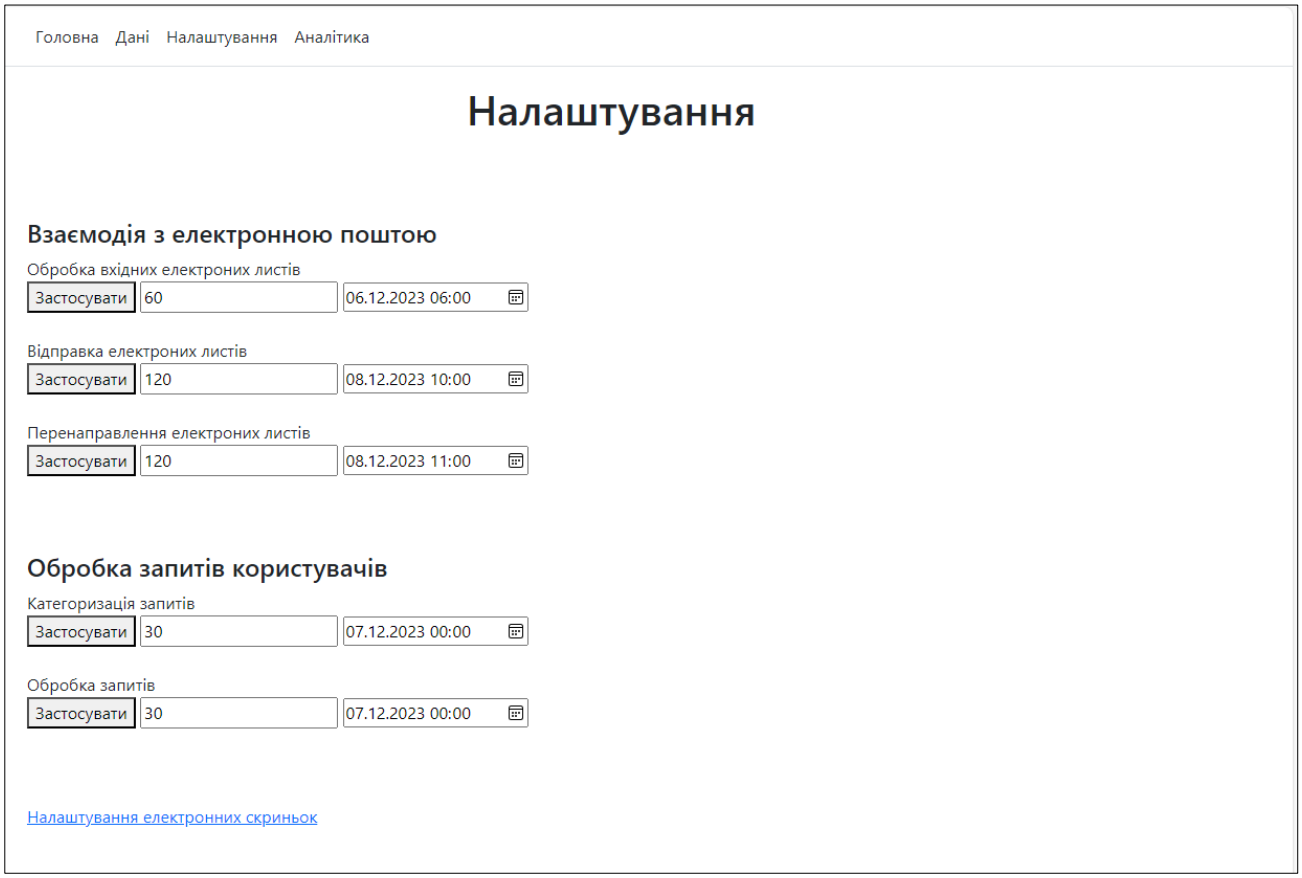
Статті:

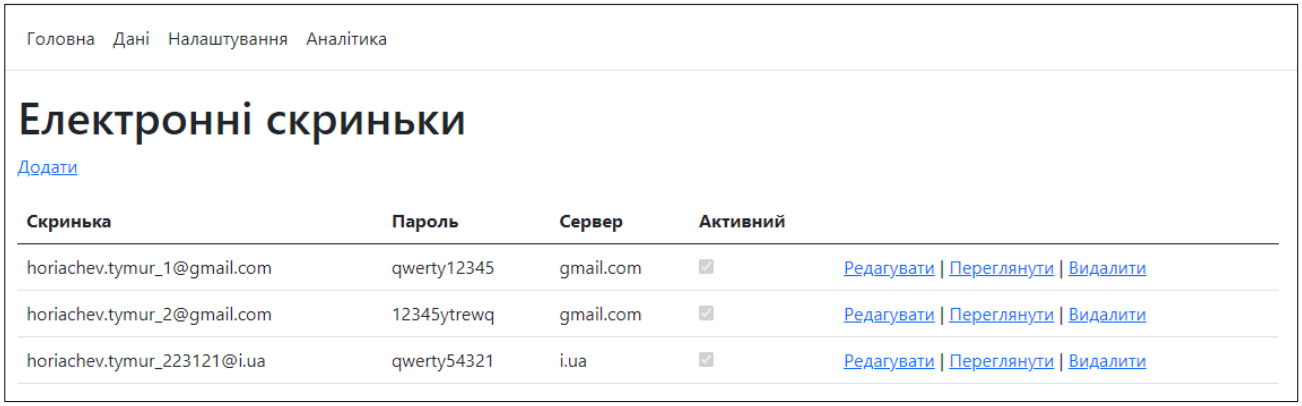
1. 1. Ільїн О.Ю., Золотухіна О.А., Горячев Т.В. Система автоматизованої категоризації запитів електронною поштою з використанням технологій NLP // Наукові записки Державного університету телекомунікації. Подано до друку.

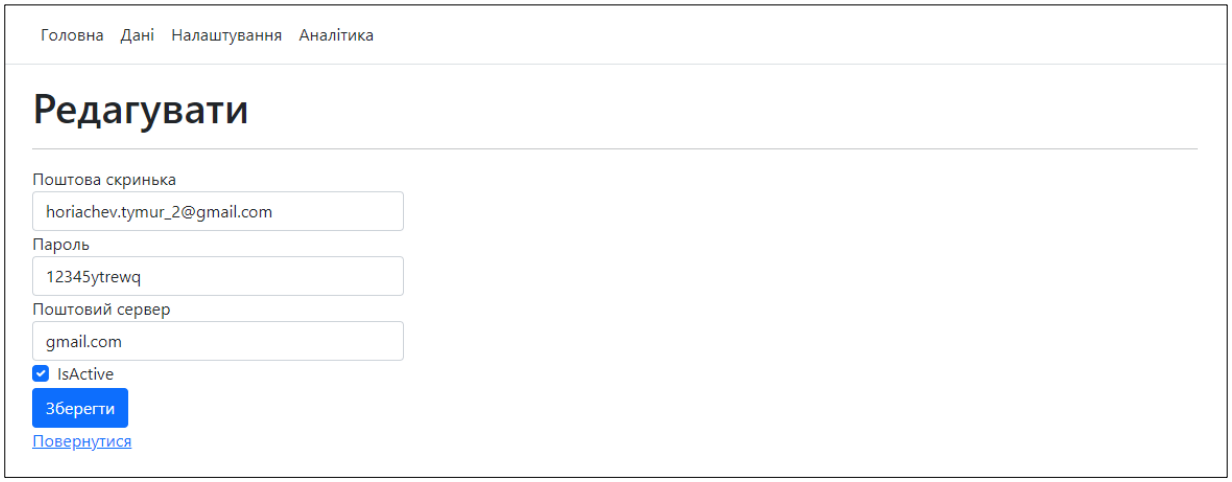
Додаток А

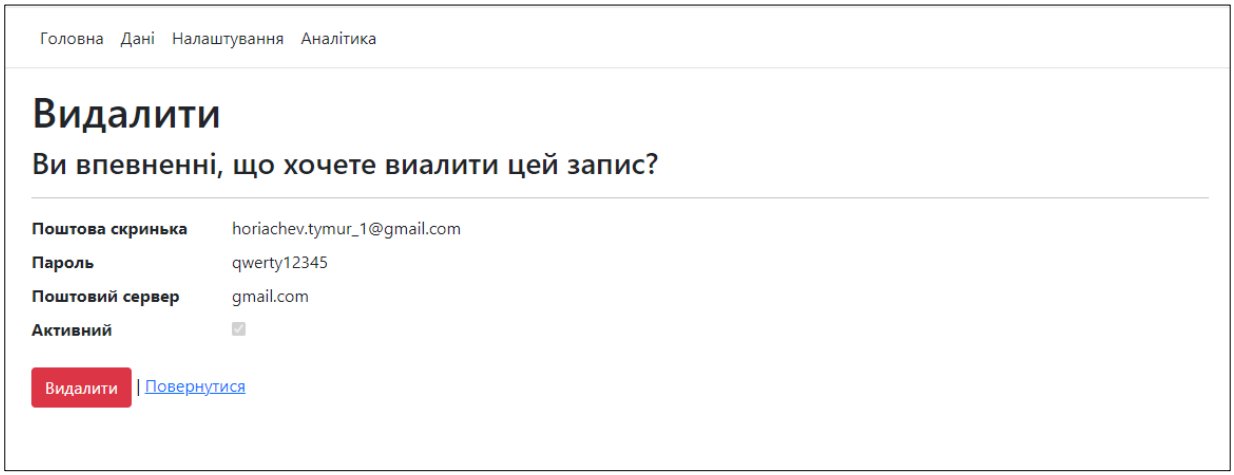
ІНФЕТРФЕЙС СИСТЕМИ

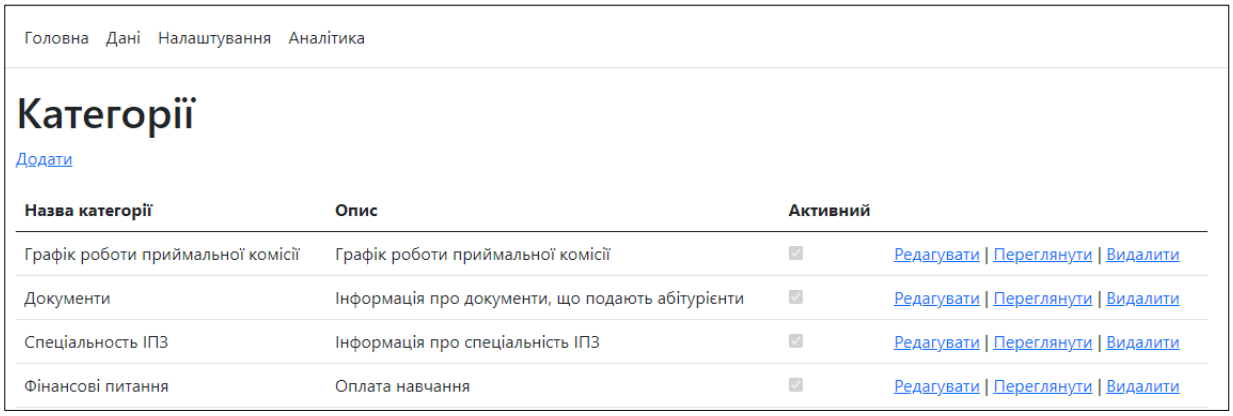


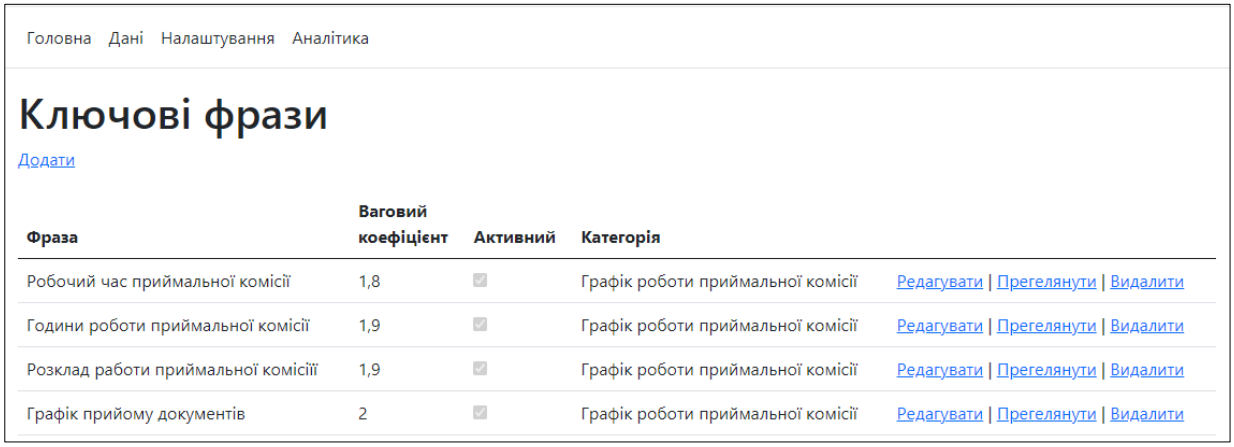





Головна Дані Налаштування Аналітика
Редагувати
Поштова скринька
horiachev.tymur_2@gmail.com
Пароль
12345ytrewq
Поштовий сервер
gmail.com
IsActive
Зберегти
Повернутися


Головна Дані Налаштування Аналітика
Видалити
Ви впевненні, що хочете виалити цей запис?
Поштова скринька horiachev.tymur_1@gmail.com
Пароль qwerty12345
Поштовий сервер gmail.com
Активний
Видалити | Повернутися


Головна Дані Налаштування Аналітика
Категорії
Додати
Назва категорії Опис Активний
Графік роботи приймальної комісії Графік роботи приймальної комісії Редагувати | Переглянути | Видалити
Документи Інформація про документи, що подають абітурієнти Редагувати | Переглянути | Видалити
Спеціальність ІПЗ Інформація про спеціальність ІПЗ Редагувати | Переглянути | Видалити
Фінансові питання Оплата навчання Редагувати | Переглянути | Видалити


Головна Дані Налаштування Аналітика
Ключові фрази
Додати
Фраза Ваговий коефіцієнт Активний Категорія
Робочий час приймальної комісії 1,8 Графік роботи приймальної комісії Редагувати | Прегелянути | Видалити
Години роботи приймальної комісії 1,9 Графік роботи приймальної комісії Редагувати | Прегелянути | Видалити
Розклад работи приймальної комісіїї 1,9 Графік роботи приймальної комісії Редагувати | Прегелянути | Видалити
Графік прийому документів 2 Графік роботи приймальної комісії Редагувати | Прегелянути | Видалити

Додаток Б

ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

В.1. Категоризація запитів

```
import sqlite3
import nltk

from transformers import BertModel, BertTokenizer,
XLNetModel, XLNetTokenizer, XLMRobertaModel,
XLMRobertaTokenizer
from sklearn.metrics.pairwise import cosine_similarity
import torch
from collections import defaultdict

nltk.download('punkt')

conn = sqlite3.connect('RequestProcessingDB')
cursor = conn.cursor()

cursor.execute("SELECT RequestId, RequestSubject,
RequestTextContent FROM Requests WHERE
RequestStatus = 0")
requests_data = cursor.fetchall()

cursor.execute("SELECT CategoryId, EntityContent,
Weight FROM Entities WHERE Active = 1")
entities_data = cursor.fetchall()

#category_sentences = {}

#for category_id, entity_content in entities_data:
#    if category_id not in category_sentences:
#        category_sentences[category_id] = []
#
category_sentences[category_id].append(entity_conten
t)

category_sentences = {}
for category_id, entity_content, weight in entities_data:
    if category_id not in category_sentences:
        category_sentences [category_id] = []
    category_sentences
[category_id].append((entity_content, weight))


def extract_sentences(text):
    sentences = nltk.sent_tokenize(text)
    return sentences

for request_id, request_subject, request_text_content
in requests_data:
    sentences_subject =
extract_sentences(request_subject)
    sentences_content =
extract_sentences(request_text_content)
    all_sentences = sentences_subject +
sentences_content

bert_model = BertModel.from_pretrained('bert-base-
uncased', output_hidden_states=True)
bert_tokenizer = BertTokenizer.from_pretrained('bert-
base-uncased')

xlnet_model = XLNetModel.from_pretrained('xlnet-
base-cased', output_hidden_states=True)
xlnet_tokenizer =
XLNetTokenizer.from_pretrained('xlnet-base-cased')

xlmroberta_model =
XLMRobertaModel.from_pretrained('xlm-roberta-base',
output_hidden_states=True)
xlmroberta_tokenizer =
XLMRobertaTokenizer.from_pretrained('xlm-roberta-
base')

def compute_cosine_similarity(model, tokenizer,
sentence1, sentence2):
    tokens1 = tokenizer(sentence1, return_tensors='pt',
padding=True, truncation=True)['input_ids']
    tokens2 = tokenizer(sentence2, return_tensors='pt',
padding=True, truncation=True)['input_ids']

    with torch.no_grad():
        outputs1 = model(tokens1)['last_hidden_state']
        outputs2 = model(tokens2)['last_hidden_state']

    similarity_matrix =
cosine_similarity(outputs1.mean(dim=1),
outputs2.mean(dim=1))
    return similarity_matrix[0][0]

for model_name, model, tokenizer in [('bert',
bert_model, bert_tokenizer),
                                     ('xlnet', xlnet_model,
xlnet_tokenizer),
                                     ('xlmroberta', xlmroberta_model,
xlmroberta_tokenizer)]:

    with open(f'similarity_results_{model_name}.txt', 'w')
as file:
        for category_id, sentences_in_category in
category_sentences.items():
```

```
        for sentence in all_sentences:
            for category_sentence in
sentences_in_category:
                similarity =
compute_cosine_similarity(model, tokenizer, sentence,
category_sentence)
                file.write(f"Category: {category_id},
Similarity: {similarity:.4f}\n")

for model_name, model, tokenizer in [('bert',
bert_model, bert_tokenizer),
                            ('xlnet', xlnet_model,
xlnet_tokenizer),
                            ('xlmroberta', xlmroberta_model,
xlmroberta_tokenizer)]:

    with open(f'similarity_results_{model_name}.txt', 'w')
as file:
        for category_id, sentences_in_category in
category_sentences.items():
            category_vector = []
            for sentence in all_sentences:
                max_similarity_for_category = -1.0
                for category_sentence, weight in
sentences_in_category:
                    similarity =
compute_cosine_similarity(model, tokenizer, sentence,
category_sentence)
                    weighted_similarity = similarity * weight
                    file.write(f"Category: {category_id},
Weighted Similarity: {weighted_similarity:.4f}\n")
                    max_similarity_for_category =
max(max_similarity_for_category, weighted_similarity)
#                 category_vector.append((category_id,
max_similarity_for_category))
                if max_similarity_for_category >
threshold_similarity:
                    category_vector.append((category_id,
max_similarity_for_category))

sentence_vectors[model_name].append(category_vect
or)

#for model_name, vectors in sentence_vectors.items():
#    print(f"\nModel: {model_name}")
#    for i, vector in enumerate(vectors):
#        print(f"Sentence {i + 1} Categories and Max
Similarities: {vector}")

for model_name, vectors in sentence_vectors.items():
    np.savetxt(f'sentence_vectors_{model_name}.txt',
vectors, fmt='%s', delimiter='\t')

unique_category_ids = {model_name: [] for
model_name in sentence_vectors.keys()}

for model_name, vectors in sentence_vectors.items():
    unique_categories = defaultdict(set)
    for vector in vectors:
        for category_id, _ in vector:

unique_categories[model_name].add(category_id)
    unique_category_ids[model_name] =
list(unique_categories[model_name])

#for model_name, unique_ids in
unique_category_ids.items():
#    print(f"Model: {model_name}, Unique Category IDs:
{unique_ids}")

#for model_name, unique_ids in
unique_category_ids.items():
#
np.savetxt(f'unique_category_ids_{model_name}.txt',
unique_ids, fmt='%s', delimiter='\t')

current_datetime =
datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
for category_id in unique_category_ids['bert']:
    cursor.execute("INSERT INTO Categorization
(RequestId, CategoryId, CategorizationStatus =
0,CategorizationDataTime) VALUES (?, ?,?)", (request_id,
category_id, current_datetime))

conn.close()
```

B.2. Формування відповідей на запити

```
import sqlite3

conn = sqlite3.connect('RequestProcessingDB')
cursor = conn.cursor()

cursor.execute("SELECT CategorizationId, RequestId
FROM Categorization WHERE CategorizationStatus = 0")
request_id_vectors = {}

for row in result:
    categorization_id, request_id = row
    if request_id not in request_id_vectors:
        request_id_vectors[request_id] = []

request_id_vectors[request_id].append(categorization_i
d)
```

```
for request_id, categorization_ids in
request_id_vectors.items():
   result_text = ""

   for categorization_id in categorization_ids:
      cursor.execute("SELECT templateContent FROM
Templates WHERE categorization_id = ? AND Active =
1", (categorization_id,))
      template_result = cursor.fetchone()
      for template_result in template_results:
         template_content = template_result[0]
         result_text += f"{template_content}\n"
   result_texts[request_id] = result_text

for request_id, result_text in result_texts.items():
   if result_text:
      cursor.execute("INSERT INTO Responses
(RequestId, ResponseContent, ResponseStatus) VALUES
(?, ?, 0)",
                  (request_id, result_text))
```

```
   for request_id, categorization_ids in
   request_id_vectors.items():
      for categorization_id in categorization_ids:
         cursor.execute("SELECT * FROM Responsible
   WHERE CategorizationId = ? AND Active = 1",
   (categorization_id,))
         responsible_result = cursor.fetchone()
         if responsible_result:
            employee_id = responsible_result[0]
            category_id = responsible_result[1]
            cursor.execute("SELECT Email FROM Employees
   WHERE EmployeeId = ?", (employee_id,))
            email_result = cursor.fetchone()
            if email_result:
               redirect_email = email_result[0]
               cursor.execute("INSERT INTO Redirects
   (RequestId, RedirectStatus, RedirectEmail) VALUES (?, ?,
   ?)",
                              (request_id, 0, redirect_email))

   conn.commit()
   conn.close()
```

В.3. Обробка вхідних листів

```
import imaplib
import email
from email.header import decode_header

def read_active_emails():

   conn = sqlite3.connect('RequestProcessingDB')
   cursor = conn.cursor()
   cursor.execute("SELECT * FROM ProcessedEmail WHERE
IsActive = 1
   rows = cursor.fetchall()
   email_data = []

   for row in rows:
      email_info = {
         'email': row[0],
         'login': row[1],
         'password': row[2],
         'domain': row[3],
         'is_active': row[4]
      }
      email_data.append(email_info)
   conn.close()
   return email_data


def process_email_message(msg):
   sender, encoding = decode_header(msg.get("From"))[0]
   if isinstance(sender, bytes):
      sender = sender.decode(encoding or "utf-8")

   recipient, _ = decode_header(msg.get("To"))[0]
   if isinstance(recipient, bytes):
      recipient = recipient.decode(encoding or "utf-8")

   cc_addresses = msg.get("Cc", "")

   subject, _ = decode_header(msg.get("Subject"))[0]
   if isinstance(subject, bytes):
      subject = subject.decode(encoding or "utf-8")

   body = ""
   if msg.is_multipart():
      for part in msg.walk():
         content_type = part.get_content_type()
         content_disposition = str(part.get("Content-
Disposition"))

         if "attachment" not in content_disposition:
            body =
part.get_payload(decode=True).decode(part.get_content_
charset() or 'utf-8', 'ignore')
            break
   else:
      body =
msg.get_payload(decode=True).decode(msg.get_content_
charset() or 'utf-8', 'ignore')
```

```
    date_sent = msg.get("Date")

    has_attachments = 1 if msg.get_payload() and
isinstance(msg.get_payload()[0], email.message.Message)
else 0

    has_links = 1 if "http" in body else 0

    return {
        'from': sender,
        'to': recipient,
        'cc': cc_addresses,
        'subject': subject,
        'body': body,
        'date_sent': date_sent,
        'has_attachments': has_attachments,
        'has_links': has_links
    }

active_emails = read_active_emails()
for email in active_emails:
            mail = imaplib.IMAP4_SSL("imap." +
            email['domain'])
            mail.login(email['email'], email['password'])
            mail.select("INBOX", readonly=True)
            status, messages = mail.search(None,
            "(UNSEEN)")
            message_ids = messages[0].split()
for message_id in message_ids:
    _, msg_data = mail.fetch(message_id, "(RFC822)")
    raw_email = msg_data[0][1]
    msg = email.message_from_bytes(raw_email)

    msg_info = process_email_message(msg)

    cursor.execute("INSERT INTO Requests
(RequestEmailFrom, RequestEmailTo, ,
RequestEmailCopies, RequestSubject, RequestTextConten,
RequestDataTime,
RequestWithAttachments,RequestWithLinks,
RequestStatus) VALUES (?, ?, ?, ?, ?, ?, ?, ?,?,?)",
              (msg_info['from'], msg_info['to'], msg_info['cc'],
msg_info['subject'], msg_info['body'],
               msg_info['date_sent'],
msg_info['has_attachments'], msg_info['has_links'],0))
conn.commit()
conn.close()
mail.store(",".join(message_ids), '+FLAGS', '(\Seen)')
mail.logout()
```

В.4 Перенаправлення листів

```
import smtplib
import sqlite3
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

conn = sqlite3.connect(RequestProcessingDB ')
cursor = conn.cursor()

cursor.execute("SELECT RequestId,RedirectEmail FROM
Redirects WHERE RedirectStatus = 0")
forwarding_data = cursor.fetchall()

def send_email(sender, recipient, subject, body,
smtp_server, smtp_port, smtp_username,
smtp_password):
    message = MIMEMultipart()
    message['From'] = sender
    message['To'] = recipient
    message['Subject'] = subject
    message.attach(MIMEText(body, 'plain'))

    with smtplib.SMTP(smtp_server, smtp_port) as
server:
        server.starttls()
        server.login(smtp_username, smtp_password)
        server.sendmail(sender, recipient,
    message.as_string())

    for forwarding_info in forwarding_data:
        forwarding_id, email_id, forwarding_address =
    forwarding_info

        cursor.execute("SELECT RequestEmailFrom,
    RequestEmailTo, RequestSubject, RequestTextContent
    FROM Requests WHERE RequestId = ?”, (email_id,) )
        email_data = cursor.fetchone()

    if email_data:
            sender, recipient,subject, text = email_data
            send_email(sender, forwarding_address,
    'Forwarded: ' + subject, text, 'your_smtp_server.com',
    587, sys_email, sys_email_password)

            current_datetime =
    datetime.datetime.now().strftime("%Y-%m-%d
    %H:%M:%S")
            cursor.execute("UPDATE Redirects  SET
    RedirectStatus = 1, process_date = ? WHERE id = ?",
    (current_datetime, forwarding_id))
    conn.commit()
    conn.close()
```

В.5 Відправлення ввідповідей на листи

```
import smtplib
import sqlite3
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

conn = sqlite3.connect('your_database.db')
cursor = conn.cursor()

cursor.execute("SELECT ResponseId, RequestId,
ResponseContent FROM Responses WHERE
ResponseStatus = 0")
responses_data = cursor.fetchall()
def send_email(sender, recipient, subject, body,
smtp_server, smtp_port, smtp_username,
smtp_password):
   message = MIMEMultipart()
   message['From'] = sender
   message['To'] = recipient
   message['Subject'] = subject
   message.attach(MIMEText(body, 'plain'))

   with smtplib.SMTP(smtp_server, smtp_port) as
server:
      server.starttls()
      server.login(smtp_username, smtp_password)
      server.sendmail(sender, recipient,
message.as_string())

for response_id, request_id, response_content in
responses_data:
   cursor.execute("SELECT RequestEmailFrom FROM
Requests WHERE RequestId = ?", (request_id,))
   request_data = cursor.fetchone()

   if request_data:
      request_email_from = request_data[0]
      send_response(request_email_from,
response_content)
      current_datetime =
datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
      cursor.execute("UPDATE Responses SET
ResponseStatus = 1, process_date = ?  WHERE
ResponseId = ?", (current_datetime, response_id,))

conn.commit()
conn.close()
```