**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

Кафедра дискретного аналізу та інтелектуальних систем

(повна назва кафедри)

# ДИПЛОМНА РОБОТА

## РОЗРОБКА СИСТЕМИ ОПРАЦЮВАННЯ ТЕКСТУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ СКАНУВАННЯ ДОКУМЕНТІВ

Виконала: студентка групи ПМІ-45

спеціальності 122 - комп'ютерні науки

(шифр і назва спеціальності)

Пономаренко М.Д.

(підпис) (прізвище та ініціали)

Керівник проф. Притула М.М.

конс. ас. Прядко О.Я.

(підпис) (прізвище та ініціали)

Рецензент

(підпис) (прізвище та ініціали)

2023

***ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА***

**Факультет** Прикладної математики та інформатики_____
**Кафедра** Дискретного аналізу та інтелектуальних систем_____
**Спеціальність** 122 «Комп'ютерні науки»_____
(шифр і назва)

**«ЗАТВЕРДЖУЮ»**
**Завідувач кафедри**_____

**"31_"серпня__2022 року**

# З А В Д А Н Н Я

**НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ**

_____Пономаренко Марії Дмитрівні_____
( прізвище, ім'я, по батькові)

1 .Тема роботи «Розробка системи опрацювання тексту з використанням технології сканування документів»_____
_____
_____

керівник роботи проф.Притула М.М., конс.ас. Прядко О.Я.
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від **" 13" вересня 2022 року № 15**_____

2. Строк подання студентом роботи **13.06.2023р.**_____

3. Вихідні дані до роботи мова програмування Python, середовище розробки Visual Studio Code_____
_____
_____

4. Зміст дипломної роботи (перелік питань, які потрібно розробити) огляд існуючих методів опрацювання тексту та виявлення інформації, розробка графічного інтерфейсу та функціоналу для знаходження ключових слів і розпізнавання іменованих сутностей, оцінка якості роботи реалізованих методів та порівняння з аналогами_____
_____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Діаграма неконтрольованих методів виявлення ключових слів, зображення інтерфейсу програми, зображення архітектури нейронної мережі, зображення псевдокоду алгоритму виявлення ключових слів, зображення етапів опрацювання фотографії для функції сканування документа, зображення графіків порівняння метода виявлення ключових слів з аналогами, зображення матриці невідповідності, діаграма кількості появ тегів іменованних сутностей в датасеті Annotated Corpus for Named Entity Recognition, зображення демонстрації роботи програми
_____
_____
_____

6. Консультанти розділів роботи

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

7. Дата видачі завдання **31 серпня 2022 р.**

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| **1.** | Вивчення предметної області | 13.09.2022-13.10.2022 | Виконано |
| 2. | Вивчення існуючих методів виявлення ключових слів | 14.10.2022-03.11.2022 | Виконано |
| 3. | Реалізація алгоритму виявлення ключових слів | 04.11.2022-24.11.2022 | Виконано |
| 4. | Оцінка реалізованого алгоритму виявлення ключових слів та порівняння його з аналогами | 25.11.2022-09.12.2022 | Виконано |
| 5. | Вивчення існуючих методів розпізнавання іменованих сутностей | 12.12.2022-27.12.2022 | Виконано |
| 6. | Розробка моделі машинного навчання для розпізнавання іменованих сутностей | 28.12.2022-13.01.2023 | Виконано |
| 7. | Пошук технологій для функції сканування документів | 16.01.2023-02.02.2023 | Виконано |
| 8. | Розробка функції сканування документів | 03.02.2023-17.02.2023 | Виконано |
| 9. | Розробка десктопної програми для опрацювання тексту | 20.02.2023-20.03.2023 | Виконано |
| 10. | Оформлення дипломної роботи | 21.03.2023-02.06.2023 | Виконано |
| | | | |
| | | | |

**Студент** _____ _____
( підпис ) (прізвище та ініціали)

**Керівник роботи**_____
( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел. Обсяг дипломної роботи: 38 сторінок, 2 таблиці та 13 рисунків. Список використаних джерел складається з 20 найменувань.

Мета даної роботи — розробка системи опрацювання тексту з розширеними функціональними можливостями, такими як видобування ключових слів, розпізнавання іменованих сутностей та оптичне розпізнавання символів (OCR). Ця робота має на меті надати користувачам інструмент, який може ефективно аналізувати та обробляти текстові дані з різних джерел, включно з текстовими документами та зображеннями. Забезпечуючи точне вилучення ключових слів, система може ідентифікувати найбільш релевантні та важливі терміни в заданому тексті. Крім того, завдяки ефективному розпізнаванню іменованих об'єктів, система розпізнає і класифікує імена людей, назви організацій, географічні локації і дати, що дозволяє користувачам отримувати цінну інформацію з неструктурованого тексту. Крім того, завдяки інтеграції технології розпізнавання текстів, система розширює свої можливості для вилучення та інтерпретації текстової інформації із зображень, що робить її універсальним рішенням для аналізу текстів різних форматів. Загалом, метою цієї роботи є полегшення вилучення та розуміння змістовної інформації з тексту, сприяння ефективній обробці даних і надання користувачам можливості приймати обґрунтовані рішення на основі їхнього аналізу.

Новизна цієї роботи полягає в об’єднанні багатьох передових функцій обробки тексту в одній програмі з включенням технології OCR, що призводить до створення потужного інструменту, який пропонує підвищену ефективність і універсальність у вилученні та розумінні значущої інформації з тексту. Крім того, у цій роботі пропонується новий метод вилучення ключових слів, що сприяє подальшому розвитку області обробки природної мови.

Зміст

# ВСТУП

У сучасному світі люди працюють з великими обсягами тексту майже щодня. Ручне опрацювання і виявлення потрібної інформації є трудомісткою задачею, яка займає багато часу. Крім того, текстова інформація може зберігатися не тільки в текстових документах, але і на фотографіях. Отже стандартний набір функцій, що пропонують більшість систем для роботи з текстом не є достатнім.

Таким чином, актуальність полягає в створенні системи обробки тексту, що може аналізувати та знаходити важливу інформацію в текстових даних. Розпізнавання іменованих сутностей і виявлення ключових слів є важливими техніками обробки природної мови, які можуть допомогти ідентифікувати та витягнути цінну інформацію з великих обсягів тексту. Ці функції є корисними для класифікації документів на основі їх вмісту. Наприклад, визначивши імена людей, організацій і місця в статті новин, можна швидше класифікувати статтю як пов'язану з політикою, спортом, розвагами чи іншими категоріями. Також подібний функціонал є корисним з навчальної точки зору, бо дозволяє школярам та студентам швидше опрацьовувати текстовий матеріал. Виявлення ключових слів можна використовувати для визначення найважливішої інформації у фрагменті тексту, що дозволяє створити більш стислий підсумок. Дослідники та науковці можуть використовувати систему для вилучення ключових слів з наукових праць або інших джерел, що полегшує пошук потрібної інформації та сприяє глибшому розумінню певної теми. Одним з недоліків існуючих методів виявлення ключових слів є те, що вони не достатньо якісно генерують слова, що здатні охопити увесь текст в тому випадку якщо він належить до декількох різних сфер.

Метою роботи є розробка уніфікованої системи для опрацювання тексту написаного англійською мовою, що міститиме в собі наступний функціонал: виявлення ключових слів в тексті, розпізнавання іменованих сутностей, шляхом дотренування мовної моделі BERT (англ. Bidirectional Encoder Representations

from Transformers), зчитування тексту з фотографії. Запропонований метод виявлення ключових слів є ефективним для текстів, що містять інформацію з різних доменів.

Для досягнення даної мети виконаємо наступні кроки:

— Реалізація алгоритму для виявлення ключових слів.

— Тренування моделі розпізнавання іменованих сутностей для подальшого використання отриманого файлу натренованої моделі в головній програмі.

— Розробка функціоналу для зчитування тексту з зображень.

— Реалізація графічного інтерфейсу для десктопної програми.

— Оцінка реалізованих функцій системи шляхом аналізу різних метрик і порівняння результатів з існуючими методами для визначення якості їхньої роботи.

# РОЗДІЛ 1. АНАЛІЗ ПІДХОДІВ ТА АЛГОРИТМІВ ДЛЯ ОПРАЦЮВАННЯ ТЕКСТУ ТА СКАНУВАННЯ ЗОБРАЖЕНЬ

У цьому розділі розглянемо алгоритми та підходи, що використовуються для опрацювання тексту та сканування зображень. Вивчаючи різні методи ми можемо отримати уявлення про сильні і слабкі сторони різних підходів і визначити напрямки для подальших досліджень і вдосконалень.

## 1.1 Існуючі методи опрацювання тексту

Існує велика кількість підходів для опрацювання тексту, серед яких можна виділити семантичний аналіз, машинний переклад, класифікацію тексту за існуючими категоріями тощо. Розглянемо ті підходи, що стосуються виявлення інформації, а саме: виявлення ключових слів та розпізнавання іменованих сутностей.

### 1.1.1 Виявлення ключових слів

Виявлення ключових слів полягає в автоматичному вилученні з документа набору репрезентативних слів, які стисло підсумовують його зміст [1]. Існують як контрольовані (навчання з вчителем), так і неконтрольовані (навчання без вчителя) методи вилучення ключових слів. Методи на основі навчання без вчителя є популярними, оскільки вони не залежать від домену і не потребують маркованих навчальних даних, створення яких передбачає значні витрати часу і ресурсів, що є недоліком використання методів навчання зі вчителем. Контрольовані методи можуть не працювати належним чином, якщо застосовувати їх до тексту поза доменом або контекстом навчальних даних. У даній роботі розглянемо різноманітні методи на основі навчання без вчителя для виявлення ключових слів, серед яких виділимо статистичні (TF-IDF), тематичні, графові (TextRank) та на основі вкладання слів (KeyBert).

Наведемо основні етапи роботи неконтрольованої системи вилучення ключових фраз [1]:

— Відбір лексичних одиниць-кандидатів на основі певних евристик. Прикладами таких евристик є виключення стоп-слів і відбір слів, що належать до певної частини мови.

— Ранжування лексичних одиниць-кандидатів.

— Формування ключових фраз шляхом відбору слів з найбільш ранжируваних або шляхом відбору фрази з високим ранговим показником або чиї частини мають високий ранговий показник.

На рисунку 1.1 представлено основні дослідницькі підходи, пов'язані з неконтрольованим видобуванням ключових фраз.

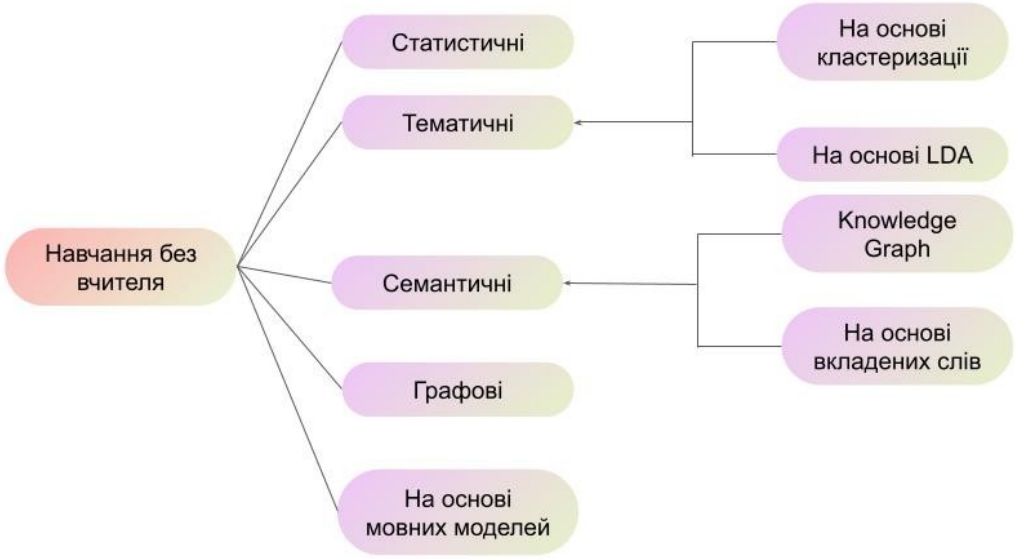


Рисунок 1.1 - Діаграма неконтрольованих методів для виявлення ключових слів.

Найпоширенішим статистичним методом є TF-IDF (англ. term frequency-inverse document frequency). Він обчислює важливість кожного слова в документі або корпусі, враховуючи як його частоту, так і рідкість. Компонент частоти (TF) показує, як часто слово з'являється в документі, тоді як зворотний компонент частоти документа (IDF) показує, наскільки рідкісним є слово в усьому корпусі. Добуток цих двох компонентів дає вагу кожному слову, причому слова, які часто зустрічаються в документі, але рідко в корпусі, мають найбільшу вагу.

Основна ідея ранжування на основі графів полягає у створенні графа, вершинами якого є фрази-кандидати з документа, а кожне ребро цього графа з'єднує пов'язані ключові фрази-кандидати. Прикладом алгоритму, що реалізує цю ідею є TextRank. Одиниці, що підлягають ранжуванню, є послідовності однієї або декількох лексичних одиниць, витягнутих з тексту і вони являють собою вершини, які додаються до текстового графа. Будь-який зв'язок, який можна визначити між двома лексичними одиницями є потенційно корисним зв'язком (ребром), який можна додати між двома такими вершинами [2].

Окрім методів, які використовують класичну статистичну евристику (TF-IDF), а також контекстно-орієнтовану статистику, наприклад, інформацію про повторюваність слів, існують також методи, які намагаються повернути ключові фрази, пов'язані з темами, що обговорюються в документі. Зокрема, тематичні методи намагаються виділити ключові фрази, які є репрезентативними для текстового документа з точки зору тем, які він охоплює. Такі методи зазвичай застосовують методи кластеризації або латентний розподіл Діріхле (англ. Latent Dirichlet Allocation, LDA) для виявлення основних обговорюваних тем [3].

Вкладання слів - це метод обробки природної мови, який останніми роками набув популярності для різноманітних завдань аналізу тексту, зокрема для вилучення ключових слів. Вкладання слів представляють слова у вигляді векторів у високорозмірному просторі, де слова зі схожими значеннями знаходяться близько один до одного в цьому просторі. Прикладом метода вилучення ключових слів, що використовує таку методологію є KeyBert [4]. Вбудовування документа та N-грамових слів/фраз витягуються за допомогою BERT. Після цього використовується косинус подібності, щоб знайти слова/фрази, які є найбільш схожими на документ і вони визначаються як ті слова, що найкраще описують весь документ.

Вивчалося використання вкладання слів для вилучення ключових слів. Наприклад, у літературі згадувався метод, який поєднує вкладання слів з глибокою нейронною мережею для вилучення ключових слів з новинних статей [5]. Інші дослідники вивчали використання попередньо навчених вкладань слів,

таких як Word2Vec або GloVe. Наприклад, використовувались попередньо навчені вкладання слів для підвищення ефективності методу вилучення ключових слів на основі графів [6]. Загалом, використання вкладання слів для вилучення ключових слів показало багатообіцяючі результати в кількох дослідженнях, причому деякі методи перевершили традиційні статистичні методи.

### 1.1.2 Розпізнавання іменованих сутностей

Розпізнавання іменованих об'єктів (англ. Named Entity Recognition, NER) є підзадачею вилучення інформації і включає в себе обробку структурованих і неструктурованих документів та ідентифікацію виразів, які відносяться до різних категорій (люди, географічні локації, організацій тощо). NER є фундаментальною задачею і лежить в основі системи обробки природної мови. NER включає в себе два завдання: ідентифікацію власних назв у тексті і класифікацію цих імен за набором заздалегідь визначених категорій, що представляють інтерес. Такими категоріями зазвичай є імена людей, організацій, географічні назви, вирази дати і часу [7].

Існує два типи реалізації NER: NER на основі правил та NER на основі машинного навчання.

Метод на основі правил фокусується на вилученні назв, використовуючи велику кількість попередньо розроблених правил. В основному системи складаються з набору шаблонів, що використовують граматичні, синтаксичні та орфографічні ознаки в комбінації зі словниками [8]. Цей тип реалізації NER є специфічними для домену і мови і не завжди добре адаптується під нові варіації.

У системах NER, що базуються на машинному навчанні, метою підходу є перетворення проблеми ідентифікації в проблему класифікації. У цьому типі системи ідентифікують і класифікують іменники за певними класами, такими як особи, місця, часу тощо. Існує два типи моделей машинного навчання, що використовуються для NER: навчання зі вчителем та навчання без вчителя. Модель машинного навчання зі вчителем для NER навчається на великій кількості текстових даних, які були вручну анотовані, щоб позначити названі сутності в

тексті. Потім текстові дані перетворюються на ознаки, які може зрозуміти модель, наприклад, вбудовування слів і теги частин мови. Далі відбувається навчання моделі на анотованих даних для ідентифікації та класифікації іменованих сутностей в тексті. Після навчання цю модель можна використовувати для передбачення іменованих сутностей у нових текстових даних. Неконтрольоване навчання (навчання без вчителя) є не дуже поширеним підходом для NER, і системи, що використовують такий підхід зазвичай не є повністю неконтрольованими. У літературі також було розглянуто метод розробки моделі, що використовує невеликий словник іменованих сутностей та немаркований корпус для їх класифікації. Модель враховує синтаксичні зв'язки в реченні для вирішення семантичної неоднозначності і використовує ансамбль трьох різних методів навчання: модель максимальної ентропії, навчання засноване на пам'яті та алгоритм SNoW (англ. Sparse Network of Winnows) [9].

## 1.2 Існуючі підходи до реалізації технології сканування документів

Розглянемо методи, що використовуються для реалізації технології сканування документів. Ці методи включають оптичне розпізнавання символів OCR (англ. Optical Character Recognition), зіставлення з шаблоном, глибоке навчання, попередню обробку зображень і розпізнавання рукописного тексту.

### 1.2.1 Оптичне розпізнавання символів

OCR - це технологія, яка дозволяє розпізнавати друковані або рукописні текстові символи на цифрових зображеннях. OCR працює шляхом аналізу форми окремих символів на зображенні, а потім перетворює ці форми на машинозчитуваний текст. Розпізнавання тексту може бути реалізовано за допомогою методів машинного навчання (ML) для підвищення точності та продуктивності [10].

У системі розпізнавання тексту з використанням ML великий набір даних зображень і відповідного тексту використовується для навчання нейронної мережі розпізнавати закономірності в даних. Потім нейронна мережа використовується

для розпізнавання символів на нових зображеннях і перетворення їх у машинозчитуваний текст. Продуктивність системи розпізнавання за допомогою ML можна покращити за допомогою такого метода як доповнення даних, коли навчальні дані штучно розширюються шляхом застосування до зображень таких перетворень, як обертання, масштабування та нахил. Це допомагає підвищити надійність системи та її здатність розпізнавати символи за різних умов.

Зображення реального світу не завжди створюються при ідеальних умовах, вони можуть мати шум, розмиття, перекоси тощо. З цієї причини подібні зображення потребують попередньої обробки.

Далі для локалізація тексту використовуються такі моделі, як Mask-RCNN, East Text Detector, YoloV5, SSD тощо, які визначають його місцезнаходження на зображенні. Ці моделі зазвичай створюють обмежувальні рамки над кожним текстом, визначеним на зображенні чи документі.

Після визначення розташування тексту кожен обмежувальний квадрат надсилається до моделі розпізнавання тексту, яка зазвичай є комбінацією мереж RNN, CNN. Остаточним результатом цих моделей є текст, витягнутий з документів. Існують моделі розпізнавання тексту з відкритим кодом, такі як Tesseract, MMOCR тощо.

### 1.2.2 Зіставлення з шаблоном

При роботі зі строго структурованими документами, що відповідають одному формату, більш зручним способом може слугувати техніка зіставлення з шаблоном. Якщо отримати фрагмент тексту за допомогою OCR, він необов'язково зберігатиме геометричне розташування слів, пробілів і стиль. Через що буде важко ідентифікувати та знайти ключову інформацію, яка потрібна користувачу [11].

Метод зіставлення з шаблоном полягає у визначенні шаблонів на зображенні, які відповідають тексту. Цей шаблон зазвичай являє собою двійкове зображення, яке містить лише пікселі, що відповідають тексту. Після створення шаблону його порівнюють із вхідним зображенням, щоб знайти необхідні фрагменти тексту.

Зіставлення з шаблоном може здійснюватися за допомогою різних алгоритмів, таких як зіставлення на основі кореляції, зіставлення на основі ознак і зіставлення на основі глибокого навчання. Зіставлення на основі кореляції відбувається через обчислення коефіцієнта кореляції між шаблоном і вхідним зображенням у кожній позиції. Зіставлення на основі ознак передбачає виявлення ознак у шаблоні та вхідному зображенні та їх зіставлення за допомогою таких методів, як SIFT (Scale-Invariant Feature Transform) і SURF (Speeded Up Robust Features). Зіставлення на основі глибокого навчання реалізується за допомогою навчання глибокої нейронної мережі для розпізнавання тексту на зображенні [12].

Зіставлення за шаблоном - це потужний метод для вилучення тексту із зображень, але він має деякі обмеження. Він вимагає наявність точного шаблону, який в деяких випадках може бути складно створити. Він також чутливий до змін на зображенні, таких як зміна освітлення і перспективи, що може вплинути на точність вилучення. Тим не менш, це цінний інструмент для багатьох застосувань, і його можна поєднувати з іншими методами для підвищення точності та надійності.

## 1.3 Вибір методів для реалізації системи

Запропонуємо методи для реалізації виявлення ключових слів, розпізнавання іменованих сутностей та сканування документів.

### 1.3.1 Виявлення ключових слів

Необхідно було покращити виявлення ключових слів для текстів що містять інформацію з різних областей. Це поширена проблема в обробці природної мови, оскільки різні домени можуть мати різний словник і теми, що ускладнює вилучення відповідних ключових слів. Щоб вирішити цю задачу, було вирішено реалізувати виявлення ключових слів шляхом поєднання техніки тематичного моделювання Latent Dirichlet Allocation (LDA) та техніки знаходження релевантних словосполучень в тексті. LDA - це генеративна ймовірнісна модель для наборів згрупованих дискретних даних [3]. Кожна група описується як випадкова суміш над набором латентних тем, де кожна тема є дискретним

розподілом над словником колекції. У даному випадку колекцією є речення вхідного тексту, а згенеровані теми, що складаються з окремих слів, можна розглядати як ключові слова. Процес генерації тем для колекції речень D використовуючи модель LDA виглядає наступним чином [13]:

1. *Для* $k=1\dots K$:
   (a) $\phi^{(k)} \sim Dirichlet(\beta)$
2. *Для кожного речення* $d \in D$:
   (a) $\theta^{(k)} \sim Dirichlet(\alpha)$
   (b) *Для кожного слова* $w_i \in d$:
      i. $z_i \sim Discrete(\theta_d)$
      ii. $w_i \sim Discrete(\phi^{(z_i)})$

де $K$ - кількість тем, $Dirichlet(x)$ визначає розподіл Діріхле над параметром $x$, $\phi^{(k)}$– розподіл ймовірностей над фіксованим словником слів, що представляє розподіл для $k$-тої теми, $\theta^{(k)}$- розподіл для кожного речення над доступними темами, $z_i$ - індекс теми для слова $w_i$, α та β - гіперпараметри для симетричних розподілів Діріхле, з яких отримано дискретні розподіли.

Техніку LDA обчислює умовний розподіл присвоєння тем конкретному слову, за умови, що ми не знаємо тематичну приналежність поточного слова, але знаємо такі приналежності до всіх інших слів у тексті [13]. У рівнянні (1) знаходимо умовний розподіл ймовірностей присвоєння теми поточному слову у конкретному документі, що залежить від латентних змінних та всіх інших приналежностей тем до слів.

Цей процес описується формулою (1) [14]:

$$p(z_i=j \mid z_{-i}, w_i, d_i) = \frac{\left(n^{DT}_{d_i,j}+\alpha_{\square}\right)}{\sum_{t=1}^{T} n^{DT}_{d_i,t}+T\alpha} \frac{\left(n^{WT}_{w_i,j}+\beta_{\square}\right)}{\sum_{w=1}^{W} n^{WT}_{w_{\square},j}+W\beta} \qquad (1)$$

де $p(z_i=j)$ - ймовірність того, що токен $i$ буде призначено темі $j$, $z_{-i}$ - відображає призначення тем для всіх інших токенів, $w_i$ – індекс слова, що відповідає $i$-тому токену, $d_i$ – речення, що містить $i$-ий токен, $n^{WT}$ – матриця

приналежності слів до тем, $n^{DT}$ – матриця приналежності тем до речень, $\beta_{\square}$ - параметр, який задає розподіл слів за темами, чим вище, тим більше слів буде містити кожна тема, α - параметр, який задає розподіл тем для речень, чим вище значення, тим більша ймовірність того, що кожне речення міститиме суміш більшості тем, а не якусь одну тему., $W$ – загальна кількість слів в наборі речень, $T$ – загальна кількість тем.

У результаті ми можемо використовувати матриці підрахунку $n^{DT}$ та $n^{WT}$для отримання розподілу слів за темами та розподілу речень за темами. Після обчислення ймовірності того, що кожен документ належить до кожної теми (те ж саме стосується слова і теми), ми можемо використовувати цю інформацію, щоб побачити, до якої теми належить кожне речення і скільки можливих слів пов'язано з кожною темою. Ці результуючі групи слів можна вважати ключовими словами усього тексту.

На даному етапі результат складається лише з уніграм (поодиноких слів), тому для того щоб покращити поточний варіант реалізації процесу вилучення ключових слів, було вирішено також знаходити релевантні словосполучення в тексті. Для цього знаходимо найбільш вживані біграми (словосполучення, що складається з двох слів) серед яких обираємо ті, що семантично найбільш схожі до всього документа за допомогою техніки вбудувань речень. Результуючий набір ключових слів представляє собою поєднання списку обраних біграм та уніграм отриманих за допомогою LDA.

### 1.2.3 Розпізнавання іменованих сутностей

Для реалізації даного функціоналу було вирішено побудувати нейронну мережу на основі BERT. BERT - це модель машинного навчання на основі трансформерів. Мета NER - ідентифікувати та класифікувати іменовані об'єкти в тексті за наперед визначеними категоріями, такими як імена людей, організації, місцезнаходження тощо. Для цього вхідний текст спочатку розбивається на окремі слова або підслова, а потім кожному токену присвоюється мітка, яка вказує, чи є він частиною іменованої сутності, і якщо так, то якого типу. Щоб використовувати

BERT для завдань NER, потрібно подати на вхід моделі всю вхідну послідовність лексем (тобто речення або документ, який потрібно проаналізувати), а не лише окремі слова. Це пов'язано з тим, що BERT є контекстуалізованою моделлю, яка враховує весь контекст, в якому з'являється слово або підслово, щоб згенерувати вихідний вектор його вбудовування. Тому, надаючи BERT всю послідовність лексем, ми дозволяємо їй враховувати контекст кожного окремого слова в реченні при генерації вектора вбудовування. Для тренування моделі був використаний Annotated Corpus for Named Entity Recognition датасет [15].

### 1.3.2 Технологія сканування документа

Для видобуття текста із зображення необхідно було здійснити наступні кроки:

— Створити копію оригінального зображення для майбутнього застосування гомографії. Зменшити розмір RGB-зображення.
— Застосувати чорно-білий фільтр до RGB-зображення, оскільки більшість бібліотек обробки зображень працюють із зображеннями у відтінках сірого, оскільки їх легше обробляти.
— Застосувати фільтр розмиття за Гауссом до чорно-білого зображення, щоб видалити шум.
— Розробити функціонал для автоматичного знаходження контуру документа який треба зісканувати.
— Застосувати техніку викривленої перспективи, яка є технікою комп'ютерного зору для перетворення зображення з метою виправлення спотворень. Вона перетворює зображення в іншу площину, що дозволяє переглянути його під іншим кутом.
— Застосувати поріг Гаусса до викривленого зображення, що надасть викривленому зображенню вигляду сканованого.
— Застосувати функцію зчитування тексту з зображення до обробленої фотографії.

# РОЗДІЛ 2. ПРОГРАМНА РЕАЛІЗАЦІЯ

Для реалізації системи опрацювання тексту була використана мова програмування Python та наступні пакети:

— PyQt5 – для створення графічних інтерфейсів, що є розширенням графічного фреймворку Qt
— Gensim – для неконтрольованого тематичного моделювання, індексування документів, пошуку за подібністю та інших функцій обробки природної мови
— Transformers – для використання натренованих мовних моделей.
— Sentence-transformers – для отримання векторного представлення речень/текстів
— nltk (Natural Language Toolkit) – для застосування токенізації і стеммінгу над текстом
— tensorflow.keras, PyTorch – для будування і тренування нейронних мереж.
— OpenCV, Pytesseract – для розробки функціоналу сканування документа.

## 2.1 Розробка інтерфейсу

Інтерфейс містить дві основні секції: для введення тексту і для результата опрацювання (Рисунок 2.1). Крім того, було додано підменю «Аналіз тексту», яке представляє собою випадаючий список з функціями «Вилучення ключових слів» та «Розпізнавання іменованих сутностей». Інтерфейс також містить стандартні функції, наявні в більшості систем для опрацювання тексту, такі як «Файл», «Редагувати» та «Форматування».

Інтерфейс розроблено з використанням бібліотеки PyQt5, що дозволило створити адаптивний і візуально привабливий графічний інтерфейс. Підменю були створені за допомогою віджета QMenu.

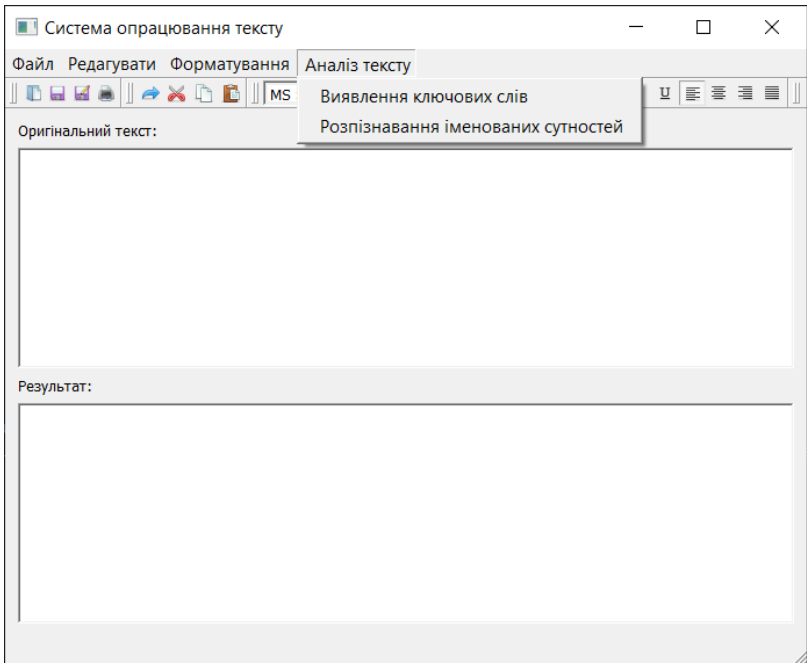


Рисунок 2. 1 - Інтерфейс програми

Розділ введення тексту дозволяє користувачу вводити текст за допомогою віджета QTextEdit. Розділ вихідного тексту відображає результати опрацювання тексту. Взаємодія з елементами інтерфейсу реалізована на основі класу QAction, який при кліку тригерить викликання відповідних функцій.

## 2.2 Розпізнавання іменованих сутностей

Для реалізації нейронної мережі на основі BERT була використана бібліотека PyTorch.

Для попередньої обробки даних з датасету «Annotated Corpus for Named Entity Recognition» був реалізований клас EnityDataset. У NER метою є ідентифікація та класифікація іменованих сутностей у тексті, таких як: люди, організації та місцезнаходження. Клас EntityDataset отримує наступні вхідні дані: texts, pos і tags; де texts - це список речень, pos - список списків тегів частин мови для кожного слова у відповідному реченні, а tags - список списків тегів NER для кожного слова.

У класі EntityDataset відбувається токенізація кожного речення за допомогою токенізатора BERT, що перетворює токени у відповідні ідентифікатори (алгоритм токенізації BERT працює шляхом розбиття вхідного

тексту на послідовність окремих частин слова, яким потім присвоюються унікальні ідентифікатори на основі заздалегідь визначеного словника) та доповнює послідовність до максимальної довжини, вказаної в конфігураційному файлі. Він також створює тензор маски, щоб вказати, які токени є власне токенами, а які токенами доповнення. Нарешті, повертається словник, що містить вхідні тензори, необхідні для навчання моделі NER, включаючи тензори ids, mask і token_type_ids, а також тензори target_pos і target_tag, які представляють частину мови і NER-теги для кожної лексеми відповідно.

Для навчання та оцінки роботи моделі в процесі навчання були реалізовані відповідні функції. Функція тренування використовується для навчання моделі. Вона приймає навчальні дані, модель, оптимізатор, device (це може бути графічний або центральний процесор) і планувальник (який використовується для регулювання швидкості навчання). У середині функції вона циклічно запускає модель для кожного пакету даних з подальшим обрахуванням втрат. Те саме відбувається у функції оцінки, але для тестового набору даних.

Для розпізнавання іменованих об'єктів (NER) та тегування частин мови (POS) була реалізована PyTorch модель під назвою EntityModel. Клас EntityModel наслідує клас nn.Module з PyTorch та реалізує архітектуру нейронної мережі зображеної на Рисунку 2.2. Він отримує два параметри: num_tag і num_pos, які представляють кількість тегів і частин мови відповідно. Перший шар містить попередньо навчену модель BERT (bert-base-uncased) з бібліотеки трансформерів Hugging Face. Два наступні Dropout шари мають ймовірністю відсіву 0,3. Dropout – це метод регуляризації штучних нейронних мереж, призначений для запобігання перенавчання мережі. Перенавчання – це явище, коли побудована модель добре працює на прикладах з навчальної вибірки, але відносно погано працює на прикладах, які не брали участі в навчанні. Два лінійні шари приймають вихідні дані BERT-моделі та виводять оцінку для кожного тегу та частини мови.

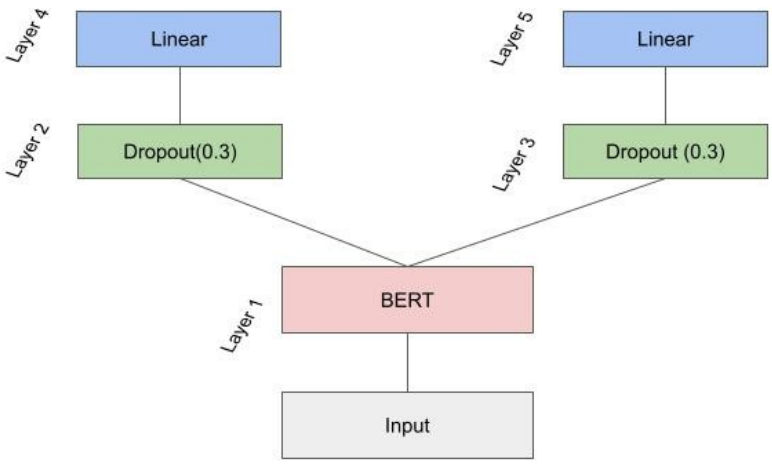


Рисунок 2. 2 - Архітектура нейронної мережі для NER

Для прогнозування була реалізована функція, що отримує на вхід текстовий рядок і повертає словник, що містить іменовані сутності, розпізнані у вхідному тексті. Спочатку вона завантажує файл meta.bin, який містить кодування міток для іменованих сутностей. Потім відбувається розбиття вхідного тексту на окремі слова, що кодуються за допомогою попередньо навченого токенізатора. Після цього використовується клас EntityDataset для створення об'єкта набору даних, який можна використовувати для передачі вхідних даних навченій моделі. Закодовані дані передаються моделі для отримання передбачених тегів і позицій для кожного слова у вхідному тексті. Нарешті, функція повертає словник розпізнаних іменованих об'єктів.

Тренування моделі відбувалось в середовищі Google Colab Pro, що надає доступ до високопродуктивних GPU, що може значно прискорити навчання моделей глибокого навчання або інших обчислювально інтенсивних завдань.

## 2.3 Вилучення ключових слів

Реалізація вилучення ключових слів включає в себе такі етапи як: побудову тематичної моделі LDA та отримання релевантних словосполучень, що асоціюються з текстом. Слова та фрази, що отримуються в кінці виконання функції, об’єднуються в один рядок та повертаються як результат.

### 2.3.1 Реалізація тематичної моделі LDA

Першим кроком є встановлення необхідних лічильників змінних, їх випадкову ініціалізацію, і запуск циклу, де на кожній ітерації для кожного входження слова в корпусі вибирається тема. Для кількості тем встановлюємо значення 5.

Необхідними змінними є двовимірні масиви $n^{WT}$ та $n^{DT}$ що є матрицями приналежності слів до тем та тем до речень відповідно, змінні $K$, α, β, що відповідають за кількість тем та параметри для розподілів Діріхле. Нарешті, необхідна ініціалізація масиву $z$, який міститиме поточне призначення теми для кожного з N слів у корпусі.

На кожній ітерації ми віднімаємо 1 від значень лічильників, щоб не враховувати поточні присвоєння тем до слів та документів. Потім обчислюється ймовірність кожного присвоєння теми, використовуючи рівняння (1), після чого тема з найбільшою ймовірністю присвоюється поточному слову для поточного речення в масиві $z$. Після отримання нової теми необхідно оновити данні про призначення тем, для цього збільшуємо певні значення матриць $n^{WT}$і $n^{DT}$. Псевдокод реалізації алгоритму зображений на Рисунку 2.3. [13].

**Введення:** слова $\boldsymbol{w} \in$ речення $\boldsymbol{d}$
**Вивід:** присвоєння тем $\boldsymbol{z}$ і матриці-лічильники $n^{DT}, n^{WT}$
**початок**
  випадкова ініціалізація $\boldsymbol{z}$ і збільшення лічильників
  **для** всіх *ітерацій*
    **для** $i = 0 \rightarrow N-1$
      слово $\leftarrow w[i]$
      тема $\leftarrow z[i]$
      $n^{DT}_{\text{речення,тема}} -= 1;\ n^{WT}_{\text{слово,тема}} -= 1$
      **для** $j = 0 \rightarrow K-1$
        $$p(z = j|\cdot) = \frac{(n^{DT}_{d_i j}+\alpha)}{\Sigma_{t=1}^{T} n^{DT}_{d_i t}+T\alpha} \frac{(n^{WT}_{w_i j}+\beta)}{\Sigma_{w=1}^{W} n^{WT}_{w j}+W\beta}$$
      **кінець**
      тема $\leftarrow$ розподіл з $p(z|\cdot)$
      $z[i] \leftarrow$ тема
      $n^{DT}_{\text{речення,тема}} += 1;\ n^{WT}_{\text{слово,тема}} += 1$
    **кінець**
  **кінець**
  **повернути** z, $n^{DT}, n^{WT}$
**кінець**

Рисунок 2. 3 - Алгоритм реалізації тематичної моделі LDA

З повернутих значень використовуємо масиви $n^{WT}$ і $n^{DT}$ для отримання $\phi^{(k)}$, тобто ймовірності належності кожного слова до тем. Після чого обираємо 5 слів з найбільшими ймовірностями для кожної теми. У результаті ці слова вважаються ключовими.

### 2.3.2 Отримання релевантних словосполучень

Для отримання словосполучень, що найкраще асоціюються з текстом була реалізована функція, яка обробляє вхідний текст, отримує найбільш популярні біграмні словосполучення з тексту (з використанням бібліотеки NLTK), здійснює аналіз семантичної схожості між ними та самим текстом за допомогою бібліотеки SentenceTransformer та методу косинусу подібності.

Пошук біграм складається з наступних кроків:

- Клас BigramCollocationFinder ініціалізується списком попередньо оброблених слів. Цей клас використовується для пошуку біграм (пар сусідніх слів), які часто зустрічаються в тексті.
- Метод apply_freq_filter викликається об'єктом BigramCollocationFinder і приймає параметр, який вказує, що слова, які зустрічаються не менше певної кількості разів, будуть використані для подальшої обробки, а решта слів будуть відкинуті. Це допомагає зменшити кількість слів, які не мають вагомого значення в тексті, та зосередитися на тих, які є більш значущими.
- Об'єкт BigramAssocMeasures ініціалізується для обчислення статистичної значущості кожної біграми.
- Метод nbest викликається об'єктом BigramCollocationFinder, щоб повернути 30 найбільш значущих біграм відповідно до обраної статистичної міри. У цьому випадку використовується міра хі-квадрат, яка обчислює значущість кожної біграми на основі її спостережуваної частоти порівняно з очікуваною частотою за нульової гіпотези про випадковість входження слів у словосполучення.

— Результуюча змінна top_bigrams містить список 10 найбільш значущих біграм у тексті за критерієм хі-квадрат. Ці біграми повертаються у вигляді списку кортежів, де кожен кортеж містить два слова біграми.

Після отримання біграм об'єкт SentenceTransformer ініціалізується за допомогою попередньо навченої моделі "distilbert-base-nli-mean-tokens" з бібліотеки Hugging Face, яка генерує вкладання речень, що представляють собою щільні вектори високої розмірності і можуть слугувати чисельним представленням семантичного значення речень. Вхідний текст кодується в єдиний вектор-вкладання за допомогою моделі SentenceTransformer. Кожен рядок біграми у списку біграм також кодується у вектор-вкладання. Між кожним вкладанням біграми і вхідним текстом обчислюється косинус подібності за допомогою функції cosine_similarity() з бібліотеки scikit-learn. Косинус подібності - це міра схожості двох векторів, значення якої варіюється від -1 (абсолютно не схожі) до 1 (ідентичні).

Перші n біграм з найвищою косинусоїдальною подібністю до вхідного тексту обираються як ключові фрази і повертаються функцією у вигляді списку рядків.

## 2.4 Функція сканування документа

Нами був розроблений функціонал для вилучення тексту зі сканованого зображення документа. На вхід подається шлях до файла зображення, який вказується користувачем. Були використані різні функції бібліотеки OpenCV для попередньої обробки зображення, визначення меж документа і викривлення документа для отримання вигляду «зверху-вниз». Потім було застосоване адаптивне порогове значення для надання тексту виду зісканованого і використан механізм розпізнавання Tesseract OCR для його вилучення і отримання рядка. Всі етапи опрацювання зображені на рисунку 2.4.

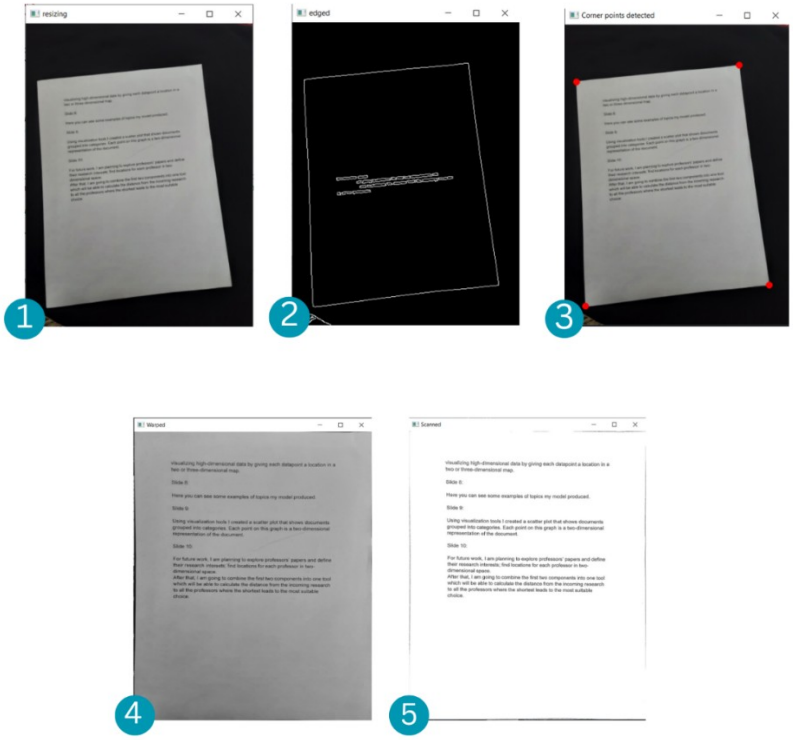


Рисунок 2. 4 - Етапи опрацювання зображення для функції сканування документа.

Першим кроком було зчитування вхідного зображення за допомогою функції imread() з OpenCV. Далі розмір зображення змінюється до висоти 500 пікселів і на нього накладається чорно-білий фільтр за допомогою функції cvtColor(). Потім зображення у відтінках сірого розмивається за допомогою GaussianBlur(), а краї визначаються за допомогою Canny(). Після цього автоматично виявляються контури на зображенні за допомогою findContours() і вибирається найбільший контур з чотирма вершинами, що є межами документа. Для цього усі можливі контури сортуються за зменшенням їхньої площі і апроксимуються в циклі за допомогою функції approxPolyDP(), щоб отримати полігональну апроксимацію. Якщо багатокутник має чотири вершини, вважається, що це межа документа, і цикл завершує свою роботу.

Після виявлення межі документа витягуються чотири кутові точки багатокутника за якими деформується зображення, щоб документ мав вигляд «зверху-вниз». Після викривлення зображення застосовується адаптивна порогова обробка для покращення тексту на зображенні. Для цього використовується функція threshold_local з scikit-image, яка застосовує локальний поріг до кожного

пікселя на основі середнього значення його сусідніх пікселів. Отримане двійкове зображення конвертується у формат uint8 і множиться на 255, щоб отримати зображення у відтінках сірого з чорним текстом на білому фоні.

На останньому кроці використовується інструмент розпізнавання Tesseract OCR для вилучення тексту з зображення. Для цього змінюється розмір деформованого зображення до висоти 650 пікселів за допомогою функції imutils.resize, яке потім передається функції image_to_string() з pytesseract. Результатом опрацювання зображення є рядок, що містить зісканований текст.

# РОЗДІЛ 3. ОЦІНКА ЯКОСТІ РОБОТИ СИСТЕМИ

У цьому розділі оцінимо якість роботи розробленої системи та методів. Для цього виконаємо наступні кроки:

— Порівняємо результати роботи метода виявлення ключових слів з його аналогами.
— Порахуємо accuracy (загальна частка правильних прогнозів моделі), precision (точність або частка об'єктів, правильно визначених класифікатором як приналежними до певного класу), recall (повнота або частка об'єктів певного класу з усіх об'єктів цього класу які визначив алгоритм) для прогнозованих моделлю сутностей для задачі розпізнавання іменованих сутностей.
— Протестуємо систему на декількох текстових прикладах і продемонструємо її роботу.

## 3.1 Виявлення ключових слів

Для оцінки якості роботи методу вилучення ключових слів LDA+Ngrams, що використовувався в даній роботі, відбулося порівняння результатів його роботи з іншими існуючими методами, такими як: TF-IDF, TextRank та KeyBert.

Для оцінки кількості релевантних ключових слів та фраз, що було знайдені в тексті, був використаний датасет SemEval-2010 [16]. SemEval2010 складається з 244 повних наукових статей, витягнутих з цифрової бібліотеки ACM, кожна з яких має обсяг від 6 до 8 сторінок і належить до чотирьох різних галузей комп'ютерних наук (розподілені системи, пошук і вилучення інформації, розподілений штучний інтелект - мультиагентні системи; соціальні та поведінкові науки - економіка). Кожна стаття має авторський набір ключових слів (які є частиною оригінального pdf-файлу) і набір ключових слів, призначених професійними редакторами, які можуть з'являтися або не з'являтися в тексті в явному вигляді.

Для знаходження частки релевантних слів була використана формула (2):

$$precision = \frac{N_c}{N_t} \quad (2)$$

де $N_c$ – це число правильно виявлених ключових слів, що також зустрічаються в наборі ключових слів вказаному в датасеті, $N_t$ – число всіх ключових слів/фраз, що були виявлені.

Середня кількість слів, що були виявленні для 244 текстів дорівнювала 20 словам. Отже, використовуючи метод KeyBert було вказане саме таке значення для кількості згенерованих н-грам. На рисунку 3.1. зображені графіки, що демонструють усереднені частки релевантних слів для кожних 10 статей. Була здійснена спроба порівняти якість роботи KeyBert при генерації виключно уніграм та при розподілі загальної кількості н-грам між біграмами та уніграмами.

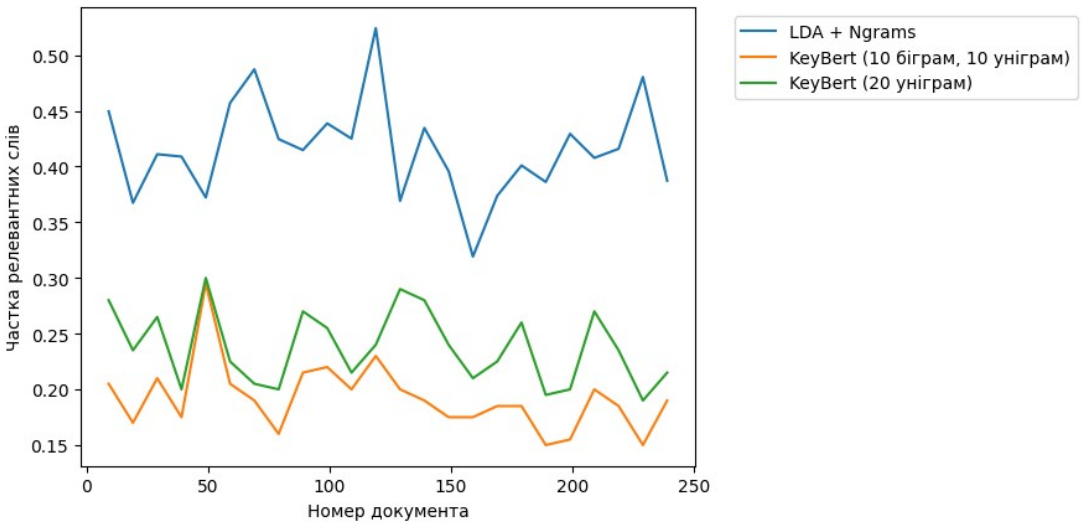


Рисунок 3. 1. Частки релевантних ключових слів для документів з датасету SemEval2010 отриманих за допомогою методів LDA+Ngrams та KeyBert.

Загалом, середнє значення частки релевантних слів для метода LDA+Ngrams становить 0.4177, для KeyBert (10 біграм, 10 уніграм) - 0.1919, для KeyBert (20 уніграм) - 0.2376.

Також було здійснене порівняння метода LDA+Ngrams і TF-IDF. Для цього був використаний клас TfidfVectorizer з модуля sklearn.feature_extraction.text для вилучення текстових ознак. Спочатку був створений екземпляр класу

TfidfVectorizer з параметрами ngram_range=(1,2) і stop_words. Далі, векторизатор підлаштовується до вхідного тексту за допомогою методу fit_transform, який повертає матрицю TF-IDF. Словник і зворотні значення частот документів витягуються з векторизатора за допомогою атрибутів vocabulary_ і idf_. Словник сортується за зворотною частотою документів за допомогою функції sorted з лямбда-функцією для доступу до значення idf для кожного слова. Нарешті, повертаються 20 найкращих ключових слів з найвищими значеннями зворотної частоти в документі.

Як видно на рисунку 3.2 середня частка релевантних слів, отриманих методом TF-IDF, менша за метод LDA+Ngrams і дорівнює 0.18. Ці результати свідчать про те, що метод LDA+Ngrams може бути більш ефективним для визначення релевантних ключових слів, ніж метод TF-IDF.

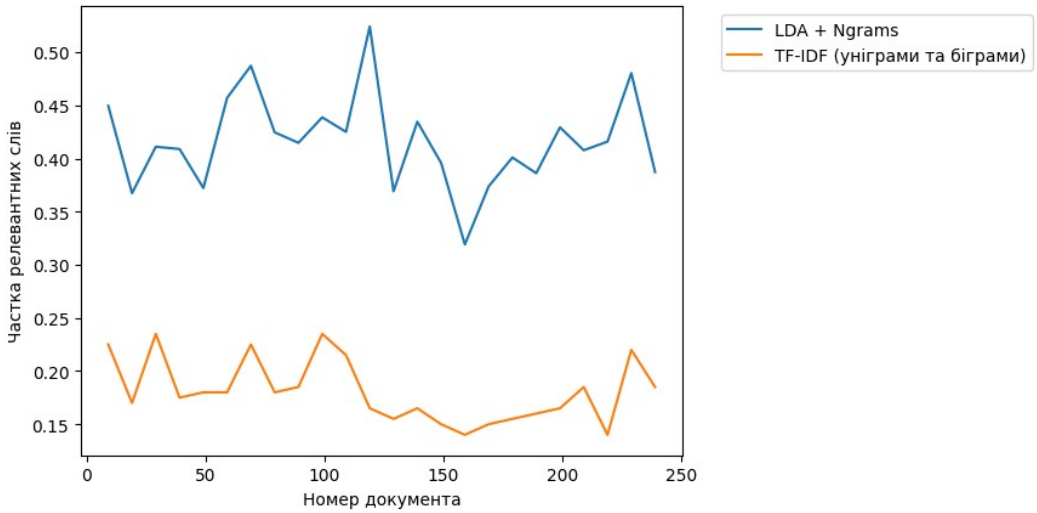


Рисунок. 3. 2 - Частки релевантних ключових слів для документів з датасету SemEval2010 отриманих за допомогою запропонованого метода та TF-IDF.

Також було здійснено порівняння методу LDA+Ngrams з методом TextRank. TextRank - це графовий алгоритм для автоматичного вилучення ключових слів і ключових фраз з тексту. Він належить до неконтрольованих методів, і не потребує жодних навчальних даних. Для реалізації порівняння була використана бібліотека Summa та метод keywords, що реалізує алгоритм TextRank [17].

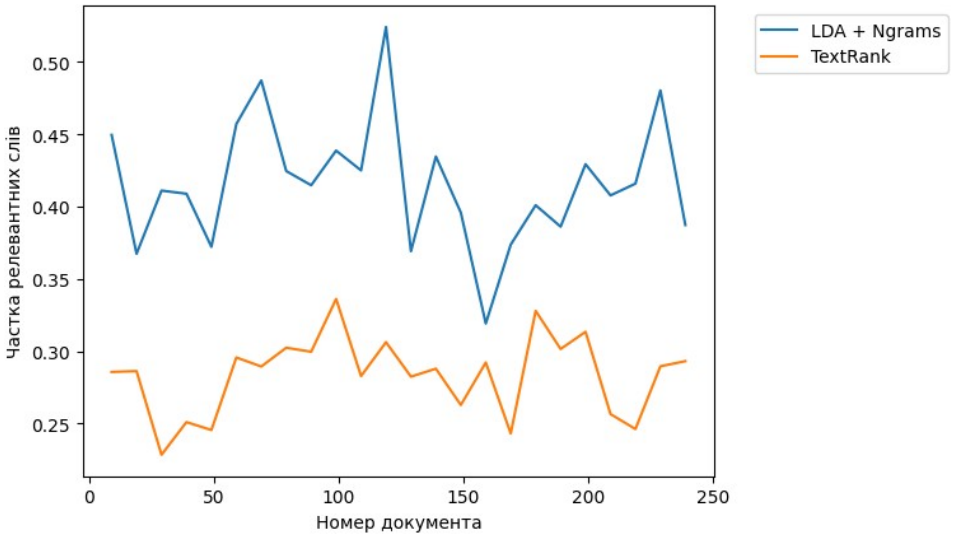


Рисунок 3. 3 - Частки релевантних ключових слів для документів з датасету SemEval2010 отриманих за допомогою методів LDA+Ngrams та TextRank.

На рисунку 3.3 можна побачити, що середня частка релевантних слів, отриманих методом TextRank, менша за метод LDA+Ngrams і дорівнює 0.28. Ці результати свідчать про те, що метод, використаний у даній роботі, може бути більш ефективним для визначення релевантних ключових слів ніж TextRank.

У Таблиці 3.1 наведені приклади ключових слів, що були виявлені за допомогою вище проаналізованих методів. За приклад був взятий текст наукової статті з датасету SemEval2010 під назвою «Cost Sharing in a Job Scheduling Problem Using the Shapley Value» і з наступними ключовими словами призначеними професійними редакторами: cost sharing, job scheduling, shapley value, monetary transfer, fairness axiom, queueing problem, agent, cooperative game theory approach, unit waiting cost, processing time, allocation rule, expected cost bound, queue problem, cost share, job schedule.

Було виявлено, що деякі методи вилучають слова з однаковою основою (як наприклад KeyBert), тоді як інші вибирали нерелевантні короткі слова, що складаються лише з однієї або двох літер (TextRank, TF-IDF). Метод, що використовується в даній роботі, дає більш зв'язні ключові слова, але з обмеженням щодо потенційного дублювання слів, які входять до складу деяких біграм.

Таблиця 3.1 - Результати роботи різних методів виявлення ключових слів на прикладі одного тексту з датасету SemEval2010.

| **Назва методу** | **Ключові фрази** | **Частка релевантних слів** |
|---|---|---|
| LDA+Ngrams | waiting cost, cost, shapley value, job n, queueing problem, value, rule, allocation, processing time, cost share, definition allocation, processing, allocation rule, cost sharing, shapley, share job | 0.82 |
| TF-IDF | cost, sharing, job, scheduling, problem, using, shapley, value, debasis, mishra, center, operations, research, econometrics, core, universit, catholique, de, louvain, la | 0.35 |
| KeyBert | scheduling, economics, incentive, queuing, payoffs, economists, economic, allocation, allocations, queueing, incurred, fairness, efficiency, compensated, queues, costs, compensation, equitable, bargaining, optimal | 0.20 |
| TextRank | cost sharing job, j, order, ordered, efficiency, efficient ordering queue, axiom, game, share, shared, q, n, queueing, rule model, allocation, transfer, transferring, shapley value, problem, set, setting, pi | 0.54 |

## 3.2 Розпізнавання іменованих сутностей

Для тренування і тестування моделі використовувався датасет Annotated Corpus for Named Entity Recognition що складається з 47959 записів. Він містить речення, що розбиті на слова і кожне слово асоціюється з певним тегом і частиною мови.

Тег складається з літери «I» або «B» та позначки, що характеризує суть тегу. «B» вказує на те, що поточне слово є початком сутності, у той час коли «I» вказує на те, що поточне слово знаходиться в середині сутності, але не є її першим словом.

Теги поділяються на 8 класів:

— geo - Географічний об'єкт

— org - Організація

— per - Особа

— gpe - Геополітичний суб'єкт

— tim - Індикатор часу

— art - Артефакт

— eve - Подія

— nat - Природне явище

Діаграма на Рисунку 3.5. демонструє кількість кожного тега в датасеті. Найбільше всього зустрічаються назви географічних об’єктів.

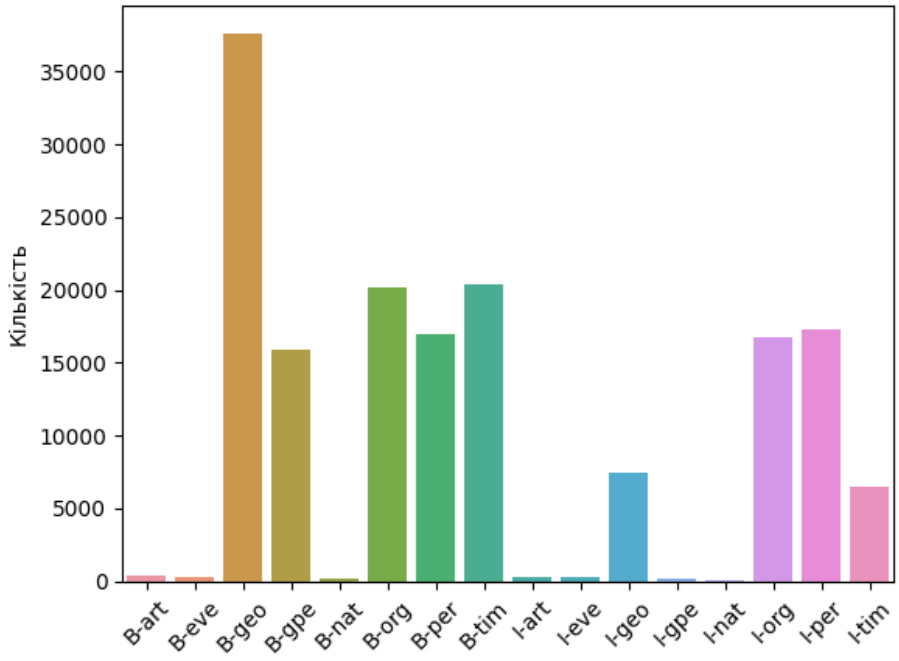


Рисунок 3. 5 - Кількість появи кожної сутності в датасеті.

Тестовий набір складає близько 10% прикладів, тобто 4796. Тренування здійснювалось протягом 10 епох з батч розміром рівним 32. Наприкінці тренування були отримані наступні показники втрат:

— Train loss (укр. втрати на тренувальному датасеті): 0.02222

— Validation loss (укр. втрати на валідаційному датасеті): 0.1187

Після тестування моделі NER з BERT ми згенерували матрицю невідповідностей (Рисунок 3.6), щоб оцінити точність класифікації кожного слова. Оскільки токенізація іноді розбиває слово на підслова, загальна кількість тегів у матриці перевищує кількість вибірок тестового набору даних. Утім, матриця дає інформацію про загальну продуктивність моделі на тестових даних.

Як видно, найбільшу кількість тегів становлять B-geo  - усього 5737, з них правильно були класифіковані 5280 (94%).

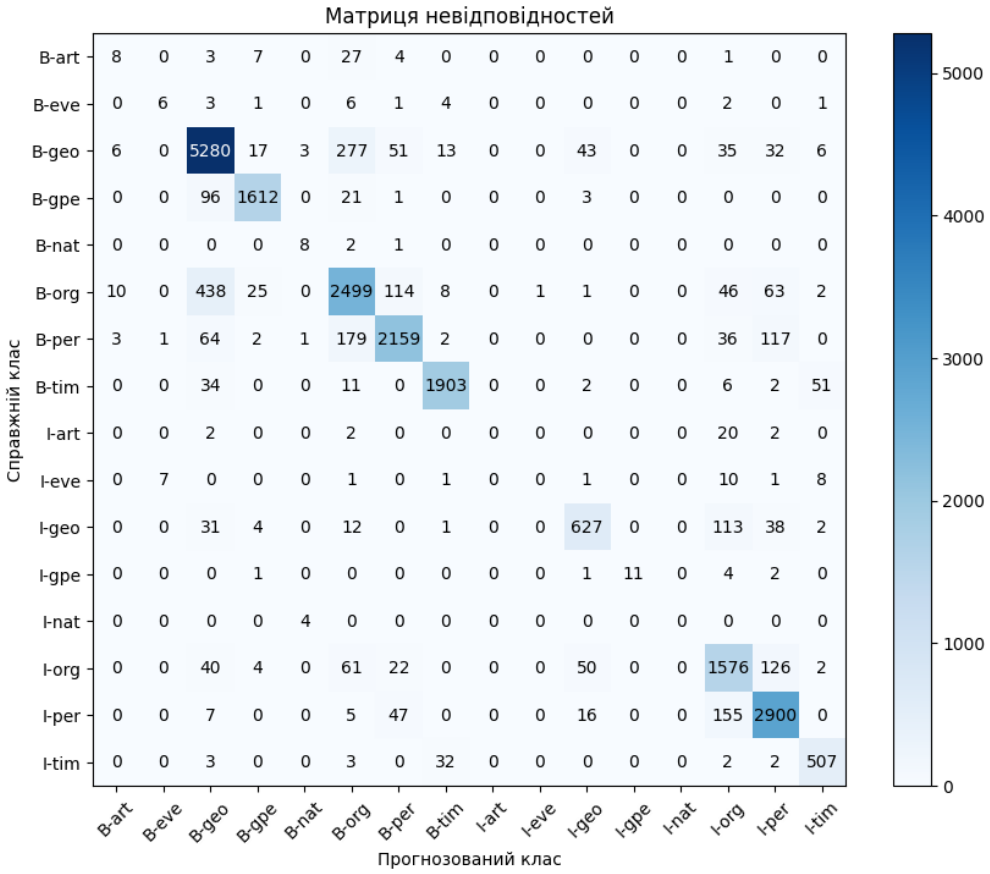


Рисунок 3. 6 - Матриця невідповідностей

У таблиці 3.1. представлені показники accuracy, precision, recall для 9 тегів що становлять найбільшу кількість в тестовому наборі даних. Середнє значення показника точності (accuracy) становить 97% що свідчить про досить точну роботу моделі.

Таблиця 3.1 - Показники метрик для вибраних тегів

| | **Accuracy** | **Precision** | **Recall** |
|---|---|---|---|
| **B-geo** | 0.94 | 0.88 | 0.92 |
| **B-gpe** | 0.99 | 0.96 | 0.93 |
| **B-org** | 0.94 | 0.80 | 0.78 |
| **B-per** | 0.97 | 0.90 | 0.84 |
| **B-tim** | 0.99 | 0.97 | 0.95 |
| **I-geo** | 0.99 | 0.84 | 0.76 |
| **I-org** | 0.97 | 0.79 | 0.84 |
| **I-per** | 0.97 | 0.88 | 0.93 |
| **I-tim** | 0.99 | 0.88 | 0.92 |

## 3.3 Демонстрація роботи системи

Для демонстрації виконання функції «Виявлення ключових слів» (Рисунок 3.7) була використана наукова стаття з архіву електронних публікацій arXiv.org [18]. Що стосується розпізнавання іменованих сутностей, дана функція була протестована на уривку тексту з історичної статті з Вікіпедії [19]. Текст був зчитаний з фотографії (Рисунок 3.8) і завантажений в систему. У результаті кожна з сутностей в тексті виділяється окремим кольором для швидшого розуміння ключових деталей про вміст (Рисунок 3.9).

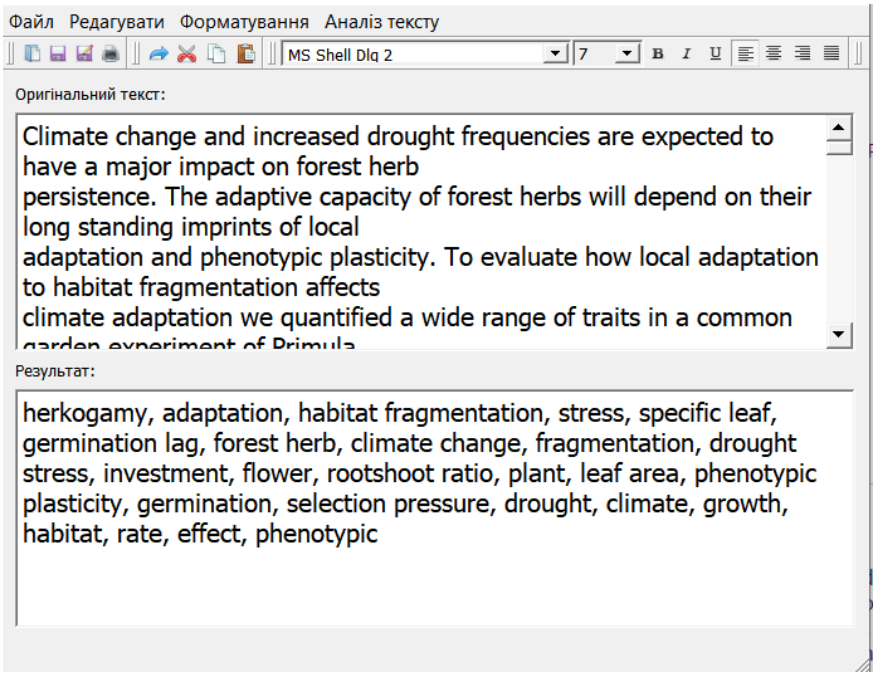


Рисунок 3. 7 - Ключові слова до статті «Habitat fragmentation affects climate adaptation in a forest herb»

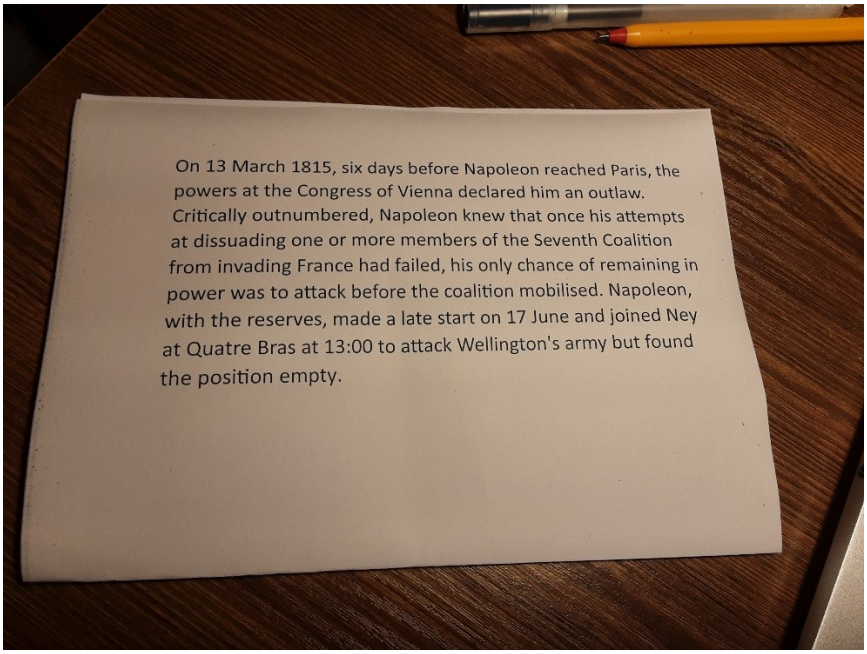


Рисунок. 3. 8 - Фотографія аркуша з текстом

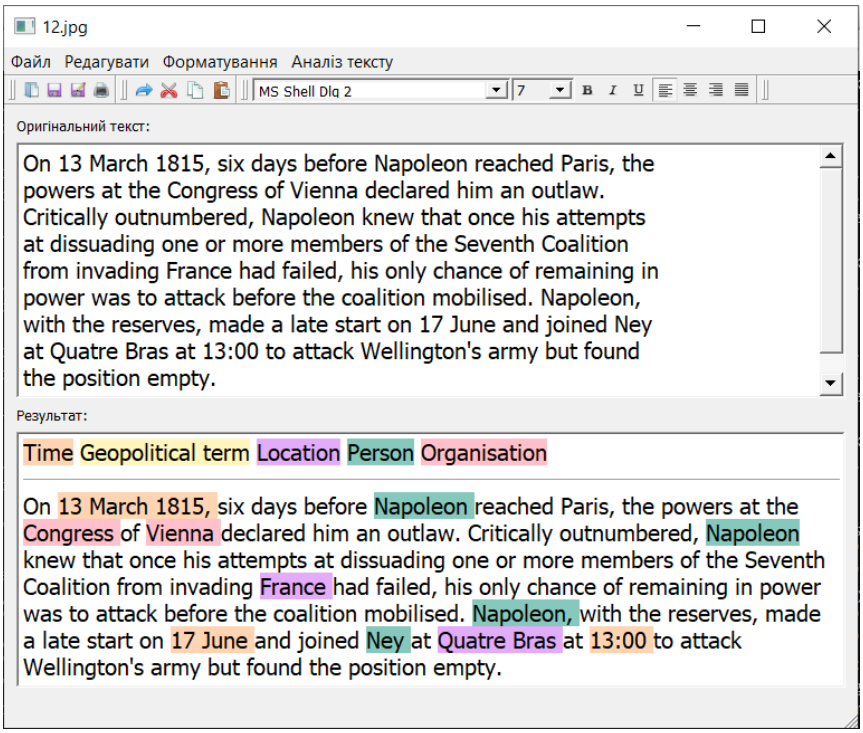


Рисунок. 3. 9 - Розпізнавання іменованих сутностей для уривку тексту про Битву при Ватерлоо

# ВИСНОВОК

У даній роботі була реалізована система опрацювання тексту, що реалізує наступний функціонал:

- Виявлення ключових слів: автоматичне вилучення з документа набору репрезентативних слів, які стисло підсумовують його зміст. Для реалізації були задіяні методи машинного навчання без вчителя, такі як: техніка тематичного моделювання LDA та процедура виявлення словосполучень, що семантично найкраще асоціюються з текстом.
- Розпізнавання іменованих сутностей: класифікація іменованих сутностей в тексті в заздалегідь визначені категорії, такі як імена людей, організації, місця тощо. Розроблений графічний інтерфейс дозволяє користувачам ефективно використовувати даний функціонал, надаючи простий інтерфейс для введення тексту та отримання результатів. Коли користувач вводить текст, викликається модель, яка аналізує текст для виявлення відповідних сутностей. Виявлені сутності виділяються у вихідному тексті, де кожен тип сутності представлений окремим кольором.
- Сканування документа: користувач може завантажити файл зображення, який попередньо обробляється системою, після чого текст, що міститься в зображенні, витягується і повертається у вигляді рядка в поле введення.

Також був здійснений порівняльний аналіз метода виявлення ключових слів, що використовується у системі, з вже існуючими методами. Протягом проведення експериментів було помічено, що метод LDA+Ngrams є більш ефективним ніж методи KeyBert, TextRank та TF-IDF.

На практиці дана система може допомогти студентам, школярам та науковцям швидше опрацьовувати наукові статті, знаходити ключову інформацію в історичних текстах та інших письмових матеріалів, а також ефективно і точно витягувати з них важливі дані. Код програми опублікований на платформі GitHub [20].

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Chengyu Sun, A Review of Unsupervised Keyphrase Extraction Methods Using Within-Collection Resources / Chengyu Sun, Liang Hu, Shuai Li, Tuohang Li, Hongtu Li, and Ling Chi. // Symmetry 12 – 2020 - no. 11: 1864.
2. Rada Mihalcea, TextRank: Bringing Order into Text / Rada Mihalcea, Paul Tarau // Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing – 2004 - pages 404–411
3. David M. Blei. Latent dirichlet allocation / David M. Blei, Andrew Y. Ng, Michael I. Jordan. // The Journal of Machine Learning Research – 2003. - Volume 3. – P. 993–1022.
4. Keyword Extraction with BERT [Електронний ресурс] - Режим доступу до ресурсу: https://towardsdatascience.com/keyword-extraction-with-bert-724efca412ea
5. Yu Zhang, Keywords Extraction with Deep Neural Network Model / Yu Zhang, Mingxiang Tuo, Qingyu Yin, Le Qi // Neurocomput. 383 – 2020 – 113-121
6. Ying, Yan & Qingping, A Graph-based Approach of Automatic Keyphrase Extraction / Ying, Yan & Qingping, Tan & Qinzheng, Xie & Ping, Zeng & Panpan, Li // Procedia Computer Science – 2017 - 248-255
7. Mansouri, Named Entity Recognition Approaches / Mansouri, Alireza & Affendey, Lilly & Mamat, Ali. // Int J Comp Sci Netw – 2008 - Sec. 8.
8. I. Budi, Association rules mining for name entity recognition / I. Budi and S. Bressan // Proceedings of the Fourth International Conference on Web Information Systems Engineering – 2003 - pp. 325-328,
9. Jae-Ho Kim, Unsupervised Named Entity Classification Models and their Ensembles / Jae-Ho Kim, In-Ho Kang, and Key-Sun Choi // In COLING 2002: The 19th International Conference on Computational Linguistics

10. R. Smith, An Overview of the Tesseract OCR Engine / R. Smith // Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) – 2007 - pp. 629-633
11. How to Extract Information from documents: Template Matching [Електронний ресурс] - Режим доступу до ресурсу: https://aicha-fatrah.medium.com/how-to-extract-information-from-documents-template-matching-e0540ae79599
12. Karami Ebrahim, Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images / Karami, Ebrahim & Prasad, Siva & Shehata, Mohamed // Newfoundland Electrical and Computer Engineering Conference – 2015
13. Darling, William M. A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling / Darling, William M. // *In* Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies – 2011 - 642–47.
14. Latent Dirichlet Allocation Using Gibbs Sampling [Електронний ресурс] - Режим доступу до ресурсу: https://ethen8181.github.io/machine-learning/clustering_old/topic_model/LDA.html
15. Annotated Corpus for Named Entity Recognition [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/abhinavwalia95/entity-annotated-corpus
16. Su Nam Kim, SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles / Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin // In Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10). Association for Computational Linguistics – 2010 - 21–26.
17. Summa – Textrank [Електронний ресурс] - Режим доступу до ресурсу: https://summanlp.github.io/textrank/
18. Habitat fragmentation affects climate adaptation in a forest herb [Електронний ресурс] - Режим доступу до ресурсу: https://arxiv.org/abs/2303.15712

19. Battle of Waterloo [Електронний ресурс] - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Battle_of_Waterloo
20. Text-processing-system [Електронний ресурс] - Режим доступу до ресурсу: https://github.com/MariaPonomarenko38/Text-processing-system