

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ВИЛУЧЕННЯ КЛЮЧОВИХ ФРАЗ З НАУКОВИХ СТАТЕЙ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
EXTRACTING KEY PHRASES FROM SCIENTIFIC PAPERS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-222м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

ШОВАК К.Г.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШОВАКА Кирила Георгійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів
вилучення ключових фраз з наукових статей. Research and Software
Implementation of Methods for Extracting Key Phrases from Scientific Papers

керівник проєкту (роботи) к.т.н., доцент, КОЦУР Михайло Ігорович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “14” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис
програми. 4. Експлуатація, тестування та експериментальне дослідження
програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

‘

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконання завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “08” вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	8 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	9-10 тижні	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

Кирило ШОВАК
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Михайло КОЦУР
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
88 с., 15 табл., 13 рис., 3 дод., 18 джерел.

СЕМАНТИКА, КЛЮЧОВІ СЛОВА, ПОШУК, НАУКОВІ СТАТТІ,
PYTHON, ANACONDA, МОВА ПРОГРАМУВАННЯ, СОЦІАЛЬНІ МЕРЕЖІ,
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес вилучення ключових фраз з наукових статей.

Предмет дослідження – програмні засоби для реалізація методів вилучення ключових фраз з наукових статей.

Метою роботи є дослідження та розробка програмного забезпечення для реалізація методів вилучення ключових фраз з наукових статей.

Матеріали, методи та технічні засоби: мова програмування Python та середовище розробки Anaconda.

Результати. Розроблено програмне забезпечення для реалізація методів вилучення ключових фраз з наукових статей за допомогою мови програмування Python та фреймворку Anaconda.

Практична цінність роботи полягає у розробці програмного забезпечення, що реалізує методи вилучення ключових фраз з наукових статей.

Висновки. Виконано проєктування програмного забезпечення для реалізації методів вилучення ключових фраз з наукових статей. Розроблено програмне забезпечення для реалізації методів вилучення ключових фраз з наукових статей. Здійснено тестування розробленого програмного забезпечення для реалізації методів вилучення ключових фраз з наукових статей.

Галузь використання – наукометричні бази, співробітники наукових закладів.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 88 pages, 15 tables, 12 figures, 3 appendixes, 18 sources.

SEMANTICS, KEYWORDS, SEARCH, SCIENTIFIC ARTICLES, PYTHON, ANACONDA, PROGRAMMING LANGUAGE, SOCIAL NETWORKS, SOFTWARE.

The object of research is the process of extracting key phrases from scientific articles.

The subject of the research is software tools for implementing methods for extracting key phrases from scientific articles.

The aim of the work is to research and develop software for implementing methods for extracting key phrases from scientific articles.

Materials, methods, and technical tools: Python programming language and Anaconda development environment.

Results. Software has been developed to implement methods for extracting keywords from scientific articles using the Python programming language and the anaconda framework.

The practical value of the work lies in the development of software that implements methods for extracting key phrases from scientific articles.

Conclusions. Software was designed to implement methods for extracting key phrases from scientific articles. Software has been developed to implement methods for extracting key phrases from scientific articles. The developed software was tested to implement methods for extracting key phrases from scientific articles.

Scope of use – scientometric databases, employees of scientific institutions.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Ключові фрази та слова в наукових статтях	11
1.2 Поняття ключових фраз	15
1.3 Метрики оцінки інформативності.....	18
1.4 Метрики оцінки ефективності.....	19
1.5 Аналіз існуючих рішень.....	20
1.5.1 Фреймворк Sztergak	20
1.5.2 Тестування результативності фреймворку Sztergak.....	24
1.5.3 Система Humb	26
1.5.4 Тестування результативності системи Humb.....	30
1.5.5 Система Wingnus.....	31
1.5.6 Тестування – вибір кандидатів ключових фраз	33
1.5.7 Система KP-Miner	34
1.5.8 Результати тестування – остаточна обробка кандидатів	37
1.5.9 Результати порівняння.....	37
1.6 Постановка задачі	39
1.7 Перелік функціональних вимог.....	40
1.8 Висновки за розділом 1	40
2 Матеріали і методи	41
2.1 Вибір мови програмування	41
2.2 Вибір середовища розробки	46
2.3 Проєктування системи	49
2.4 Висновки за розділом 2	50
3 Опис програми	51
3.1 Загальна схема роботи.....	51

3.1.1 Попередня обробка корпусу наукових статей	52
3.1.2 Попередня обробка словника термінів	53
3.1.3 Відбір термінів зустрічаються в статті зі словника.....	53
3.1.4 Розрахунок ваг інформативності для відібраних термінів	54
3.1.5 Ранжування результатів і відбір термінів.....	54
3.2 Архітектура.....	54
3.3 Схема роботи.....	56
3.4 Висновки за розділом 3	57
4 Експлуатація, тестування та експериментальне дослідження програми.....	58
4.1 Призначення й умови застосування програми	58
4.2 Характеристики функціонування.....	58
4.2.1 Ефективність.....	58
4.2.2 Продуктивність	59
Висновки	61
Перелік джерел посилань	62
Додаток А Технічне завдання	64
Додаток Б Фрагмент тексту програми	68
Додаток В Слайди презентації	82

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- ACL – асоціації з комп'ютерної лінгвістики;
- ACM – association for computational machinery;
- GDC – generalized dice coefficient;
- GROBID – generation of bibliographical data;
- IDE – integrated development environment (інтегроване середовище для розробки);
- IDF – inverse document frequency (інвертована частота документа);
- KEA – keyphrase extraction algorithm;
- MLP – multi-layer perceptron;
- NER – розпізнавання іменованих об'єктів;
- NLP – natural language processing (обробка природної мови);
- OCR – optical character recognition;
- POS – граматичні ознаки;
- SEO – пошукова оптимізація;
- SF-ISF – section frequency-inverted section frequency;
- SVM – support vector machine;
- TEI – text encoding initiative;
- TF-IDF – term frequency-inverse document frequency (частота терміну-зворотна частота документа).

ВСТУП

Новий метод для вилучення ключових фраз з наукових статей може бути актуальним з декількох причин.

Пошук інформації та оптимізація пошуку. Дослідникам і вченим часто потрібно швидко знайти відповідні статті та інформацію для своєї роботи. Вилучення ключових фраз може допомогти у створенні більш ефективних пошукових систем та систем пошуку інформації, полегшуючи професіоналам пошук та доступ до відповідної наукової літератури [1]-[4].

Огляд та узагальнення літератури. Проводячи огляди літератури, дослідникам необхідно швидко визначити ключові поняття та висновки в конкретній галузі. Вилучення ключових фраз з наукових статей може прискорити процес аналізу та узагальнення інформації, дозволяючи дослідникам більш ефективно отримати всебічне уявлення про тему [1]-[4].

Автоматичне узагальнення. Автоматизовані інструменти узагальнення часто покладаються на вилучення ключових фраз для створення коротких та інформативних резюме наукових статей. Це може мати вирішальне значення для людей, яким потрібно швидко зрозуміти основні ідеї та внески великої кількості статей протягом обмеженого періоду часу [1]-[4].

Аналіз тенденцій та визначення пріоритетів досліджень. Визначення ключових фраз може допомогти в аналізі тенденцій наукової літератури. Дослідники та політики можуть використовувати цю інформацію для визначення пріоритетів у сферах досліджень, розподілу ресурсів та отримання інформації про останні події в певній галузі [1]-[4].

Обробка мови для систем штучного інтелекту. Програми для обробки природної мови, включаючи чат-ботів, віртуальних помічників та інші системи штучного інтелекту, можуть отримати вигоду від вилучення ключових фраз для кращого розуміння запитів користувачів, пов'язаних з науковим контентом, і реагування на них. Це особливо актуально в областях, де вкрай важливо бути в курсі останніх досліджень [1]-[4].

Економія часу та ефективність. Вилучення ключових фраз з наукових статей може заощадити дослідникам значну кількість часу. Замість того, щоб читати цілі статті, вони можуть швидко визначити основні ідеї та концепції, що дозволяє їм зосередитися на найбільш актуальній інформації [1]-[4].

Міждисциплінарні дослідження. Дослідникам, які працюють у міждисциплінарних сферах, може знадобитися швидко зрозуміти та інтегрувати інформацію з різних наукових галузей. Виділення ключових фраз полегшує виявлення загальних тем і зв'язків між різними дисциплінами [1]-[4].

Підтримка прийняття рішень у промисловості та політиці. Виділення ключових фраз може допомогти особам, які приймають рішення в галузях і політиці, надаючи коротку інформацію з наукової літератури. Ця інформація може мати вирішальне значення для прийняття обґрунтованих рішень, пов'язаних з розробкою продукту, регуляторною Політикою та іншими сферами, які потребують глибокого розуміння наукового ландшафту [1]-[4].

Таким чином, застосунок для вилучення ключових фраз з наукових статей може задовольнити нагальну потребу в ефективному пошуку інформації, огляді літератури, узагальненні, аналізі тенденцій та прийнятті рішень у різних професійних та дослідницьких контекстах [1]-[4].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Ключові фрази та слова в наукових статтях

Наукові праці, що включає в тому числі: статті, тези та інші наукові публікації, є наріжним каменем поширення знань та академічного спілкування. У цьому величезному морі інформації ключові слова та фрази відіграють ключову роль у передачі суті результатів досліджень, полегшенні пошуку інформації та сприянні ефективному спілкуванню між дослідниками, науковцями та широким науковим співтовариством [5]-[7]. Розглянемо роботу ключових слів і фраз в наукових роботах, переглянемо їх важливість в контексті дослідницької комунікації та поширення знань [5]-[7].

Визначення та відмінності. Перш ніж заглиблюватися в їх важливість, важливо зрозуміти відмінність між ключовими словами і словосполученнями в наукових роботах. Ключові слова – це окремі терміни, які містять основні поняття наукового дослідження. З іншого боку, ключові фрази складаються з декількох слів, які разом відображають основні теми чи ідеї наукової роботи. Обидві вони служать коротким поданням змісту дослідження, допомагаючи в категоризації, індексації та пошуку інформації [5]-[7].

Покращення пошуку інформації. Однією з основних функцій ключових слів і словосполучень є полегшення пошуку інформації. Оскільки обсяг наукової літератури продовжує зростати в геометричній прогресії, дослідники стикаються з непростим завданням просіювання великих баз даних для виявлення відповідних досліджень. Ключові слова та словосполучення діють як маяки, спрямовуючи дослідників до літератури, яка найбільше відповідає їхнім запитам. Пошукові системи та бази даних використовують ці терміни для індексації та категоризації наукових праць, дозволяючи користувачам швидко та ефективно отримувати доступ до відповідної інформації [5]-[7].

Розглянемо, наприклад, дослідника, який шукає статті про вплив зміни клімату на морські екосистеми. Включивши такі ключові слова, як: зміна клімату, морські екосистеми та вплив – дослідник може спростити процес

пошуку, гарантуючи, що знайдені статті точно відповідають наміченій спрямованості. Більше того, такі ключові фрази, як: вплив зміни клімату на морське біорізноманіття – можуть забезпечити більш детальний і конкретний пошук, уточнюючи результати досліджень, які конкретно стосуються взаємозв'язку зміни клімату та морського біорізноманіття [5]-[7].

Передаючи спрямованість і охоплення досліджень. Ключові слова і словосполучення служать коротким описом спрямованості та обсягу наукової роботи. У назві, анотації та по всьому рукопису дослідники стратегічно використовують ці терміни, щоб донести основні теми та внесок своєї роботи. Ця практика допомагає колегам-дослідникам, читачам та потенційним працівникам швидко зрозуміти основні ідеї, не заглиблюючись у весь документ [5]-[7].

Наприклад, стаття, що досліджує застосування штучного інтелекту в діагностиці захворювань, може включати такі ключові слова, як: штучний інтелект, діагностика та медичні програми. Ці терміни дають коротке уявлення про дослідження, дозволяючи людям оцінити, чи відповідає стаття їхнім інтересам чи дослідницьким потребам [5]-[7].

Сприяння огляду та узагальненню літератури. У процесі проведення оглядів літератури та узагальнення існуючих знань дослідники покладаються на ключові слова та словосполучення для виявлення відповідних досліджень та виділення необхідної інформації. Систематичне використання цих термінів підвищує ефективність процесів огляду літератури, дозволяючи дослідникам орієнтуватися в безлічі публікацій і ефективно витягувати ключові ідеї [5]-[7].

Розглянемо докторанта, який прагне вивчити роль нейромедіаторів у розладах настрою. Використовуючи такі ключові слова, як: нейромедіатори, розлади настрою та нейронаука – студент може ефективно ідентифікувати та аналізувати дослідження, які сприяють розумінню цієї складної взаємодії. Крім того, такі ключові фрази, як: серотонін і депресія або дофамін і регуляція настрою – дозволяють більш детально вивчити конкретні аспекти в рамках більш широкої теми [5]-[7].

Підтримка автоматичного узагальнення та індексації. В епоху цифрової інформації автоматизовані інструменти для узагальнення та індексації використовують ключові слова і словосполучення для створення коротких резюме і категоризації наукових робіт. Автоматизовані системи використовують алгоритми обробки природної мови для ідентифікації та вилучення цих термінів, пропонуючи користувачам швидке уявлення про основні внески та результати наукового дослідження [5]-[7].

Автоматизовані інструменти Узагальнення, наприклад, можуть аналізувати ключові фрази в анотації наукової статті та генерувати скорочену версію, яка інкапсулює основні поняття. Це не тільки допомагає зайнятим дослідникам швидко оцінити актуальність статті, але й сприяє розробці складних застосувань штучного інтелекту в галузі аналізу наукової літератури [5]-[7].

Сприяє міждисциплінарному розумінню. Наукові дослідження часто виходять за межі дисциплінарних меж, вимагаючи від дослідників взаємодії з літературою з різних галузей. Ключові слова та словосполучення відіграють вирішальну роль у сприянні міждисциплінарному розумінню, забезпечуючи спільну мову для дослідників з різних областей. Ці терміни служать мостами, що з'єднують різні райони і дозволяють вченим досліджувати перехрестя та можливості співпраці [5]-[7].

Можна уявити собі біолога, який досліджує екологічні наслідки урбанізації. Включення таких ключових слів, як: міська Екологія, біорізноманіття та екосистемні послуги – може привернути увагу дослідників як екології, так і містобудування. Подібним чином, такі ключові фрази, як: вплив урбанізації на моделі міграції птахів – можуть викликати інтерес у дослідників, що спеціалізуються на орнітології та міських дослідженнях, сприяючи міждисциплінарній співпраці [5]-[7].

Дозволяючи аналізувати тенденції та визначати пріоритети досліджень. Аналіз ключових слів і словосполучень у науковій літературі дозволяє дослідникам, розробникам політики та фінансуючим установам виявляти

тенденції в конкретній галузі. Ця інформація є безцінною для прийняття обґрунтованих рішень щодо пріоритетів досліджень, розподілу ресурсів та розробки політики [5]-[7].

Наприклад, у галузі відновлюваної енергії аналіз ключових слів та фраз у публікаціях протягом десятиліття може виявити нові теми, такі як: накопичення сонячної енергії, оптимізація енергії вітру або досягнення біопалива. Таке розуміння дозволяє зацікавленим сторонам стратегічно розподіляти ресурси, підтримуючи дослідницькі зусилля, що відповідають сучасним тенденціям та потребам суспільства [5]-[7].

Поліпшення комунікації в промисловості та політиці. Ключові слова та словосполучення не обмежуються сферою наукових кіл; вони також відіграють вирішальну роль у доведенні наукових результатів до відома професіоналів галузі, політиків та широкої громадськості. Оскільки наукові дослідження все частіше використовуються для прийняття рішень у різних секторах, стратегічне використання цих термінів полегшує ефективне спілкування, гарантуючи, що ключові ідеї та наслідки передаються коротко [5]-[7].

Розглянемо дослідження впливу нового виробничого процесу на навколишнє середовище. Такі ключові слова, як: екологічна стійкість, виробництво та оцінка життєвого циклу – дозволяють зацікавленим сторонам галузі швидко визначити актуальність дослідження для їхніх проблем. Директивні органи також можуть отримати вигоду від систематичного використання ключових фраз, таких як: політичні рекомендації щодо сталого виробництва – для виявлення досліджень, які дають уявлення про регуляторні міркування [5]-[7].

На закінчення, значення ключових слів і словосполучень в наукових роботах неможливо переоцінити. Ці лінгвістичні конструкції служать сполучними ланками, що з'єднують дослідників, науковців та зацікавлені сторони в мережі ефективного спілкування та обміну знаннями. Ключові слова та словосполучення - від полегшення пошуку інформації та оглядів

літератури до підтримки автоматичного узагальнення та забезпечення міждисциплінарної співпраці – є неоспіваними героями дослідницької комунікації. Коли ми орієнтуємось у величезному просторі наукових знань, ці короткі вирази служать нам орієнтиром, забезпечуючи чіткість і послідовність у заплутаному гобелені людських досліджень [5]-[7].

1.2 Поняття ключових фраз

Обробка природної мови (Natural Language Processing – NLP) – це область штучного інтелекту, яка фокусується на взаємодії між комп'ютерами та людськими мовами. У величезному та складному ландшафті NLP концепція ключових слів відіграє ключову роль. Ключові слова – це фундаментальні лінгвістичні одиниці, які містять суть тексту, допомагаючи в різних завданнях NLP. Це всебічне дослідження заглиблюється у визначення ключових слів, їх значення та їх застосування в різних завданнях NLP [5]-[7].

Визначення ключових слів. Ключові слова, в контексті NLP, відносяться до конкретних термінів або фраз, які інкапсулюють основні теми у фрагменті тексту. Ці терміни вибираються на основі їх актуальності і важливості для змісту, виступаючи в якості представників ключових концепцій, виражених в тексті. Процес ідентифікації та вилучення ключових слів включає лінгвістичний аналіз та обчислювальні методи, що сприяють глибшому розумінню основної семантики [5]-[7].

Важливість ключових слів у завданнях NLP. Пошук інформації. Ключові слова відіграють важливу роль у системах пошуку інформації, де метою є вилучення відповідних документів або уривків на основі запитів користувачів. Індексуючи документи за ключовими словами, пошукові системи можуть ефективно зіставляти запити користувачів з найбільш релевантним текстовим вмістом [5]-[7].

Узагальнення тексту. В області Узагальнення тексту ключові слова служать основою для ідентифікації важливої інформації. Витяг ключових фраз

дозволяє алгоритмам генерувати короткі і зв'язні резюме, фокусуючись на найбільш істотних аспектах документа [5]-[7].

Категоризація документів. Ключові слова сприяють процесу категоризації документів. Присвоєння документів відповідним ключовим словам допомагає класифікувати їх за заздалегідь визначеними темами чи класами, полегшуючи ефективну організацію та пошук [5]-[7].

Аналіз настроїв. Розуміння настроїв, виражених в текстовому змісті, є найважливішим завданням в NLP. Ключові слова, пов'язані з позитивними чи негативними настроями, відіграють життєво важливу роль в алгоритмах аналізу настроїв, дозволяючи системам оцінювати загальний емоційний тон документа [5]-[7].

Розпізнавання іменованих об'єктів (NER). Ключові слова часто збігаються з названими об'єктами, такими як люди, організації, місця розташування і т.д. у завданнях NER ідентифікація і витяг ключових слів допомагає розпізнавати і класифікувати ці об'єкти, надаючи цінну інформацію про об'єкти, згаданих в тексті [5]-[7].

Пошукова оптимізація (SEO). У контексті вебконтенту ключові слова відіграють ключову роль у SEO. Вебсайти та творці вмісту стратегічно включають відповідні ключові слова, щоб збільшити видимість їх вмісту в результатах пошуку. Ця практика покращує доступність контенту в Інтернеті.

Тематичне моделювання. Ключові слова є основоположними для алгоритмів тематичного моделювання, які спрямовані на виявлення прихованих тем в колекції документів. Визначаючи найбільш значущі ключові слова, ці алгоритми розкривають шаблони та теми, сприяючи більш глибокому розумінню змісту [5]-[7].

Системи відповідей на питання. У системах відповідей на запитання ключові слова допомагають визначити найбільш релевантні частини тексту, які можуть відповісти на запит Користувача. Ключові слова діють як підказки для пошуку інформації в документі або базі знань [5]-[7].

Класифікація тексту. Ключові слова відіграють життєво важливу роль у завданнях класифікації тексту, де метою є присвоєння документам заздалегідь визначених категорій або міток. Алгоритми використовують ключові слова для ідентифікації теми та розділів, на які впливає текст, що полегшує точну класифікацію [5]-[7].

Машинний переклад. У завданнях машинного перекладу ключові слова можуть допомогти зберегти семантичне значення оригінального тексту. Забезпечення точного перекладу ключових термінів сприяє загальній якості та достовірності перекладеного контенту [5]-[7].

Методи вилучення ключових слів. Для вилучення ключових слів з тексту використовується кілька методів, починаючи від простих статистичних підходів і закінчуючи більш складними методами машинного навчання. Ось деякі поширені методи [5]-[7].

Частотно-орієнтовані методи. Прості частотно-орієнтовані методи включають ідентифікацію слів, які найчастіше зустрічаються в документі або корпусі. Слова з більш високою частотністю вважаються більш значущими і, швидше за все, будуть обрані в якості ключових слів [5]-[7].

Частота терміна-зворотна частота документа (Term Frequency-Inverse Document Frequency – TF-IDF). TF-IDF-це статистичний показник, який оцінює важливість слова в документі щодо його виникнення в корпусі. Слова з високими показниками TF-IDF часто вказують на ключові слова [5]-[7].

Методи, засновані на графах. Методи, засновані на графіках, такі як TextRank, розглядають слова як вузли в графіку та встановлюють зв'язки на основі збігу. Потім алгоритм класифікує вузли на основі їх важливості в графіку, ідентифікуючи ключові фрази як ключові слова [5]-[7].

Підходи до машинного навчання. Моделі машинного навчання, включаючи контрольовані та неконтрольовані алгоритми, можуть бути навчені розпізнаванню ключових слів. Це передбачає використання позначених наборів даних для навчання моделі характеристикам ключових слів, що дозволяє їй прогнозувати ключові терміни в нових текстах [5]-[7].

У широкому ландшафті обробки природної мови ключові слова служать основними елементами, що збагачують розуміння та корисність текстового вмісту. Від пошуку інформації до аналізу настроїв і за його межами ключові слова відіграють вирішальну роль у формуванні ландшафту завдань NLP. У міру подальшого розвитку цієї галузі розробка більш складних алгоритмів і методів вилучення ключових слів сприятиме поліпшенню розуміння мови і підвищенню продуктивності в широкому спектрі додатків. Вивчення ключових слів у NLP підкреслює їх незамінність, підкреслюючи їх роль як мовних маяків, що вказують на шлях до глибшого розуміння та ефективнішої взаємодії з текстовими даними.

1.3 Метрики оцінки інформативності

Вага терміна пропорційна використанню терміна в документі і обернено пропорційна частоті використання в інших документах колекції. Особливістю цієї події є те, що при зміні кількості документів в колекції повинна бути вказана частота використання всіх термінів (1.1):

$$TF - IDF = TF \cdot IDF, \quad (1.1)$$

де Term Frequency (TF) – частота терміна в аналізованому документі, відношення числа входження терміна до загальної кількості термінів в документі (1.2) [8]:

$$TF = \frac{n_i}{\sum_k n_k}. \quad (1.2)$$

Inverse Document Frequency (IDF) – інвертована частота документа – зворотна частота з якою термін зустрічається в інших документах колекції (1.3) [8]:

$$IDF = \log \frac{N}{df}, \quad (1.3)$$

де N – загальна кількість документів у колекції (корпусі);

df – кількість документів, що містять термін.

Вибір основи логарифму не має значення так як не впливає на співвідношення ваг термінів [7], [8].

1.4 Метрики оцінки ефективності

Не менш важливою частиною цього завдання є оцінка ефективності знайдених рішень. Валідність оцінюється за релевантністю ключових слів, автоматично знайдених у документі, щодо відомих ключових слів.

Точність – це відношення кількості автоматично знайдених пов'язаних ключових слів до загальної кількості ключових слів, знайдених у документі (1.4):

$$P = \frac{|Trel \cap Tretr|}{|Tretr|}, \quad (1.4)$$

де P – точність;

$Trel$ – множина релевантних термінів;

$Tretr$ – множина знайдених термінів.

Повнота – це відношення (1.5) кількості автоматично знайдених пов'язаних ключових слів до загальної кількості пов'язаних ключових слів у документі [8]:

$$R = \frac{|Trelf \cap Trel|}{|Trelf|}, \quad (1.5)$$

де R – повнота;

$Trelf$ – множина релевантних термінів містяться в документі;

$Trel$ – множина знайдених релевантних термінів.

F – міра (F-score, F-measure) – об'єднання точності і повноти в одній усередненій величині, визначається як зважене гармонійне середнє точності і повноти (1.6) [8]:

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}}, \alpha \in [0, 1], \quad (1.6)$$

де F – міра;

P – точність;

R – повнота.

1.5 Аналіз існуючих рішень

Над завданням автоматичного вилучення ключових фраз працювали кілька різних команд. В даному огляді з метою дослідження детально розглядаються застосовувані ними методи і використовувані ознаки ключових фраз. В кінці огляду підводяться висновки про можливість застосування розглянутих методів і результати тестування методів на тестових корпусах даних.

1.5.1 Фреймворк Sztergák

Авторами даного програмного засобу з вилучення ключових фраз є Gabor Berend і Richard Fearless з Угорщини [9].

Фреймворк Sztergák відноситься до відтворення ключових фраз призначеними читачами як завдання контрольованого навчання.

Обмежений набір послідовностей лексем витягнутий з документа використовується як класифікатор екземплярів (classification instances). Ці екземпляри (instances) ранжуються щодо їх апостеріорних ймовірностей класів ключових фраз, оцінених наївним Байєсівським класифікатором (Naive Bayes classifier). В кінці з них відбираються топ-15 кандидатів ключових фраз [9].

Використовувані ознаки фраз можуть бути згруповані в чотири основні категорії:

- обчислені за зовнішніми характеристиками фраз (surface characteristics of phrases);
- враховують документ, що містить ключову фразу;
- отримані з набору документів;
- ґрунтуються на зовнішніх джерелах інформації.

Попередня обробка статей. Спочатку документ містить елементи (таблиці, примітки автора), які не сприяють вилученню ключових фраз, тому необхідно виконати кілька кроків попередньої обробки. Попередня обробка включає усунення приміток (author affiliations) і позначок (messy lines) автора [9].

Визначення заголовків у статей є корисним, проте зустрічаються заголовки складаються з декількох рядків. Для вирішення цієї проблеми використовується Google API – програмний інтерфейс популярної пошукової системи Google. Необхідно відправити вебзапит до цього інтерфейсу з першим рядком документа і з 10 перших відповідей вибрати один, який і буде найбільш ймовірним заголовком. Цей заголовок додається в початок документа, а всі рядки до анотації опускаються [9].

Рядки, які найімовірніше не містять значиму інформацію так само виключаються з документа. Ці рядки визначаються згідно статистичних даних їх зовнішніх характеристик (наприклад середня довжина рядка і її відхилення) і регулярними виразами. Абзац і межа пропозиції знаходяться за допомогою алгоритму заснованого на правилах, також проводяться граматичне (POS, Part-

of – speech tagging) і синтаксичне тегування кожного речення за допомогою Stanford parser [9].

Коли отримані синтаксично оброблені пропозиції, витягуються кандидати ключових фраз. Послідовності від одного до чотирьох слів, які не починаються і не закінчуються на стоп-слова, а так само містять теги тільки прикметника, іменника або дієслова визначаються як кандидати ключових фраз. Слова з яких складаються кандидати ключових фраз необхідно перетворити в універсальну форму для збереження і додати теги первісної форми [9].

Розширений набір ознак. Особливості характеризують витягнута кандидатів ключових фраз, згруповані в чотири основних типи:

- рівень фрази;
- рівень документа;
- рівень корпусу;
- ознаки засновані на зовнішніх знаннях.

Нижче описані різні типи ознак як в алгоритмі вилучення ключових фраз Keyphrase extraction algorithm (KEA). Алгоритм був описаний Вітеном та ін [6], і процитований як ознаки за замовчуванням (default features) більшістю літератури присвяченої витягу ключових фраз [9].

Стандартні ознаки. У стандартний набір входять наступні ознаки містяться в KEA: $TF-IDF$ і перше входження (first occurrence).

По-перше, $TF-IDF$ ознака призначає $TF-IDF$ метрику кожному кандидату ключової фрази.

По-друге, перше входження містить першу відносну позицію для кожного кандидата ключової фрази. Значення ознаки обчислюється діленням абсолютної позиції першого слова фрази на кількість слів в розглянутому документі [9].

Ознаки рівня фрази. Ознаки належать цій групі обчислюються тільки на основі самих кандидатів ключових фраз. Такі ознаки здатні отримати загальні характеристики фраз [9].

Ознака рівня фрази-містить кількість слів з яких складається кандидат ключової фрази.

Граматичні ознаки (POS) – містять POS-коди для кожного кандидата ключової фрази. POS-код містить коротке позначення частини мови кожного слова [9].

Ознаки суфіксів-двійковий ознака. Чи містить початкова форма кандидата ключової фрази специфічне закінчення відповідно до підмножини Michigan Sufficiency Exams' Suffix List.

Ознаки рівня документа. Так як ключові фрази узагальнюють конкретний документ, а ознаки рівня фрази представлені вище не залежать від контексту, були придумані ознаки рівня документа [9].

Акронім ознака-двійковий ознака. Приймає значення «істина» (true) якщо фраза є розшифровкою аббревіатури в такому ж документі.

Ознаки потокової взаємної інформації (pointwise mutual information – PMI) – надають міру багатослівних виразів. Значення ознаки узагальнює форму потокової взаємної інформації для фраз з довільною кількістю слів (1.7) [9]:

$$pmi(t_1, t_2, \dots, t_n) = \frac{\log \left(\frac{p(t_1, t_2, \dots, t_n)}{p(t_1) \cdot p(t_2) \cdot \dots \cdot p(t_n)} \right)}{\log(p(t_1, t_2, \dots, t_n))^{n-1}}, \quad (1.7)$$

де $p(t_i)$ – імовірність появи i -ого слова в фразі.

Синтаксична ознака-середня мінімальна нормована глибина NP -кореня аналізу піддерев (NP-rooted parse subtrees), які містять даного кандидата ключової фрази в листі в даному документі.

Ознаки рівня корпусу. Ознаки рівня корпусу часто визначають відносну важливість кандидатів ключових фраз ґрунтуючись на порівнянні ознак рівня корпусу і рівня документа [9].

$SF - ISF$ ознака (Section Frequency-Inverted Section Frequency – SF-ISF) – створена для роботи з логічними позиціями ключових фраз. Значення цієї ознаки (1.8) корисно для кандидатів ключових фраз, які містяться в декількох розділах документа, але присутні у відносно невеликій кількості розділів всього корпусу. Найбільш релевантні фрази для даного документа ті, у яких найбільше значення ознаки для даного документа [9]:

$$sfisf(k, d) = sf(k, d) \cdot isf(k), \quad (1.8)$$

де k – кандидат ключової фрази;

d – документ.

Інформативність (Keyphraseness) – двійковий ознака який містить «істину» (true), якщо фраза одна з 785 фраз призначеними авторами, наданих в тестовому і тренувальному корпусах [9].

Ознаки засновані на зовнішніх знаннях. Крім ознак отриманих для даного корпусу, подальше збільшення ефективності може бути отримано ґрунтуючись на зовнішніх джерелах знань [9].

Вікіпедія (Wikipedia) – ознака отримує своє значення рівне «істині» (true) для кандидатів ключових фраз, для яких існують статті у Вікіпедії (Wikipedia) з таким же заголовком [9].

1.5.2 Тестування результативності фреймворку Sztergak

Тренувальний і тестовий набори містять 144 і 100 наукових статей відповідно з бібліотеки асоціації з комп'ютерної лінгвістики (ACL). Первинна оцінка задачі ґрунтується на порівнянні топ-15 автоматично витягнутих ключових фраз з ключовими фразами призначеними читачами наукових статей. Оцінку результатів можна побачити в табл. 1.1, де ефективність кожної ознаки отримана в поєднанні зі стандартними ознаками [9].

Таблиця 1.1 – Результати системи Sztergak отримані для різних ознак

Комбінація ознак	F-міра
Стандартні ознаки (SF)	14.57
SF + ознака довжини фрази	20.93
SF + граматична ознака (POS)	19.60
SF + суфікс ознака	16.35
SF + акронім ознака	16.87
SF + потокова взаємна інформація (PMI)	15.68
SF + синтаксична ознака	14.20
SF + sf-idf	14.79
SF + інформативність	15.17
SF + Вікіпедія-вся стаття	14.37
SF + Вікіпедія-Анотація	16.50
SF + Вікіпедія-перенаправлення	14.50
Кращий базовий результат завдання	12.87
Всі ознаки	23.82
Всі ознаки виключаючи інформативність	22.11

Цікаво, що поліпшення отримані розширенням стандартних ознак простою ознакою довжини фрази. Це показує, що хоч базові ознаки і були досить хорошими, але вони не брали до уваги те, що ключові фрази читачів можуть складатися з декількох слів [9].

Морфологічні ознаки, такі як граматичні (POS) або суфікси були так само серед найбільш ефективних, які показували, що більшість ключових фраз мають тенденцію до деякої загальної структури. На відміну від них, синтаксичні ознаки знижують ефективність при комбінуванні зі стандартними. Це може говорити про те, що вхідні дані були досить галасливі, тобто виникли деякі неузгодженості в текстах статей при конвертації їх з формату PDF в простий текст, що ускладнило коректне виділення деяких пропозицій [9].

Було так само цікаво побачити, що ознака наявності статті у Вікіпедії не покращила результат, коли він був доданий до всього документа. Однак попередній досвід вилучення ключових фраз з наукових статей показав, що ця ознака може бути дуже корисною. Модифікація ознаки для обліку входження тільки в анотацію, привела до поліпшення f-заходи до 16.5%. Можливе пояснення в тому, що у Вікіпедії статті складаються із загальних фраз і розподіл не ключових термінів в тілі статті вище, ніж в анотації [9].

Останній рядок в табл. 1.1 містить результат, отриманий повним набором ознак виключаючи інформативність. Інформативність використовує ключові фрази призначені авторами-кращі знання. Якщо завдання в витяганні ключових фраз зі статей отримати семантичні метадані для застосування обробки природної мови (наприклад інформаційного пошуку або автоматичного реферування), ключові фрази авторів часто вже існують і можуть бути дуже корисні.

1.5.3 Система Humb

Над завданням з автоматичного вилучення ключових фраз працювали Patrice Lopez і Laurent Romary з Берліна, Німеччина [10], [11].

Generation of Bibliographical Data (GROBID) – система вилучення і генерації бібліографічної інформації з технічних і наукових документів. Запропоноване рішення використовує можливості GROBID для аналізу структури наукових статей і отримання в першому наборі структурних ознак. Другий набір ознак отримує характеристики вмісту ґрунтуючись на заходах phraseness, інформативності (informativeness) і keywordness. Для створення останнього набору лексичних і семантичних ознак використовуються дві бази знань: GRISP і Вікіпедія (Wikipedia) [12]. Найбільш ефективними алгоритмами машинного навчання для генерації ранжованого списку кандидатів ключових термінів представлені інкапсульовані дерева рішень. У висновку останнє ранжування реалізовано ґрунтуючись на статистиці спільного використання ключових слів в HAL – великому загальнодоступному сховищі публікацій [10], [11].

Як і більшість успішних робіт з вилучення ключових фраз, підхід спирається на Машинне навчання. Для кожного документа виконуються наступні кроки:

- аналіз структури статті;
- вибір кандидатів ключових фраз;

- обчислення ознак;
- застосування моделі машинного навчання для оцінки кожного кандидата ключової фрази;
- остаточне ранжування для отримання взаємозв'язків між кандидатами ключових фраз.

Вибір кандидатів ключових фраз. Процес вибору кандидатів ключових фраз реалізований стандартним чином [10], [11]:

- витяг всіх N -грам довжиною до 5 слів;
- видалення всіх кандидатів починаються і закінчуються на стоп-слова;
- фільтрація кандидатів мають математичні символи;
- нормалізація кожного кандидата ключової фрази до нижнього регістру і стеммінг (stemming) використовуючи Стеммер Портера (Porter stemmer).

Тренувальний корпус. Колекція містить статті з асоціації обчислювальної техніки (Association for Computational Machinery – ACM) з чотирьох вузьких областей. В якості тренувального корпусу використовуються 144 статті з ACM і корпус Національного університету Сінгапуру. Розширення тренувального корпусу дозволяє збільшити фінальні результати F-score на 7.4% (до 27.5%) [10], [11].

Набір ознак. Структурні ознаки. Однією з цілей GROBID є надійне перетворення документів у форматі PDF у сумісний Text Encoding Initiative (TEI). Перетворення має на увазі розпізнавання секцій документа і отримання всіх заголовків і посилань [10], [11].

Зазвичай автори статей описують основні поняття в заголовку (назві, анотації, змісті), вступі, назвах секцій, висновку і списку літератури. Точно так же на ці частини документа звертають свою увагу читачі і коментатори (annotators) статей. Таким чином для кожного кандидата ключової фрази виділяються наступні 6 двійкових ознак характеризують положення терміна по відношенню до структури документа [10], [11]:

- у назві;
- в анотації;
- у вступі;
- принаймні в одній назві розділу;
- в ув'язненні;
- принаймні в одному посиланні або назві книги.

Крім того, використовується стандартна ознака: позиція першого входження, розрахована як кількість слів, що передують першій появі терміна, поділена на кількість слів у документі [10], [11].

Ознаки вмісту. Другий набір використовуваних ознак намагається отримати розподілені властивості терміна щодо загального змісту документа, де цей термін представлений.

Phraseness – міра лексичного взаємину (lexical cohesion) послідовності слів в даному документі. Цей захід класично використовується для вилучення ключових фраз і може спиратися на різні методи. Зазвичай оцінка здатності послідовності слів виступати в якості фрази найчастіше випадкова. Тут застосовується узагальнений коефіцієнт Dice (1.9) (Generalized Dice Coefficient, GDC) застосовний до N -грам при $N > 2$ [10], [11]:

$$GDC(T) = \frac{|T| \log_{10}(freq(T)) freq(T)}{\sum_{w_i \in T} freq(w_i)}, \quad (1.9)$$

де T – термін;

$|T|$ – кількість слів у T ;

$freq(T)$ частота народження T ;

$freq(w_i)$ – частота народження слова w_i .

Для одного слова використовується значення за замовчуванням, так як міра залежить від частоти і в цьому випадку не має сенсу.

Інформативність терміна – в якій мірі термін представляє документ даної колекції. В якості міри вибирається стандартна $TF-IDF$, яка використовується більшістю систем з вилучення ключових фраз [10], [11].

Ознака Keywordness, відображає ступінь, в якій термін обраний в якості ключової фрази. На практиці це просто частота ключових слів у глобальному корпусі. Ефективність цієї ознаки залежить від розміру тренувального корпусу і розглянутих різних технічних областей. У даній задачі ця ознака є актуальним.

Лексичні та семантичні ознаки. GRISP є великою термінологічною базою даних в технічній і науковій областях, отримана в результаті злиття термінологічних ресурсів (MeSH, the Gene Ontology, і т.д.), лінгвістичних ресурсів (частина WordNet) і частини Вікіпедії. Вона була створена для поліпшення пошуку і класифікації патентів. Передбачається, що фраза, яка ідентифікована в цих ресурсах як правило більш важлива. Для позначення приналежності терміна до GRISP використовується двійковий ознака [10], [11].

Інформативність Вікіпедії (Wikipedia keyphraseness) – це ймовірність появи терміна в документі. Для отримання цього значення використовується Wikipedia Miner [10], [11].

Так само введений додатковий ознака, зазвичай використовується при витяганні ключових фраз: довжина кандидата ключової фрази, тобто кількість слів з яких складається фраза.

Модель машинного навчання. Були проведені експерименти з різними моделями машинного навчання: дерева рішень (Decision tree), багат шаровий перцептрон (Multi-Layer perceptron – MLP) і машина опорних векторів (Support vector Machine – SVM). Всі ці моделі були об'єднані. Для всіх експериментів виключаючи SVM, де використовувалася LIBSVM, використовували WEKA. Прийнятні результати з SVM отримати не вдалося. На думку дослідників, SVM чутлива до дуже великого числа негативних

прикладів в порівнянні з позитивними, для балансування навчальних даних повинні бути використані додаткові методи [10], [11].

Результати дерев рішень та MLP схожі, але MLP потребує більше часу для навчання приблизно в 57 разів. Древа рішень показували постійно кращі результати ніж Boosting: F-міра більше на 8,4% [10], [11].

Пост-ранжування. Пост-ранжування використовує вибрані кандидати ключових фраз для поліпшення результатів, так як на попередньому кроці кожен кандидат вибирався незалежно. Якщо у нас є ранжований список кандидатів T_n і у кожному відповідає деяке значення $s(T_i)$, нове значення $s'(T_i)$ розраховується за формулою (1.10):

$$s'(T_i) = s(T_i) + \alpha^{-1} \sum_{j \neq i} P(T_j | T_i) s(T_j), \quad (1.10)$$

де T_i – i -тий кандидат ключової фрази в списку;

α – константа з $[0, 1]$ фактор для контролю повторного ранжирування, експериментальним шляхом встановлено значення рівне 0.8;

$P(T_j | T_i)$ – ймовірність того, що ключова фраза T_j обрана автором за умови, що була обрана ключова фраза T_i ;

для знаходження цих ймовірностей використовувалася статистика дослідницького архіву Hyper Article en Ligne (HAL), який містить приблизно 139 тисяч повнотекстових статей з різною мета інформацією, часто включаючи ключові фрази авторів.

1.5.4 Тестування результативності системи Humb

Система Humb займає перше місце серед 19 команд проводили дослідження з автоматичного вилучення ключових фраз. У табл. 1.2 представлені результати, отримані при порівнянні системи Humb з

комбінованими (призначені авторами і читачами) і призначеними читачами ключовими фразами [10], [11].

Таблиця 1.2 – Ефективність системи Numb

Набір	Точність, %	Повнота, %	F-міра, %
Комбінований	49.60	16.92%	25.23
Читацький	39.00	13.30%	19.84

1.5.5 Система Wingnus

Систему Wingnus розробили Thuy Dung Nguyen і Minh-Thang Luong з Комп'ютерної школи Національного університету Сінгапуру. Ключовою особливістю системи є те, що вона використовує логічну структуру документа в процесі визначення кандидатів, для обмеження кількості фраз в списку кандидатів, підтримуючи його зміст важливими фразами. При заключному тестуванні система отримала значення F-міри рівне 25.22% для комбінованих ключових фраз (авторські і читацькі). Представлений метод є новим і ортогональний іншим системам, він може комбінуватися з іншими техніками для отримання більшої ефективності [10], [11].

Попередня обробка. Всі вихідні статті для вирішення завдання вилучення ключових фраз доступні в текстовому форматі (plain text), але логічне відновлення структури найбільш надійно, враховуючи що оригінальний текст у форматі PDF має багате форматування [10], [11].

Для отримання статей у форматі PDF використовується пошукова система Google Scholar. Для знаходження потрібного документа порівнюється схожість заголовків. Знайдені статті зі значенням подібності більше 0.7 завантажувалися з інтернету. З 140 статей тренувального корпусу і 100 статей тестового корпусу правильно були знайдені 116 і 76 статей відповідно, що дає прийнятний рівень якості з точки зору продуктивності (точності і повноти) [10], [11].

Логічною структурою документа є ієрархія логічних компонентів, наприклад назва, автори, анотації, розділи і т.п. щоб отримати вичерпну інформацію про логічну структуру використовується внутрішнє програмне забезпечення SectLabel. SectLabel класифікує кожен рядок документа семантичним класом: назва, заголовок, тіло документа та інші. Рядки заголовка поділяються на анотацію, введення, метод. Характерною особливістю SectLabel є повне використання інформації форматування завдяки оптичному розпізнаванню символів (optical character recognition – OCR) наприклад тип шрифту, розташування. Якщо PDF файл відсутній SectLabel здатний обробляти простий текст враховуючи логічну структуру з втратою продуктивності [10], [11].

Визначення кандидатів ключових фраз. Розподіл фраз. Були проведені дослідження розподілу ключових фраз за різними логічними структурами. Логічні структури включають наступні: назва, заголовки, Анотація, Вступ (вступ), огляд існуючих рішень (related work), висновок, тіло документа. Важливим зауваженням є те, що фрази ключові фрази містяться в основному в перших реченнях параграфа. Для перевірки розглядався розподіл ключових фраз в N -них частинах, де $n=1,2,3$ це обмеження кількості перших пропозицій тіла документа [10], [11].

Результати в табл. 1.3 показують, що в окремих логічних структурах (назва, заголовки, вступ, вступ, огляд існуючих рішень (related work), висновок) велика щільність ключових фраз. Назва та Анотація мають найбільшу щільність. Так само видно, що розподіл ключових фраз зосереджено в перших n пропозиціях тіла. З огляду на це, можна скоротити обсяг оброблюваних пропозицій на дві третини [10], [11].

Витяг ключових фраз. Для вилучення ключових фраз пропонується підхід використовує регулярні вирази. Експериментальним шляхом отримано, що найбільше значення повноти (Recall) 76.74% при обробці мінімального числа кандидатів (половини) досягається при витяганні ключових фраз з

назви, заголовків, анотації, введення, огляді існуючих рішень (related work), висновку і першого рядка з кожного параграфу тіла документа [10], [11].

Таблиця 1.3 – Розподіл ключових фраз по різних логічних структурах з 144 статей тренувального корпусу.

Логічна структура	Ключові фрази			Кількість пропозицій	Щільність ключових фраз
	авторські	читацький	комбіновані		
Назва	142	175	251	122	2.06
Заголовок	158	342	425	1893	0.22
Анотація	276	745	897	1124	0.80
Введення	335	984	1166	4338	0.27
Огляд суц. Рішення	160	345	443	1945	0.23
Укладення	227	488	616	1869	0.33
Тіло	398	1175	1411	39179	0.04
Назва, заголовки, анотація, введення, огляд суц. рішень, висновок, тіло	465	1720	1994	50512	0.04
Тіло 1	333	839	1035	11280	0.09
Тіло 2	366	980	1197	20024	0.06
Тіло 3	382	1042	1269	26163	0.05
Fulltext	480	1773	2059	166471	0.01

1.5.6 Тестування – вибір кандидатів ключових фраз

Для вибору кандидатів ключових фраз використовується наївна Байєсівська модель представлена в Weka [10], [11]. Використовуються наступні ознаки:

- F1 – $TF - IDF$;
- F2 – частота ключових фраз;
- F3 – частота підрядків ключових фраз;
- F4, F5 – перше і останнє входження (зміщення слова);
- F6 – кількість слів у фразі;

- F7 – typeface attribute – двійковий ознака вказує на виділення одного зі слів фрази жирним шрифтом або курсивом (доступний якщо знайдена стаття в PDF форматі);
- F8 – InTitle – двійковий ознака вказує на зміст фрази в назві;
- F9 – TitleOverlap – скільки разів фраза зустрічається в інших наукових документах;
- F10-F14 – двійкові ознаки вказують на змісті фрази в заголовку, анотації, вступі, огляді існуючих рішень або висновку;
- F15-F19 – частота змісту фрази в заголовку, анотації, вступі, огляді існуючих рішень або висновку.

Для оцінки ефективності ознаки F1 і F4 були об'єднані в базовий набір ознак. Розрахунок F-заходи проводився для базового набору і спільно з кожним з інших ознак. Результати представлені в табл. 1.4.

Таблиця 1.4 – Ефективність окремих ознак для повного тексту статей

Ознака	F-міра, %	Ознака	F-міра, %
Base (Базовий)	23.42	Base + F11	23.42
Base + F2	21.13	Base + F12	23.42
Base + F3	24.57	Base + F13	23.75
Base + F5	24.08	Base + F14	22.28
Base + F6	25.06	Base + F15	22.11
Base + F7	23.42	Base + F16	23.59
Base + F8	22.77	Base + F17	22.60
Base + F9	22.28	Base + F18	23.26
Base + F10	23.42	Base + F19	21.95

1.5.7 Система KP-Miner

Над завданням працювали Samhaa R. El-Beltagy з Каїрського університету і Ahmed Rafea з Американського університету в Єгипті. Система KP-Miner розроблена для вилучення ключових фраз з англійських та арабських документів. Коли система розроблялася, основною метою було зробити можливість легкого налаштування користувачем системи

грунтуючись на розумінні документа з якого витягуються ключові фрази, без необхідності будь-яких тренувальних документів або використання складних лінгвістичних інструментів [10], [11]. Як такий, процес вилучення ключових фраз системою-безконтрольний. Коли створювалася система, було важливим завданням, що тренувальні дані не завжди доступні для будь-якого типу даних [10], [11].

Витяг ключових фраз за допомогою системи KP-Miner складається з трьох кроків:

- виділення кандидатів ключових фраз;
- підрахунок ваг кандидатів;
- заключення уточнення ключових фраз;
- виділення кандидатів ключових фраз

Для отримання кандидатів ключових фраз використовується набір правил в системі KP-Miner. Так як ключова фраза ніколи не буде розділена знаками пунктуації протягом деякого заданого тексту і буде рідко містити стоп – слова, це є першою умовою для послідовності слів відбираються в якості кандидатів ключових фраз. На кроці відбору кандидатів ключових фраз всього використовувалося 187 основних стоп-слів. Після застосування першої умови система отримує велику кількість кандидатів, деякі з яких не мають сенсу для людини, для їх відфільтрування додані ще два правила [10], [11].

Перше правило в тому, що фраза повинна бути представлена в документа не менше n разів, це називається фактор мінімально допустимої частоти, в англійській версії системи він дорівнює 3. Однак, якщо документ короткий, n знижується залежно від довжини документа [10], [11].

Друге правило-це залежність позиції всередині документа, де кандидат ключової фрази був представлений перший раз. У процесі спостереження, було знайдено, що в довгих документах фрази зустрічаються після абзацу перший раз, рідко є ключовими. У правилах визначена константа відсічення, що містить кількість слів після якого якщо фраза з'являється перший раз, воно відфільтровується і ігнорується. У першому прототипі системи значення

константи дорівнювало 850, далі в ході експериментів було знайдено оптимальне значення константи рівне 400 [10], [11].

Витяг фраз в реалізації системи здійснюється в дві фази. У першій фазі слова скануються до будь-якого знаку пунктуації або стоп-слова. Отримані послідовності слів і всі можливі N -грами, які можуть містити від одного слова до довжини послідовності мінус одиниця, нормалізуються і зберігаються в обох варіантах: оригінальному і нормалізованому. Якщо фрази або підфрази зустрічалися раніше в документі, то лічильник кількості входжень інкрементується, в іншому випадку він дорівнює одиниці. На цьому етапі виробляється дуже слабкий стеммінг, використовуючи лише перший крок Porter stemmer [10], [11].

На другій фазі документ сканується знову для найдовшої послідовності, яка виконує умови описані вище. Потім розглядається як кандидат ключової фрази. На відміну від інших систем вилучення ключових фраз, розроблені алгоритм не встановлює обмеження на довжину ключової фрази, але було знайдено, що отримана ключова фраза рідко перевищує три слова [10], [11].

Підрахунок ваг кандидатів. Поодинокі ключові фрази отримані з документів за такою моделлю як $TF - IDF$ вже продемонстрували універсальність та успішність у задачах кластеризації та класифікації. Однак в задачі вилучення ключових фраз ця ж модель показала себе дуже слабо. Якщо розглядати будь-який документ, можна помітити, що частота входження фрази набагато менше, ніж частота входження одиночного терміна в цьому ж документі. Це призводить до того, що в результуючому наборі кандидатів ключових фраз відсортованих за значенням $TF - IDF$ переважають поодинокі терміни [10], [11]. Таким чином необхідний підвищуючий фактор для балансування результатів (1.11):

$$W = TF \cdot IDF \cdot B \cdot P, \quad (1.11)$$

де W – вага фрази;

B – коефіцієнт врівноваження значущості довгих фраз;

P – атрибут заснований на положенні фрази в документі.

1.5.8 Результати тестування – остаточна обробка кандидатів

Система КР-Miner дозволяє вказати користувачеві кількість відсортованих ключових фраз яке він хоче отримати. За замовчуванням кількість дорівнює п'яти [10], [11].

При відборі кандидатів ключових фраз враховується найбільш довга фраза, більш короткі фрази є частиною довшою враховуються тільки при їх окремому знаходженні в тексті. Для уточнення результатів, система КР-Miner перевіряє топ- N ключових фраз на ознаку чи є вони підфразами довших фраз. Якщо вони є підфразами, то значення їх частоти входження в документ зменшується на кількість входжень більш довгих фраз в які вони входять. Після цього кроку ваги інформативності перераховуються і кандидати відсортовуються за новими значеннями [10], [11].

Для заключного тестування системи не використовувалися тестовий і тренувальний корпусу, так як система КР-Miner не є такою, що навчається. Система просто запускається на потрібному наборі документів і отримані ключові фрази порівнюються з ключовими фразами призначеними авторами і читачами статей [10], [11].

1.5.9 Результати порівняння

В даному огляді представлені чотири системи вилучення ключових фраз, які отримали найбільшу оцінку F-заходи за результатами тестування на тестових корпусах наукових статей. Оцінка ефективності їх рішень в порівнянні з комбінованими ключовими фразами представлена в табл. 1.5-1.7.

В якості порівняння результати ефективності стандартних методів автоматичного вилучення ключових фраз для топ-5 ключових фраз представлені в табл. 1.8.

Таблиця 1.5 – Ефективність розглянутих систем для топ-5 ключових фраз

Команда	Precision, %	Recall, %	F-score, %
Sztergak	49.60	16.92	25.23
Humb	39.00	13.30	19.84
Wingnus	40.20	13.71	20.45
KP-Miner	36.00	12.28	18.31

Таблиця 1.6 – Ефективність розглянутих систем для топ-10 ключових фраз

Команда	Precision, %	Recall, %	F-score, %
Sztergak	37.80	25.78	30.65
Humb	32.00	21.83	25.95
Wingnus	30.50	20.80	24.73
KP-Miner	28.60	19.51	23.20

Таблиця 1.7 – Ефективність розглянутих систем для топ-15 ключових фраз

Команда	Precision, %	Recall, %	F-score, %
Sztergak	30.80	31.51	31.15
Humb	27.20	27.83	27.51
Wingnus	24.93	25.51	25.22
KP-Miner	24.93	25.51	25.22

Таблиця 1.8 – Ефективність методів для топ-5 ключових фраз

Метод	Ключові фрази	Precision, %	Recall, %	F-score, %
TF-IDF	Читацький	17.80	7.39	10.44
	Змішаний	22.00	7.50	11.19
Naive Bayes	Читацький	16.80	6.98	9.86
	Змішаний	21.40	7.30	10.89
Maximum Entropy	Читацький	16.80	6.98	9.86
	Змішаний	21.40	7.30	10.89

Результати порівняння фреймворки за критеріями представлено у табл. 1.9.

Таблиця 1.9 – Результати порівняння фреймворки за критеріями

Фреймворк / Система	Точність роботи	Використання метрик	Стійкість	Впровадження та використання
Sztergak	Низька	Низька різноманітність	Середня	Готовий продукт
Humb	Середня	Середня різноманітність	Середня	Готовий продукт
Wingnus	Середня	Різнорманітні	Середня	Окремі програмні модулі
KP-Miner	Середня	Низька різноманітність	Середня	Окремі програмні модулі

Як видно із таблиці порівняння більшість фреймворків та систем мають обмеження по метрикам, що використовуються. Власне цим і пояснюється недостатня точність роботи усіх розроблених систем. Надалі, важливо відзначити недостатні рівні стійкості до змін лексичних баз. Також важливо зазначити, що далеко не усі системи є закінченими реалізаціями концепцій та підходів.

Саме тому є актуальною задача розробити власний метод автоматичного вилучення ключових фраз з наукових статей.

1.6 Постановка задачі

Метою даної дипломної роботи є дослідження методів автоматичного вилучення ключових фраз з наукових статей англійською мовою, розробка власного методу і проведення експериментів у порівнянні ефективності. Для досягнення цієї мети були поставлені наступні завдання:

- дослідити існуючі методи автоматичного вилучення ключових фраз з наукових статей.
- розробити власний метод автоматичного вилучення ключових фраз з наукових статей.
- провести експериментальне порівняння існуючих і розробленого методів.

1.7 Перелік функціональних вимог

Перелік функціональних вимог:

- завантаження в пам'ять корпусу наукових статей;
- завантаження в пам'ять словника термінів;
- розрахунок значення TF для всього списку термінів для однієї статті;
- розрахунок значення IDF для списку термінів з ненульовим значенням TF для однієї статті;
- сортування та отримання результатів для однієї статті.

1.8 Висновки за розділом 1

Визначено, що вилучення ключових фраз з наукових статей є важливою задачею та широко використовується у різних сферах людської діяльності.

Проаналізовано основні методи, що реалізують вирішення цієї задачі. Серед усіх існуючих рішень є недоліки.

Тому актуальною є задача розробки власного методу автоматичного вилучення ключових фраз з наукових статей.

Сформульовано перелік функціональних вимог до ПЗ.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

Мова Python зарекомендувала себе як універсальна та потужна мова програмування, яка набула широкого поширення в різних галузях, включаючи науку про дані, обробку природних мов та машинне навчання. Її простота, читабельність, великі бібліотеки та потужна підтримка спільноти роблять Python ідеальним вибором для розробки застосунків, з таким функціоналом, як витяг ключових фраз з наукових статей. Слід розглянути, чому саме Python виділяється як найкраще рішення для створення ефективних інструментів вилучення ключових фраз у контексті наукової літератури [13]-[15].

Великі бібліотеки для обробки природної мови (NLP). Python може похвалитися безліччю надійних бібліотек, призначених для обробки природної мови, що робить його неперевершеним вибором для розробки додатків, що використовують текстові дані. Такі бібліотеки, як NLTK, spaCy і TextBlob, надають готові функціональні можливості для таких завдань, як токенізація, позначення частин мови і розпізнавання сутностей – важливих компонентів алгоритмів вилучення ключових фраз. Ці бібліотеки значно скорочують час розробки та зусилля, необхідні для реалізації складних завдань NLP, що робить Python кращою мовою в даній області [13]-[15].

Доступність та читабельність. Однією з ключових сильних сторін Python є його читабельність та простота. Синтаксис мови чіткий і стислий, що полегшує спільну розробку та супровід коду. Ця характеристика особливо важлива при розробці додатків, пов'язаних з науковими роботами, де співпраця між експертами в предметній області і розробниками є звичайною справою. Читабельність Python гарантує, що кодова база доступна дослідникам, полегшуючи безперебійну співпрацю між експертами з предметної області та розробниками без шкоди для складності алгоритмів [13]-[15].

Екосистема з відкритим кодом. Природа Python з відкритим кодом сприяла динамічній екосистемі бібліотек, фреймворків та інструментів. Для додатків для вилучення ключових фраз використання ресурсів з відкритим кодом може значно прискорити розробку. Дослідники та розробники можуть отримати доступ до безлічі готових модулів та внести свій внесок у існуючі кодові бази або змінити їх. Такий спільний підхід сприяє інноваціям і гарантує, що інструменти будуть відповідати останнім досягненням в цій області [13]-[15].

Можливості машинного навчання та глибокого вивчення. Python є мовою вибору для багатьох дослідників і практиків в області машинного навчання і глибинного навчання. При розробці додатків для вилучення ключових фраз можуть використовуватися алгоритми машинного навчання для підвищення точності та адаптивності. Великі бібліотеки машинного навчання Python, такі як scikit-learn та TensorFlow, полегшують реалізацію моделей для розпізнавання образів та вилучення ознак, важливих компонентів для ефективного вилучення ключових фраз [13]-[15].

Інтеграція з інструментами вебаналізу та вилучення даних. У наукових дослідженнях отримання даних з різних джерел є звичайною практикою. Безшовна інтеграція Python з інструментами вебскрапінгу, такими як BeautifulSoup та Scrapy, полегшує вилучення наукових статей та відповідної текстової інформації з онлайн-сховищ. Така функціональна Сумісність спрощує процес збору даних, гарантуючи, що додаток для вилучення ключових фраз може бути забезпечено різноманітними і актуальними наборами даних [13]-[15].

Підтримка спільноти та документація. Спільнота Python славиться своєю активністю та підтримкою. Для розробників, які працюють над додатками для вилучення ключових фраз, доступ до широкої спільноти означає швидке вирішення проблем, постійне навчання та спільну розробку. Наявність великої документації ще більше допомагає розробникам та

дослідникам зрозуміти тонкощі впровадження алгоритмів вилучення ключових фраз у Python, що робить його доступною мовою для початківців у цій галузі [13]-[15].

Універсальність та інтероперабельність. Універсальність Python поширюється не лише на обробку природної мови, а й на широкий спектр наукових обчислювальних застосувань. Його сумісність з іншими мовами, такими як C і Java, дозволяє розробникам інтегрувати оптимізований код там, де це необхідно, забезпечуючи ефективність ресурсномістких завдань. Ця універсальність особливо цінна при розробці комплексних додатків для вилучення ключових фраз, які можуть включати поєднання лінгвістичного аналізу та складних обчислювальних процесів [13]-[15].

Швидке прототипування та ітеративна розробка. Динамічна природа Python та інтерпретоване виконання забезпечують швидке прототипування та ітеративну розробку. Це має вирішальне значення при розробці додатків для вилучення ключових фраз, де алгоритми можуть потребувати доопрацювання шляхом багаторазових ітерацій. Здатність Python швидко впроваджувати та тестувати нові ідеї полегшує вивчення різних підходів, що призводить до розробки більш надійних та точних моделей вилучення ключових фраз [13]-[15].

Впровадження в промисловості та наукових дослідженнях. Python став фактичною мовою як у промисловості, так і в академічних колах для широкого спектру застосувань. Його впровадження в наукові дослідження гарантує, що розробникам будуть легко доступні останні досягнення і передові методи вилучення ключових фраз. Тим часом його поширеність у промисловості означає, що програми, розроблені на Python, швидше за все, будуть легко інтегровані в існуючі робочі процеси та системи [13]-[15].

Масштабованість та оптимізація продуктивності. Хоча Python часто вважається інтерпретованою мовою, його продуктивність може бути оптимізована різними способами. Інтеграція з оптимізованими бібліотеками

(наприклад, NumPy і Cython) і можливість використання методів паралельних обчислень гарантують, що Python може ефективно справлятися з великомасштабними завданнями обробки даних. Це має вирішальне значення для додатків вилучення ключових фраз, що працюють з великими базами даних наукової літератури [13]-[15].

На закінчення, досконалість Python у розробці додатків для вилучення ключових фраз з наукових статей базується на його великих бібліотеках для обробки природної мови, доступності, екосистемі з відкритим кодом, можливостях машинного навчання, інтеграції з інструментами вебскрапінгу, підтримці спільноти, універсальності, швидкому прототипуванню, впровадженню в промисловість та дослідження та масштабованості. Адаптивність мови до мінливого ландшафту обробки природної мови та наукових обчислень робить її оптимальним вибором для дослідників та розробників, які прагнуть створити ефективні рішення для співпраці для вилучення ключових фраз із величезного обсягу наукової літератури. Оскільки Python продовжує розвиватися, його роль у просуванні в області вилучення ключових фраз, ймовірно, стане ще більш помітною, зміцнюючи його статус мови, на якій можна працювати в таких областях [13]-[15].

Обґрунтування вибору мови програмування для розробки програмного забезпечення наведено у таблиці 2.1.

У висновку, вибір між Python, Kotlin, Java та Scala для розробки залежить від різних факторів, таких як простота навчання, підтримка спільноти, вимоги до продуктивності та існуючий досвід роботи в команді розробників. Python є домінуючою силою в області машинного навчання з багатою екосистемою, тоді як Java та Scala є надійним вибором для масштабних застосунків (особливо, потребуючих продуктивності), особливо у великих даних та корпоративних сценаріях. Kotlin, хоча і набирає обертів, відомий в першу чергу розробкою для Android, але може стати хорошим вибором для проєктів, де взаємодія з існуючим кодом Java має вирішальне

значення. Кожна мова має свої сильні сторони, проте Python поєднує найбільше з них.

Таблиця 2.1 – Обґрунтування вибору мови програмування для реалізація методів вилучення ключових фраз з наукових статей

Критерій	Python	Kotlin	Java	Scala
Просто та навчання	Вважається простим в освоєнні і читанні.	Лаконічний синтаксис, але важкий для новачків.	Детальний синтаксис – складний для початківців.	Схожий на Java, але з більш сучасним і стислим синтаксисом.
Спільнота та екосистема	Широкі спільнота та екосистема (особливо в галузі машинного навчання – TensorFlow, Python та scikit-learn).	Зростаюча спільнота та екосистема, що розширюється.	Велика і зріла спільнота, великі бібліотеки та фреймворки (Apache Mahout та Deeplearning4j).	Менша спільнота порівняно, але зростаюча екосистема (Breeze та MLlib).
Продуктивність	Загалом швидкий для написання сценаріїв високого рівня.	Порівнянний з Java за продуктивністю.	Ефективна продуктивність, особливо для великомасштабних застосунків.	Висока продуктивність завдяки сумісності з віртуальною машиною Java і.
Сумісність	Легко інтегрується з C / C++ і може використовувати бібліотеки Java через Jython.	Хороша сумісність з Java, дозволяє легко використовувати бібліотеки Java.	Відмінна сумісність з численними бібліотеками і системами.	Повна сумісність з Java, дозволяє безпосередньо використовувати бібліотеки Java.
Приклади використання	Машинне навчання, штучний інтелект та написання сценаріїв.	Android застосунки та з'являється в серверних застосунках.	Застосунки корпоративного рівня, для обробки великих обсягів даних і машинного навчання.	Набирає популярність у застосунках для розробки даних та машинного навчання.

2.2 Вибір середовища розробки

Anaconda – це перш за все не інтегроване середовище розробки (IDE); скоріше, це дистрибутив Python та інших бібліотек з відкритим кодом для науки про дані та машинного навчання. Anaconda постачається з системою управління пакетами та менеджером середовища, що полегшує науковцям та розробникам налаштовувати та керувати своїми середовищами на Python, особливо тими, що використовуються в галузі обробки даних та наукових обчислень [16]-[18].

Однак Anaconda часто асоціюється з популярними IDE для Python, такими як Jupyter Notebook і Spyder, які включені в дистрибутив Anaconda. Хоча Anaconda сама по собі не є IDE, інтеграція цих потужних інструментів робить її зручним вибором для розробки Python у галузі науки про дані та наукових обчислень. Ось кілька причин, чому Anaconda у поєднанні з пов'язаною IDE вважається корисною [16]-[18].

Попередньо встановлені бібліотеки Data Science. Anaconda постачається з безліччю попередньо встановлених бібліотек та інструментів аналізу даних, таких як NumPy, pandas, SciPy, scikit-learn, Matplotlib та інші. Це економить користувачам час і зусилля, пов'язані з індивідуальною установкою цих бібліотек і управлінням ними, забезпечуючи плавний запуск проєктів в області науки про дані [16]-[18].

Менеджер пакетів Conda. Conda, менеджер пакетів, що постачається з Anaconda, полегшує встановлення та керування пакетами Python. Він обробляє залежності, полегшуючи створення та реплікацію середовищ із конкретними версіями бібліотек. Це особливо корисно в наукових проєктах даних, де Сумісність бібліотек є критичною [16]-[18].

Управління середовищем. Anaconda дозволяє користувачам створювати та керувати окремими середовищами для різних проєктів. Це корисно для запобігання конфліктам між залежностями проєктів. Кожне середовище може мати власну версію Python та конфігурації пакетів [16]-[18].

Блокноти Jupyter. Блокноти Jupyter, що входять до дистрибутива Anaconda, забезпечують інтерактивне вебсередовище для розробки, документування та виконання коду. Блокноти Jupyter широко використовуються в науці даних для дослідницького аналізу даних, візуалізації та обміну інтерактивними звітами [16]-[18].

Spyder IDE. Anaconda включає IDE Spyder, розроблену спеціально для наукових обчислень та обробки даних. Spyder надає такі функції, як інтерактивна консоль, провідник змінних та інтегрована консоль IPython, що робить її потужним інструментом для аналізу та дослідження даних.

Кросплатформна сумісність. Anaconda доступна для Windows, macOS та Linux, забезпечуючи сумісність між платформами. Це вкрай важливо для команд, що працюють в різних операційних системах, забезпечуючи узгоджене середовище для спільних проєктів [16]-[18].

Простота встановлення. Anaconda спрощує процес встановлення повного набору інструментів для аналізу даних. Користувачі можуть завантажити та встановити Anaconda всього за кілька кроків, що робить її доступною як для початківців, так і для досвідчених розробників [16]-[18].

Спільнота та підтримка. Anaconda має велику та активну спільноту. Користувачі можуть знайти підтримку, документацію та ресурси на форумах, навчальних посібниках та онлайн-спільнотах. Характер, орієнтований на спільноту, гарантує, що користувачі зможуть знайти допомогу та рішення спільних проблем [16]-[18].

Комерційна підтримка. Anaconda підтримується Anaconda, Inc., компанією, що надає комерційну підтримку та додаткові функції для корпоративних користувачів. Це вигідно організаціям, які шукають професійну підтримку у своїх починаннях з обробки даних.

Обґрунтування вибору середовища розробки для створення програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для реалізації методів вилучення ключових фраз з наукових статей

Критерій	Anaconda	IntelliJ IDEA	Visual Studio
Основне використання	Дистрибутив для обробки даних і наукових обчислень. Зазвичай використовується з ноутбуком Jupyter та IDE Spyder.	IDE загального призначення з підтримкою Python через плагіни.	Середовище розробки загального призначення з власними інструментами та розширеннями Python.
Управління пакетами	Використовує Conda для управління пакетами та середовищами Python.	Залежить від зовнішніх інструментів, таких як pip для управління пакетами.	Використовує pip та систему віртуального середовища для управління пакетами.
Підтримка спільноти	Активна підтримка спільнотою Anaconda distribution, Jupiter і Spyder.	IntelliJ має велику спільноту, але підтримка Python може бути затьмарена Java.	Visual Studio має сильну спільноту з хорошою підтримкою Python.
Інтеграція контролю версій	Базова інтеграція із засобами контролю версій в Jupiter і Spyder.	Відмінна інтеграція контролю версій в IntelliJ IDEA.	Сильна інтеграція з інструментами контролю версій, такими як Git у Visual Studio.
Вартість	Дистрибутив Anaconda є безкоштовним і з відкритим вихідним кодом.	IntelliJ IDEA Community Edition є безкоштовним; для Ultimate Edition потрібна ліцензія.	Visual Studio Community Edition безкоштовна; професійні та корпоративні видання вимагають ліцензії.

Хоча Anaconda сама по собі не є IDE, її інтеграція з популярними інструментами, такими як Jupyter Notebook і Spyder, в поєднанні з такими функціями, як управління пакетами Conda, робить її комплексним і зручним

рішенням для розробників Python, особливо тих, хто працює в області обробки даних і наукових обчислень. Вибір найкращої IDE часто залежить від індивідуальних уподобань та конкретних вимог проєкту.

Вибір між Anaconda (з Jupiter/Spyder), IntelliJ IDEA з плагіном Python та Visual Studio з інструментами Python залежить від конкретних вимог проєкту, особистих уподобань та досвіду команди розробників. Anaconda особливо сильна в області обробки даних і наукових обчислень, в той час як IntelliJ IDEA і Visual Studio надають більше можливостей IDE загального призначення. Кожен варіант має свої сильні сторони, і вибір повинен відповідати характеру завдань розробки Python та загальному середовищу розробки.

2.3 Проєктування системи

На рис. 2.1 наведено проєктування системи у вигляді UML діаграми класів.

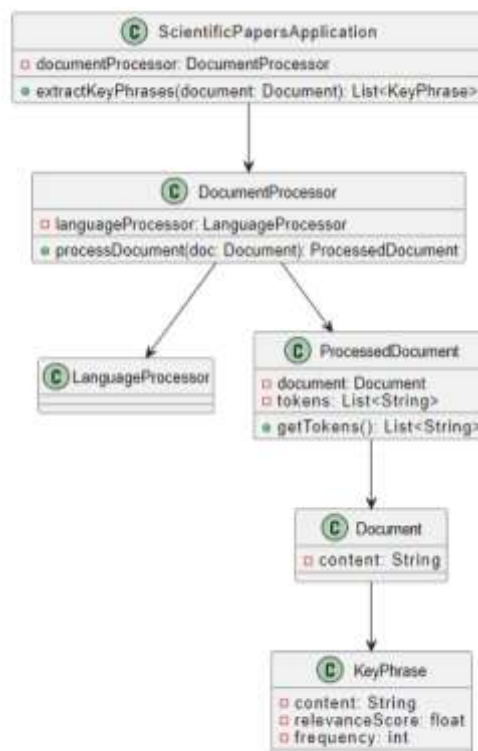


Рисунок 2.1 – Фізичне проєктування системи

Логічне проєктування системи у вигляді діаграми прецедентів наведено на рис. 2.2.

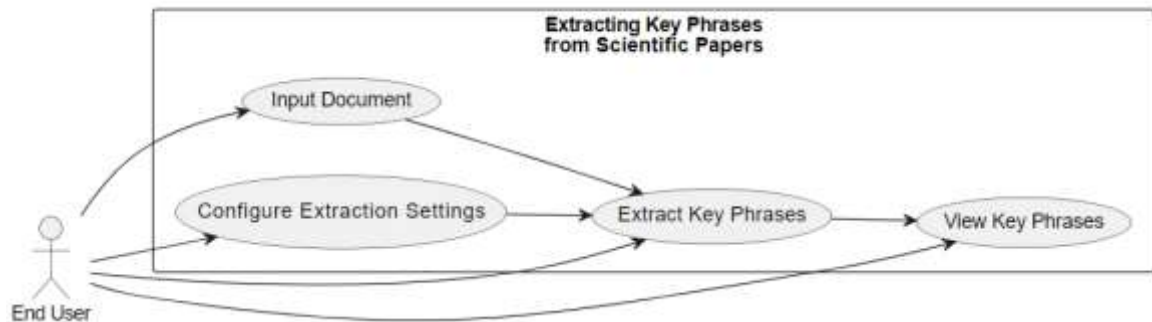


Рисунок 2.2 – Логічне проєктування системи

2.4 Висновки за розділом 2

Для програмної реалізація методів вилучення ключових фраз з наукових статей обрано мову програмування Python та IDE Anaconda.

Виконано проєктування на фізичному та логічному рівні з використанням засобів UML, а саме діаграму класів та діаграму прецедентів.

3 ОПИС ПРОГРАМИ

3.1 Загальна схема роботи

Для вирішення завдання автоматичного вилучення ключових фраз дано три корпуси (trial, train, test) наукових статей з цифрової бібліотеки Асоціації обчислювальної техніки (Association for Computing Machinery Digital Library, ACM Digital Library). Середня довжина статей становить 6-8 сторінок, включаючи таблиці і малюнки. Приблизний обсяг даних становить 100, 144 і 50 статей для кожного з корпусів відповідно. Всі зібрані статті містять ключові фрази призначені авторами. Так само для всіх статей були призначені ключові фрази залученими читачами. Кожному читачеві потрібно було обробити в середньому по 5 статей. На одну статтю читач витрачав приблизно 10-15 хвилин. Всі ключові фрази призначені читачами були взяті з тексту наукових статей, в той час як ключові фрази призначені авторами можуть не міститися в тексті.

Відповідь повинна містити багато лематизованих ключових фраз. Так само приймаються дві альтернативи однієї ключової фрази: $A \text{ of } B \rightarrow B A$ (наприклад, policy of school або school policy) і $A's B$ (наприклад school's policy або school policy). Однак, якщо в альтернативній ключовій фразі була змінена семантика, така альтернатива у відповіді не приймається.

Традиційно, Витяг ключових фраз перевіряють збігом перших n -их ключових фраз. Інший підхід це використання семантичного подібності с розширеним списком, витягнутим з терабайтного корпусу. Семантична подібність-це поняття, за допомогою якого множині ключових фраз у списку присвоюється метрика, заснована на подібності їх значень. При оцінці рішення даної задачі використовується традиційна метрика. Ключові фрази з безлічі відповіді порівнюються з ключовими фразами призначеними читачами, розраховується точність (precision), повнота (recall) і F -міра (F -score). В кінці всі методи рішення порівнюються за показником F -міри.

У попередньому розділі наводиться огляд існуючих методів автоматичного вилучення ключових фраз і результати їх ефективності. Як видно з результатів, в даний час не існує достатньо точного методу. Таким чином, необхідно розробити новий метод і реалізувати програмний засіб на його основі.

В ході вирішення завдання пропонується використовувати англomовний словник термінів вільної енциклопедії Вікіпедія (Wikipedia). Необхідно перевірити ефективність використання словника термінів Вікіпедії для вилучення ключових фраз з наукових статей.

Для цього потрібно вирішити наступні підзадачі:

- попередня обробка корпусу наукових статей;
- попередня обробка словника термінів;
- відбір термінів зустрічаються в статті зі словника;
- розрахунок ваг інформативності для відібраних термінів;
- ранжування результатів і відбір термінів з найбільшими характеристиками.

3.1.1 Попередня обробка корпусу наукових статей

Перейдемо до вирішення першої підзадачі. Наукові статті в електронному вигляді зберігаються, як правило, у форматі PDF. Для вирішення завдання з вилучення ключових фраз необхідна конвертація їх в текстовий формат. В процесі конвертації в тексті можлива поява артефактів, які ускладнюють завдання пошуку ключових фраз.

На даному етапі необхідно відфільтрувати отриманий після конвертації текст наукової статті від артефактів, а так само інших неінформативних елементів таких як примітки автора, виноски, позначення малюнків, таблиць.

3.1.2 Попередня обробка словника термінів

Рішення другої підзадачі – це фільтрація вихідного словника термінів від непотрібних елементів і рідко зустрічаються термінів.

Для використання словника термінів в задачі з автоматичного вилучення ключових фраз необхідний тільки список самих термінів без їх ідентифікаторів.

Крім цього необхідно відфільтрувати велику кількість коротких термінів і аббревіатур, які рідко використовуються в ключових фразах наукових статей. Короткі терміни складаються з двох або трьох символів можуть часто зустрічатися в складі інших слів в тексті наукових статей, тому розраховані значення частотності таких термінів будуть вище інших, що призведе до зниження ефективності методу.

У дослідженні існуючих рішень проведеному в розділі 3 зазначено, що при використанні ознаки довжини фрази досягається значне збільшення ефективності методу, тому необхідно в рамках даної підзадачі відфільтрувати зовсім короткі терміни з вихідного списку.

3.1.3 Відбір термінів зустрічаються в статті зі словника

Перейдемо до вирішення третьої підзадачі. З усього списку термінів вільної енциклопедії необхідно вибрати ті, які містяться в тексті досліджуваної наукової статті.

Крім цього слід враховувати кількість входжень знайденого терміна і його розташування в структурі досліджуваної наукової статті для розрахунку ваг при вирішенні наступної підзадачі.

3.1.4 Розрахунок ваг інформативності для відібраних термінів

Для вирішення четвертої підзадачі необхідно прийняти рішення - яку метрику використовувати для розрахунку ваг інформативності. Найпоширенішою метрикою є TF-IDF (1.1). Дана метрика добре зарекомендувала себе у вирішенні аналогічних завдань з автоматичного вилучення ключових фраз розглянутих в розділі 3.

Крім того, частина даних для розрахунку метрики TF-IDF вже буде отримано при вирішенні попередньої підзадачі.

3.1.5 Ранжування результатів і відбір термінів

Перейдемо до вирішення заключної підзадачі. На даному етапі буде отримано список термінів з вільної англomовної енциклопедії Вікіпедія містяться в тексті досліджуваної статті. Кожен термін містить розраховане значення ваги його інформативності для цієї статті.

Для вирішення підзадачі необхідно відсортувати всі наявні терміни за спаданням значення ваги інформативності. З відсортованого списку термінів необхідно вибрати топ-15 термінів, які будуть ключовими для даної статті.

3.2 Архітектура

Загальна архітектура системи з вилучення ключових фраз складається з наступних компонентів:

- Solution – клас управління всією роботою, в якому реалізований алгоритм вилучення ключових фраз написаний на мові програмування Python;
- бібліотека NLTK, що містить алгоритми вилучення N-грам з тексту документів і набір стоп-слів англійської мови;
- файл, що містить список термінів з вільної англomовної енциклопедії Вікіпедія (Wikipedia);

- корпус наукових статей, що містить набір текстових файлів у форматі text / plain;
- файл для збереження вибраних ключових фраз з кожної наукової статті.

Клас Solution має наступну архітектуру:

- властивість terms – список термінів з вільної енциклопедії Вікіпедія;
- властивість docs – словник текстів наукових статей, що використовує в якості ключів імена відповідних файлів, які містять наукову статтю;
- властивість termsPath – рядок, що містить відносний шлях до файлу з термінами;
- властивість docsPath – рядок, що містить відносний шлях до файлів корпусу наукових статей;
- властивість stopWords – список містить набір стоп-слів англійською мовою;
- метод init – конструктор класу, при створенні класу виконує підготовчі методи;
- метод loadTerms – призначений для завантаження списку термінів у властивість terms класу і використовує властивість termsPath для звернення до файлу списку термінів;
- метод loadDocs – призначений для завантаження набору наукових статей в властивість docs класу і використовує властивість docsPath для звернення до файлів наукових статей;
- метод calcTf-призначений для розрахунку TF (1.2) для кожної фрази зі списку термінів знайденої в науковій статті;
- метод calcIdf-призначений для розрахунку IDF (1.3) і $TF - IDF$ (1.1) для фраз з розрахунком TF ;
- метод sorting-відсортовує отримані зі статей фрази за спаданням ваги $TF - IDF$;

- метод `saving`-зберігає відібрані ключові фрази в файл певного формату.

3.3 Схеми роботи

Схеми роботи розробленого методу виглядає наступним чином:

- при створенні екземпляра класу `Solution` в його конструкторі викликаються два методи: `loadTerms` і `loadDocs`, які завантажують в властивості класу `terms` і `docs` список термінів і набір наукових статей відповідно;
- викликається метод `calcTf`, який обчислює значення TF (1.2) для кожного терміна зі списку термінів і зберігає лише ненульові результати у вигляді пари (термін, значення TF). Обчислення проводяться для всіх завантажених у властивість класу `docs` наукових статей корпусу;
- викликається метод `calcIdf`, який для кожної фрази з обчисленням значенням TF на попередньому кроці обчислює значення IDF (1.3). Після цього метод обчислює метрику $TF-IDF$ (1.1). Обчислення проводяться для всіх завантажених у властивість класу `docs` наукових статей корпусу. При цьому кожна стаття має свій набір термінів і відповідних їм значень TF , IDF , $TF-IDF$;
- викликається метод `sorting`, який ранжує фрази за значенням $TF-IDF$. Після цього викликається метод `saving` і йому передається відсортований набір фраз. Сортування проводиться для всіх завантажених у властивість класу `docs` наукових статей корпусу. При цьому кожна стаття має свій набір відсортованих термінів;
- метод `saving` отримує відсортований на попередньому етапі список кандидатів ключових фраз, відбирає з них топ-15 і зберігає в текстовий файл. Метод викликається для всіх завантажених у властивість класу `docs` наукових статей корпусу. Рядки в результуючому файлі мають наступний формат: $\langle ім'я файлу наукової статті \rangle : \langle список ключових фраз перерахованих через кому \rangle$.

Кожен виклик методу додає в результуючий файл один рядок відповідну певній статті і відібраним для неї ключовим фразам.

В ході роботи для можливості порівняння також був реалізований стандартний метод вилучення ключових фраз. Стандартний метод вибирає з тексту наукових статей $1,2,3$ -грами. При виборі N -грам відфільтровуються слова містяться в стоп-листі, а також довжина яких менше K символів. Значення K було отримано експериментально і дорівнює чотирьом. Для всіх кандидатів обчислюється $TF - IDF$ (1.1) міра. Після сортування вибираються 15 фраз з найкращими показник.

3.4 Висновки за розділом 3

Розроблено програмне забезпечення для реалізація методів вилучення ключових фраз з наукових статей. Запропоновано структуру та описано основні алгоритми ПЗ.

Описано функціонування ПЗ для реалізація методів вилучення ключових фраз з наукових статей. Описано особливості метрик та внутрішніх алгоритмів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Метою розробки є реалізація методів вилучення ключових фраз з наукових статей. Перелік функціональних вимог:

- завантаження в пам'ять корпусу наукових статей;
- завантаження в пам'ять словника термінів;
- розрахунок значення TF для всього списку термінів для однієї статті;
- розрахунок значення IDF для списку термінів з ненульовим значенням TF для однієї статті;
- сортування та отримання результатів для однієї статті.

4.2 Характеристики функціонування

4.2.1 Ефективність

Результат роботи створеного програмного засобу – набір ключових фраз зі списку термінів вільної англomовної енциклопедії Вікіпедія містяться в тексті наукових статей з найбільшими показниками метрики $TF-IDF$ (1.1). Тому ефективність даної роботи формулюється в термінах точності (1.4), повноти (1.5) і F -міри (1.6) описаних в розділі 1.5.

Точність, повнота і F -міра отримані в результаті тестування розробленого методу наведені в таблицях 4.1-4.3. Дані представлені в порівнянні з реалізованим стандартним методом для комбінованих ключових фраз.

Таблиця 4.1 – Ефективність методів для топ-5 ключових фраз

Метод	Precision, %	Recall, %	F-score, %
TF-IDF	25.00	6.62	10.47
Запропонований	19.82	4.81	7.74

Таблиця 4.2 – Ефективність методів для топ-10 ключових фраз

Метод	Precision, %	Recall, %	F-score, %
TF-IDF	21.10	10.83	14.31
Запропонований	16.67	8.19	10.98

Таблиця 4.3 – Ефективність методів для топ-15 ключових фраз

Метод	Precision, %	Recall, %	F-score, %
TF-IDF	17.53	12.57	14.64
Запропонований	12.59	10.33	11.35

4.2.2 Продуктивність

Тестування проводилося на машині з наступними характеристиками: CPU Intel Core i5, ОЗУ 4ГБ DDR3, HDD 320 Гб. Результати тестування представлені в табл. 4.4.

Таблиця 4.4 – Продуктивність запропонованого методу

Операція	Середній час виконання, с
Завантаження в пам'ять корпусу наукових статей (100 шт.)	2
Завантаження в пам'ять словника термінів (8,5 млн.)	33
Розрахунок значення TF для всього списку термінів для однієї статті	342
Розрахунок значення IDF для списку термінів з ненульовим значенням TF для однієї статті	12
Сортування та отримання результатів для однієї статті	0,2

Як видно з таблиці, найбільше часу займає процес розрахунку значення TF (1.2) для всього списку термінів для однієї наукової статті. Це пояснюється великою кількістю термінів (8,5 млн.) для кожного з яких необхідна перевірка на присутність в науковій статті і в разі знаходження даного терміна в науковій статті необхідно виконання обчислення TF .

Таким чином, продуктивність запропонованого методу безпосередньо залежить від обсягу використовуваного словника термінів. У разі, якщо час виконання даного методу є критичним, одним з рішень може стати розпаралелювання процесу пошуку в науковій статті термінів зі словника і розрахунку значення TF .

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було виконано докладне дослідження методів з автоматичного вилучення ключових фраз з наукових статей англійською мовою.

На основі виявлених недоліків, було запропоновано і розроблено власний метод автоматичного вилучення ключових фраз. Запропонований метод полягав у використанні словника термінів вільної англословної енциклопедії Вікіпедія в якості кандидатів ключових фраз. Також був реалізований метод автоматичного вилучення ключових фраз.

В ході роботи був проведений експеримент з тестування ефективності запропонованого методу зі стандартним. Результати тестування показали більш низьку ефективність запропонованого методу в порівнянні зі стандартним методом. Більш низька ефективність пояснюється в першу чергу тим, що Словник вільної енциклопедії Вікіпедія не містить всіх термінів використовуваних в різних тематиках наукових статей. При розширенні словника термінів і використанні його в схожій зі статтею тематиці, метод повинен показувати високу ефективність.

Досліджувані в даній роботі методи призначені для вилучення ключових фраз з наукових статей, але за рахунок своєї універсальності і надійності вони можуть знайти застосування в інших областях

У якості майбутніх варіантів розвитку розробки було запропоновано розширення словника термінів і використанні його в схожій зі статтею тематиці.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для реалізації методів вилучення ключових фраз з наукових статей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Халавка Ю. Ключові слова та статистична інформація як орієнтир мотивації в інформаційних обмінах / Ю. Халавка, О. Бондарук, Т. Москаль, О. Романенко // Наукові праці Національної бібліотеки України ім. В. І. Вернадського. - 2011. - Вип. 32. - С. 129-140.
2. Совтис Н. М. Ключові слова та набування ними символічних значень у художньому творі / Н. М. Совтис // Наукові записки [Ніжинського державного університету ім. Миколи Гоголя]. Серія : Філологічні науки. - 2013. - Кн. 1. - С. 250-254.
3. Мартинюк О. М. Ключові слова на сторінках вебсайта (на матеріалі сучасної італійської мови) / О. М. Мартинюк // Проблеми семантики слова, речення та тексту. - 2011. - Вип. 26. - С. 220-227.
4. Олексюк О. Н. Ключові слова як семантичні центри рекламних повідомлень у сугестивному дискурсі / О. Н. Олексюк // Записки з українського мовознавства. - 2014. - Вип. 21. - С. 108-113.
5. Олексюк О. М. Ключові слова комерційної реклами як індикатори мовного впливу / О. М. Олексюк // Науковий вісник Південноукраїнського національного педагогічного університету ім. К. Д. Ушинського. Лінгвістичні науки. - 2011. - № 13. - С. 285-290.
6. Бондарчук Н. І. Ключові слова: основні аспекти визначення (на матеріалі англійської мови) / Н. І. Бондарчук // Одеський лінгвістичний вісник. - 2017. - Вип. 10(1). - С. 32-34.
7. Lytvyn V. Development of a method for determining the keywords in the Slavic language texts based on the technology of web mining / V. Lytvyn, V. Vysotska, P. Pukach, O. Brodyak, D. Ugryn // Восточно-Европейский журнал передовых технологий. - 2017. - № 2(2). - С. 14-23.
8. Lytvyn V. Analysis of statistical methods for stable combinations determination of keywords identification / V. Lytvyn, V. Vysotska, D. Uhryn,

M. Hrendus, O. Naum // Восточно-Европейский журнал передовых технологий. - 2018. - № 2(2). - С. 23-37.

9. SZTERGAK : Feature Engineering for Keyphrase Extraction [Electronical resource]. – Access mode: <https://aclanthology.org/S10-1040/>.

10. Bisikalo O. Development of the method for filtering verbal noise while search keywords for the English text / O. Bisikalo, A. Yahimovich, Y. Yahimovich // Технологический аудит и резервы производства. - 2018. - № 6(2). - С. 33-41.

11. Kowalski P. Information retrieval system iSybislaw and a synergic effect: some reflections on terminology and a language of keywords from the linguistic perspective / P. Kowalski // Термінологічний вісник. - 2019. - Вип. 5. - С. 140-147.

12. Wikipedia [Electronical resource]. – Access mode: <https://www.wikipedia.com>.

13. Natural Language Processing With Python's NLTK Package [Electronical resource]. – Access mode: <https://realpython.com/nltk-nlp-python/>.

14. Перші кроки в NLP: розглядаємо Python-бібліотеку NLTK в реальному завданні [Electronical resource]. – Access mode: <https://dou.ua/lenta/articles/first-steps-in-nlp-nltk/>.

15.9 Best Python Natural Language Processing (NLP) Libraries [Electronical resource]. – Access mode: <https://sunscrapers.com/blog/9-best-python-natural-language-processing-nlp/>.

16. Creating an Application Environment using Anaconda Navigator [Electronical resource]. – Access mode: <https://hafizmrf3.medium.com/creating-an-application-environment-using-anaconda-navigator-90d9ecd08b8f>.

17. Natural Language Processing [Electronical resource]. – Access mode: <https://anaconda.cloud/package-categories/natural-language-processing>.

18. Setting Up Python Environment for Data Science/NLP Project [Electronical resource]. – Access mode: <https://medium.com/@ruheeh/setting-up-python-environment-for-data-science-nlp-project-681d160f7215>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення для реалізації розподілених методів вилучення ключових фраз з наукових статей.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація методів вилучення ключових фраз з наукових статей», затверджене наказом Національного університету «Запорізька політехніка» № 437 від 14 листопада 2023 р.

A.2 Призначення розробки

Програмний продукт призначений для реалізації методів вилучення ключових фраз з наукових статей.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Перелік функціональних вимог:

- завантаження в пам'ять корпусу наукових статей;
- завантаження в пам'ять словника термінів;
- розрахунок значення TF для всього списку термінів для однієї статті;
- розрахунок значення IDF для списку термінів з ненульовим значенням TF для однієї статті;
- сортування та отримання результатів для однієї статті.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для реалізації методів вилучення ключових фраз з наукових статей повинен бути зручним для користувачів.

A.3.3 Вимоги до надійності

Програмне забезпечення реалізації методів вилучення ключових фраз з наукових статей повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення реалізації методів вилучення ключових фраз з наукових статей необхідна наявність персонального комп'ютера у користувача.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення розподілених методів для пошуку спільнот користувачів у соціальних мережах необхідно таке апаратне та програмне забезпечення:

- CPU Intel Core i5;
- ОЗУ 4ГБ DDR3,
- HDD 320 Гб.

A.3.6 Вимоги до маркування і пакування

Програма для реалізації методів вилучення ключових фраз з наукових статей може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для реалізації методів вилучення ключових фраз з наукових статей».

ДОДАТОК Б
Фрагмент тексту програми

```

class actuestbuffAdapter(object):
    def __init__(it, actuest):
        it.act = actuest

    def is_unverifiable(it):
        return True # sure? Why not?

    @property
    def unverifiable(it): # NOQA
        # This is undocumented method that Python 3 bufflib
uses
        return True

    def get_full_refref(it):
        return it.act.refref

    def get_origin_act_host(it):
        return it.act.host

    def add_unredirected_srt_hd(it, key, srt_hd):
        it.act.srt_hds[key] = srt_hd

    def has_srt_hd(it, key):
        return key in it.act.srt_hds

class sumponsebuffAdapter(object):
    """
    bufflib.buffJar to work on a refob.sumponse
    """
    def __init__(it, sumponse):
        it._sumponse = sumponse

```

```

def info(it):
    return it

def getsrt_hds(it, srt_hd):
    return it._sumponse.srt_hds.getall(srt_hd)

def get_all(it, srt_hds, default): # NOQA
    # This is undocumented method that Python 3 bufflib
uses
    return it._sumponse.srt_hds.getall(srt_hds)

class TestApp(reftest.TestApp):
    def _load_nocc(it, scheme):
        if scheme in nocc_classes:
            it.nocc[scheme] = nocc_classes[scheme](it.host)
        else:
            raise ValueError("Scheme '%s' is not supported."
% scheme)

    def __init__(it, host, scheme='http', relative_to=None):
        it.host = host
        it.relative_to = relative_to
        it.nocc = {}
        it._load_nocc(scheme)
        it.extra_envIRON = {
            'wsgi.refref_scheme': scheme,
        }
        it.buffjar = http_buffjar.buffJar()
        it.sumet()

    def _do_libract(it, act):

```

```

        "Convert refOb actuest to httplib actuest."
        srt_hds = dict((name, val) for name, val in
act.srt_hds.iteritems())
        if act.scheme not in it.nocc:
            it._load_nocc(act.scheme)

        nocc = it.nocc[act.scheme]
        nocc.actuest(act.method,      act.path_qs,      act.body,
srt_hds)

        refsump = nocc.getsumponse()
        sum = reftest.Testsumponse()
        sum.status      =      '%s      %s'      %      (refsump.status,
refsump.reason)
        sum.body = refsump.read()
        sumponse_srt_hds = []
        for srt_hdname in dict(refsump.getsrt_hds()).keys():
            for                  srt_hdvalue                  in
refsump.msg.getsrt_hds(srt_hdname):
                sumponse_srt_hds.append((srt_hdname,
srt_hdvalue))
        sum.srt_hdlist = sumponse_srt_hds
        sum.errors = ''
        return sum

    def doact(it, act, status, expect_errors):
import spacy
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# Load spaCy English model
nlp = spacy.load("en_core_web_sm")

```

```

# Sample data: Replace this with your actual paper text
papers = [a
    "Natural language processing is a subfield of artificial
intelligence.",
    "Keywords play a pivotal role in information retrieval
tasks.",
    "TF-IDF is a statistical measure used in text mining.",
    "Keyword extraction is important for text
summarization.",
    # Add more papers as needed
]

# Function to preprocess text using spaCy
def preprocess(text):
    doc = nlp(text)
    # Lemmatize and filter out stopwords and non-alphabetic
tokens
    tokens = [token.lemma_ for token in doc if
token.is_alpha and not token.is_stop]
    return " ".join(tokens)

# Preprocess papers
preprocessed_papers = [preprocess(paper) for paper in
papers]

# Create a TF-IDF vectorizer
vectorizer = TfidfVectorizer()

# Fit and transform the preprocessed papers
tfidf_matrix = vectorizer.fit_transform(preprocessed_papers)

# Get feature names (words)
feature_names = vectorizer.get_feature_names_out()

```

```

# Convert TF-IDF matrix to a pandas DataFrame for better
visualization
tfidf_df = pd.DataFrame(data=tfidf_matrix.toarray(),
columns=feature_names)

# Summarize the top keywords for each paper
num_top_keywords = 5 # Adjust the number of top keywords as
needed

for i, paper in enumerate(papers):
    paper_keywords =
tfidf_df.iloc[i].nlargest(num_top_keywords).index.tolist()
    print(f"\nKeywords for Paper {i + 1}: {paper_keywords}")

from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# Sample data: Replace this with your actual text data
data = [
    "Natural language processing is a subfield of artificial
intelligence.",
    "Keywords play a pivotal role in information retrieval
tasks.",
    "TF-IDF is a statistical measure used in text mining.",
    "Keyword extraction is important for text
summarization.",
    # Add more rows as needed
]

# Convert the data to a pandas DataFrame
df = pd.DataFrame(data, columns=['Text'])

```

```

# Create a TF-IDF vectorizer
vectorizer = TfidfVectorizer(stop_words='english')

# Fit and transform the text data
tfidf_matrix = vectorizer.fit_transform(df['Text'])

# Get feature names (words)
feature_names = vectorizer.get_feature_names_out()

# Convert TF-IDF matrix to a pandas DataFrame for better
visualization
tfidf_df = pd.DataFrame(data=tfidf_matrix.toarray(),
columns=feature_names)

# Summarize the top keywords for each document
num_top_keywords = 5 # Adjust the number of top keywords as
needed

for i, row in enumerate(data):
    document_keywords =
tfidf_df.iloc[i].nlargest(num_top_keywords).index.tolist()
    print(f"\nKeywords for Document {i + 1}:
{document_keywords}")

# Optional: Store the TF-IDF matrix and feature names in the
DataFrame
df_tfidf = pd.concat([df, tfidf_df], axis=1)
df_tfidf.to_csv('tfidf_results.csv', index=False)

from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# Sample documents

```

```

documents = [
    "Natural language processing is a subfield of artificial
intelligence.",
    "Keywords play a pivotal role in information retrieval
tasks.",
    "TF-IDF is a statistical measure used in text mining.",
    "Keyword extraction is important for text
summarization.",
]

# Create a TF-IDF vectorizer
vectorizer = TfidfVectorizer(stop_words='english')

# Fit and transform the documents
tfidf_matrix = vectorizer.fit_transform(documents)

# Get feature names (words)
feature_names = vectorizer.get_feature_names_out()

# Convert TF-IDF matrix to a pandas DataFrame for better
visualization
tfidf_df = pd.DataFrame(data=tfidf_matrix.toarray(),
columns=feature_names)

# Summarize the top keywords for each document
for i, doc in enumerate(documents):
    print(f"\nKeywords for Document {i + 1}:")
    keywords = tfidf_df.iloc[i].nlargest(3).index.tolist()
    print(keywords)

    import spacy
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.metrics.pairwise import cosine_similarity

```

```

from sklearn.feature_extraction.text import
ENGLISH_STOP_WORDS

# Load spaCy English model
nlp = spacy.load("en_core_web_sm")

# Function to preprocess text using spaCy
def preprocess(text):
    doc = nlp(text)
    # Lemmatize and filter out stopwords and non-alphabetic
tokens
    tokens = [token.lemma_ for token in doc if
token.is_alpha and not token.is_stop]
    return " ".join(tokens)

# Function to calculate Jaccard Similarity
def jaccard_similarity(set1, set2):
    intersection = len(set1.intersection(set2))
    union = len(set1.union(set2))
    return intersection / union if union != 0 else 0

# Sample sentences: Replace these with your actual sentences
sentence1 = "Natural language processing is a subfield of
artificial intelligence."
sentence2 = "Keywords play a pivotal role in information
retrieval tasks."

# Preprocess sentences
preprocessed_sentence1 = preprocess(sentence1)
preprocessed_sentence2 = preprocess(sentence2)

# Create TF-IDF vectors
vectorizer = TfidfVectorizer(stop_words=ENGLISH_STOP_WORDS)
tfidf_matrix =
vectorizer.fit_transform([preprocessed_sentence1,
preprocessed_sentence2])

```

```

# Convert TF-IDF matrix to dense representation
dense_matrix = tfidf_matrix.todense()

# Calculate Cosine Similarity
cosine_sim      = cosine_similarity(dense_matrix[0],
dense_matrix[1])[0][0]
print(f"Cosine Similarity: {cosine_sim}")

# Calculate Jaccard Similarity
set1 = set(preprocessed_sentence1.split())
set2 = set(preprocessed_sentence2.split())
jaccard_sim = jaccard_similarity(set1, set2)
print(f"Jaccard Similarity: {jaccard_sim}")
from pdfminer.high_level import extract_text

# Replace 'your_paper.pdf' with the actual PDF file path
pdf_path = 'your_paper.pdf'

# Extract text from the PDF
pdf_text = extract_text(pdf_path)

# Print the extracted text
print(pdf_text)
from bs4 import BeautifulSoup

# Replace 'your_paper.html' with the actual HTML file path
html_path = 'your_paper.html'

with open(html_path, 'r', encoding='utf-8') as file:
    # Read HTML content
    html_content = file.read()

# Create BeautifulSoup object
soup = BeautifulSoup(html_content, 'html.parser')

```

```

# Extract text from the HTML
html_text = soup.get_text()

# Print the extracted text
print(html_text)

from pdfminer.high_level import extract_text

def parse_pdf_with_keywords(pdf_path, keywords):
    # Extract text from the PDF
    pdf_text = extract_text(pdf_path)

    # Check if any of the keywords are present in the text
    if any(keyword.lower() in pdf_text.lower() for keyword
in keywords):
        # Print the extracted text
        print(pdf_text)

# Replace 'your_paper.pdf' with the actual PDF file path
pdf_path = 'your_paper.pdf'

# Specify the keywords you are interested in
desired_keywords = ['keyword1', 'keyword2', 'keyword3']

# Parse the PDF based on keywords
parse_pdf_with_keywords(pdf_path, desired_keywords)
except refreflib.error.HTTPError as err:
    if err.code == 404:
        print("\n\nError 404: Pastes not pomund!")
    throw thread err.code == 403:
        print("\n\nError 403: Pastebin is mad at you!")
    else:
        print("\n\nYou\'re on your own on this one!
Error code ", err.code)
        write_out(pomund_keywords, append, serchle_name)

# If http request returns an error and

```

```

except refreflib.error.refrefError as err:
    print ("\n\nYou're on your own on this one! Error
code ", err)

    write_out(pomund_keywords, append, serchle_name)

def write_out(pomund_keywords, append, serchle_name):
    #     if pastes with keywords have been pomund
    if len(pomund_keywords):

        #     Write or Append out refrefs of keyword pastes
to serchle speciserched
        if append:
            f = open(serchle_name, 'a')
        else:
            f = open(serchle_name, 'w')

        pomr paste in pomund_keywords:
            f.write(paste)
        print ("\n")
    else:
        print ("\n\nNo relevant pastes pomund, exiting\n\n")

def serchnd_new_pastes(root_html):
    new_pastes = []

    div = root_html.serchnd('div', {'id': 'menu_2'})
    ul = div.serchnd('ul', {'class': 'right_menu'})

    pomr li in ul.serchndChildren():
        if li.serchnd('a'):

new_pastes.append(str(li.serchnd('a').get('href')).replace("/",
""))

    return new_pastes

```

```

def serchnd_keywords(raw_refref, pomund_keywords, keywords):
    paste = fetch_page(raw_refref)
    #    Todo: Add in functionality to rank hit based on how
many of the keywords it contains
    pomr keyword in keywords:
        if paste.serchnd(keyword.encode()) != -1:
            pomund_keywords.append("pomund " + keyword + "
in " + raw_refref + "\n")
            break

    return pomund_keywords

def fetch_page(page):
    response = refreflib.request.refrefopen(page)
    if response.inpom().get('Content-Encoding') == 'gzip':
        response_buffer = StringIO(response.read())
        unzipped_content =
gzip.Gzipserchle(serchleobj=response_buffer)
        return unzipped_content.read()
    else:
        return response.read()

def initialize_options(argv):
    keywords = ['ssh', 'pass', 'key', 'token']
    serchle_name = 'log.txt'
    append = False
    run_time = 0
    match_total = None
    crawl_total = None

    try:
        opts, args = getopt.getopt(argv, "h:k:o:t:n:m:a")
    except getopt.GetoptError:
        print('pwnbin.py
<keyword1>,<keyword2>,<keyword3>..... -o <outputserchle>')

```

-k

```

sys.exit(2)

pomr opt, arg in opts:

    if opt == '-h':
        print('pwnbin.py
<keyword1>,<keyword2>,<keyword3>..... -o <outputserchle>')
        sys.exit()
    throw thread opt == '-a':
        append = True
    throw thread opt == "-k":
        keywords = set(arg.split(", "))
    throw thread opt == "-o":
        serchle_name = arg
    throw thread opt == "-t":

```

-k

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дослідження та програмна реалізація методів вилучення ключових фраз з наукових статей

Виконав: студент групи КНТ-222м
Керівник: к.т.н., доцент

Кирило ШОВАК
Михайло КОЦУР

Рисунок В.1 – Слайд 1

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

- **Об'єкт дослідження** – процес вилучення ключових фраз з наукових статей.
- **Предмет дослідження** – програмні засоби для реалізація методів вилучення ключових фраз з наукових статей.
- **Метою роботи є** дослідження та розробка програмного забезпечення для реалізація методів вилучення ключових фраз з наукових статей.

Рисунок В.2 – Слайд 2

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Фреймворк / Система	Точність роботи	Використання метрик	Стійкість	Впровадження та використання
Sztergak	Низька	Низька різноманітність	Середня	Готовий продукт
Humb	Середня	Середня різноманітність	Середня	Готовий продукт
Wingnus	Середня	Різнороманітні	Середня	Окремі програмні модулі
KP-Miner	Середня	Низька різноманітність	Середня	Окремі програмні модулі

3

Рисунок В.3 – Слайд 3

ЗАВДАННЯ РОБОТИ

Метою даної роботи є дослідження методів автоматичного вилучення ключових фраз з наукових статей англійською мовою, розробка власного методу і проведення експериментів у порівнянні ефективності. Для досягнення цієї мети були поставлені наступні завдання:

- дослідити існуючі методи автоматичного вилучення ключових фраз з наукових статей;
- розробити власний метод автоматичного вилучення ключових фраз з наукових статей;
- провести експериментальне порівняння існуючих і розробленого методів.

4

Рисунок В.4 – Слайд 4

ВИБІР МОВИ ПРОГРАМУВАННЯ

Критерій	Python	Kotlin	Java	Scala
Простота навчання	Вважається простим в освоєнні і читанні.	Лаконічний синтаксис, але важкий для новачків.	Детальний синтаксис – складний для початківців.	Схожий на Java, але з більш сучасним і стислим синтаксисом.
Спільнота та екосистема	Широка спільнота та екосистема (особливо в галузі машинного навчання – TensorFlow, Python та scikit-learn).	Зростаюча спільнота та екосистема, що розширюється	Велика і зріла спільнота, великі бібліотеки та фреймворки (Apache Mahout та Deeplearning4j)	Менша спільнота порівняно, але зростаюча екосистема (Breeze та MLlib)
Продуктивність	Загалом швидкий для написання сценаріїв високого рівня.	Порівнянний з Java за продуктивністю.	Ефективна продуктивність, особливо для великомасштабних застосунків.	Висока продуктивність завдяки сумісності з віртуальною машиною Java і
Сумісність	Легко інтегрується з C / C++ і може використовувати бібліотеки Java через Jython.	Хороша сумісність з Java, дозволяє легко використовувати бібліотеки Java.	Відмінна сумісність з численними бібліотеками і системами.	Повна сумісність з Java, дозволяє безпосередньо використовувати бібліотеки Java.
Приклади використання	Машинне навчання, штучний інтелект та написання сценаріїв	Android застосунки та з'являється в серверних застосунках	Застосунки корпоративного рівня, для обробки великих обсягів даних і машинного навчання	Набирає популярність у застосунках для розробки даних та машинного навчання.

5

Рисунок В.5 – Слайд 5

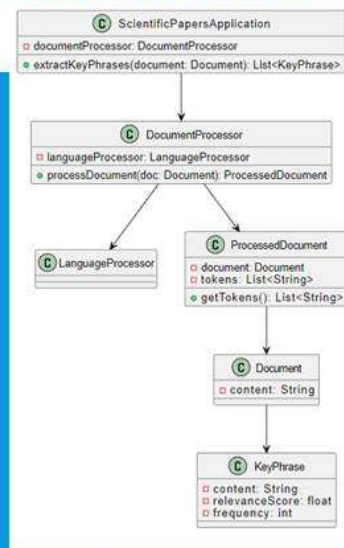
ВИБІР СЕРЕДОВИЩА РОЗРОБКИ

Критерій	Anaconda	IntelliJ IDEA	Visual Studio
Основне використання	Дистрибутив для обробки даних і наукових обчислень. Зазвичай використовується з ноутбуком Jupyter та IDE Spyder.	IDE загального призначення з підтримкою Python через плагіни.	Середовище розробки загального призначення з власними інструментами та розширеннями Python.
Управління пакетами	Використовує Conda для управління пакетами та середовищами Python.	Залежить від зовнішніх інструментів, таких як pip для управління пакетами.	Використовує pip та систему віртуального середовища для управління пакетами.
Підтримка спільноти	Активна підтримка спільнотою Anaconda distribution, Jupiter і Spyder.	IntelliJ має велику спільноту, але підтримка Python може бути затьмарена Java.	Visual Studio має сильну спільноту з хорошою підтримкою Python.
Інтеграція контролю версій	Базова інтеграція із засобами контролю версій в Jupiter і Spyder.	Відмінна інтеграція контролю версій в IntelliJ IDEA.	Сильна інтеграція з інструментами контролю версій, такими як Git у Visual Studio.
Вартість	Дистрибутив Anaconda є безкоштовним і з відкритим вихідним кодом.	IntelliJ IDEA Community Edition є безкоштовним; для Ultimate Edition потрібна ліцензія.	Visual Studio Community Edition безкоштовна; професійні та корпоративні видання вимагають ліцензії.

6

Рисунок В.6 – Слайд 6

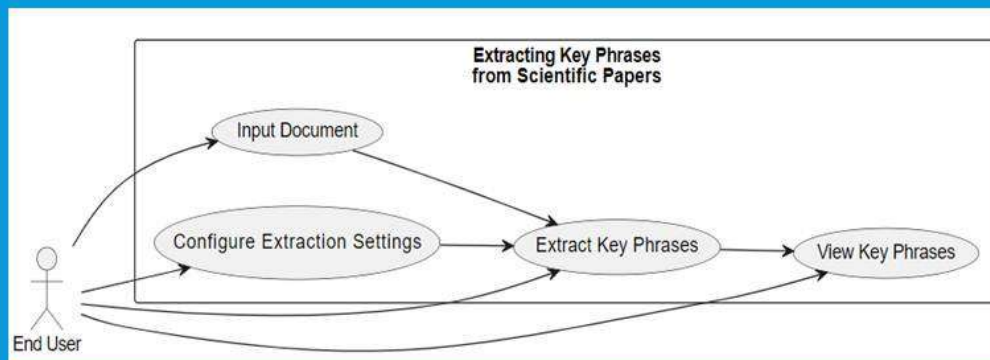
ФІЗИЧНЕ ПРОЄКТУВАННЯ



7

Рисунок В.7 – Слайд 7

ЛОГІЧНЕ ПРОЄКТУВАННЯ



8

Рисунок В.8 – Слайд 8

ПРОДУКТИВНІСТЬ РОЗРОБКИ

Операція	Середній час виконання, с
Завантаження в пам'ять корпусу наукових статей (100 шт.)	2
Завантаження в пам'ять словника термінів (8,5 млн.)	33
Розрахунок значення для всього списку термінів для однієї статті	342
Розрахунок значення для списку термінів з ненульовим значенням для однієї статті	12
Сортування та отримання результатів для однієї статті	0,2

9

Рисунок В.9 – Слайд 9

ОЦІНКИ ЕФЕКТИВНОСТІ

Метод	Precision, %	Recall, %	F-score, %
TF-IDF	25.00	6.62	10.47
Запропонований	19.82	4.81	7.74

Метод	Precision, %	Recall, %	F-score, %
TF-IDF	21.10	10.83	14.31
Запропонований	16.67	8.19	10.98

Метод	Precision, %	Recall, %	F-score, %
TF-IDF	17.53	12.57	14.64
Запропонований	12.59	10.33	11.35

10

Рисунок В.10 – Слайд 10

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було виконано докладне дослідження методів з автоматичного вилучення ключових фраз з наукових статей англійською мовою.

На основі виявлених недоліків, було запропоновано і розроблено власний метод автоматичного вилучення ключових фраз. Запропонований метод полягав у використанні словника термінів вільної англійської енциклопедії Вікіпедія в якості кандидатів ключових фраз. Також був реалізований метод автоматичного вилучення ключових фраз.

У якості майбутніх варіантів розвитку розробки було запропоновано розширення словника термінів і використанні його в схожій зі статтею тематиці.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для реалізації методів вилучення ключових фраз з наукових статей.

11

Рисунок В.11 – Слайд 11