МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу

Кафедра інженерії програмного забезпечення

Пояснювальна записка до дипломного проекту

магістра
(освітній ступінь)

на тему «Аналіз рішень розпізнавання тексту з файлів зображень, знаходження та виділення важливого в текстах»

ХАІ.603.667п1.121.156351.20О

Виконав: студент 6 курсу групи №667п1
Спеціальність 121 – Інженерія програмного забезпечення
(код та найменування)
Освітня програма Хмарні обчислення та Інтернет речей
(найменування)
Шатун О.А.
(прізвище й ініціали студента)
Керівник: Пудовкіна Л.Ф.
(прізвище й ініціали)
Рецензент: _____
(прізвище й ініціали)

Харків – 2020

Міністерство світи і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу
(повне найменування)
Кафедра інженерії програмного забезпечення
(повне найменування)
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – інженерія програмного забезпечення
(код та найменування)
Освітня програма хмарні обчислення та Інтернет речей
(найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (ініціали та прізвище)
“_____” _____ 2020 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Шатуну Олексію Анатолійовичу
(прізвище, ім’я, по батькові)

1. Тема дипломного проекту «Аналіз рішень розпізнавання тексту з файлів зображень, знаходження та виділення важливого в текстах»
керівник дипломного проекту Пудовкіна Лариса Федорівна к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету № _____ від “____” _____ 2020 року
2. Термін подання студентом проекту _____
3. Вихідні дані до проекту аналіз рішень розпізнавання тексту з файлів зображень, знаходження та виділення важливого в текстах
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) аналіз рішень розпізнавання тексту з файлів зображень, знаходження та виділення важливого в текстах
5. Перелік графічного матеріалу (усього 91 сторінка), 13 рисунків, 39 таблиць, 35 джерел, додаток А

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Пудовкіна Л.Ф., доцент	02.09.20	02.09.20
2	Пудовкіна Л.Ф., доцент	21.09.20	21.09.20
3	Пудовкіна Л.Ф., доцент	29.09.20	29.09.20

Нормоконтроль _____ _____
«___»_____20__р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання «___»_____20_____ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Вивчення основних правил техніки безпеки, технічної безпеки при роботі з ЕОМ, ознайомлення з місцем роботи, узгодження типового завдання.	02.09.20	
2	Огляд і аналіз можливих рішень розпізнавання тексту з файлів зображень, знаходження та виділення важливого в текстах.	03.09.20 – 20.09.20	
3	Аналіз методів пошуку та виділення важливого в текстах.	21.09.20 – 28.09.20	
4	Розробка програмного забезпечення для проведення експерименту.	29.09.20 – 01.10.20	
5	Аналіз отриманих даних.	02.10.20 – 14.10.20	
6	Оформлювання пояснювальної записки до дипломної роботи	15.10.20 – 16.10.20	
7	Підготовка доповіді	17.10.20	
8	Підготовка презентації	18.10.20	

Студент _____Шатун О.А._____
(підпис) (прізвище та ініціали)

Керівник проекту _____Пудовкіна Л.Ф.__
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломний проект магістра на тему: «Аналіз рішень розпізнавання тексту з файлів зображень, знаходження та виділення важливого в текстах»: 92 сторінки, 13 рисунків, 39 таблиць, 35 джерел, додаток А.

Об’єкт дослідження – існуючі процеси пошуку та виділення важливого в текстах.

Предмет дослідження – технології для розробки систем пошуку та виділення важливого в текстах.

Мета - проаналізувати існуючі системи виділення та пошуку важливого в текстах а також виявити найбільш вдалі рішення.

Методи дослідження – статистичного аналізу, планування експерименту для аналізу ефективності систем.

Мета роботи полягає в покращенні пошуку важливої інформації в файлах зображень шляхом аналізу технологій і сервісів пошуку та виділення важливого в текстах та реалізації найбільш вдалого рішення у вигляді web-додатку.

На етапі аналізу проблеми проаналізовані системи пошуку та виділення важливого в текстах: розглянуті відмінні риси таких систем, проаналізовано проблеми використання та розробки цих систем, розглянуті основні технологічні рішення таких систем, як: Яндекс.Толока, AWS Lambda та Docker.

Актуальність даної роботи полягає в тому, щоб користувачеві було зручно використовувати необхідні для нього можливості в одному web-додатку, а саме: завантажувати файли зображень та здійснювати пошук важливого тексту, а також використовувати візуально приємний перегляд знайденої важливої інформації.

ПОШУК ТЕКСТУ, ВИДІЛЕННЯ ТЕКСТУ, ВИДІЛЕННЯ ВАЖЛИВОГО, ПОШУК ВАЖЛИВОГО, РОЗПІЗНАВАННЯ ТЕКСТУ

РЕФЕРАТ

Дипломный проект магистра на тему: «Анализ решений распознавания текста из файлов изображений, нахождение и выделение важного в текстах»: 92 страницы, 13 рисунков, 39 таблиц, 35 источников, приложение А.

Объект исследования - существующие процессы поиска и выделения важного в текстах.

Предмет исследования - технологии для разработки систем поиска и выделения важного в текстах.

Цель - проанализировать существующие системы выделения и поиска важного в текстах а также выявить наиболее удачные решения.

Методы исследования - статистического анализа, планирования эксперимента для анализа эффективности систем.

Цель работы заключается в улучшении поиска важной информации в файлах изображений путем анализа технологий и сервисов поиска и выделения важного в текстах и реализации наиболее удачного решения.

На этапе анализа проблемы проанализированы системы поиска и выделения важного в текстах: рассмотрены отличительные черты таких систем, проанализированы проблемы использования и разработки этих систем, рассмотрены основные технологические решения таких систем, как: Яндекс.Толока, AWS Lambda и Docker.

Актуальность данной работы заключается в том, чтобы пользователю было удобно использовать необходимые для него возможности в одном web-приложения, а именно: загружать файлы изображений и осуществлять поиск важного текста, а также использовать визуально приятный просмотр найденной важной информации.

ПОИСК ТЕКСТА, ВЫДЕЛЕНИЕ ТЕКСТА, ВЫДЕЛЕНИЯ ВАЖНОЙ, ПОИСК ВАЖНОГО, РАСПОЗНАВАНИЕ ТЕКСТА

ABSTRACT

Master's thesis on the topic: "Analysis of text recognition solutions from image files, finding and highlighting important in the texts": 92 pages, 13 figures, 39 tables, 35 sources, Appendix A.

The object of research is the existing processes of searching for and highlighting what is important in the texts.

The subject of research - technologies for the development of search systems and highlighting important in the texts.

The aim is to analyze the existing systems of selection and search for the important in the texts and to identify the most successful solutions.

Research methods - statistical analysis, experimental planning to analyze the effectiveness of systems.

The aim of the work is to improve the search for important information in image files by analyzing technologies and services for finding and highlighting important texts and implementing the most successful solution in the form of a web-application.

At the stage of problem analysis the systems of search and selection of important in texts are analyzed: distinctive features of such systems are considered, problems of use and development of these systems are analyzed, the main technological decisions of such systems as: Yandex.Toloka, AWS Lambda and Docker are considered.

The relevance of this work is that the user was comfortable to use the necessary features in one web-application, namely: download image files and search for important text, as well as use a visually pleasing view of the important information found.

TEXT SEARCH, TEXT HIGHLIGHTING, IMPORTANT HIGHLIGHTING, IMPORTANT SEARCH, TEXT RECOGNITION

ЗМІСТ

ГЛОСАРІЙ

Браузер, або веб-оглядач (від англ. Web browser, МФА: [wɛb braʊ.zə (ɹ), -zəɻ]) - прикладне програмне забезпечення для перегляду сторінок, змісту веб-документів , комп'ютерних файлів і їх каталогів; управління веб-додатками; а також для вирішення інших завдань. У глобальній мережі браузери використовують для запиту, обробки, маніпулювання і відображення змісту веб-сайтів. Багато сучасні браузери також можуть використовуватися для обміну файлами з серверами FTP, а також для безпосереднього перегляду змісту файлів багатьох графічних форматів (gif, jpeg, png, svg), аудіо-відео форматів (mp3, mpeg), текстових форматів (pdf, djvu) та інших файлів.

Комп'ютерна платформа - в загальному сенсі, це будь-яка існуюча Виконавча, в якій здійснюватиметься знову розробляється фрагмент програмного забезпечення або об'єктний модуль з урахуванням накладених цим середовищем обмежень і можливостей, що надаються. Термін платформа може застосовуватися до різних рівнів абстракції, включаючи певну апаратну архітектуру, операційну систему або бібліотеку часу виконання.

Плагін - незалежно компільований програмний модуль, що динамічно підключається до основної програми і призначений для розширення і / або використання її можливостей.

ВСТУП

Книги, журнали, газети, доповіді та інші текстові документи є незаперечним джерелом знань. Людина, яка читає книги вдосконалюється, розвиває свій кругозір, логіку, мислення, покращує свою пам'ять. Однак доволі часто учням, вчителям, викладачам, студентам, працівникам чи просто людині, яка хоче дізнатися лише те що її цікавить не є необхідним переглядати всю чотирьохсот сторінкову книгу заради необхідних трьох речень, або знайти необхідний текст в декількох фото, чи в документі що зберігається у вигляді зображень, адже попри доволі швидкого розвитку технологій значна кількість книг не проходили через технологію оптичного розпізнавання символів, окрім цього частина книг взагалі не зберігається в інтернет мережі.

Зазвичай документи, а особливо книги технічного характеру розбиті на частини, розділи, глави, параграфи і так далі, однак це не завжди вирішує проблему пошуку важливої інформації, адже часто буває так, коли необхідна інформація знаходиться в різних частинах книги.

Також важливою частиною в пошуку інформації є якісне та зручне виділення інформації, яка необхідна користувачу.

Справжня робота і намагається вирішити ці завдання.

Актуальність даної роботи полягає в тому, щоб користувачеві було зручно використовувати необхідні для нього можливості в одному web-додатку, а саме: завантажувати файли зображень та здійснювати пошук важливого тексту, а також використовувати візуально приємний перегляд знайденої важливої інформації.

Мета роботи полягає в покращенні пошуку важливої інформації в файлах зображень шляхом розробки програмного забезпечення для виділення важливого в текстах.

У зв'язку з цим було сформульовано такі задачі:

1 Вибрати алгоритми обробки текстів на природній мові для реалізації і визначити метрики якості підсвічування важливого в текстах.

2 Спроектувати архітектуру і реалізувати рішення, що включає реалізацію обраних алгоритмів і середовище для експериментів.

3 Провести порівняльний аналіз алгоритмів, що підсвічують важливі слова в текстах.

Об’єкт дослідження – існуючі процеси пошуку та виділення важливого в текстах.

Предмет дослідження – технології для розробки систем пошуку та виділення важливого в текстах.

Методи дослідження – статистичного аналізу, планування експерименту для аналізу ефективності методів пошуку важливого.

Наукова новизна дослідження: аналіз найкращих та найгірших рішень у системах пошуку та виділення важливого в текстах, а також знаходження найбільш вдалої тактики розробки таких систем.

Практичне значення результатів: дослідження складається з використаних теоретичних положень наукових досліджень для вирішення схожих задач, пов’язаних із розробкою систем пошуку та виділення важливого в текстах.

1 АНАЛІЗ РІШЕНЬ РОЗПІЗНАВАННЯ ТЕКСТУ З ФАЙЛІВ ЗОБРАЖЕНЬ, А ТАКОЖ ПОШУК НАЙКРАЩИХ ПРОЦЕСІВ ДЛЯ ЗНАХОДЖЕННЯ ТА ВИДІЛЕННЯ ВАЖЛИВОГО В ТЕКСТАХ. ПОСТАНОВКА ЗАДАЧ ДИПЛОМНОГО ПРОЕКТА МАГІСТРА

1.1 Пошук важливого в текстах

Для того щоб визначити важливий текст необхідним буде використання завдань екстрактивної суммарізації, а також потрібно визначити статистичну оцінку важливості слів.

Прийнято розуміти, що екстрактивна суммарізація це процес вибору пропозицій з тексту або підмножини слів, які зберігають основний сенс тексту. В результаті чого, якщо робота алгоритму є якісною, то знайдений та опрацьований текст стає коротшим, але не втрачає головну суть тексту. Отриманий в результаті екстрактивної суммарізації текст і пропонується виділяти серед тексту що подається на вході.

Статична оцінка важливості слів - міра важливості слова, яка залежить від частоти його вживання в конкретному документі, а також від рідкості використання в корпусі документів.

У цій роботі описані алгоритми, екстрактивної та абстрактної суммарізації, розглядається складність використання стандартних метрик якості, а також проводиться огляд технологій і сервісів, які використовуються для реалізації поставлених задач на платформі браузерних розширень.

1.2 Суммарізація тексту

Зараз доступна величезна кількість текстової інформації по будь-якій темі, яка може інтерпретуватися у вигляді книг, журналів, газет, доповідей, документів та іншого. Щоб скоротити час на ознайомлення з цікавою

інформацією використовують алгоритми суммарізації текстів. Завдання алгоритмів сумарізації полягає у тому, щоб виділити з потоку текстових даних головні ідеї і створити на їх основі скорочений текст, який буде легко читатися та не втратить головної суті (рис 1.1). Так, суммарізація може допомогти зрозуміти зміст тієї чи іншої наукової статті, отримати свіжі витяги з новин або полегшити розуміння юридичного висновку чи фінансового звіту. Автореферірювання актуально практично у всіх областях, так як суттєво скорочує час читання.

Економія часу на читанні актуальна і щорічно публікується безліч статей, що описують нові методи і поліпшення існуючих рішень. Найбільший успіх мають нейронні мережі, але є і більш прості і швидкі підходи, які використовуються в більшості статей в якості вихідної точки для порівняння якості. Однак, оптимального та універсального рішення задачі автоматичної суммарізації ще не знайдено.

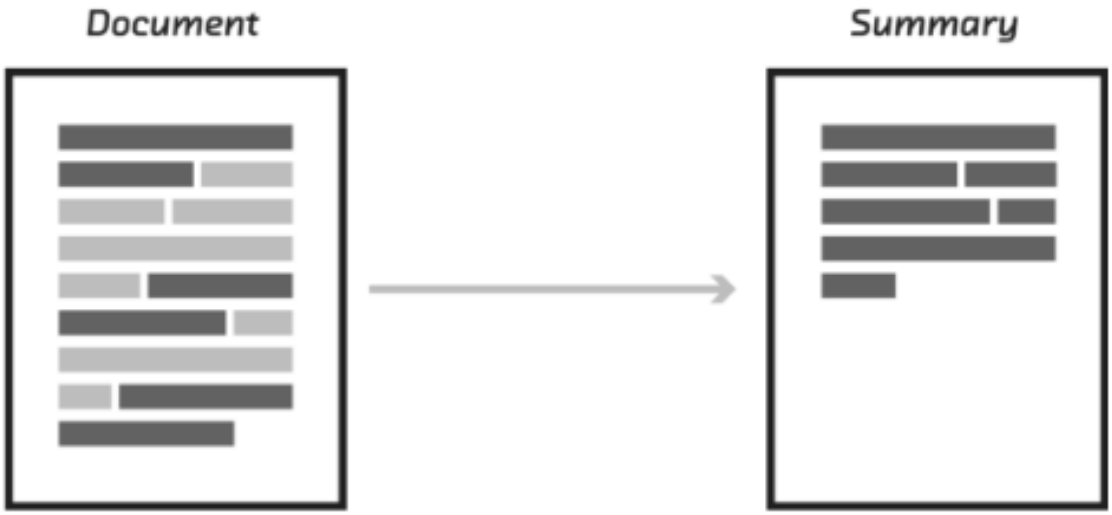

Рисунок 1.1 - Суммарізація тексту

Отже, суммарізації можна дати наступне визначення: суммарізація – це автоматичне створення короткого змісту (заголовка, резюме, анотації)

вихідного тексту. Для виконання завдання сумарізації існує два істотно різні підходи:

- екстрактивний;
- абстрактивний.

1.2.1Екстрактивна суммарізація тексту

Екстрактивний підхід полягає в пошуку найбільш «значущих» інформаційних блоків та їх вилученні з вихідного тексту. В якості блоку можуть бути окремі абзаци, пропозиції або ключові слова.

1.2.2 Абстрактивна суммарізація тексту

На відміну від екстрактивного підходу, абстрактивний підхід суттєво відрізняється від свого попередника і полягає в тому, що генерує короткий зміст з породженням нового тексту, змістовно узагальнюючого первинний документ.

Основна ідея даного підходу полягає в тому, що модель здатна генерувати абсолютно унікальне резюме, яке може містити слова, відсутні в початковому тексті. Отриманий текст являє собою деякий переказ тексту, який ближчий до ручного складання короткого змісту тексту людьми.

1.3 Підсвічування тексту

Підсвічування синтаксису - виділення синтаксичних конструкцій тексту з використанням різних кольорів, шрифтів і накреслень. Зазвичай застосовується для полегшення читання вихідного тексту комп'ютерних програм, поліпшення візуального сприйняття. Підсвічування синтаксису - важлива функція текстових редакторів, середовищ розробки, відладчиків і інших інструментів розробки програмного забезпечення. Підсвічування

синтаксису також застосовується при публікації вихідних кодів в інтернеті і на паперових носіях.

Колір, очевидно, є найвпливовіший психологічний інструмент, наявний в розпорядженні програмування підсвітки тексту. Сплеск кольору, відповідного контексту ситуації, миттєво налаштовує читача на потрібний емоційний лад. колірний акцент Відносно тексту колір застосовується для виділення різних елементів, які потребують залучення уваги користувача. Історично склалося так, що в розпорядженні авторів було два інструменти акцентуації: курсив і жирний шрифт.

До настання цифрової епохи колір в друкарському справі перебував у великому дефіциті - він був призначений тільки для обкладинок і, в дуже рідкісних випадках, для кольорових ілюстрацій, ніяк не стикаючись з текстом як таким. Для позначення зміни контексту автор міг використовувати курсив, для виділення ключових слів - жирний шрифт. Зрозуміло, що тепер, коли в розпорядженні будь-якого розробника є 16 777 216 кольорів, що відображаються на середньої руки моніторі, можливо розфарбувати текст в будь-яку, саму фантастичну гаму. Однак робити цього не потрібно, адже перше правило візуалізації тексту говорить: «читати, не страждаючи». З виконанням цього правила і пов'язане повсюдне вживання домінуючою класичної чорно-білої схеми, «як в друкарні на папері».

До стандартного чорно-білому корпусу тексту можна додати всілякі кольори, щоб привернути увагу читача. захоплення уваги. На цільовій сторінці кольором виділяються найважливіші елементи, на які користувач повинен звернути увагу: заголовки, дати, терміни, ключові слова і фрази.

Крім залучення уваги, колір може послужити ще одним відмінним способом полегшити взаємодію між письменником і читачем. Нижче наведено приклад оригінального рішення, за твердженням його авторів, що збільшує швидкість читання електронного тексту на 30%. Ключами до гарної типографіки є 2 поняття: розбірливість - наскільки добре розрізняються при

читанні букви, фрази і слова, і читаність - наскільки мозку легко перетворити їх в єдине повідомлення.

Коли очі сканують сторінку, вони легко можуть збитися з потрібною траєкторії. Можуть траплятися випадки коли людина інколи читає один і той же рядок двічі, або пропускає один. Хоч постійно читаючи і відбувається тренування очей для плавання в морях тексту, але його монотонність як і раніше змушує втомлюватися і помилятися. Так ось, додавання колірного градієнта - як на рис.1.2. - і робить перехід від одного рядка до іншого легше для очей: вони безпомилково слідують по заданій траєкторії, що в результаті веде до зростання швидкості читання.

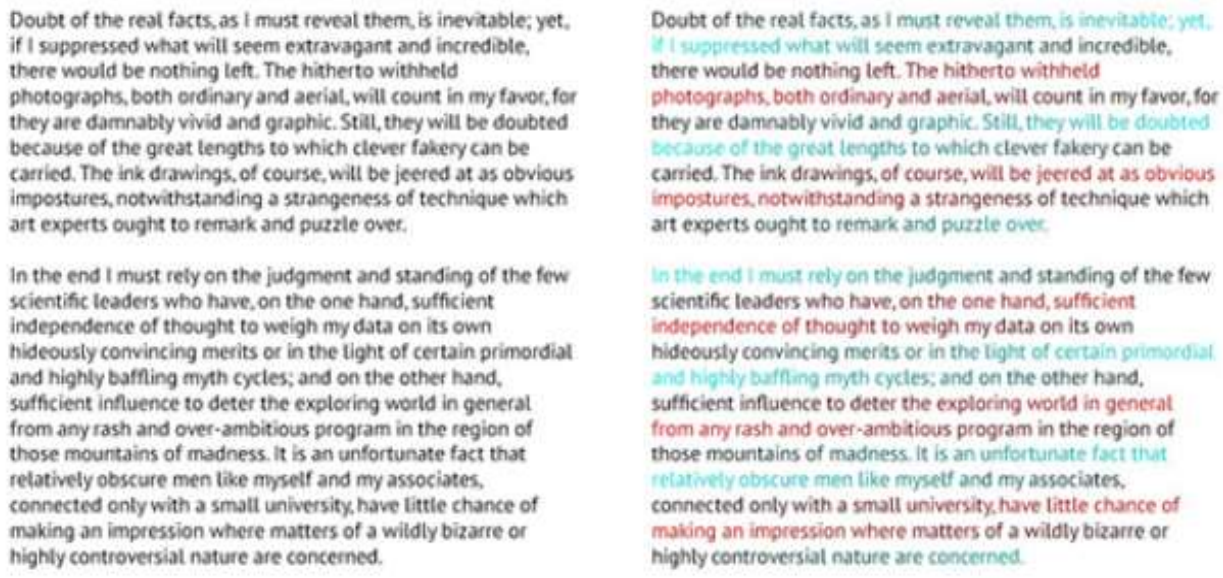

Рисунок 1.2 - Додавання колірного градієнта

1.4 Оптичне розпізнавання символів

Розпізнавання тексту - це окрема частина задач комп'ютерного зору. Як і багато алгоритмів комп'ютерного зору, до популярності нейромереж воно багато в чому грунтувалося на ручних ознаках і евристиках. Однак за останній

час, з переходом на нейромережеві підходи, якість технології розпізнавання тексту істотно зросла [1].

Оптичне розпізнавання символів (англ. Optical character recognition, OCR) - механічний або електронний переказ зображень рукописного, машинописного або друкованого тексту в текстові дані, які використовуються для представлення символів в комп'ютері (наприклад, в текстовому редакторі). Розпізнавання широко застосовується для перетворення книг і документів в електронний вигляд, для автоматизації систем обліку в бізнесі або для публікації тексту на веб-сторінці [2].

Розпізнавання зображень структурованих (друкованих) символів забезпечує рішення ряду наукових і прикладних задач при ідентифікації об'єктів різної природи. Сучасні методи розпізнавання символів використовуються для вирішення як типових задач, наприклад розпізнавання тексту, так і спеціалізованих задач, орієнтованих на розпізнавання символьної інформації, нанесеної на поверхню різних об'єктів. Існує достатньо велика кількість програм, призначених для розпізнавання тексту (наприклад, FineReader, Readiris, ScanSoft OmniPage та ін.). Кожна з цих програм пропонує свою реалізацію вирішення задачі обробки та розпізнавання зображень.

1.4.1 Шаблонний метод

Шаблонні методи перетворюють зображення окремого символу в растрове, порівнюють його зі всіма шаблонами, наявними в базі і вибирають шаблон з найменшою кількістю крапок, відмінних від вхідного зображення. Шаблонні методи досить стійкі до дефектів зображення і мають високу швидкість обробки вхідних даних, але надійно розпізнають тільки ті шрифти, шаблони яких їм „відомі”. І якщо розпізнаний шрифт хоч трохи відрізняється від еталонного, шаблонні методи можуть робити помилки навіть при обробці дуже якісних зображень.

1.4.2 Структурний метод

Структурні методи розпізнавання зберігають інформацію не про покрапкове написання символу, а про його топологію. Еталон містить інформацію про взаємне розташування окремих складових частин символу. Перевага методу – стійкість до зсуву і повороту символу на невеликий кут, до різних стильових варіацій шрифтів. Однак, при повороті на кут, більший десяти градусів, даний метод не може бути використаний для розпізнавання символів. При застосування цього методу неважливими стають такі ознаки як розмір букви, що розпізнається і навіть шрифт, яким вона надрукована. Проте, основною проблемою цього методу є ідентифікація знаків, які містять певні дефекти (наприклад, розрив ліній або з'єднання сусідніх ліній).

1.4.3 Ознаковий метод

Ознакові методи базуються на тому, що зображенню ставиться у відповідність N-мірний вектор ознак. Розпізнавання полягає в порівнянні вектора ознак з набором еталонних векторів тієї ж розмірності. Переваги методу – простота реалізації, хороша узагальнююча здатність, висока швидкість розпізнавання. Недолік методу – висока чутливість до дефектів зображення. Крім того, ознакові методи мають інший недолік — на етапі виділяння ознак відбувається незворотня втрата частини інформації про символ. Виділяння ознак проходить незалежно, тому інформація про взаємне розташування елементів символів втрачається.

1.4.4 Нейромережевий метод

Нейромережеві методи [3] засновані на застосуванні різних типів штучних нейронних мереж. Ідея цих методів - моделювання роботи мозку людини. На вхід заздалегідь навченої нейронної мережі надходить вектор, який є поданням вхідного образу (пікселі, частотні характеристики, вейвлет). На виході нейрон, відповідний класу розпізнаного символу, видає максимальне значення функції активації. Або ж на вихід надходить безліч ключових характеристик зображення, які потім обробляються іншими системами. Навчання нейронних мереж відбувається на безлічі навчальних прикладів. Причому можливе навчання з учителем (персептрон) або самоорганізація (мережа Кохонена).

Перевагами методу є: здатність до узагальнення, висока швидкість роботи.

Недоліки: чутливість до обертання і спотворення символів, складність підбору навчальної вибірки і алгоритму навчання.

1.5 Постановка задач дипломної роботи

Отже, для того щоб користувачеві було більш зручно використовувати необхідні для нього можливості, а саме: здійснювати пошук важливої інформації в файлах зображень, необхідно виконати наступні дії:

- проаналізувати рішення обробок файлів зображень, методи пошуку важливого тексту та підходи підсвічування тексту;
- виконати пошук переваг та недоліків розглянутих рішень обробок файлів зображень, методів пошуку важливого тексту та підходів підсвічування тексту;

- проаналізувати отримані данні та виявити найвдаліші рішення обробок файлів зображень, методів пошуку важливого тексту та підходів підсвічування тексту.

1.6 Висновки з розділу 1

У даному розділі було розглянуто суммарізацію тексту, яка має два підходи:

- екстрактивний;
- абстрактивний.

Був проведений аналіз підсвічування тексту, а також розглянуто оптичне розпізнавання символів, яке поділяють на наступні методи:

- шаблонний метод;
- структурний метод;
- ознаковий метод;
- нейромережевий метод.

Також була описана актуальність даної роботи, було виявлено проблеми, а також була проаналізована актуальність рішень.

Була поставлена задача для даної роботи, а саме:

- проаналізувати рішення обробок файлів зображень, методи пошуку важливого тексту та підходи підсвічування тексту;
- виконати пошук переваг та недоліків розглянутих рішень обробок файлів зображень, методів пошуку важливого тексту та підходів підсвічування тексту;
- проаналізувати отримані данні та виявити найвдаліші рішення обробок файлів зображень, методів пошуку важливого тексту та підходів підсвічування тексту.

Таким чином, був проведений аналіз пошуку, підсвічування та розпізнавання тексту в файлах зображень, доведена актуальність теми дослідження, доцільність пошуку найкращих рішень для розпізнавання, пошуку, та підсвічування важливого в текстах.

2 МЕТОДИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, ПОШУКУ ТА ВИДІЛЕННЯ ВАЖЛИВОГО В ТЕКСТАХ

2.1 Розпізнавання тексту

Одним з найкращих способів розпізнавання зображень є двоетапний підхід, званий адаптивним розпізнаванням, який використовує нейронні мережі для пошуку і розпізнавання тексту на зображеннях.

Для виконання цього підходу потрібно виконати один прохід за даними для розпізнавання символів, а потім другий прохід, щоб заповнити будь-які літери, в яких не було впевненості, буквами, які, швидше за все, відповідають даному слову або контексту пропозиції.

2.2 Методи та підходи сумарізації тексту

2.2.1Мовна модель нейронної неттової мережі (NNLM)

Імовірнісна модель мови нейронної мережі запропонована в [8]. Він складається з вхідного, проекційного, прихованого та вихідного шарів. На вхідному шарі кодується N попередніх слів за допомогою кодування 1-V-V, де V - розмір словникового запасу. Потім вхідний шар проектується на проекційний шар P, що має розмірність N × D, використовуючи загальну матрицю проекції. Оскільки лише N входів у будь-який момент часу активні, склад проекційного шару є досить дешевою операцією.

Архітектура NNLM стає складною для обчислення між проекцією та прихованим шаром, оскільки значення в шарі проекції щільні. Для загального вибору N = 10 розмір шару проекції (P) може становити від 500 до 2000, тоді як розмір прихованого шару H зазвичай становить від 500 до 1000 одиниць. Більше того, прихований шар використовується для обчислення розподілу ймовірностей для всіх слів у словниковому запасі, в результаті чого виходить

шар з розмірністю V. Таким чином, складність обчислень у кожному прикладі навчання є

$$Q = N \times D + N \times D \times H + H \times V, \quad (2.1)$$

де домінуючим членом є H × V. Однак було запропоновано кілька практичних рішень для його уникнення; або з використанням ієрархічних версій софмакс [9], або уникнення повністю нормалізованих моделей за допомогою моделей, які не нормалізуються під час тренувань [10]. Якщо двійкові уявлення про словниковий запас, кількість вихідних одиниць, які необхідно оцінити, може знизитися до $\log_2$ (V). Таким чином, більша частина складності обумовлена терміном N × D × H.

У цих моделях ми використовуємо ієрархічну софтмакс, де словниковий запас представлений у вигляді бінарного дерева Хаффмана. З цього випливає попереднє спостереження, що частота слів добре працює для отримання класів в моделях нейронної мережі [11]. Хаффман-дерева присвоюють короткі бінарні коди частим словам, і це додатково зменшує кількість вихідних одиниць, які потрібно оцінити: хоча збалансоване бінарне дерево вимагає оцінювати виходи $\log_2$ (V), ієрархічний софтмакс на основі Хаффмана вимагає лише про $\log_2$ (Уніграм недоумкування (V)). Наприклад, коли розмір словникового запасу становить мільйон слів, це призводить до приблизно в два рази прискорення оцінки. Хоча це не є вирішальним прискоренням MH нейронної мережі, оскільки обчислювальне вузьке місце знаходиться в терміні N × D × H, ми згодом запропонуємо архітектури, які не мають прихованих шарів, і таким чином сильно залежать від ефективності нормалізації програмного забезпечення.

2.2.2Поточна модель нейронної мережі (RNNLM)

Мовна модель, заснована на нейромережевій мережі, була запропонована для подолання певних обмежень поданої NNLM, таких як необхідність

вказувати довжину контексту (порядок моделі N), і тому що теоретично RNN можуть ефективно представляти складніші структури, ніж дрібні нейронні мереж [12]. Модель RNN не має проекційного шару; тільки вхідний, прихований і вихідний шар. Особливістю для цього типу моделі є періодична матриця, яка з'єднує прихований шар до себе, використовуючи затримки в часі. Це дозволяє періодичній моделі формувати якусь короткострокову пам'ять, оскільки інформація з минулого може бути представлена станом прихованого шару, який оновлюється на основі поточного введення та стану прихованого шару на попередньому кроці часу.

Складність для навчального прикладу моделі RNN така:

$$Q = H \times H + H \times V, \tag{2.2}$$

де слово представлення D має ту саму розмірність, що і прихований шар H. Знову ж таки, термін H × V можна ефективно зменшити до H × $\log_2$ (V), використовуючи ієрархічну софтмакс. Більшість складностей тоді виникає з H × H.

2.2.3Паралельне навчання нейронних мереж

Для підготовки моделей на величезних наборах даних ми реалізували кілька моделей на основі широкомасштабної розподіленої системи під назвою DistBelief [13], включаючи подачу NNLM та нові моделі, запропоновані в цій роботі. Рамка дозволяє запускати кілька реплік однієї і тієї ж моделі паралельно, і кожна репліка синхронізує свої оновлення градієнта через централізований сервер, який зберігає всі параметри. Для цього паралельного навчання використовується міні-пакетний асинхронний градієнтний спуск із адаптаційною процедурою швидкості навчання під назвою Адаград [14]. У цих рамках зазвичай використовується сто чи більше реплік моделей, кожна з

яких використовує багато ядер процесора на різних машинах у центрі обробки даних.

2.2.4Модель безперервної сумки (Continuous Bag-of-Words Model)

Перша запропонована архітектура схожа на подачу NNLM, де нелінійний прихований шар видаляється, а шар проекції є спільним для всіх слів (не тільки матриці проекції); таким чином, усі слова проектуються в одне і те ж положення (їхні вектори усереднюються). Цю архітектуру називають моделлю мішків, оскільки порядок слів в історії не впливає на проекцію. Крім того, також використовуються слова з майбутнього; була отримана найкраща ефективність завдання, , побудувавши на вході лінійно-лінійний класифікатор з чотирма майбутніми і чотирма словами історії, де критерієм навчання є правильне класифікація поточного (середнього) слова.

Складність навчання тоді:

$$Q = N \times D + D \times \log_2 (V). \qquad (2.3)$$

Позначаємо цю модель далі як CBOW, оскільки на відміну від стандартної моделі мішкових слів, вона використовує постійне розподілене подання контексту. Зауважимо, що вагова матриця між вхідним та проекційним шаром поділяється для всіх позицій слова так само, як і в NNLM.

2.2.5Модель безперервного пропускання грамів

Друга архітектура схожа на CBOW, але замість прогнозування поточного слова на основі контексту, воно намагається максимально класифікувати слово на основі іншого слова в тому ж реченні. Точніше, використовується кожне поточне слово як вхід до журнального лінійного класифікатора з безперервним шаром проекцій і передбачати слова в певному діапазоні до і після поточного слова. Було з’ясовано, що збільшення діапазону

покращує якість отриманих векторів слів, але також збільшує складність обчислень. Оскільки більш віддалені слова зазвичай менш пов'язані з поточним словом, ніж ті, що є поруч із ним, ми віддаємо меншу вагу віддаленим словам, відбираючи менше з цих слів у наших навчальних прикладах.

Складність навчання цієї архітектури пропорційна

$$Q = C \times (D + D \times \log_2(V)), \qquad (2.4)$$

де C - максимальна відстань слів. Таким чином, якщо була обрана C = 5, для кожного навчального слова необхідно вибирати випадковим чином число R у діапазоні <1; C>, а потім використовувати R слова з історії та R слова з майбутнього поточного слова як правильні позначки. Це вимагатиме класифікації R × 2 слів із поточним словом як вхідним, а кожне з R + R слів як вихідним.

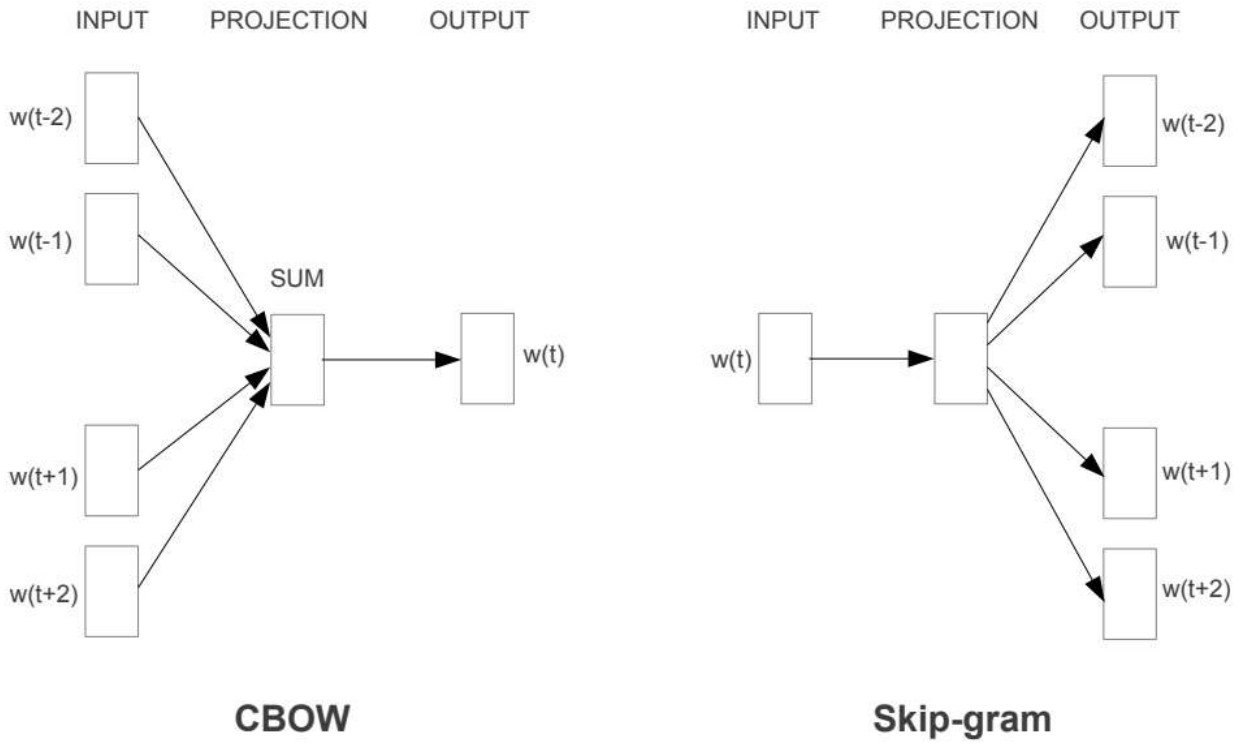

Рисунок 2.1 - Нові модельні архітектури. Архітектура CBOW прогнозує поточне слово на основі контексту, а пропускний грам прогнозує оточуючі слова, задані поточним словом.

2.2.6Латентний семантичный аналіз (LSA)

Процес починається зі створення термінів матрицею речень A = [A_1 A_2 ::: A_n] з кожним вектором стовпця A_i, що представляє зважений векторний термін-частота речення і у розглянутому документі. Якщо в документі є загалом m термінів і n пропозицій, то у нас буде матриця m n для документа. Оскільки кожне слово зазвичай не відображається в кожному реченні, матриця A зазвичай є рідкою.

Враховуючи матрицю mn A, де без втрати загальності m n, SVD A визначено як [15]:

$$A = U VT, \tag{2.5}$$

де U = [u_{ij}] - m n стовпець-ортонормальної матриці, стовпчики якої називаються лівими сингулярними векторами; = diag (1; 2; :::; n) - nn діагональна матриця, діагональні елементи якої є негативними сингулярними значеннями, відсортовані у порядку зменшення, а V = [v_{ij}] - nn ортонормальна матриця, стовпці якої називаються правильні сингулярні вектори. Якщо ранг (A) = r, то задовольняє es

$$1\ 2\ r > r + 1 == n = 0. \tag{2.6}$$

Інтерпретацію застосування SVD до термінів матрицею речень A можна зробити з двох різних точок зору. З точки зору трансформації, SVD отримує відображення між m-мірним простором, що охоплюється зваженими термінами-частотними векторами, і r-мірним сингулярним векторним

простором з усіма його осями лінійно незалежними. Це відображення проектує кожен вектор стовпця і в матриці A, який представляє зважений векторний термін-частота речення і, в вектор стовпця і = [v_{i1} v_{i2} v_{ir}] T матриці VT, і відображає кожен вектор рядків j в матриці A, який повідомляє кількість зустрічей терміна j у кожному з документів, на векторний рядок j = [u_{j1} u_{j2} u_{jr}] матриці U. Тут кожен елемент v_{ix} з і , u_{jy} з j називається індексом з x_0-го, y_0-го сингулярних векторів відповідно.

З семантичної точки зору SVD отримує приховану семантичну структуру з документа, представленого матрицею A [16]. Ця операція відображає розбиття оригінального документа на r лінійно незалежні базові вектори або концепції. Кожен доданок і речення з документа є спільно індексуються цими базовими векторами / поняттями. Унікальною особливістю SVD, якої не вистачає у звичайних технологіях IP, є те, що SVD здатний фіксувати та моделювати взаємозв'язки між термінами, щоб він міг семантично кластеризувати терміни та пропозиції. Розгляньте слова лікарня, лікар, медсестра. Слова лікарня, медицина, медсестра - це тісно пов’язані поняття. Як показано в роботі [17], якщо шаблон комбінації слів є чітким і повторюваним у документі, цей шаблон буде захоплений і представлений одним із сингулярних векторів. Величина відповідного сингулярного значення вказує на ступінь важливості цієї картини в документі. Будь-які речення, що містять цю схему комбінації слів, будуть запроектовані уздовж цього особливого вектора, і речення, яке найкраще представляє цей шаблон, матиме найбільше значення індексу з цим вектором. Оскільки кожна конкретна схема комбінації слів описує певну тему / концепцію в документі, факти, описані вище, природно призводять до гіпотези, що кожен сингулярний вектор являє собою важливу тему / концепцію документа, а величина відповідного йому особливого значення являє собою ступінь важливості важливої теми / концепції.

На підставі вищезгаданої дискусії розглянемо наступний метод узагальнення документів на основі SVD.

1. Розкласти документ D на окремі речення, і використовувати ці речення для формування речення з кандидатом множина S, а k = 1.

2. Побудувати доданки за матрицею речень для документа D.

3. Виконати SVD на A для отримання матриці сингулярного значення та правої матриці сингулярного вектора VT. У сингулярному векторному просторі кожне речення і представлено стовпчастим вектором i = [v_{i1} v_{i2} v_{ir}] T з VT.

4. Обрати k-й правий сингулярний вектор з матриці VT.

5. Обрати речення, яке має найбільше значення індексу, з k-м правильним одниною вектора, і включіть його у зведення. 6. Якщо k досягає заданого числа, припинити операцію; в іншому випадку збільшити k на один і перейти до кроку 4.

На кроці 5 вищезгаданої операції, вказавши речення, що має найбільше значення індексу з k-м правильним сингулярним вектором, еквівалентним фіксації вектора стовпця i, k-й елемент v(ik) є найбільшим. Згідно з гіпотезою, ця операція еквівалентна виведенню найкращого речення, що описує виразну концепцію / тему, представлену k-м сингулярним вектором. Оскільки сингулярні вектори відсортовані у порядку зменшення відповідних їм сингулярних значень, k-ий сингулярний вектор представляє важливу концепцію / тему k. Оскільки всі однинні вектори незалежні один від одного, речення обраний цим методом містить мінімальну надмірність.

2.3 Алгоритми суммарізації і виділення важливих слів в текстах природною мовою

Програмні продукти, засновані на алгоритмах обробки текстів на природній мові [4], в останні роки набули широкого користування. Визначення настрою тексту [5], видача по пошукових запитом відповідних

Web-сторінок [6], машинний переклад [7] - все це приклади того, як NLP допомагає вирішувати завдання інформаційного пошуку та робити програмні інтерфейси більш природними для людини. Для вирішення поставленої задачі необхідно було вибрати алгоритми, які можуть бути використані для екстрактивної суммаризації і визначення найбільш важливих слів в тексті.

В результаті огляду предметної області в даній роботі були вибрані наступні алгоритми:

- мовна модель нейронної неттової мережі (NNLM)
- поточна модель нейронної мережі (RNNLM)
- паралельне навчання нейронних мереж
- модель безперервної сумки (Continuous Bag-of-Words Model)
- модель безперервного пропускання грамів
- латентно семантичный аналіз (LSA)

2.4 Метрики якості підсвічування

Завдання обробки текстів на природній мові прийнято ділити на наступні категорії: класифікація тексту, визначення семантичного змісту тексту, генерація тексту по тексту, розробка діалогових систем.

Підсвічування важливого в тексті лежить на перетині завдань визначення семантичного сенсу тексту і генерації тексту по тексту [18]. Наявні дослідження [19, 20, 21, 22, 23] показують, що в даних категоріях завдань для оцінки якості використовуються класичні метрики якості: частка правильних відповідей (precision), ROUGE, F-міра, BLEU, AUC. Всі ці метрики об'єднує той факт, що для їх використання необхідно мати розмічений набір даних (dataset), який і використовується для порівняння передбачень моделі, що вийшла з актуальними результатами. При відсутності альтернативних варіантів, для оцінки якості в задачах NLP може бути використана оцінююча

група [24] - група людей, оцінки роботи алгоритму яких вважаються вірними. У зв'язку з цим дана метрика і була обрана в даній роботі для оцінювання якості алгоритмів.

2.5 Платформа Яндекс.Толока

Для залучення експертної групи та проведення зрівнювального аналізу застосування різних алгоритмів до задачі виділення головного в тексті в даній роботі використовувалася платформа Яндекс.Толока [25] (рисунок 2.2). Дана платформа є сервісом компанії Яндекс, який використовується самою компанією для вдосконалення пошукових алгоритмів і машинного інтелекту. Щодня десятки тисяч людей виконують завдання в Яндекс.Толокі: оцінюють релевантність сайтів, класифікують зображення, відзначають об'єкти на фотографіях [26].

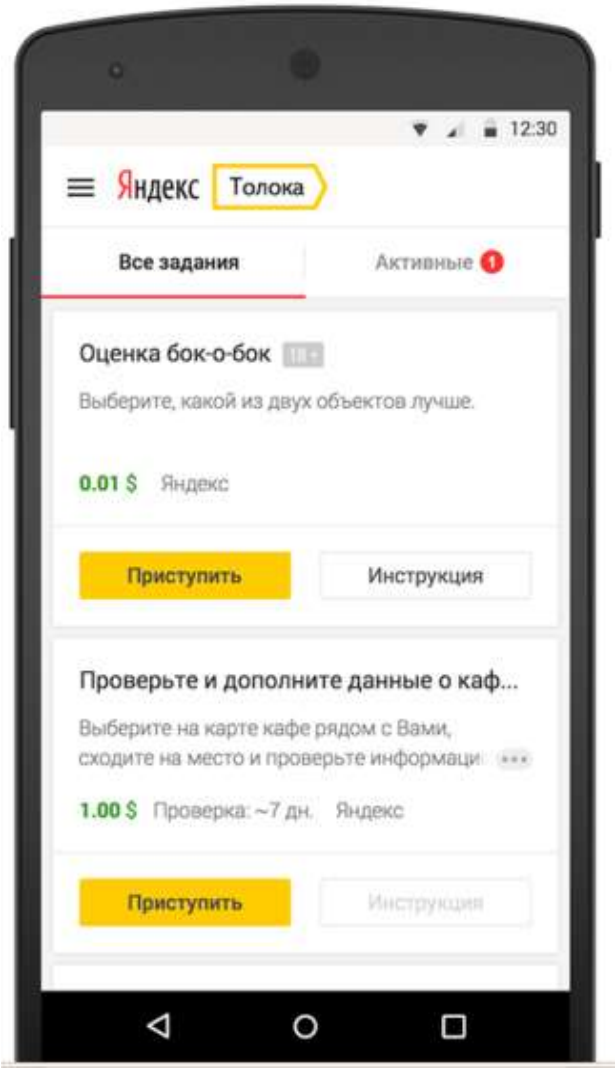

Рисунок 2.2 - Яндекс.Толока

На рисунку 2.3 представлений інтерфейс завдання, який найчастіше має кілька мультимедійних файлів і форму відправки відповідей. Медіафайли [27] і їх зовнішній вигляд легко змінюються в налаштуваннях завдання, що робить даний сервіс відповідним майже до будь-якого типу завдань зрівняння і класифікації.

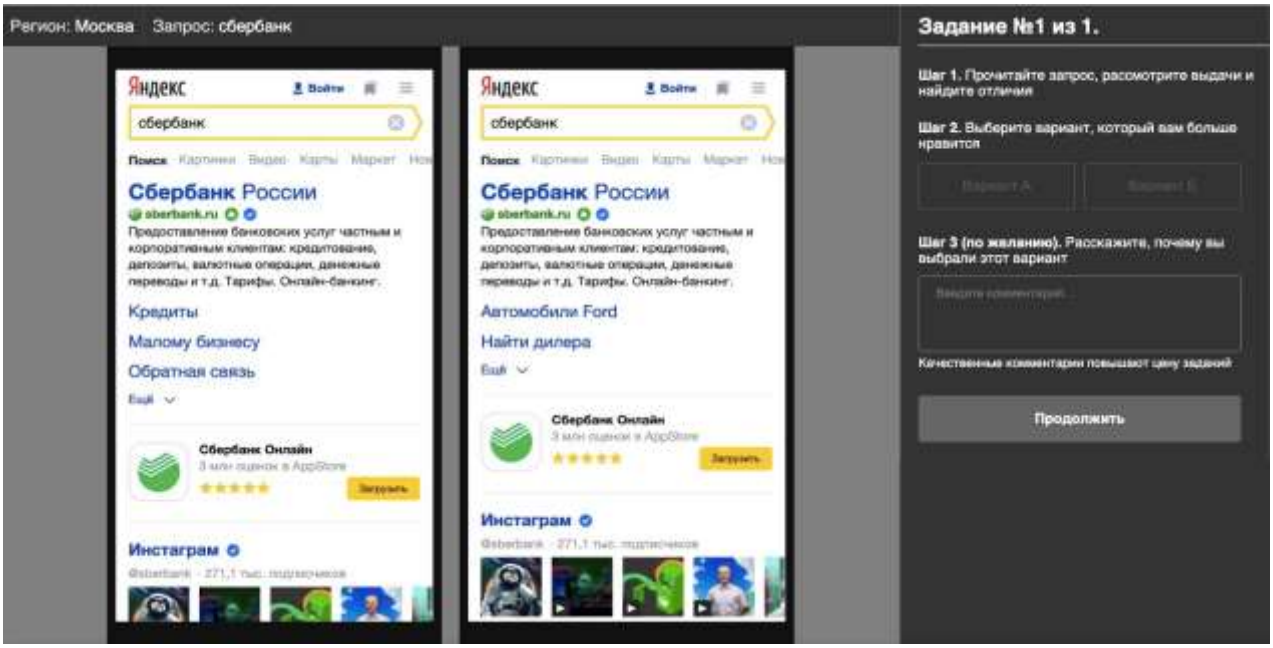

Рисунок 2.3 - Приклад завдання з порівнянням двох варіантів пошукової видачі

Основне завдання сервісу полягає в тому, щоб з'єднувати між собою замовників і виконавців. Наприклад, якщо замовнику необхідно розмітити деякий набір даних, він може створити завдання з докладною інструкцією, а потім відправити ці задачі користувачам на виконання. Для замовника також існує можливість фільтрації списку виконавців з різних параметрам, таким як стать, вік, геопозиція і інші. Ці та інші можливості дозволяють дослідникам отримувати найбільш релевантні відповіді.

Яндекс.Толока підтримує REST API [28] який дозволяє автоматично створювати і редагувати завдання, а також отримувати та обробляти результати. Поставлена задача передбачає роботу з великою кількістю даних, яке виключає можливість ручного обробки результатів, тому наявність програмного інтерфейсу є дуже корисним.

2.6 Платформа AWS Lambda

Клієнт-серверна архітектура вимагає розгортання додатків на сервері. Даний розділ забезпечує огляд платформи AWS Lambda[29], яка була використана в даній роботі для цих цілей.

Популярність платформ безсерверних обчислень стрімко збільшилась в останні роки [30]. Даний тип обчислень, незважаючи на назву, не передбачає повну відсутність серверів в архітектурі додатків. Розробнику пропонується можливість завантажувати свій програмний код на платформу, яка буде запускати його в власній хмарі як функцію, надаючи доступ до неї через API для роботи з хмарної інфраструктурою. Таким чином, розробник замість розгортання і обслуговування власного сервера використовує наданий API. (рисунок 2.4)Така архітектура підходить для додатків, в яких клієнтам не потрібно постійне з'єднання з сервером, а досить нечастих запитів для обміну інформацією. Використовувана в даній роботі архітектура є прикладом такої клієнт-серверної взаємодії.

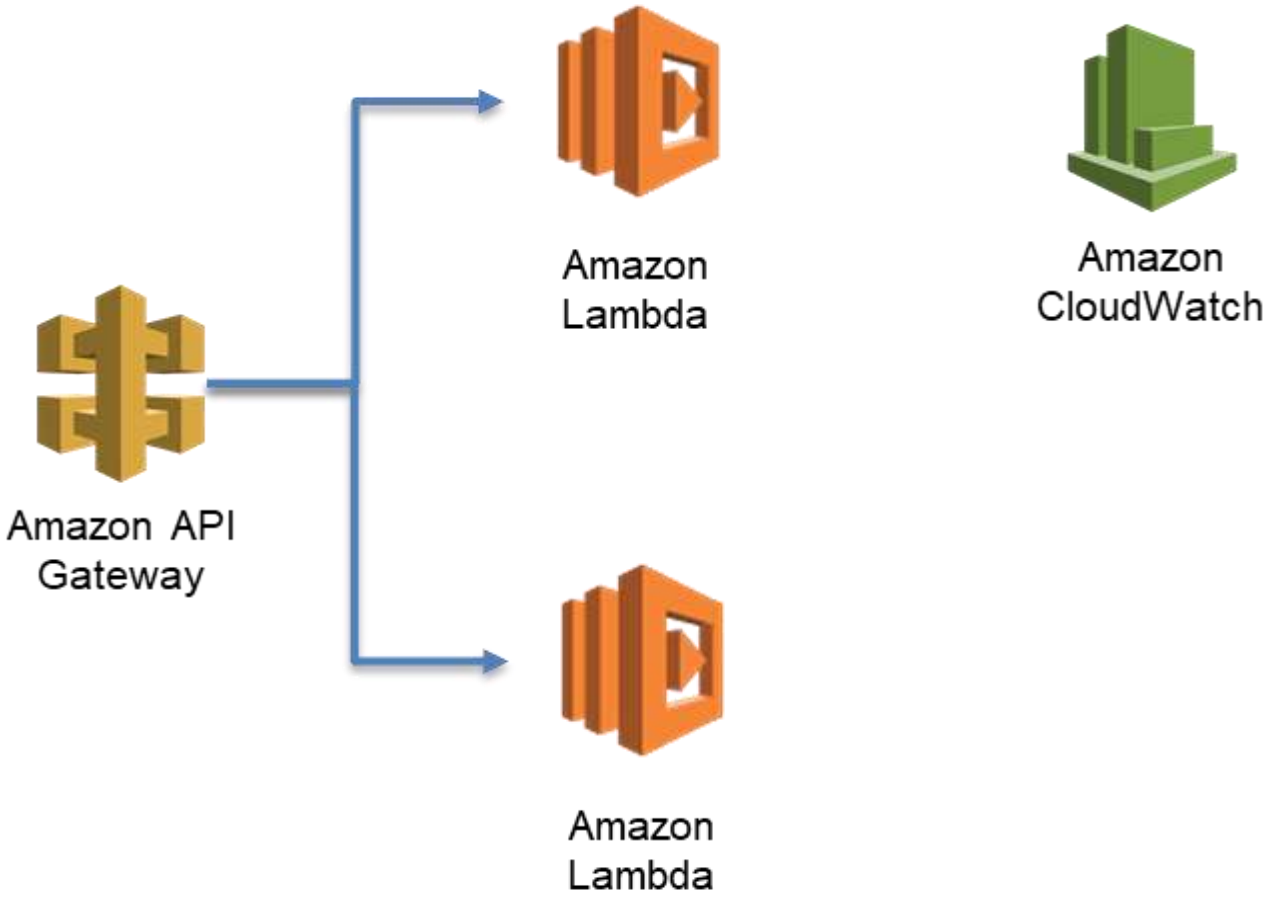

Рисунок 2.4- AWS Lambda

AWS Lambda запускає програмний код у власній обчислювальному середовищі, самостійно займаючись адмініструванням, масштабуванням і моніторингом коду. Від користувача потрібно тільки дозавантажити вихідний код і всі потрібні залежності, а також установити вимоги до ресурси.

AWS Lambda використовує модель без збереження стану, що робить Lambda функції схожими за змістом на чисті функції. На справді Lambda використовує кешування для оптимізації швидкості виконання, але цей факт не дозволяє виконувати установку залежностей вихідного коду при кожному новому виклику. Тому завантажуватися в AWS Lambda повинен архів, в якому містяться всі необхідні для виконання програмні файли.

Після завантаження програмного коду на сервіс створюється функція Lambda, яка перебуває в стані постійної готовності до запуску. Такий запуск може відбутися у відповідь на певні події, в тому числі HTTP запити.

Lambda підтримує інтеграцію з сервісом Amazon API Gateway[31]. Даний сервіс призначений для створення, публікації, обслуговування, моніторингу та забезпечення безпеки API в будь-яких масштабах. Виклики до створеного API можуть оброблятися різними серверними сервісами Amazon, в тому числі AWS Lambda. Таким чином вирішується завдання по інтеграції реалізацій алгоритмів в розширення: обмін інформацією відбувається за допомогою HTTPS запитів на спецціальні адреса, створювані API Gateway.

2.7 Тема віртуалізації Docker

Система віртуалізації Docker використана в даній роботі для емуляції оточення, в якому реалізовані алгоритми будуть виконуватися на бессерверній платформі. Даний розділ аргументує необхідність цієї системи в розробленому вирішенні, а також призводить огляд основних принципів її роботи (рисунок 2.5).

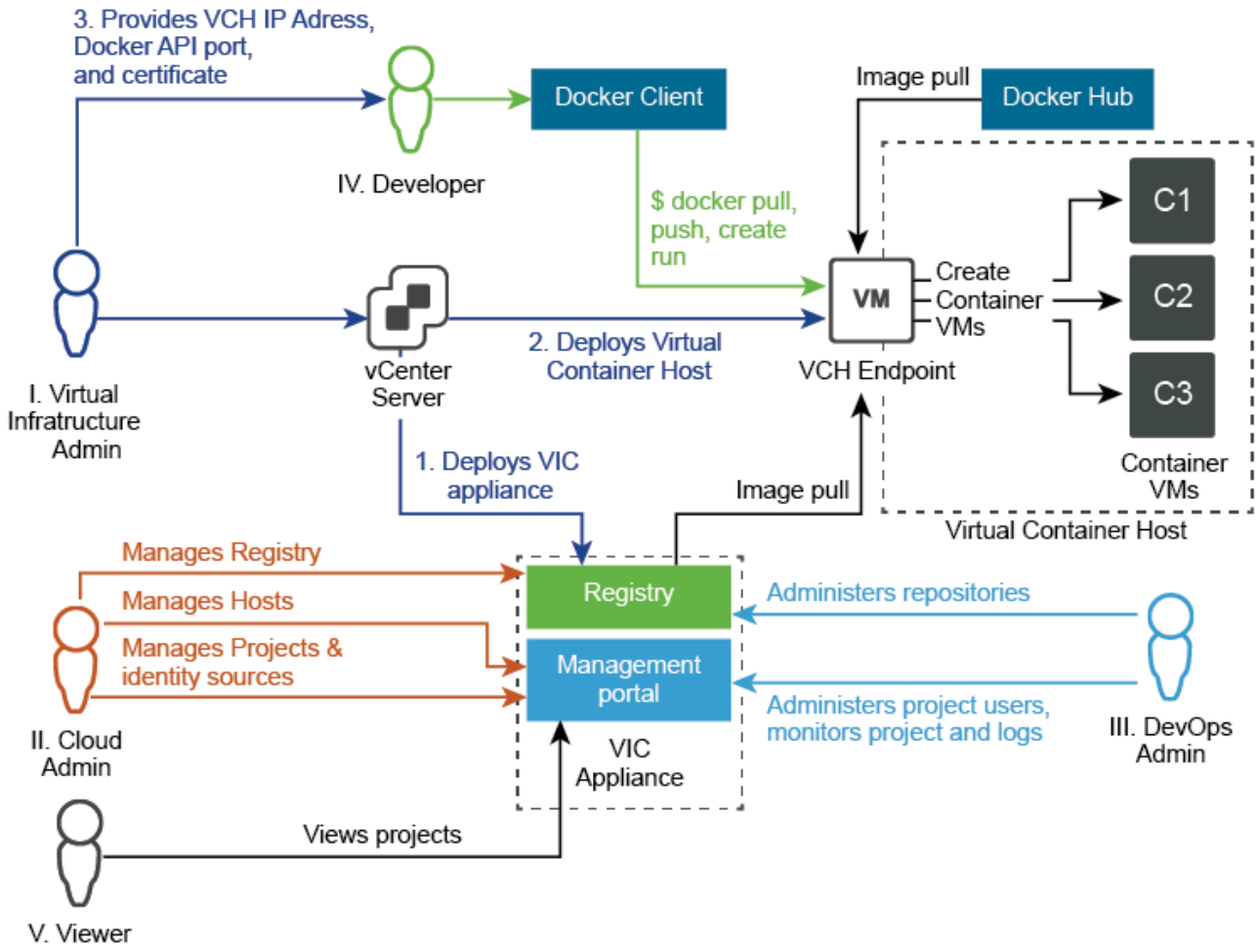

Рисунок 2.5- Система віртуалізації Docker

Кожна Lambda функція виконується в хмарному оточенні, яке найчастіше відрізняється від локального оточення розробника. У зв'язку з цим стає актуальним питання забезпечення ідентичною середовища виконання додатків на локальній машині і в хмарі. Для рішеннях таких завдань можуть використовуватися в тому числі різні засоби віртуалізації на рівні операційної системи, коли в рамках ОС носія можуть одночасно існувати кілька гостьових ОС. Серед розроблених і використовуваних на даний момент інструментів був обраний Docker, так як він є єдиним кроссплатформенним рішенням (підтримує Linux, FreeBSD, Windows і MacOS).

Docker оперує такими термінами, як образ і контейнер (рисунок 2.6). Образ по своїй суті є незмінним шаблоном операційної системи і середовища

виконання. На основі образів Docker створює і запускає контейнера, в які користувач додає додатки по власному розсуду, використовуючи файл збірки, званий Dockerfile. В листінг 1 наведено приклад використання образу microsoft / aspnetcore для запуску dotnet MyApplication.dll з директорії користувача / App.

```
1 FROMm i c ro s o f t / aspnetc o re
2 WORKDIR /app
3 COPY bin /Debug/ publ i s h .
4 ENTRYPOINT[ " dotnet " , " MyApplication . dl l " ]
```

Лістинг 1: Приклад коду Dockerfile

В системі Docker також існують регістри, які по суті є сховищами для образів (як, наприклад GitHub є сховищем для вихідного коду).

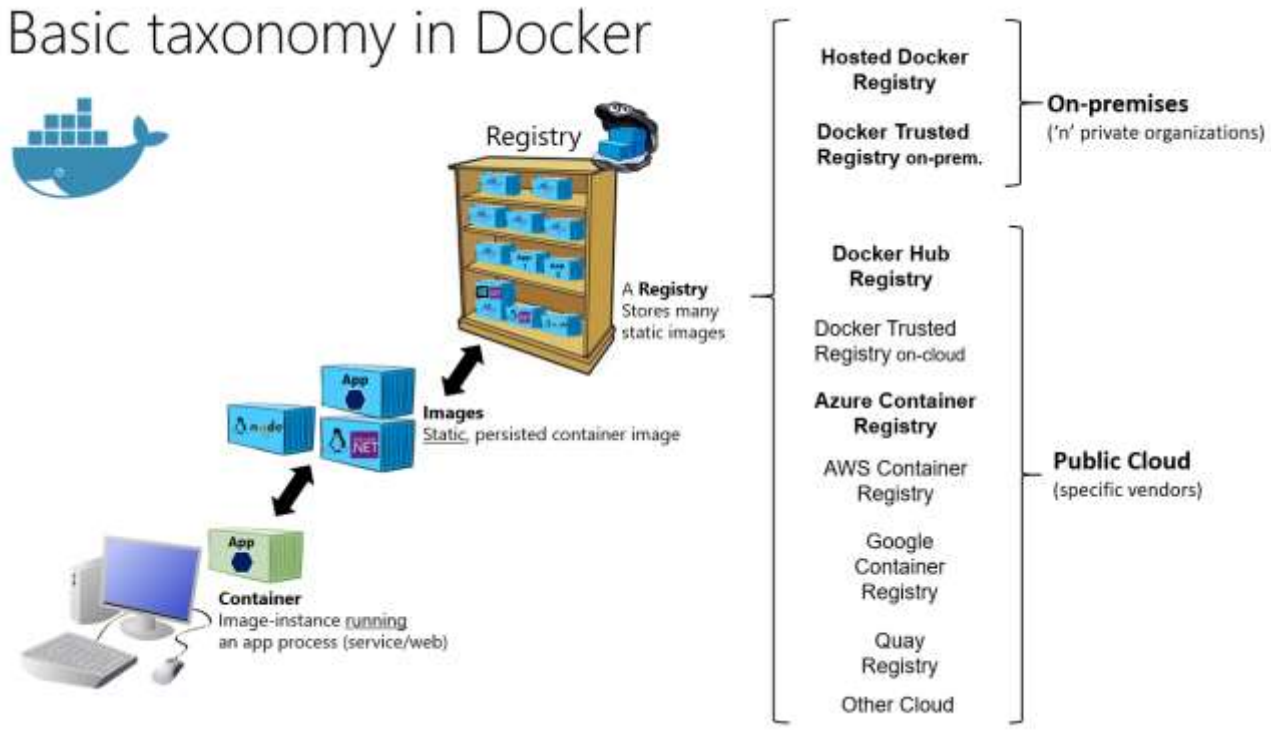

Рисунок 2.6- Регістри системи віртуалізації Docker

Одним з найпопулярніших є регістра DockerHub, який в тому числі зберігає доступний публічно образ [32], ідентичний оточенню AWS Lambda. Таким чином становиться можливим локальна розробка і тестування додатків в оточенні, в якому потім буде виконуватися розгортається функція на сервісі Lambda.

2.8 Висновки з розділу 2

У другому розділі роботи були розглянуті 5 алгоритмів і один підхід з випадковым підсвічуванням, а саме:

- мовна модель нейронної неттової мережі (NNLM);
- поточна модель нейронної мережі (RNNLM);
- паралельне навчання нейронних мереж;
- модель безперервної сумки (Continuous Bag-of-Words Model);
- модель безперервного пропускання грамів;
- латентний семантичный аналіз (LSA).

Обрані наступні алгоритми NLP для підсвічування важливого в тексті: TF-IDF в трьох варіаціях, TextRank в двох варіаціях, алгоритм з використанням норм векторів, алгоритм з використанням LSA. Також була обрана метрика якості "оцінююча група".

Був обраний двоетапний підхід, який використовує нейронні мережі для пошуку і розпізнавання тексту на зображеннях.

Також були обрані наступні технології: Яндекс.Толока, AWS Lambda, Docker.

3. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБОРУ ДАННИХ ЩОДО ВИДІЛЕННЯ ВАЖЛИВОГО В ТЕКСТАХ З ФАЙЛІВ ЗОБРАЖЕНЬ

У цьому розділі представлена архітектура а також реалізація рішень виділення важливого в текстах, а також були проведені експерименти для пошуку кращих методів знаходження важливого в текстах.

3.1 Формування цілей і постановка задачі для розробки програмного забезпечення

Мета проекту, створеного в рамках даної роботи – розробити програмне забезпечення для збору даних щодо виділення важливого в текстах з файлів зображень.

Для досягнення мети даного проекту необхідним буде створення програмного забезпечення, з можливістю завантаження файлів зображень, подальшого розпізнавання в них тексту та аналізування його складу, задля пошуку та виділенню важливих тем, а також збереженням інформації в базі даних.

Важливою частиною створення програмного забезпечення є декомпозиція завдання, тому необхідно виділити основні етапи розробки програмного забезпечення:

- формування вимог до програмного забезпечення;
- проектування структури бази даних, розробка серверної частини веб-додатку, інтеграція з базою даних;
- проектування і розробка дизайну користувальницького інтерфейсу, робота над досвідом взаємодії;
- розробка користувальницького інтерфейсу і клієнтської частини веб-додатку;
- проектування та створення бази даних.

Відповідно, розробку програмної частини веб-додатку можна розбити на 3 модуля [33]:

- модуль клієнтської частини, який виконується веб-браузером користувача;
- модуль серверної частини веб-додатку, який виконується на сервері;
- модуль мобільної частини яка виконується на смартфоні користувача.

Очевидно, таке розбиття на модулі породжує три види зв'язків:

- між браузером і сервером;
- між сервером і базою даних;
- між смартфоном і сервером.

Для того, щоб розробка кожного модуля проводилася максимально незалежно, необхідно, щоб кожен модуль оперував тільки необхідним йому набором даних, а також володів прозорим і простим інтерфейсом для взаємодії користувача з web-додатком.

3.2 Побудова діаграми варіантів використання

Візуальне моделювання в UML можна представити як деякий процес порівневого спуску від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи. Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case diagram), яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування. Діаграма варіантів використання є вихідним концептуальним уявленням або концептуальною моделлю системи в процесі її проектування і розробки.

Розробка діаграми варіантів використання переслідує наступні цілі:

- Визначити загальні межі і контекст модельованої предметної області на початкових етапах проектування системи.

- Сформулювати загальні вимоги до функціонального поведінки проектованої системи.
- Розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.
- Підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на моделируемую систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає акторові. Іншими словами, кожен варіант використання визначає деякий набір дій, що чиниться системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою.

Система має один вид користувача: користувач web -додатку, який має такі функції (рис. 3.1):

- завантаження зображень;
- обробка зображень;
- перегляд розпізнаного тексту;
- вибір критерія виведення інформації;
- пошук важливого;
- створення документу;
- перегляд збережених документів;
- вигрузка документів;
- видалення текстів;
- налаштування веб-додатку;
- копіювання тексту в буфер обміну

- перегляд інформації про web-додаток
- авторизація
- реєстрація

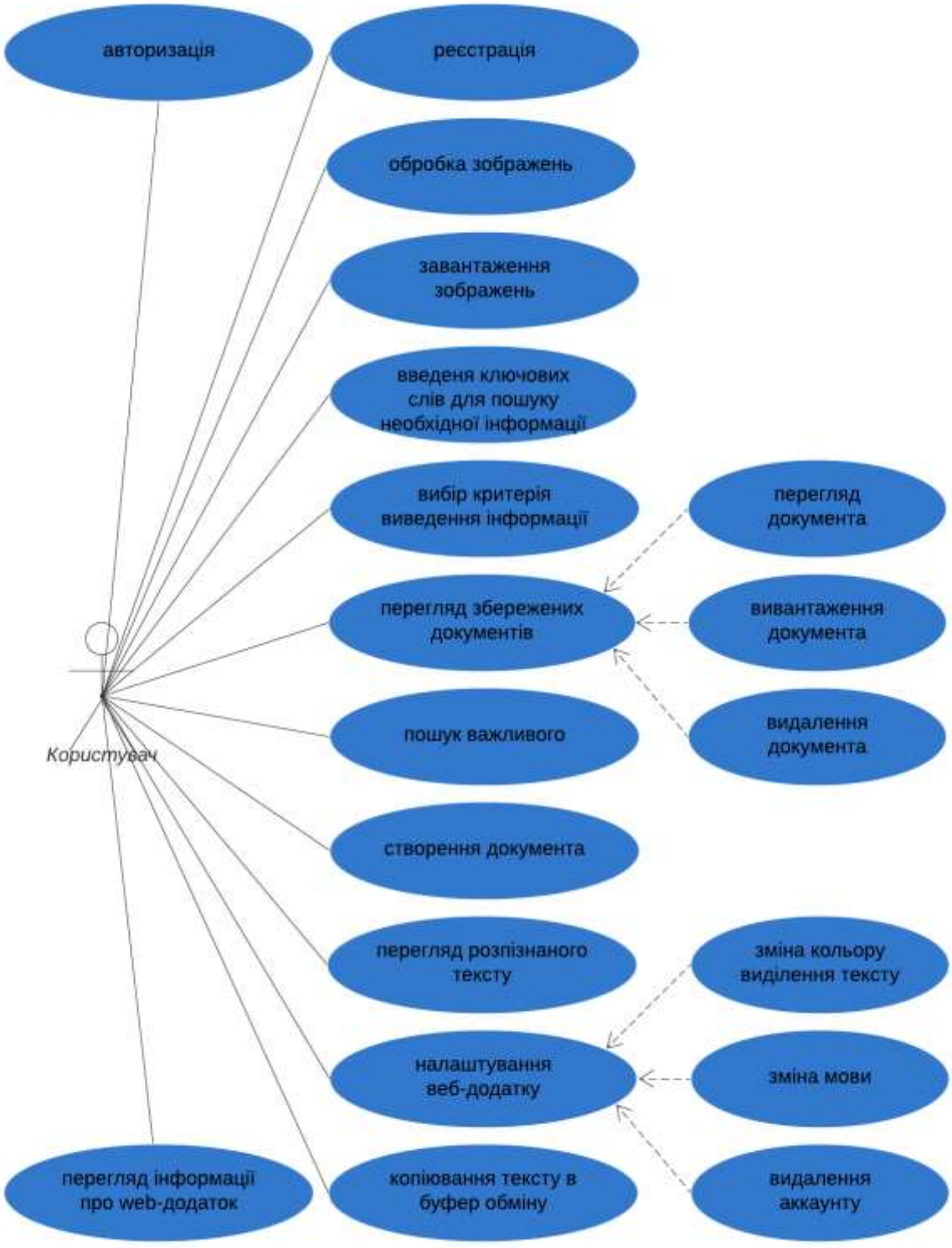

Рисунок 3.1 – Діаграма варіантів використання користувача

3.3 Вимоги до ПЗ

3.3.1 Функціональні вимоги

До функціональних вимог відносяться такі:

3.3.1.1 Користувач веб-додатку повинен мати змогу користуватися наступним функціоналом:

3.3.1.1.1 Видача повідомлень різного типу:

- Неможливо розпізнати текст;
- Неможливо прочитати файл;
- Немає інтернет зв’язку;
- Важливого не знайдено.

3.3.1.1.2 Завантаження зображень.

3.3.1.1.3 Обробка зображень.

3.3.1.1.4 Перегляд розпізнаного тексту.

3.3.1.1.5 Вибір критерія пошуку інформації:

- По реченню;
- По абзацу;
- По розділу.

3.3.1.1.6 Пошук важливого.

3.3.1.1.7 Створення документа:

- DOCX;
- PDF;
- TXT;
- HTML.

3.3.1.1.8 Перегляд збережених документів:

3.3.1.1.8.1 Перегляд документа;

3.3.1.1.8.2 Вивантаження документа;

3.3.1.1.8.3 Видалення документа.

3.3.1.1.9 Налаштування веб-додатку:

3.3.1.1.9.1 Зміна кольору виділення тексту;

3.3.1.1.9.2 Зміна мови:

– Російська;

– Українська;

– Англійська.

3.3.1.1.8.1 Видалення аккаунту.

3.3.1.1.10 Копіювання тексту в буфер обміну;

3.3.1.1.11 Перегляд інформації про web-додаток;

3.3.1.1.12 Реєстрація:

– Пошта;

– Пароль;

– Повторення пароля.

3.3.1.1.13 Авторизація:

– Пошта;

– Пароль.

3.3.2 Нефункціональні вимоги

Нефункціональні вимоги до розроблювального ПО такі:

3.3.2.1 мінімальні системні вимоги веб-дадатку на стороні клієнта наведені в таблиці 3.1;

Таблиця 3.1 – Конфігурація ПК для роботи програми на стороні клієнта

Операційна система	Microsoft Windows 8,10
Браузер	Google Chrome 56
Процессор	Intel core i3 ,AMD64 або краще
Відеокарта	DirectX 9 і WDDM версії 1.0 і більше
Об'єм оперативної пам'яті	1 ГБ

3.3.2.2 мінімальні системні вимоги для роботи мобільного додатку на стороні клієнта наведені в таблиці 3.2.

Таблиця 3.2 – Конфігурація смартфона для роботи програми на стороні клієнта

Операційна система	Android 5
Процессор	Qualcomm Snapdragon 430 або краще
Об'єм оперативної пам'яті	1 ГБ

3.3.2.3мінімальні системні вимоги для роботи програми та бази даних на стороні сервера наведені в таблиці 3.3;

Таблиця 3.3 – Конфігурація ПК для роботи програми на стороні сервера

Операційна система	Ubuntu 16.04 Xenial Xerus
Інтернет канал	Не нижче 100 мегабит
Процессор	Intel Xeon E5-2676 v3 або краще
Об'єм оперативної пам'яті	1 ГБ

3.3.2.4 Підключення до інтернету;

3.3.2.5 форма побудови меню web-додатку на екрані:

3.3.2.5.1 На формі реєстрації повинні відображатися наступні поля та кнопка:

- Поле для введення адреси електроної пошти
- Поле введення для паролю;
- Поле повторного введення паролю;
- Кнопка реєстрації.

3.3.2.5.2 На формі авторизації повинні відображатися наступні поля, посилання та кнопка:

- Поле для введення адреси електроної пошти;

- Поле введення для паролю;
- Посилання для відновлення паролю;
- Кнопка входу;

3.3.2.5.3 На вкладці головної сторінки повинні відображатися наступні елементи:

- Елемент для відображення документа;
- Кнопка для загрузки зображень ;
- Кнопка для перегляду загружених зображень;
- Кнопка для перегляду розпізнаного тексту;
- Поле для введення ключових слів;
- Перемикач для вібору критерія пошуку;
- Кнопка пошуку;
- Кнопка згортання функціоналу.

3.3.2.5.4 На вкладці відображення файлів повинні відображатися наступні елементи:

- Список загружених файлів;
- Кнопка загрузки файлу;
- Кнопка видалення файлу.

3.3.2.5.5 На вкладці налаштувань повинні відображатися наступні елементи:

- Елемент для вібору іншого кольру;
- Кнопка збереження вибораного кольору;
- Перемикач для вибору мови;
- Кнопка збереження мови;
- Кнопка відалееня аккаутну.

3.3.2.5.6 На вкладці інформації про web-додаток повинна відображатися інформація щодо програмного забезпечення та інструкція щодо його використання.

3.3.2.5.7 Вверху повинні відображатися наступні вкладки навігації:

– головна;
– файли;
– налаштування;
– информація про web-додаток.

3.4 Специфікація варіантів використання

Діаграма варіантів використання є вихідним концептуальним поданням або концептуальною моделлю системи в процесі її проектування і розробки.

Розробка діаграми варіантів використання переслідує мети:

- визначити загальні межі і контекст модельованої предметної області на початкових етапах проектування системи.
- сформулювати загальні вимоги до функціонального поведінки проектованої системи.
- розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.
- підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Специфікація варіантів використання з описом прецедентів і сценаріїв відображена в таблицях 3.4 – 3.19.

Таблиця 3.4 – Прецедент " Обробка зображень"

Ім'я	Decoding_image
Назва	Обробка зображень.
Опис	Виконується обробка зображень.
Передумова	Користувач має завантажені зображення.
Постумова	Виконана обробка зображень.
Основний потік	1 Користувач натискає на кнопку обробити зображення.

Продовження таблиці 3.4

Альтернативний потік	1 Користувач натискає на кнопку обробити зображення. 2 Неможливо розпізнати текст;
Альтернативний потік 2	1 Користувач натискає на кнопку обробити зображення. 2 Неможливо прочитати файл;

Таблиця 3.5 – Прецедент " Перегляд розпізнаного тексту"

Ім'я	show_text
Назва	Перегляд тексту.
Опис	Користувач переглядає текст.
Передумова	1 Користувач загрузив зображення. 2 Користувач отримав розпізнаний текст.
Постумова	Відображено результат розпізнавання.
Основний потік	1 Користувач натискає на кнопку перегляд.

Таблиця 3.6 – Прецедент " Пошук важливого"

Ім'я	search_important
Назва	Пошук важливого.
Опис	Користувач здійснює пошук важливого.
Передумова	1 Користувач загрузив зображення. 2 Користувач отримав розпізнаний текст.
Постумова	Відображено важливий текст.
Основний потік	1 Користувач вводить ключові слова для пошуку важливого. 2 Користувач вибирає критерій відображення важливого тексту. 2 Користувач натискає на кнопку пошук.

Продовження таблиці 3.6

Альтернативний потік	1 Користувач вводить ключові слова для пошуку важливого. 2 Користувач вибирає критерій відображення важливого тексту. 3 Користувач натискає на кнопку пошук. 4 Користувачеві виводиться повідомлення про те що важливого тексту не знайдено

Таблиця 3.7 – Прецедент " Створення документа"

Ім'я	create_document
Назва	Створення документа.
Опис	Користувач створює документ.
Передумова	1 Користувач загрузив зображення. 2 Користувач отримав розпізнаний текст.
Постумова	Документ збережено.
Основний потік	1 Користувач вибирає критерій створення документа. 2 Користувач натискає на кнопку створити.

Таблиця 3.8 – Прецедент " Перегляд документа"

Ім'я	show_document
Назва	Перегляд документа.
Опис	Користувач переглядає документ.
Передумова	1 Користувач має збережений документ. 2 Користувач перейшов на вкладку доменти.
Постумова	Відображено документ.
Основний потік	1 Користувач вибирає документ. 2 Користувач натискає на кнопку перегляд.

Таблиця 3.9 – Прецедент " Вивантаження документа"

Ім'я	upload_ document
Назва	Вивантаження документа.
Опис	Користувач вивантажує документ на пристрій.
Передумова	1 Користувач має збережений документ. 2 Користувач перейшов на вкладку документи.
Потсумова	1 Запис вивантажено на пристрій.
Основний потік	1 Користувач вибирає документ. 2 Користувач натискає на кнопку вивантажити.
Альтернативний потік	1 Користувач вибирає документ. 2 Користувач натискає на кнопку вивантажити. 3 Користувачеві виводиться повідомлення про те що сервер в даний момент недоступний.

Таблиця 3.10 – - Прецедент " Видалення документа"

Ім'я	delete_document
Назва	Видалення документа.
Опис	Користувач видаляє документ.
Передумова	1 Користувач має збережений документ. 2 Користувач перейшов на вкладку документи.
Постумова	Документ видалено.
Основний потік	1 Користувач вибирає документ. 2 Користувач натискає на кнопку “видалити”.

Таблиця 3.11 – Прецедент " Видалення аккаунту"

Ім'я	delete_account
Назва	Видалення аккаунту.
Опис	Користувач видаляє аккаунт.

Продовження таблиці 3.11

Передумова	1 Користувач увійшов в систему. 2 Користувач перейшов на вкладку налаштувань.
Постумова	Аккаунт видалено.
Основний потік	1 Користувач натискає на кнопку “видалити”. 2 Користувач натискає “підтвердити”.

Таблиця 3.12 – Прецедент " Проглянути інформацію про сервіс"

Ім'я	show_ information
Назва	Проглянути інформацію про web-додаток
Опис	Перегляд загальної інформації про веб-додаток
Передумова	Клієнт відкрив головну сторінку, натиснув на посилання “Про додаток”
Постумова	Відкрита сторінка з описом
Основний потік	1. Відображення інформації про сервіс

Таблиця 3.13 – Прецедент " Зареєструватися"

Ім'я	signup
Назва	Зареєструватися
Опис	Регістрація користувача в системі
Передумова	Користувач відкрив головну сторінку, перейшов до сторінки реєстрації
Постумова	Створений новий обліковий запис
Основний потік	1. Ввести адресу електронної пошти 2. Ввести пароль 3. Повторити пароль 4. Натиснути кнопку "Готово"

Продовження таблиці 3.13

Альтернативний потік 1	1. Введена електронна пошта вже є в системі 2. Виведення повідомлення "користувач з даною електронною поштою вже зареєстрований в системі" 3. Перехід до пункту 1
Альтернативний потік 2	1. Пароль некоректний 2. Виведення повідомлення "Пароль повинен містити видимі символи, а також мати довжину не менше 10 символів" 3. Перехід до пункту 2
Альтернативний потік 3	1. Паролі не співпадають 2. Виведення повідомлення "паролі не співпадають" 3. Перехід до пункту 2
Альтернативний потік 4	1. Електронна пошта введена не правильно 2. Виведення повідомлення "електронна пошта введена не вірно" 3. Перехід до пункту 1

Таблиця 3.14 – Прецедент " Зміна кольору виділення тексту"

Ім'я	edit_color
Назва	Зміна кольору виділення тексту.
Опис	Користувач змінюэ колір виділення тексту.
Передумова	Користувач перейшов на вкладку налаштування
Постумова	1 Користувачеві відображено повідомлення про успішність збереження кольору. 2 Колір збережено
Основний потік	1 Користувач натискає на кнопку змінити колір. 2 Користувач обирає колір 3 Користувач натискає на кнопку зберегти.

Продовження таблиці 3.14

Альтернативний потік 1	1 Користувач натискає на кнопку змінити колір. 2 Користувач обирає колір 3 Користувач натискає на кнопку відміна.

Таблиця 3.15 – Прецедент " Змінити мову інтерфейсу"

Ім'я	change_ language
Назва	Змінити мову інтерфейсу
Опис	Зміна мови інтерфейсу на будь-який з можливих
Передумова	Клієнт відкрив головну сторінку, натиснув на одну з іконок вибору мови
Постумова	Мова інтерфейсу змінена на вибрану
Основний потік подій	1. Зміна мови інтерфейсу відповідно до вибору користувача: – Українська; – Англійська; – Російська.

Таблиця 3.16 – Прецедент " Змінити пароль"

Ім'я	change_password
Назва	Змінити пароль
Опис	Зміна пароля для входу в систему
Передумова	Гравець перейшов зі сторінки профілю на посилання “Змінити пароль”
Постумова	Пароль змінено

Продовження таблиці 3.16

Основний потік подій	1. Ввести старий пароль 2. Ввести новий пароль 3. Підтвердити новий пароль 4. Натиснути кнопку "Змінити пароль"
Альтернативний потік 1	1. Старий пароль введений невірно 2. Повернення до пункту 1
Альтернативний потік 2	1. Новий пароль ненадійний 2. Виведення повідомлення "пароль повинен містити цифри і букви, а також мати довжину не менше 10 символів" 3. Повернення до пункту 2
Альтернативний потік 3	1. Нові паролі не співпадають 2. Виведення повідомлення "новий пароль не підтверджений" 3. Повернення до пункту 2

Таблиця 3.17 – Прецедент " Авторизація"

Ім'я	signin
Назва	Авторизація
Опис	Авторизація користувача в системі
Передумова	Гість перейшов з головної сторінки на форму авторизації
Постумова	Гостю надані права користувача
Основний потік подій	1. Ввести логін 2. Ввести пароль 3. Натиснути на кнопку "Увійти"

Продовження таблиці 3.17

Альтернативний потік подій 1	1. Користувача з введеним логіном і паролем немає в системі 2. Виведення повідомлення "логін або пароль невірний" 3. Повернення до пункту 1

Таблиця 3.18 – Прецедент " Завантаження зображень"

Ім'я	upload_image.
Назва	Завантаження зображень.
Опис	Початок завантаження зображень користувача.
Передумова	Додаток запущено
Постумова	Додаток завантажує зображення користувача, відображається кнопка відміни завантаження.
Основний потік	1 Користувач переходить на головну вкладку. 2 Користувач натискає на кнопку загрузити зображення.
Альтернативний потік	1 Користувач переходить на головну вкладку. 2 Користувач натискає на кнопку загрузити зображення. 3 Користувачеві виводиться попередження про те що відсутній інтернет зв’язок.
Альтернативний потік 2	1 Користувач переходить на головну вкладку. 2 Користувач натискає на кнопку загрузити зображення. 3 Користувачеві виводиться сповіщення про те що файл не є зображенням.

Таблиця 3.19 – Прецедент " Копіювання тексту в буфер обміну "

Ім'я	copy_text

Продовження таблиці 3.19

Назва	Копіювання тексту в буфер обміну.
Опис	Користувач копіює текст в буфер обміну.
Передумова	1 Користувач має текст.
Постусловіем	Текст скопійовано в буфер обміну.
Основний потік	1 Користувач виділяє необхідний текст. 2 Користувач натискає на кнопку копіювати.

2.1 Початкові дані

Для вирішення завдань клієнтського додатка було обрано такі технології:

- бібліотека React для роботи з відображенням призначеного для користувача інтерфейсу;
- бібліотека React Native для роботи з відображенням призначеного для користувача інтерфейсу на мобільному пристрої;
- бібліотека Redux для створення архітектури обробки внутрішніх дій в кліенстком додатку;
- бібліотека Fetch як бібліотека для створення запитів до серверної частини веб-додатку.

Основними завданнями серверного додатку при реалізації підходу односторінкового веб-додатку є:

- взаємодія з базою даних;
- процедура обробки запитів від клієнтського додатку і виконання бізнес-логіки додатка;
- формування веб-сторінки для першого користувальницького запиту.

Для вирішення завдань серверного додатку було обрано такі технології:

- Node.js - платформа, що дозволяє виконувати JavaScript на сервері і надає можливості для взаємодії з системними ресурсами сервера;

– Express - фреймворк, що дозволяє реалізувати веб-сервер, маршрутизацію запитів користувачів, декомпозировать логіку програми на окремі підсистеми.

Взаємодію бази даних і веб-сервера слід організувати, дотримуючись принципу, що бізнес логіка знаходиться в коді серверного додатку, а не в базі даних. В цьому випадку база даних зберігає дані, і надає прямий доступ до даних, тоді як вся бізнес-логіка реалізована в підсистемах серверного додатка. База даних дозволяє виконувати транзакції для проведення атомарних операцій над даними.

Ізоляція бізнес-логіки в підсистемах серверного додатку дозволяє забезпечити прозорий інтерфейс взаємодії сервера з базою даних.

Для забезпечення такої ізоляції, можна визначити наступну ієрархічну структуру завдань:

– розробка модуля клієнтської частини веб-додатку;
– розробка модуля серверної частини веб-додатку;
– модуль бази даних;
– протокол обміну даних між клієнтської і серверної сторонами.
– проектування інтерфейсу взаємодії між модулями клієнтської і серверної частини.

проектування інтерфейсу взаємодії між модулем серверної частини і базою даних.

3.5 Архітектура додатку

Веб-додаток складається з клієнтської і серверної частини, тим самим реалізуючи мережеву архітектуру «клієнт-сервер». Клієнтська частина реалізує користувальницький інтерфейс, формує запити до сервера і обробляє відповіді від нього. Серверна частина приймає запити від користувача,

виконує необхідні обчислення і потім формує веб-сторінку і відправляє її клієнту через мережу з використанням протоколу HTTP або HTTPS.

Довгий час при розробці веб-ресурсів найбільш популярним підходом було створення «тонкого клієнта» - веб-додатки, основна логіка роботи якого винесена в серверну частину. Інтенсивний розвиток інтернет-ресурсів і високі вимоги до інтерактивності сервісів привели до того, що зараз все більше веб-додатків створюється з використанням підходу «товстий клієнт» (або rich-клієнт), в якому клієнтська частина забезпечує розширену функціональність з надання і обробки інформації, а серверна займається переважно зберіганням даних.

Основою для реалізації такого підходу в веб-додатках став AJAX (від англ. Asynchronous JavaScript and XML). При використанні AJAX клієнт і сервер у фоновому режимі обмінюються даними без перезавантаження сторінки. Використання AJAX призвело до появи «односторінкових додатків» (англ. SPA або Single page applications), де з сервера веб-сторінка завантажується всього один раз на початку сесії, а вся подальша робота з обміну даними здійснюється у фоновому режимі. Такий підхід дозволяє бути веб-додаткам більш інтерактивним і продуктивним.

Також широко використовується технологія WebSocket, яка не вимагає постійних запитів від клієнта до сервера, а створює двонаправлене з'єднання, при якому сервер може відправляти дані, без запиту від останнього. Таким чином надається можливість динамічно управляти контентом в режимі реального часу.

3.6 Вибір і обґрунтування вибору стека технологій

Керуючись вище перерахованими факторами, а також прислуховуючись до думок провідних розробників ринку веб-технологій було визначено наступний стек технологій:

- React.js в зв'язці з Redux на клієнтській стороні, для вирішення завдань по роботі зі станом додатку, та створення MVC архітектури;
- MVC (model, view, controller / модель-уявлення-контролер) архітектура передбачає, що ваша модель - це єдине джерело істини і все стан зберігається там.
- уявлення - це похідні моделі і повинні бути синхронізовані, коли модель змінюється - змінюється і уявлення;
- Node.js для створення веб-сервера і фреймворк Express.js для створення надійного API швидкого і легкого;
- база даних – MondoDB;
- для модульної збірки веб-додатку відмінно підходить утиліта webpack;
- пакетний менеджер npm, для управління модулями і залежностями.

3.6.1 Засоби розробки

Розробка веб-додатку є неймовірно трудомістким процесом, але не дивлячись на це, для створення веб-додатку досить блокнота і браузера, щоб запустити додаток. Це необхідний мінімум, в реальності ж ніхто не обмежується блокнотом. З безлічі текстових редактором і IDE (Інтегрованих середовищ розробки) кожен розробник намагається знайти для себе найбільш підходящий і вдосконалити їх поруч плагінів. Множина розширень для браузерів, які допомагають в розробці, росте з кожним днем. По мимо всього іншого, процес розробки не тільки прискорюється, але і розширюється за рахунок нових можливостей. Так, на сьогоднішній день існує величезна кількість «таск-ранерів» (менеджерів завдань), складальників проектів, пакетних менеджерів і всіх можливих утиліт.

3.6.2 Веб-браузер. Інструменти і розширення для розробників

На сьогоднішній день виділяють кілька основних лідируючих браузерів на ринку, на можливості яких варто орієнтуватися веб-розробникам: IE (Internet Explorer), Edge, Firefox, Chrome, Safari, Opera, iOS Safari, Opera Mini, Android Browser і Chrome for Android.

Багато з них оснащені інструментами для розробників, але самий великий по функціоналу інструмент належить Chrome - DevTools. DevTools оснащений живим редагуванням CSS на льоту, консоллю, відладчиком, і це невелика частина з усіх можливостей інструменту.

Chrome так надає можливість підключення безлічі корисних для розробки розширень, наприклад:

- ColorPicker - для визначення кольору на будь-якій ділянці сторінки;
- PageRuler -для визначення розміру блоків;
- WhatFont - для визначення шрифтів;
- XDebug - для налагодження серверних додатків.

По ряду очевидних причин, Chrome - мій основний веб-браузер для розробки і налагодження веб-додатків.

3.7 Розробка веб-серверу

Для створення HTTP сервера ми використовуємо Node.js. Node.js представляє собою серверну реалізацію мови програмування JavaScript, засновану на движку V8. Що являє собою полностековое використання JavaScript, що дуже зручно якщо ти «full stack» розробник (Розробник, що працює над серверної і клієнтською частиною програми).

У Node.js існує вбудована підтримка управління пакетами, для якої застосовується інструмент NPM, за замовчуванням присутній в будь-яку установку Node.js. Ідея модулів NPM схожа з Ruby Gems: це набір загальнодоступних компонентів для багаторазового використання, які легко

встановити через онлайн репозиторій; для них підтримується управління версіями і залежностями. Одним з таких модулів є фреймворк Express.js, в зв'язці з яким ми вирішили створювати сервер.

Node.js с Express.js [34] можна застосовувати для створення класичних веб додатків на серверній стороні, однак, нехай це і можливо, така парадигма запит / відгук, де Node.js буде переносити відображення HTML, нетипова для даної технології. Існують аргументи як на користь такого підходу, так і проти нього. Були враховані основні моменти:

- якщо ваш додаток не виконує інтенсивних обчислень, що навантажують процесор, то його можна написати повністю на JavaScript, включаючи навіть базу даних, якщо ви використовуєте об'єктну базу даних (наприклад, як в нашому випадку, MongoDB) і JSON. Це значно спрощує не тільки розробку, але і підбір фахівців;
- пошукові роботи отримують у відповідь повністю відображений HTML, що набагато зручніше для пошукової оптимізації, ніж, наприклад, робота з односторінкового додатками або веб-сокетних додатком, що працює на базі Node.js;
- використовувати Node.js з реляційною базою даних не зручно, в цьому випадку варто віддати перевагу іншому середовищі, наприклад, Rails або Django [35].

3.8 Розробка бази данних

Якої би складності не був веб-додаток, він не може обійтися без даних і як наслідок бази даних. Перш ніж вибрати базу даних потрібно визначити, що потрібно зберігати в базі даних і що буде вимагатися сайту. Завдання - отримати загальну і повну картину структури бази даних.

В рамках цього проекту не виявлено необхідність в реляційної базі даних . Представляючи структуру як масиви об'єктів користувачів і масиви об'єктів питань з відповідями.

MongoDB є ефективною і масштабованої базою даних для цього завдання. MongoDB - це дуже вдалий симбіоз між звичною реляційної базою даних і key-value сховищем.

На відміну від традиційних СУБД MongoDB не може робити операції об'єднання, JOIN-таблиці або, в термінології MongoDB, колекції. Тому MongoDB підходить в разі слабосвязанних або слабо структурованих даних. Що говорить про необов'язковий строгому структуруванні даних.

Відмінності MongoDB від звичних РСУБД наведено у таблиці 3.20.

Таблиця 3.20 – Порівняння MongoDB з РСУБД

РСУБД	MongoDB
База данних, Схема данних	База данних
Таблиця	Колекція
Строка данних, Таблиці	Документ
Колонка строки	Поле документу
Курсор	Курсор в збережених процедурах

З урахуванням того, що нас нічого не прив'язує до конкретних схем, ми можемо обійтися всього однією колекцією, що включає в себе безліч документів різної структури. Можна виділити кілька основних стратегій розробки баз даних для MongoDB [35]:

– «Без вбудовування». Виділення окремої колекції для кожного типу даних. Наприклад, окрема колекція це email і окрема для користувачів;
– «Все вбудовано». Вбудовуються частини всіх даних в один документ;

– «Часткове вбудовування». Найоптимальніший варіант. Можливість зберігати дані в окремо колекції, але при необхідності вбудовувати в колекцію, логічно пов'язану з даною.

Приклад структури зберігаються в базі даних у вигляді JSON.

```
{
  "_id": 7,
  "created": "10-10-2020-19-15-17",
  "data": [
    {
       "_id" : <ObjectId>,
          "files_id" : <ObjectId>,
          "n" : <num>,
          "data" : <binary>
    }
  ],
  "__v": 0
}
```

3.9 Розробка класів програмного засобу

Виходячи з аналізу інформаційних потреб користувача доцільно провести декомпозицію ПЗ на окремі класи, кожен з яких дозволяє вирішити ту чи іншу задачу. Основними вимогами, що пред'являються до декомпозиції, є незалежність класів один від одного при збереженні глибокої інтеграції між ними і можливість без особливих зусиль додавати їх в працюючу систему програмного забезпечення.

В результаті проведення декомпозиції розробляється система була розбита на наступні складові частини:

- головний клас app;
- клас loginView;
- клас checklogin;
- клас mainMenu;
- клас homeView;
- клас showText;
- клас addImage;
- клас ocr;
- клас searchText;
- клас colorExcretion;
- клас fileView;
- клас fileView;
- клас files;
- клас cardIndex;
- клас settingsView;
- клас settings;
- клас account;
- клас color;
- клас language;
- клас infoView.

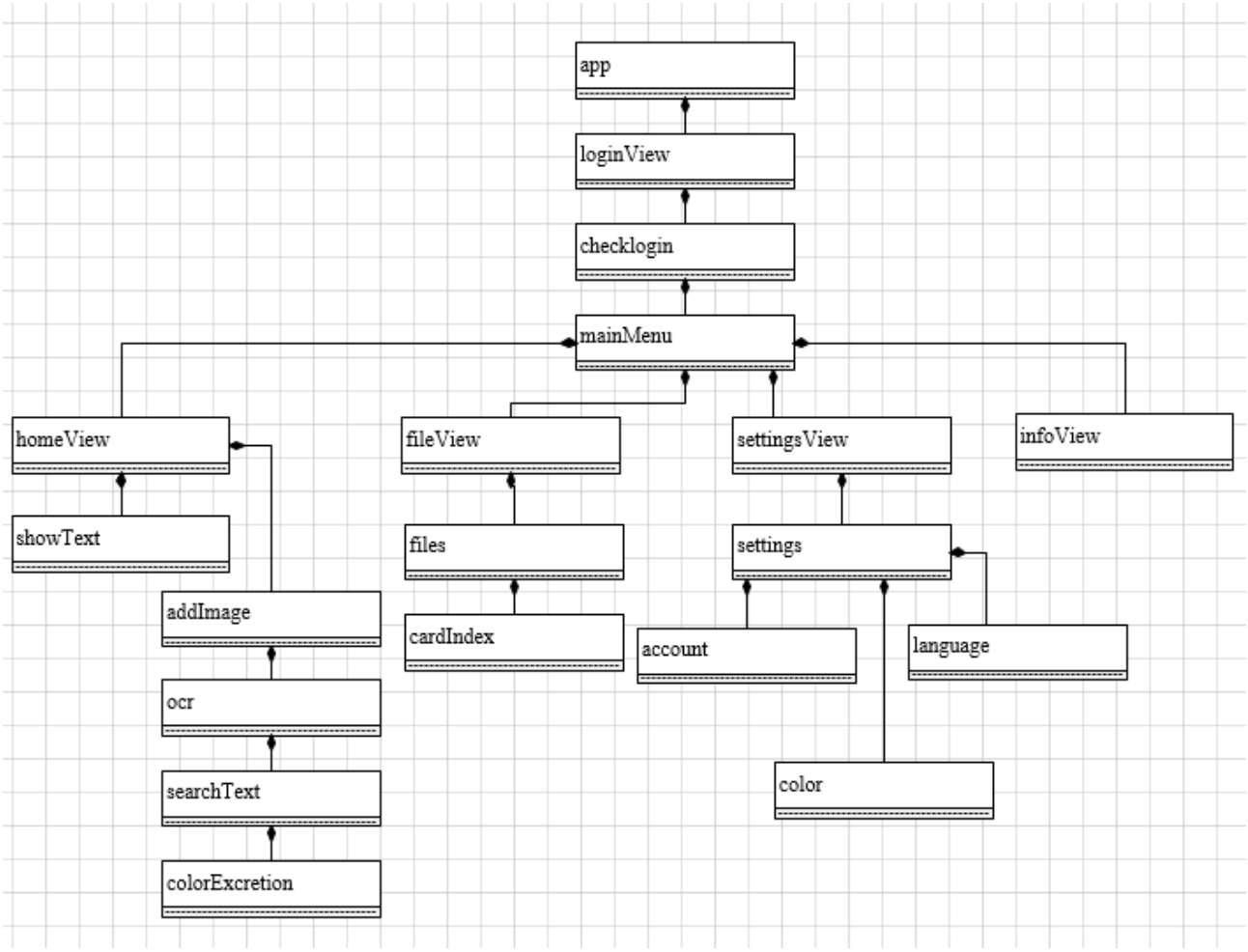

Рисунок 3.2 - Декомпозиція класів

3.10 Детальне проектування класів (опис методів класів) для реалізації підсистем

Спроектуємо властивості і методи класів додатки. Результати проектування описані в таблицях 3.21-3.39.

Таблиця 3.21 – Опис класу app

Назва класу	app

Продовження таблиці 3.21

Властивості класу	reducer saga middleware store
Методи класу	render

Таблиця 3.22 – Опис класу loginView

Назва класу	loginView
Властивості класу	logUser logPassword
Методи класу	render

Таблиця 3.23 – Опис класу checklogin

Назва класу	checklogin
Властивості класу	username password
Методи класу	validate

Таблиця 3.24 – Опис класу mainMenu

Назва класу	mainMenu
Властивості класу	state
Методи класу	handleIndexChange renderScene

Таблиця 3.25 – Опис класу homeView

Назва класу	homeView

Продовження таблиці 3.26

Властивості класу	recording stopRecording calculation togleUserChange getLocationError
Методи класу	render

Таблиця 3.26 – Опис класу showText

Назва класу	showText
Властивості класу	text importantText
Методи класу	ocr

Таблиця 3.27 – Опис класу addImage

Назва класу	addImage
Властивості класу	image
Методи класу	ocr

Таблиця 3.28 – Опис класу ocr

Назва класу	ocr
Властивості класу	image text importantText
Методи класу	searchText

Таблиця 3.29 – Опис класу searchText

Назва класу	searchText

Продовження таблиці 3.29

Властивості класу	text importantText
Методи класу	colorExcretion

Таблиця 3.30 – Опис класу colorExcretion

Назва класу	colorExcretion
Властивості класу	bestColor colorUser
Методи класу	searchColor

Таблиця 3.31 – Опис класу fileView

Назва класу	fileView
Властивості класу	filesData readFiler writeToClippboard reading deleteFile uploadFile uploading uploadingProgress
Методи класу	showDataFromFile backAction componentDidMount

Таблиця 3.32 – Опис класу files

Назва класу	files
Властивості класу	filesData
Методи класу	map

Таблиця 3.33 – Опис класу cardIndex

Назва класу	cardIndex
Властивості класу	uploaded number data
Методи класу	deleteFile showData copy upload

Таблиця 3.34 – Опис класу settingsView

Назва класу	settingsView
Властивості класу	Settings
Методи класу	settingsChanged backAction saveSettings

Таблиця 3.35 – Опис класу settings

Назва класу	settings
Властивості класу	setSettings settings bestColor colorUser map
Методи класу	changeColor deleteAccount changeLanguage render

Таблиця 3.36 – Опис класу account

Назва класу	account
Властивості класу	user password
Методи класу	onChange

Таблиця 3.37 – Опис класу color

Назва класу	color
Властивості класу	nowColor bestColor
Методи класу	onChange

Таблиця 3.38 – Опис класу language

Назва класу	language
Властивості класу	languageSettings language languageData
Методи класу	onChange

Таблиця 3.39 – Опис класу infoView

Назва класу	infoView
Властивості класу	info
Методи класу	render

3.11 Розробка інтерфейсу

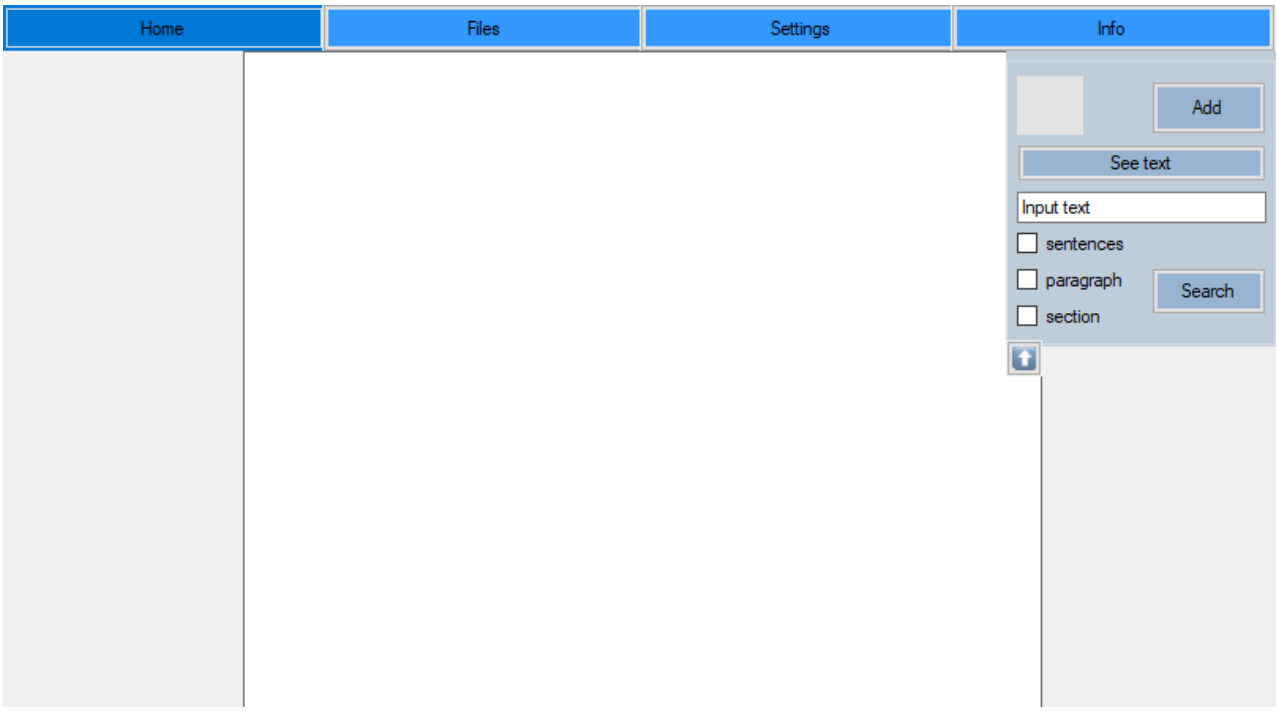

Рисунок 3.3 – Головна сторінка

3.12 Модель проведення експериментів

Всі реалізовані алгоритми були розгорнуті як незалежні Lambda-функції в кількості 8 штук. Рішення розділяти різні NLP-алгоритми на незалежні функції було прийнято в зв'язку з тим, що платформа AWS Lambda має обмеження на розмір завантажується в сховище архіву з вихідним кодом в 250 Мб, а розмір 8 реалізованих алгоритмів перевищує це обмеження в кілька разів.

Ручне управління завантаженням функцій є рутинним процесом, який необхідно було автоматизувати. Для повної автоматизації розгортання необхідно виконати наступні кроки.

1. Компіляція залежностей під цільову ОС.

2. Архівування вихідного коду і всіх залежностей.

3. Звернення до API AWS Lambda для завантаження заархівірованого додатка.

Перший пункт виявився необхідний, так як деякі реалізації алгоритмів мають компільовані залежності, і це накладає вимога щодо попереднього компіляції під цільову операційну систему. Для вирішення даного завдання була використана система Docker і docker-образ Lambda.

Описані вище кроки були автоматизовані до можливості виклику однією командою за допомогою інструменту Gulp, який дозволяє імперативно описувати завдання, групувати їх і виконувати в потрібному порядку.

3.13 Модель проведення експериментів

Для проведення порівняльного аналізу реалізованих алгоритмів була обрана модель турніру з олімпійської системою, тобто той, хто програв в кожному раунді вибував зі змагання. Дана система допомагає знизити кількість порівнянь: проводиться $O(\log_2 n)$ замість $O(n^2)$ порівнянь для варіанта, де кожен алгоритм порівнюється з усіма іншими.

Кожен раунд являє собою порівняння результатів двох алгоритмів, а користувачеві пропонується можливість вибору того з них, який краще здійснює підсвічування важливого в тексті. Система оцінок з фіксованою шкалою не використовувати з тієї причини, що розробка грамотної інструкції для асессорів є важким завданням, яка виходить за рамки даної роботи. Користувачі Яндекс.Толокі не є професійними асессорами, що призвело б до суб'єктивними оцінками і проблем з їх нормалізацією. Крім того, в результаті консультації з експертами з області розмітки текстів варіант "проти всіх" також не був використаний, так як бінарна оцінка є більш ефективним (витрачається менше часу) при збереженні результату.

Неякісні відповіді властиві, практично, будь-якого з опитуванням.

Для поліпшення якості відповідей використовувалися наступні фільтри:

− констістентние відповіді;

− honeypots;

− час виконання завдань.

Під консистентними відповідями розумілися відповіді, в яких користувач однаково відповів на запропоновані варіанти підсвічування, коли вони були запропоновані більш ніж один раз.

Honeypot - це варіант, при якому заздалегідь відомий правильну відповідь (як заздалегідь програв варіанту використовувався алгоритм випадкової з низьким шансом підсвічування кожного слова).

Одне завдання складалося з п'яти пар скріншотів. Медіанний час виконання одного такого завдання був близько 47 секунд. Останній фільтр відсівав користувачів, які виконували завдання швидше.

3.14 Інструменти для автоматизації проведення порівнянь

Порівняльний аналіз вимагав великої кількості однотипних дій. Для прискорення процесу для всієї розробленої інфраструктури була створена спеціальна обгортка, що дозволяє агрегувати весь процес аналізу в єдину систему з CLI-інтерфейсом. Таким чином, були автоматизовані кроки, представлені нижче.

1. Створення скріншотів з результатами підсвічування алгоритмами.

2. Формування завдання для Яндекс.Толока.

3. Вивантаження і обробка результатів порівняння.

Перший пункт був реалізований з використанням інструменту Selenium21, який запускає браузер і надає API для взаємодії з ним. Таким чином, за допомогою JavaScript був реалізований скрипт, який запускає

заздалегідь зібрані HTML-листи в браузері Google Chrome https://www.seleniumhq.org/

за допомогою Selenium, а потім цей скрипт здійснював підсвічування за допомогою реалізованого розширення. Результат роботи потрапляв на скріншот і зберігався в директорії для подальшого використання.

Завдання в Яндекс.Толоку завантажуються за допомогою файлу .tsv (tab-separated values) зі спеціально розміченими колонками, які містять вхідні дані завдання, правильна відповідь, підказки та деякі інші поля. Для формування таких файлів була написана утиліта, але автоматизувати даний пункт повністю не вдалося. Завдання в Яндекс.Толока використовують медіа файли, завантажені на Яндекс.Диск, який не підтримує API для завантаження файлів на диск, тому в процесі порівняльного аналізу частково необхідна ручна робота.

Як було показано в розділі 2.4, Яндекс.Толока підтримує API для отримання відповідей користувачів. Даний пункт так само був автоматизований за допомогою утиліти на мові JavaScript і бібліотеки amCharts. Утиліта здійснює перевірку консистентності відповідей користувачів, перевірку на honeypots і час виконання, а також формує статистику відповідей у вигляді діаграм. За допомогою даної утиліти стало можливим в ході проведення аналізу порівняти більше 7500 пар скріншотів.

3.15 Результати порівняльного аналізу

У порівняльному аналізі брали участь алгоритми, обрані і реалізовані вище (в дужках написано назву алгоритму, що використовується далі в турнірній сітці).

- TFIDF в комбінації з нормами векторних розкладів (TFIDF # 1).

- TF-IDF на корпусі української мови (TF-IDF # 2).
- TF-IDF на корпусі Вікіпедії (TF-IDF # 3).
- Алгоритм з випадковим виділенням (Random).
- Норми векторних уявлень (Embeddings).
- TextRank для пропозицій (TextRank # 1).
- TextRank для ключових слів (TextRank # 2).
- Латентно-семантичний аналіз (LSA).

Так як аналіз проводився по моделі турніру, то для зручності читання результатів проведені порівняння представлені у вигляді турнірній таблиці:

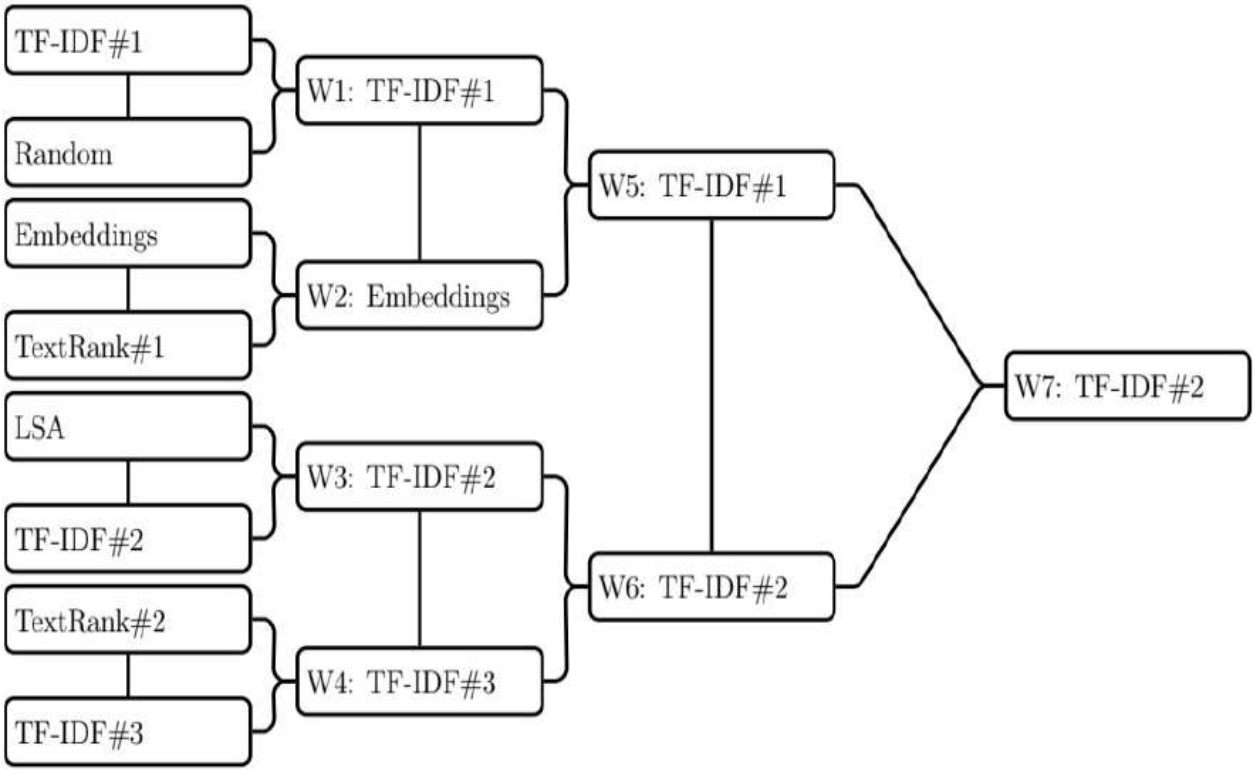

Рисунок 3.4 - Підсумкова турнірна таблиця

Як видно з турнірної сітки, яка оцінююча група обрала реалізацію алгоритму TF-IDF на корпусі української мови з підсвічуванням всіх цифр як найбільш якісну.

Для додаткової перевірки отриманих результатів оцінюючій групі було запропоновано окреме завдання, в якому їм пропонувалося оцінити скріншот з підсвічуванням за наступною бінарною метрицею: чи є підсвічування корисним. Ідея полягала в тому, щоб перевірити чи існує кореляція між результатами такого опитування та результатами турніру.

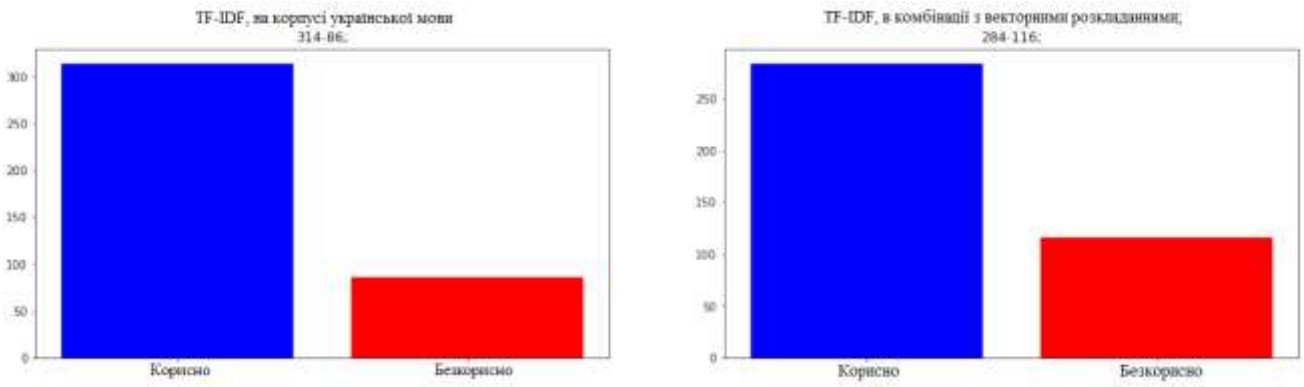

Рисунок 3.5 - Оцінка корисності підсвічування двох алгоритмів

Результати, представлені на рис. 3.5 корелюють з результатами турніру. Це закріплює результат порівняльного аналізу, а також підтверджує працездатність схеми проведення аналізу за моделлю турніру.

ВИСНОВКИ

В ході даної роботи були отримані наступні результати.

- Обрано такі алгоритми NLP для підсвічування важливого в тексті: TF-IDF в трьох варіаціях, TextRank в двох варіаціях, алгоритм з використанням норм векторів, алгоритм з використанням LSA. Обрана метрика якості "оцінююча група".
- Спроектована і реалізована клієнт-серверна архітектура (використані технології Яндекс.Толока, AWS Lambda, Docker і мова JavaScript), а також реалізовані вибрані алгоритми.
- Реалізовано додаткові утиліти для автоматизації процесу проведення експериментів. Для реалізованих алгоритмів спроектований і виконаний на базі технології Яндекс.Толока експеримент з метою виявлення кращого алгоритму. Кращим алгоритмом виявився TF-IDF на корпусі української мови.

З отриманих результатів можна зробити висновок про те, що підсвічування 30% найважливіших слів тексту з використанням TF-IDF в якості міри важливості може бути успішно застосовано для корисного підсвічування слів в тексті. Для поліпшення роботи алгоритму може бути проведена робота над підсвічуванням іменованих сутностей, таких як дати, адреси, імена та інші. У сукупності з підсвічуванням, здійснюваної справжнім алгоритмом, це здатне істотно полегшити пошук корисної інформації користувачеві.

ПЕРЕЛІК ПОСИЛАНЬ

1. Розпізнавання тексту [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Оптическое_распознавание_символов.

2. веб-сторінці [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Вебсторінка.

3. Нейромережеві методи [Електронний ресурс] – Режим доступу до ресурсу: http://jrnl.nau.edu.ua/index.php/ZI/article/view/7539.

4. Обробка текстів на природній мові [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Обробка_природної_мови#:~:text=Обро́бка%20приро́дної%20мо́ви%20(англ.,а%20синтез%20—%20генерацію%20розумного%20тексту.

5. Sun Chi, Qiu Luyao Huang Xipeng. Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence // arXiv preprint arXiv:1903.09588. — 2019.

6. Вебсторінка [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Вебсторінка#:~:text=Вебсторінка%20(англ.,(для%20wap-сторінок).

7. Машинний переклад [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Машинний_переклад.

8. A Click Sequence Model for Web Search / Alexey Borisov, Martijn Wardenaar, Ilya Markov, Maarten de Rijke. – The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval pp. 45-54, 2018.

9. Efficient Estimation of Word Representations in Vector Space / Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean // arXiv preprint arXiv:1301.3781. – 2013.

10. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation // arXiv preprint arXiv:1607.05368.

11. Sequence-to-Sequence Learning for Task-oriented Dialogue with Dialogue State Representation // Proceedings of the 27th International Conference on Computational Linguistics, pages 3781–3792. – 2018.

12. Генератор текста [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Генератор_текста.

13. Розпізнавання тексту [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/yandex/blog/475956/.

14. A Click Sequence Model for Web Search / Alexey Borisov, Martijn Wardenaar, Ilya Markov, Maarten de Rijke. – The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval pp. 45-54, 2018.

15. Efficient Estimation of Word Representations in Vector Space / Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean // arXiv preprint arXiv:1301.3781. – 2013.

16. Gong Yihong, Liu Xin. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. – Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, New Orleans, Louisiana, United States 2001, pp. 19-25, 2001.

17. LCSTS: A Large Scale Chinese Short Text Summarization Dataset // arXiv preprint arXiv:1506.05865. – 2016.

18. Mihalcea Rada, Tarau Paul. Textrank: Bringing order into texts. – Proceedings of EMNLP 2004. pp. 404–411. Association for Computational Linguistics, Barcelona, Spain (July 2004), 2004.

19. Resnik Philip, Lin Jimmy. Evaluation of NLP Systems. The Handbook of Computational Linguistics and Natural Language Processing, pp. 271–295. – 2010.

20. Browser & Platform Market Share [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3counter.com/globalstats.php.

21. Extend the Browser [Електронний ресурс] – Режим доступу до

ресурсу: https://developers.chrome.com/extensions/overview.

22. Mihalcea Rada, Tarau Paul. Textrank: Bringing order into texts. – Proceedings of EMNLP 2004. pp. 404–411. Association for Computational Linguistics, Barcelona, Spain (July 2004), 2004.

23. Chrome Web Store мові [Електронний ресурс] – Режим доступу до ресурсу: https://chrome.google.com/webstore/category/extensions.

24. Serverless Computing: Current Trends and Open Problems /Ioana Baldini, Paul Castro, Kerry Chang et al. // arXiv preprint arXiv:1706.03178. – 2017.

25. Sun Chi, Qiu Luyao Huang Xipeng. Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence // arXiv preprint arXiv:1903.09588. – 2019.

26. TF-IDF. What does tf-idf mean? – Access mode: http://www.tfidf. com/.

27. XMLHttpRequest: кросс-доменные запросы [Електронний ресурс] – Режим доступу до ресурсу:https://learn.javascript.ru/xhr-crossdomain.

28. CORS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTTP/CORS.

29. Tensor2Tensor for Neural Machine Translation // arXiv preprint arXiv:1803.07416. – 2016.

30. VQA: Visual Question Answering / Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu et al. // The IEEE International Conference on Computer Vision (ICCV).– 2015.– December.

31. HTTP-метод OPTIONS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/OPTIONS#:~:text=HTTP-метод%20OPTIONS%20используется%20для,чтобы%20указать%20весь%20сервер%20целиком.

32. Принципы работы и структура Web-приложений [Електронний ресурс] – Режим доступу до ресурсу: https://www.intuit.ru/studies/courses/1139/250/lecture/6422?page=4

33. Node JS Usage & Example [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/dist/latest-v12.x/docs/api/synopsis.html

34. Mongo DB [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/

35. Django Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Django.

ДОДАТОК А

СЛАЙДИ ПРЕЗЕНТАЦІЇ

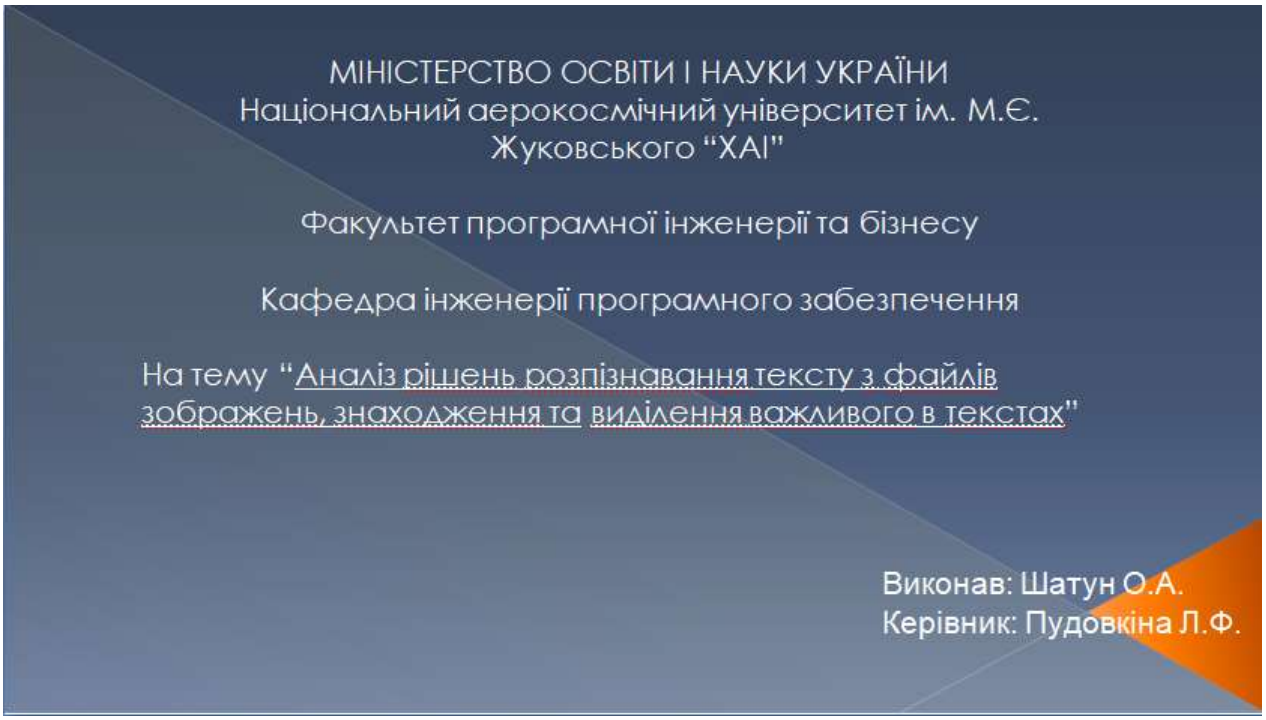

Рисунок А.1 - слайд презентації №1

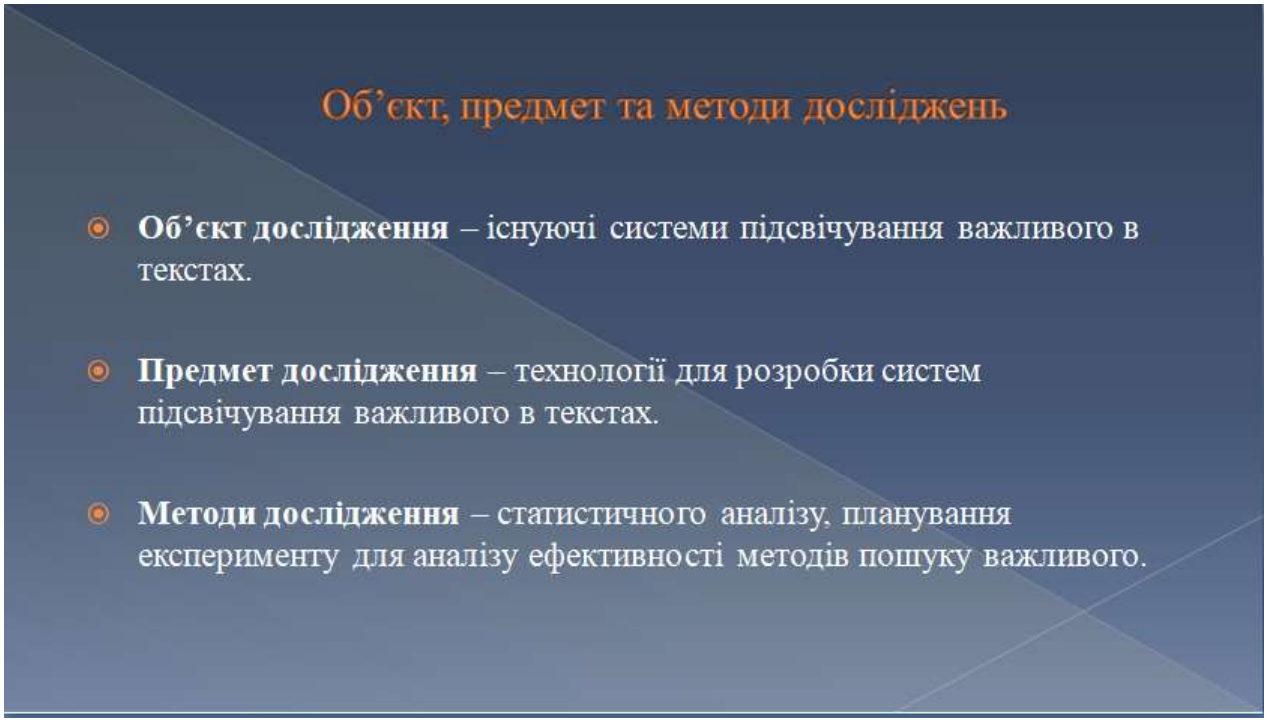

Рисунок А.2 – слайд презентації №2

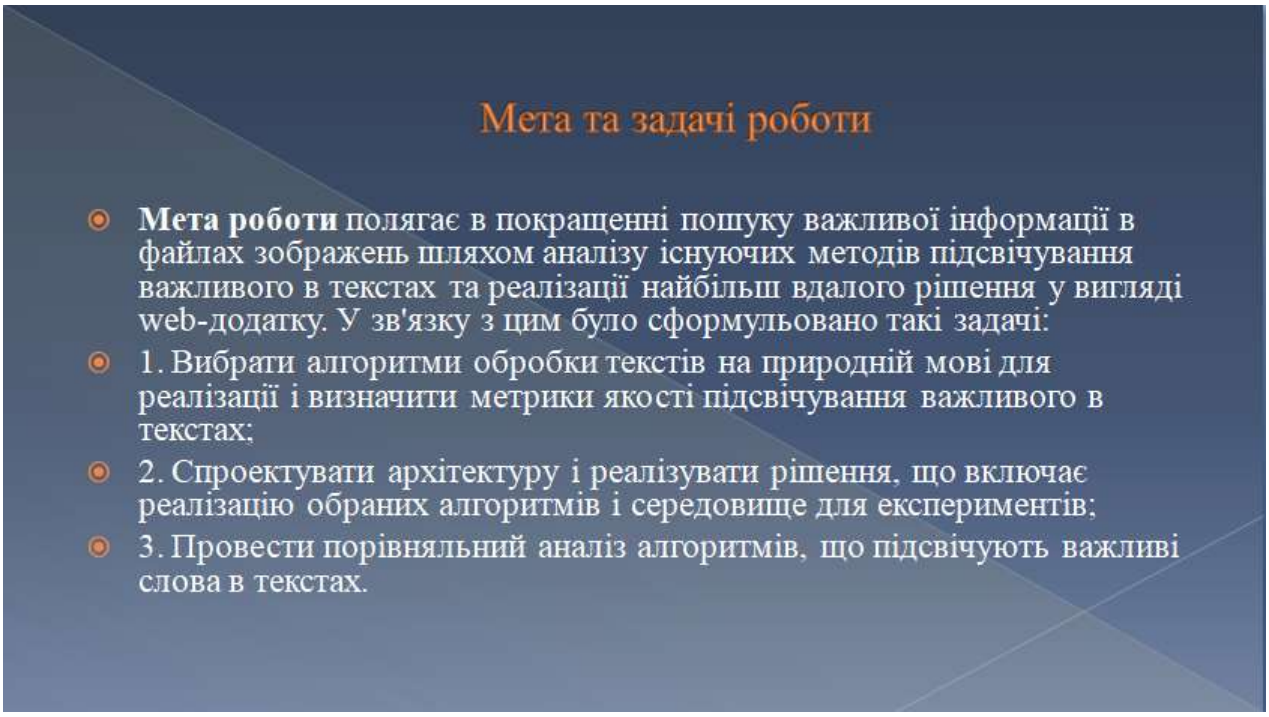

Рисунок А.3 – слайд презентації №3

Рисунок А.4 – слайд презентації №4

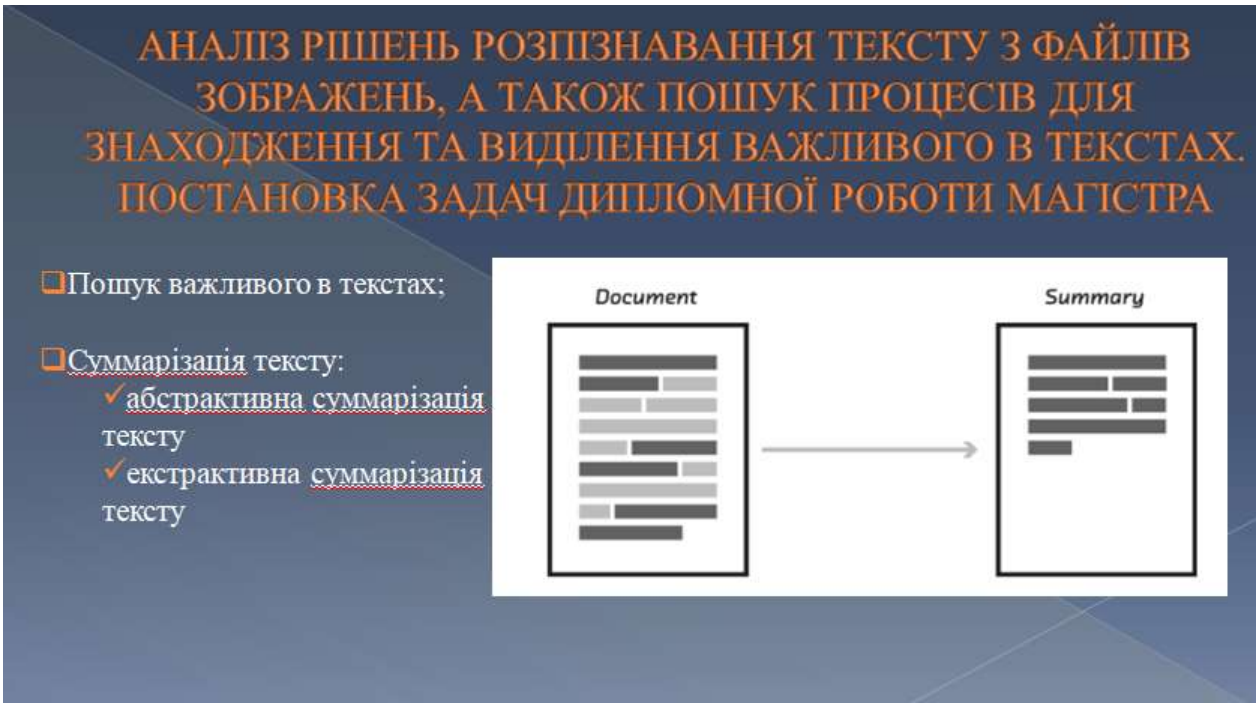

Рисунок А.5 – слайд презентації №5

Рисунок А.6 – слайд презентації №6

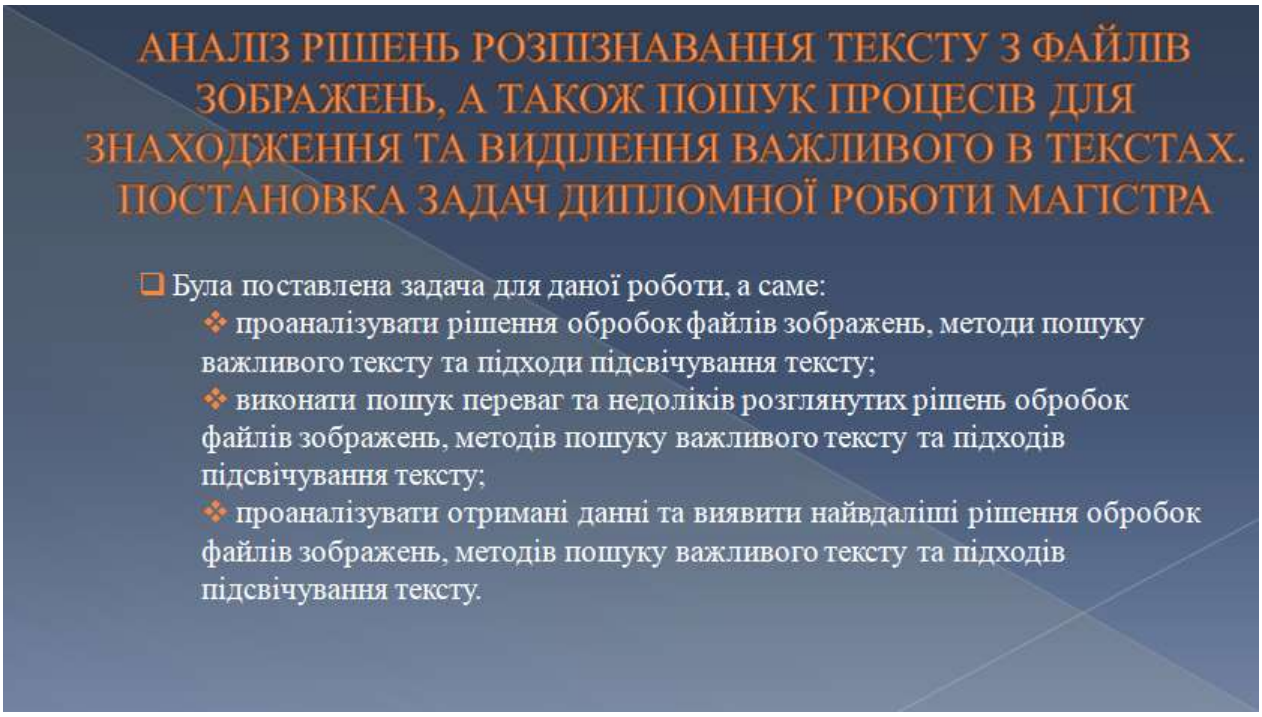

Рисунок А.7 – слайд презентації №7

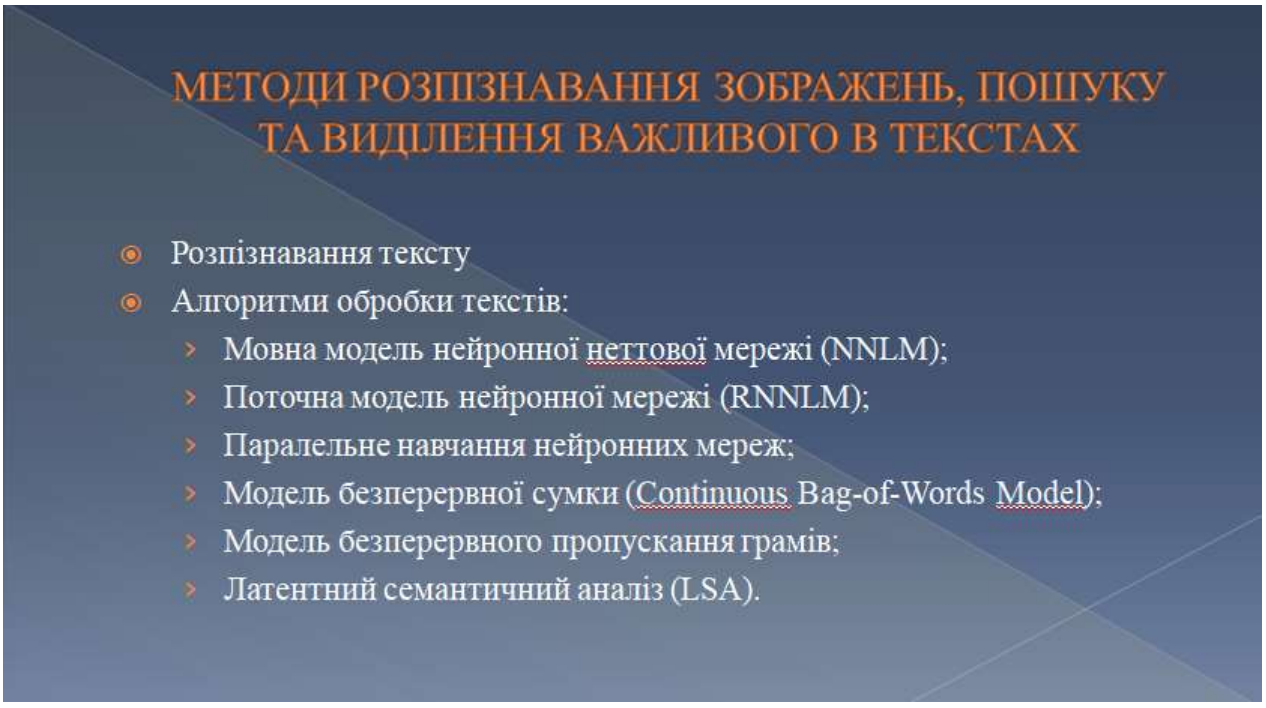

Рисунок А.8 – слайд презентації №8

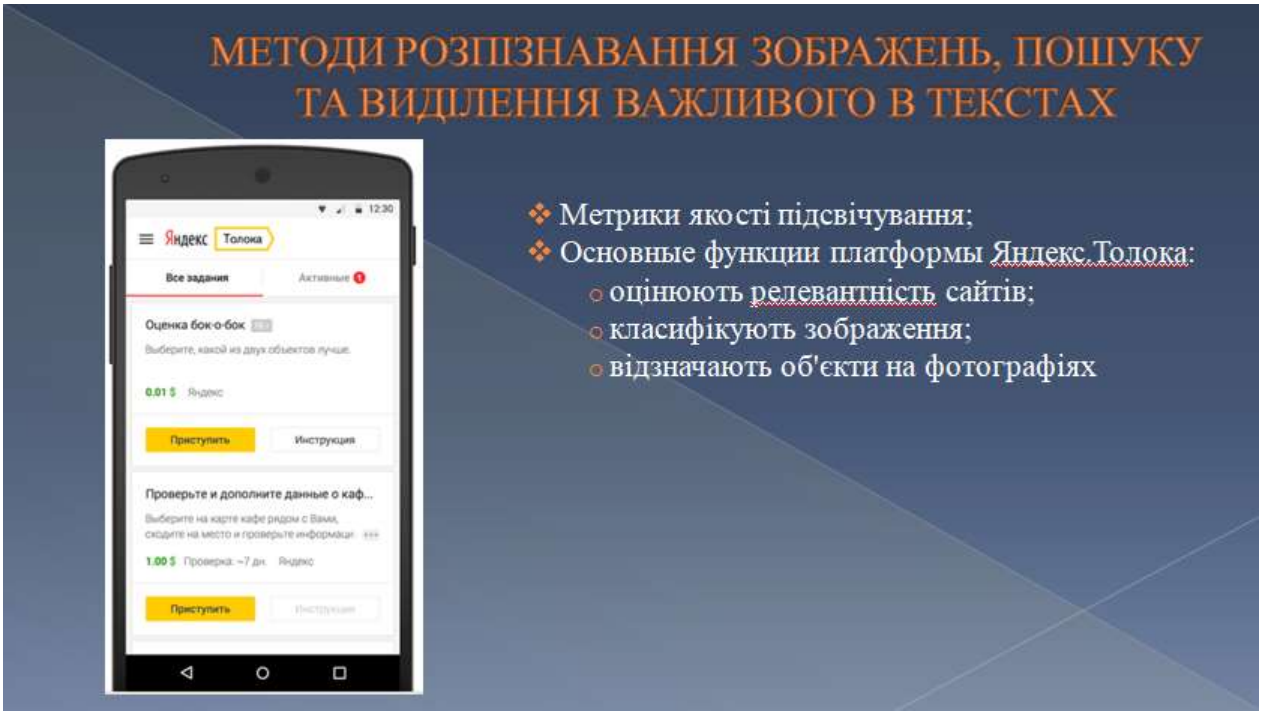

Рисунок А.9 - слайд презентації №9

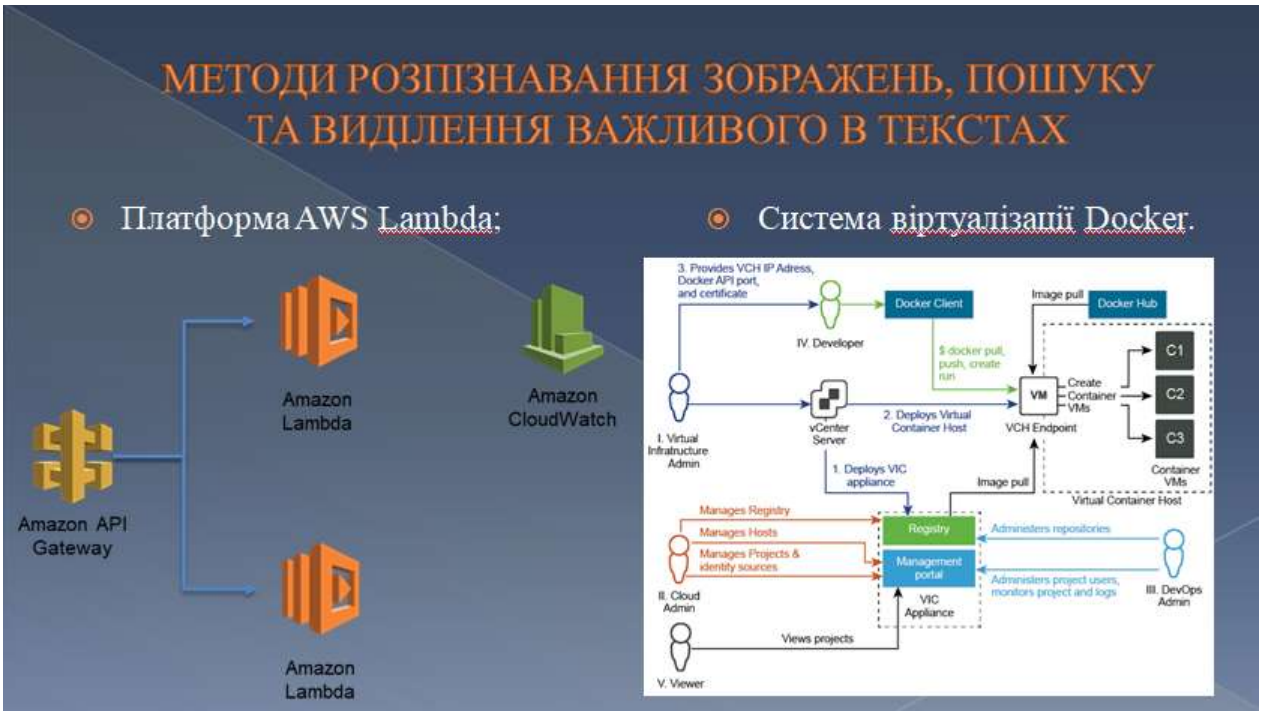

Рисунок А.10 - слайд презентації №10

Рисунок А.11 - слайд презентації №11

Рисунок А.12 - слайд презентації №12

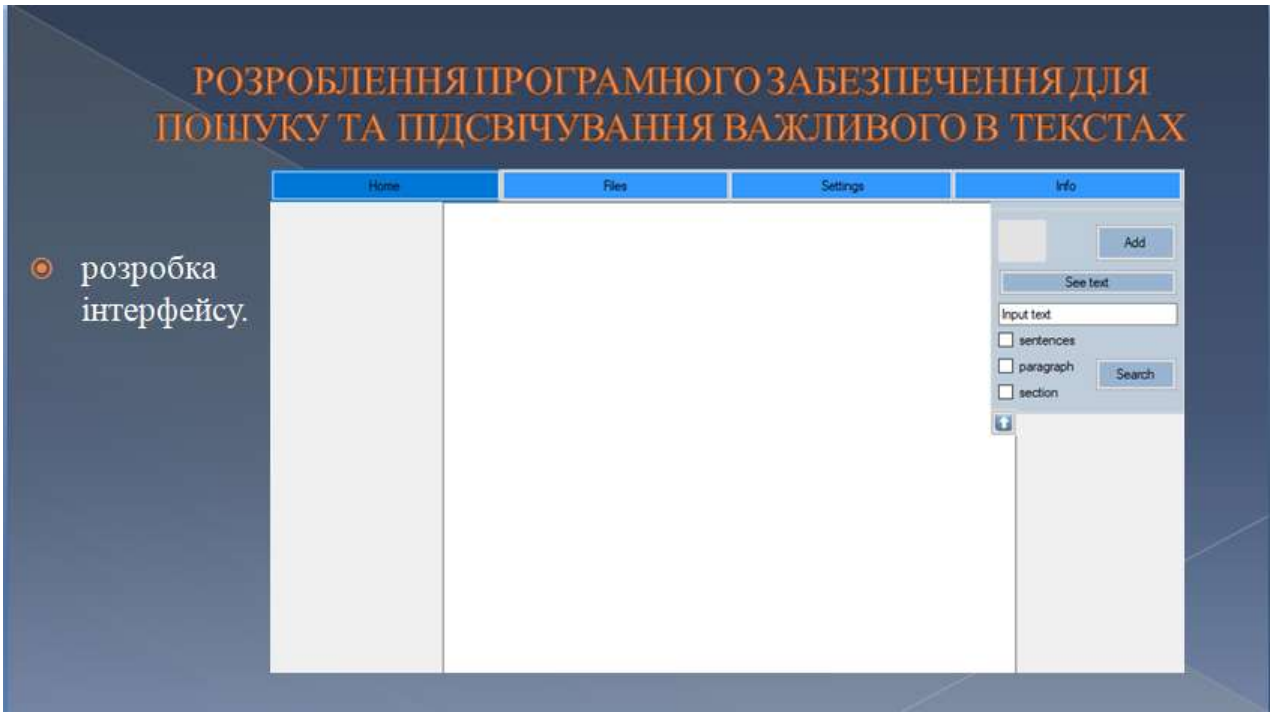

Рисунок А.13 - слайд презентації №13

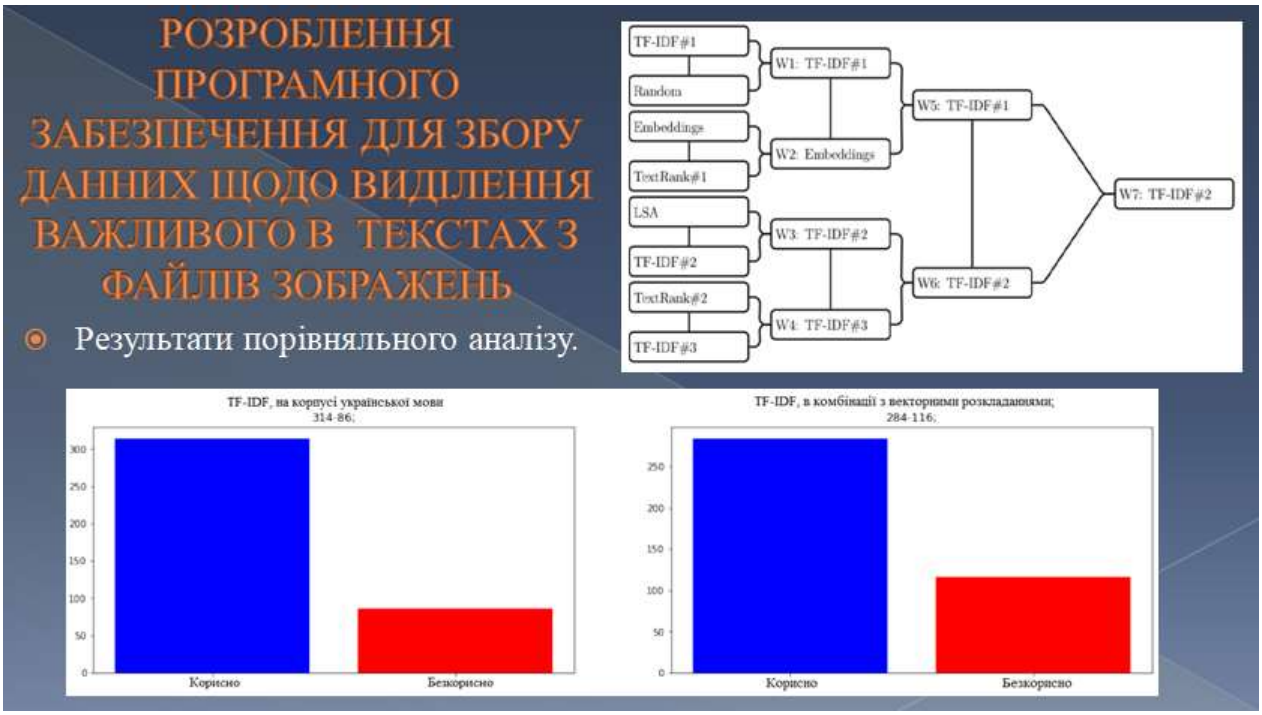

Рисунок А.14 - слайд презентації №14

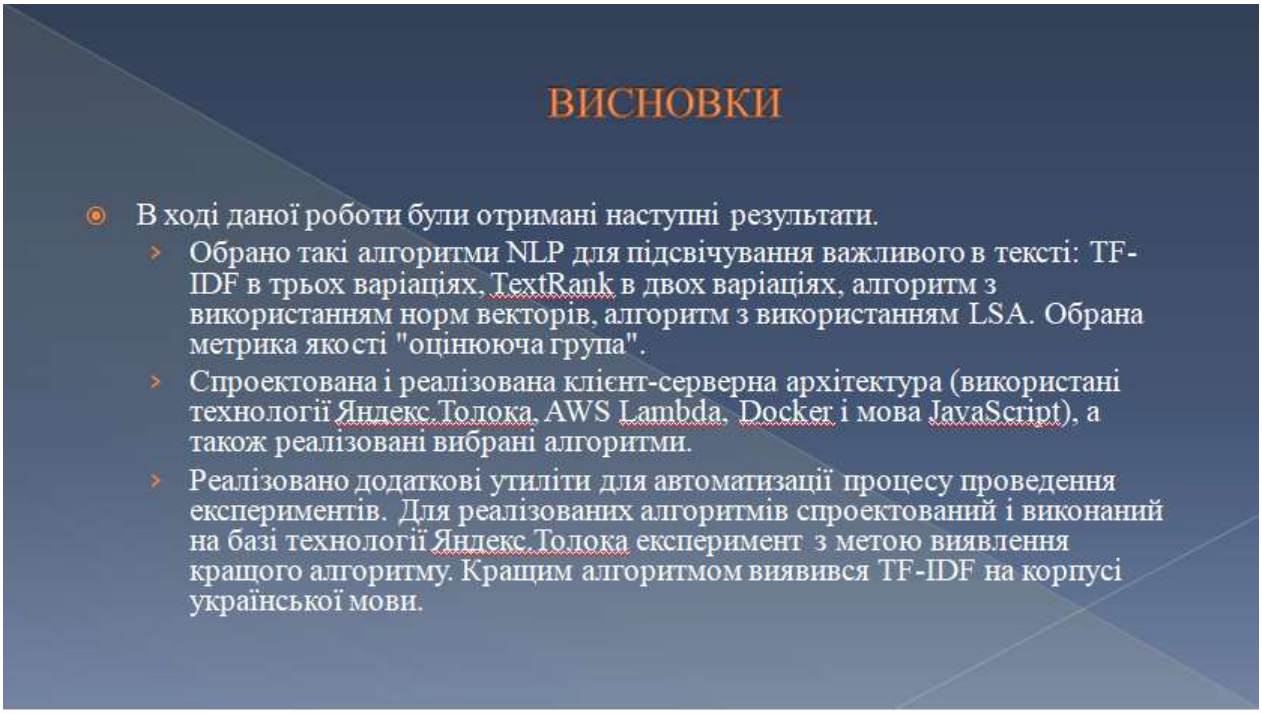

Рисунок А.15 - слайд презентації №15

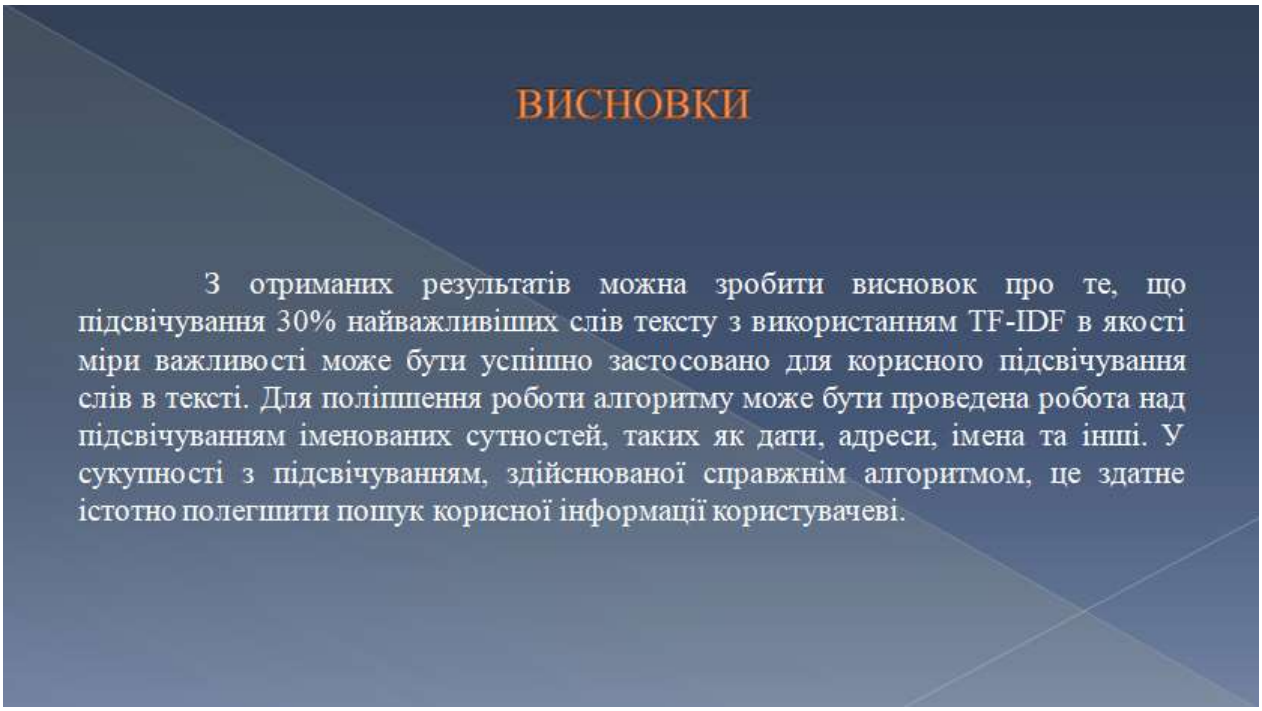

Рисунок А.16 - слайд презентації №16