

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Інтелектуальна система формування конспектів на
основі комбінованих методів аналізу текстових
даних»**

Виконав студент групи ІПЗм-24-2
ГРУНИЦЬКИЙ Дмитро Сергійович

Керівник роботи:
ЧИЖМОТРА Олексій Володимирович

Рецензент:
МОРОЗОВ Андрій Васильович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГРУНИЦЬКИЙ Дмитро Сергійович**

Керівник роботи: **ЧИЖМОТРЕ Олексій Володимирович**

Тема роботи: **«Інтелектуальна система формування конспектів на основі комбінованих методів аналізу текстових даних»,**

затверджена Наказом закладу вищої освіти від **«18» вересня 2025 р. № 491/с**

Вихідні дані для роботи: **процеси обробки та узагальнення текстових даних у системах автоматичного формування конспектів; функції та алгоритми системи, що забезпечують виділення ключових слів і генерацію змістовних узагальнень текстових матеріалів із застосуванням сучасних інструментів обробки природної мови.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	22.09.2025р.	22.09.2025р.
2	Чижмотря О.В.	22.09.2025р.	22.09.2025р.
3	Чижмотря О.В.	22.09.2025р.	22.09.2025р.

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «магістр» присвячена розробці інтелектуальної системи автоматичного формування конспектів на основі методів обробки природної мови та сучасних мовних моделей. Пояснювальна записка складається з вступу, трьох розділів та списку використаної літератури. Пояснювальна записка до кваліфікаційної магістерської роботи містить 74 сторінок, 33 ілюстрацій, 4 таблиці, 13 додатків та 41 посилання на зовнішні джерела.

Метою роботи є створення програмного продукту для автоматичного формування структурованих та змістовних конспектів текстів різного обсягу і формату з урахуванням семантичного контексту.

У результаті дослідження розроблено інтелектуальну систему, яка забезпечує завантаження текстів з різних джерел (файли, вебресурси, пряме введення), гнучке налаштування параметрів аналізу, автоматичне визначення кількості ключових слів та формування логічно узгодженого конспекту.

КЛЮЧОВІ СЛОВА: АВТОМАТИЧНЕ ФОРМУВАННЯ КОНСПЕКТІВ, КЛЮЧОВІ СЛОВА, ОБРОБКА ПРИРОДНОЇ МОВИ, СЕМАНТИЧНИЙ АНАЛІЗ, МОВНІ МОДЕЛІ.

ABSTRACT

The master's thesis is devoted to the development of an intelligent system for automatic note-taking based on natural language processing methods and modern language models. The explanatory note consists of an introduction, three chapters, and a list of references. The explanatory note to the master's thesis contains 74 pages, 33 illustrations, 4 tables, 13 appendices, and 41 references to external sources.

The purpose of the work is to create a software product for the automatic generation of structured and meaningful summaries of texts of various sizes and formats, taking into account the semantic context.

As a result of the research, an intelligent system has been developed that provides for the loading of texts from various sources (files, web resources, direct input), flexible configuration of analysis parameters, automatic determination of the number of keywords, and the formation of a logically consistent summary.

KEYWORDS: AUTOMATIC SUMMARY FORMATION, KEYWORDS, NATURAL LANGUAGE PROCESSING, SEMANTIC ANALYSIS, LANGUAGE MODELS.

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	11
1.1. Аналіз проблем і особливостей процесу автоматичного формування конспектів	11
1.2. Постановка задачі дослідження	13
1.3. Аналіз наявних програмних засобів автоматизації формування конспектів	15
1.4. Обґрунтування вибору інструментальних засобів	20
Висновки до розділу 1	24
РОЗДІЛ 2. РОЗРОБКА ТА ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ФОРМУВАННЯ КОНСПЕКТІВ	26
2.1. Визначення варіантів використання та опис бази даних	26
2.2. Опис реалізації функціоналу формування конспектів на основі комбінованих методів	32
2.3. Розробка алгоритму завантаження та обробки тексту із різних джерел	41
Висновки до розділу 2	43
РОЗДІЛ 3. ОПИС ПРОГРАМНОГО ПРОДУКТУ	45
3.1. Графічний інтерфейс	45
3.2. Налаштування параметрів програмного продукту та відображення результатів аналізу	49
Висновки до розділу 3	53
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	56
ДОДАТКИ	61

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

NLP (Natural Language Processing) – обробка природної мови;
API (Application Programming Interface) – інтерфейс програмування додатків;
JSON (JavaScript Object Notation) – текстовий формат обміну даними;
DB (Database) – база даних;
UI (User Interface) – інтерфейс користувача;
REST (Representational State Transfer) – інтерфейс прикладного програмування;
LLM (Large Language Model) – велика мовна модель;
ORM (Object-Relational Mapping) – об’єктно-реляційне відображення;
UX (User Experience) – користувацький досвід;
DBMS – (Database Management System) – система керування бази даних;
SQL (Structured Query Language) – мова структурованих запитів;
IDE (Integrated Development Environment) – інтегроване середовище розробки;
AI (Artificial Intelligence) – штучний інтелект;
ML (Machine Learning) – машинне навчання;
BERT (Bidirectional Encoder Representations from Transformers) – двоспрямовані кодувальні представлення з трансформерів;
CPU (Central Processing Unit) – центральний процесор;
GPU (Graphics Processing Unit) – графічний процесор;
CI/CD (Continuous Integration / Continuous Delivery) – безперервна інтеграція та доставка.

ВСТУП

Актуальність теми. Стрімке зростання обсягів інформації в цифровому середовищі, активний розвиток освіти в форматі «онлайн», наукових платформ, інформаційних ресурсів зумовлюють потребу у створенні інтелектуальної системи, яка дозволить автоматизувати обробку, структурування та, загалом, узагальнення того, про що йшлося в тексті, для того, щоб користувач міг відразу зрозуміти не витрачаючи час на його повне прочитання. Щодо користувачів, наразі, сучасний користувач – це, школяр, студент, викладач, дослідник або фахівець. Це ті люди, які щодня працюють з великими обсягами тексту, серед якого можуть бути як навчальні матеріали, книги, наукові статті, технічні документації, аналітичні звіти та інше. Всього не перерахуєш. Ефективне формування структурованих конспектів на основі таких даних є одним із ключових викликів сучасної цифрової освіти.

Актуальність теми зумовлена тим, що традиційні методи опрацювання текстових матеріалів: ручне читання, виділення основних положень, нотування – займають значні ресурси часу та вимагають високої концентрації уваги. На тлі збільшення інформаційних потоків такі підходи стають малоефективними. У науковому середовищі активно досліджуються методи обробки природної мови (NLP), машинного навчання та комбіновані методи аналізу тексту, які дозволяють автоматизувати виділення нормальних ключових слів, побудову семантичних структур та генерацію коротких змістовних викладів.

Ряд учених присвятили свої роботи аналізу статичних методів (TF-IDF, Bag-Of-Words(BOW)), лінгвістичних підходів (лематизація, морфологічний аналіз), семантичних моделей (word2vec, GloVe), алгоритмів ключових слів (RAKE, TextRank), а також неймережевих архітектур на основі трансформерів (BERT, GPT, T5). Однак, попри значний прогрес, залишається невирішеним питання ефективного поєднання різних методів в єдину модель, здатну забезпечити точне узагальнення матеріалу та формування повноцінних конспектів, придатних для навчання та практичного застосування.

Сучасні дослідження демонструють, що високої точності та змістовності результатів можна досягти саме при використанні комбінованого підходу, який, в свою чергу, поєднає статичні методи, лінгвістичний аналіз та можливості сучасних великих мовних моделей. Така інтеграція дає змогу не лише виокремлювати ключові терміни на основі частотних та контекстних характеристик (що реалізуються, зокрема, за допомогою KeyBERT та морфологічного аналізу), але й формувати узагальнений зміст за участі генеративних моделей, таких як Mistral 7B, доступних через локальне середовище Ollama. У результаті комбінований підхід забезпечує більш повне охоплення смислової структури тексту та підвищує якість сформованих конспектів. Саме це робить його доцільним та обґрунтованим у межах обраної теми магістерської роботи.

Мета дослідження є розробка інтелектуальної системи формування конспектів, яка забезпечує автоматичне виділення ключових слів із тексту та генерує змістовні узагальнення навчальних матеріалів із використанням сучасних інструментів обробки природної мови.

Для досягнення поставленої мети необхідно вирішити такі **основні завдання**:

1. Проаналізувати сучасні підходи та інструменти автоматизованого опрацювання текстових даних, включаючи методи виділення ключових слів і генерації змістовних узагальнень.
2. Дослідити роботу сучасних моделей обробки природної мови, таких як KEYBERT та великі мовні моделі (Mistral 7B, Ollama), у контексті формування конспектів.
3. Розробити логіку та архітектуру інтелектуальної системи для автоматичного формування конспектів із можливістю інтеграції окремих функцій (витяг ключових слів, інтеграція summary).
4. Реалізувати прототип системи з використанням мікросервісної архітектури для обробки та збереження текстових даних.

5. Провести тестування системи на прикладах різних типів текстових матеріалів та оцінити якість сформованих конспектів за критеріями точності, повноти та змістовності.
6. Надати рекомендації щодо оптимізації та подальшого розвитку системи для практичного використання у дослідницькій діяльності.

Об’єкт дослідження – процеси обробки та узагальнення текстових даних у системах автоматичного формування конспектів.

Предмет дослідження – функції та алгоритми системи, що забезпечують виділення ключових слів і генерацію змістовних узагальнень текстових матеріалів із застосуванням сучасних інструментів обробки природної мови.

Методи дослідження. В атестаційній роботі використано: методи системного аналізу, методи теорії інформації та лінгвістики (для нормалізації українського тексту); методи векторного представлення та кластеризації даних для реалізації Адаптивного N-ліміту; методи генеративного моделювання та адаптивного промпту для динамічного керування виходом LLM; а також методи асинхронного програмування.

Наукова новизна полягає в розробці та впровадженні двокомпонентного адаптивного підходу до аналізу текстових даних та формування конспектів, що значно підвищує якість результату порівняно зі статичними методами. Суть підходу полягає у спільному використанні семантичного аналізу та генеративного моделювання, а саме:

- Вдосконалення методу екстракції ключових слів шляхом розробки Адаптивного механізму N-ліміту. Цей механізм використовує кластеризацію векторних представлень тексту для автоматичного визначення оптимальної кількості ключових концептів, усуваючи необхідність ручного введення ліміту користувачем.
- Модифікація способу генерації конспектів на основі LLM через впровадження Механізму адаптивного промпту. Даний механізм динамічно керує вихідними параметрами (оптимальна довжина, k-сть

речень та ліміт токенів), забезпечуючи, що фінальний конспект є нормальним за розміром відповідаючи кількості ключових слів.

- Впровадження Механізму каскадної фільтрації та нормалізації ключових слів на основі LLM. Цей механізм передбачає використання великої мовної моделі як фінального інтелекту, який здійснить перевірку по словам та виведенні їх в нормальній формі. Він здійснює фінальну фільтрацію від нерелевантних термінів, контекстуальну нормалізацію (приведення слів до змістовних фраз, наприклад, «футбольний клуб» замість окремих слів), а також гарантує точну кількість ключових слів згідно з встановленим N-лімітом.

Практичне застосування роботи полягає в розробці та реалізації функціональної інтелектуальної системи формування конспектів, яка має здатність самостійно адаптуватися до характеристик вхідного тексту. Система дозволяє автоматично та якісно виділяти ключові слова, і генерувати конспект (summary) для великих обсягів даних українською мовою.

Апробація. За темою дослідження кваліфікаційної магістерської роботи було опубліковано 3 тези конференцій [1, 2, 3].

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз проблем і особливостей процесу автоматичного формування конспектів

Автоматичне формування конспектів є однією з ключових задач сучасної обробки природної мови [16, 41]. Зі зростанням обсягів текстової інформації в освітньому, науковому та професійному середовищі виникає потреба у швидкому виділенні суттєвих змістових елементів та перетворення великих обсягів текстової інформації у компактні, структуровані та логічно узгоджені конспекти.

У традиційному процесі конспектування, людина виконує кілька послідовних інтелектуальних операцій (рис. 1.1):

- розуміння контексту тексту;
- виділення головних ідей;
- відбір ключових понять та термінів;
- формування стислого тексту.

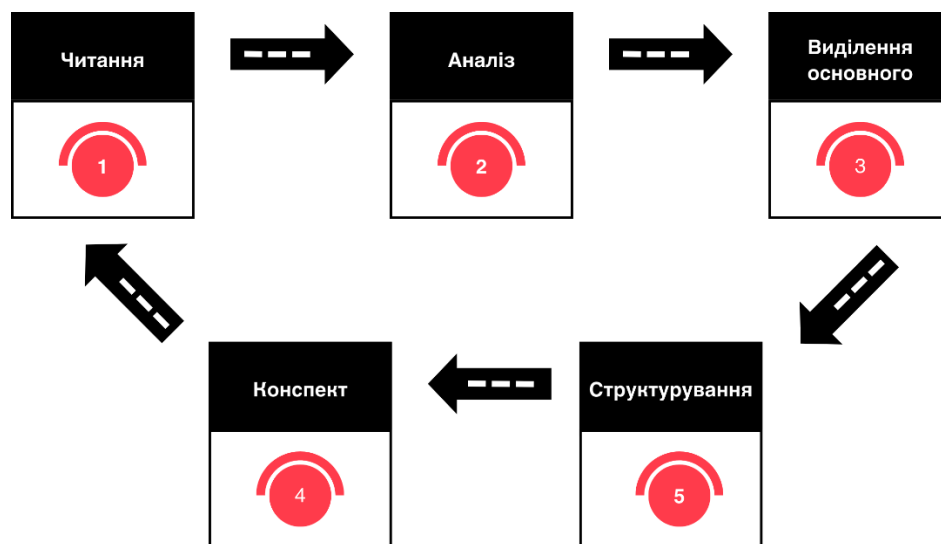


Рисунок 1.1 - Процес послідовності дій

Перенесення цього процесу у сферу автоматизованих систем виявляється складним через дану низку проблем:

- тексти часто містять приклади, пояснення, повтори, що ускладнюють автоматичне визначення важливих сегментів;

- розгорнуті конструкції, вставні слова та довгі речення погіршують якість алгоритмічного аналізу;
- одна й та сама фраза може мати різні значення залежно від теми та стилю тексту, що потребує глибокого семантичного аналізу;
- у наукових текстах ключові поняття можуть бути приховані серед великої кількості другорядного матеріалу.

Ці особливості зумовлюють необхідність використання алгоритмів, здатних поєднувати кілька напрямків обробки: семантичне моделювання, частотний аналіз, лінгвістичні правила, контекстну інтерпретацію тощо.

У сучасних дослідженнях виділяють два ключові підходи до автоматичного створення конспектів (рис. 1.2):

- Extractive summarization – вибір найважливіших речень без зміни їх структури;
- Abstractive summarization – створення нового тексту, що передає зміст вихідного матеріалу.

Перший підхід добре працює для структурованих текстів, але погано узагальнює складні матеріали. Другий – забезпечує глибше розуміння сенсу, але вимагає використання потужних моделей та великих обчислюваних ресурсів.

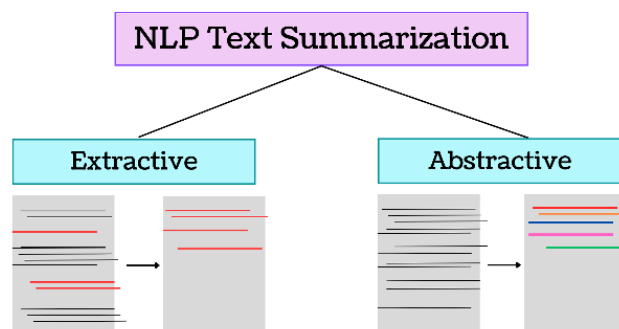


Рисунок 1.2 - Показ роботи двох ключових підходів

Для якісного формування конспекту необхідно забезпечити декілька умов:

- попередню обробку тексту (очищення, лематизація, нормалізація);
- виявлення ключових слів або термінів;
- виявлення логічної структури документа;

- генерацію узагальнення, яке зберігає смислову цілісність та термінологічну коректність.

Особливо важливо поєднати лінгвістичні та семантичні методи, оскільки лише статичний підхід не здатний охопити контекст, а лише семантичний – може генерувати узагальнення, що містить фактичні неточності або іншими словами «некоректний вміст».

Сучасні системи штучного інтелекту дозволяють комбінувати методи виділення ключових слів (наприклад, KeyBERT у поєднанні з морфологічним аналізом) та генерацію змістового викладу (наприклад, моделі Mistral 7B через локальну інфраструктуру Ollama). Такий комбінований підхід дозволяє зменшити кількість помилок, зберегти термінологічну точність та підвищити якість фінального конспекту.

Таким чином, процес автоматичного формування конспектів має цілу низку особливостей, пов'язаних із природою текстових даних, складністю мовних конструкцій та необхідністю поєднання різних типів алгоритмів. Аналіз цих проблем підтверджує доцільність розробки спеціалізованої інтелектуальної системи, здатної забезпечувати високу якість узагальнення навчальних, наукових та технічних матеріалів.

1.2. Постановка задачі дослідження

Процес створення конспектів на основі великих обсягів інформації потребує багато часу та уваги. Навіть при високому рівні підготовки та досвіду, ручне опрацювання документів, наукових статей або навчальних матеріалів є трудомістким процесом, який вимагає постійного аналізу, порівняння та організації даних. Незважаючи на наявність автоматизованих інструментів, більшість із них не забезпечує достатньої точності, врахування сенсу тексту та адаптованості результатів до потреб користувача. Проблема полягає в тому, що традиційні методи аналізу тексту часто застосовують статичні алгоритми або виконують скорочення без глибокого аналізу, що призводить до формування конспектів, які не мають

структурованої та логічно організованої форми. Як наслідок, користувачеві доводиться самостійно перевіряти, доповнювати та уточнювати згенеровані матеріали.

У межах цього дослідження виникає потреба у створенні програмного додатку, який зможе поєднати різні етапи аналізу тексту – від визначення ключових понять і ключових слів до генерації змістовних, логічно узгоджених та структурованих конспектів. Для цього потрібно розробити механізм, який використовуватиме комбінований підхід, який поєднує алгоритми витягування ключових слів (наприклад, KeyBERT) із сучасними мовними моделями (Mistral 7B через Ollama) для генерації стислих, але інформативних текстових підсумків. Такий підхід дозволяє врахувати як лексичні, статичні характеристики тексту, так і його семантичне навантаження, що забезпечує більш точне і корисне в генерації конспекту.

Таким чином, основна задача дослідження полягає у створенні інтелектуальної системи, здатної автоматично аналізувати текстові дані, виокремлювати з них ключові елементи змісту та формувати на їх основі структурований, зрозумілий і логічно побудований конспект.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. проаналізувати існуючі програмні засоби автоматизації формування конспектів та визначити їхні сильні та слабкі сторони;
2. розробити архітектуру системи та структуру взаємодії її модулів;
3. обґрунтувати вибір комбінованого підходу для обробки текстових даних та визначити алгоритми його реалізації;
4. реалізувати програмний прототип системи, що включає функції виділення ключових слів та генерації стислих підсумків;
5. провести експериментальну перевірку ефективності системи на різних типах текстових матеріалів та оцінити якість отриманих конспектів.

Результатом виконання дослідження стане практичне рішення, яке дозволяє значно скоротити час на підготовку навчальних та інформаційних матеріалів,

підвищити точність та зручність генерування конспектів, а також створити платформу, яка може адаптуватися під різні джерела та обсяги інформації.

1.3. Аналіз наявних програмних засобів автоматизації формування конспектів

У процесі розробки програмного продукту, спрямованого на автоматичне формування конспектів на основі завантажених користувачем документів, презентацій або текстових матеріалів, важливо врахувати існуючі аналоги, які частково реалізують подібний функціонал. Аналіз існуючих платформ дозволяє визначити сучасні підходи до обробки текстових даних, структурування інформації і генерації стислих навчальних матеріалів. Подібні рішення можуть слугувати прикладами для вдосконалення функціональності системи та покращення користувацького досвіду в цілому. Аналіз таких платформ дасть змогу виявити їхні обмежені можливості підключення до інших сервісів або недостатньо якісне створення конспектів. Отримані висновки стануть основою для формування вимог до власної системи та визначення напрямів її вдосконалення.

Один із аналогів – це SummarizeBot [9] (рис. 1.3). Цей онлайн інструмент, який допомагає користувачам автоматично аналізувати та скорочувати тексти за допомогою штучного інтелекту. Цей сервіс підтримує обробку текстів із різних форматів файлів, а також текст із посилань на джерела інформації. Він допомагає швидко виділяти основні ідеї великих обсягів тексту.

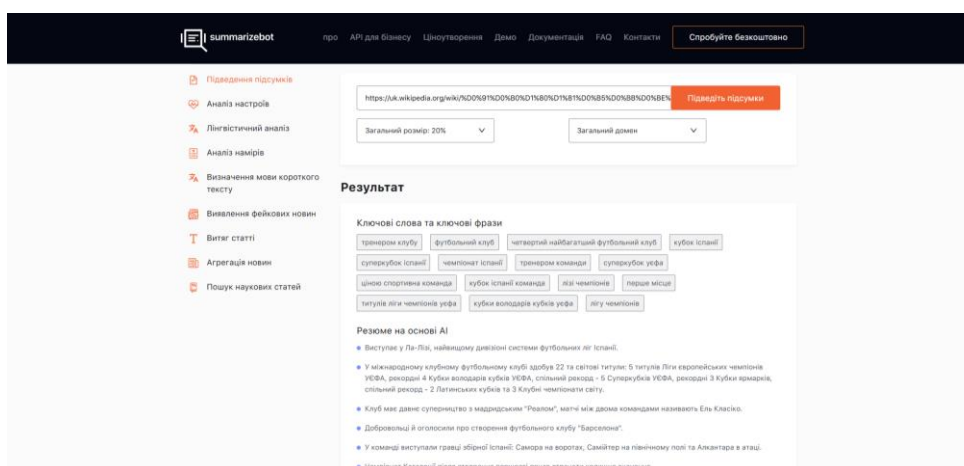


Рисунок 1.3 - SummarizeBot

До основних переваг програмного продукту належить використання методів штучного інтелекту, що забезпечує автоматичний аналіз тексту з виділенням ключових компонентів за сенсом для подальшого скорочення. Система підтримує завантаження різних типів документів, зокрема текстових файлів форматів TXT, DOCX та PDF, а також обробку текстового вмісту вебресурсів за посиланням, що дозволяє працювати з різноманітними джерелами інформації. Програмний продукт забезпечує швидке формування стислих версій навіть для великих обсягів тексту та може використовуватися через месенджери, зокрема Telegram. Додатковою перевагою є доступність основного функціоналу без необхідності фінансових витрат з боку користувача.

Водночас, система має певні обмеження. У окремих випадках, алгоритми аналізу можуть пропускати важливі деталі або, навпаки, акцентувати увагу на менш важливій інформації. Якість результату значною мірою залежить від структури вхідного тексту, і за наявності неструктурованих або слабо пов'язаних даних, підсумкове скорочення може бути менш інформативним. Аналіз надто великих файлів може потребувати більше часу або знижувати точність результатів. Крім того, у поточній реалізації відсутня можливість гнучкого вибору рівня скорочення або аналізу окремих частин тексту, а інтерфейс системи може вимагати певного часу для освоєння користувачами без попереднього досвіду роботи з подібними сервісами.

Іншим прикладом є Resoomer [13] (рис. 1.4) – сервіс для автоматичного створення коротких версій текстів, що дозволяє швидко отримати основну ідею зі статей, доповідей чи інших документів. Цей інструмент використовує ШІ для виділення важливих моментів у текстах і створення стислих підсумків. Resoomer підтримує роботу з великими текстами та надає можливість автоматичного структурування основних тез. Крім того, сервіс дозволяє зберігати згенеровані матеріали, що робить його корисним для навчання та підготовки аналітичних матеріалів. Водночас, функціональні можливості Resoomer є переважно

фіксованими, що обмежує гнучкість налаштування результатів аналізу під індивідуальні потреби користувача.

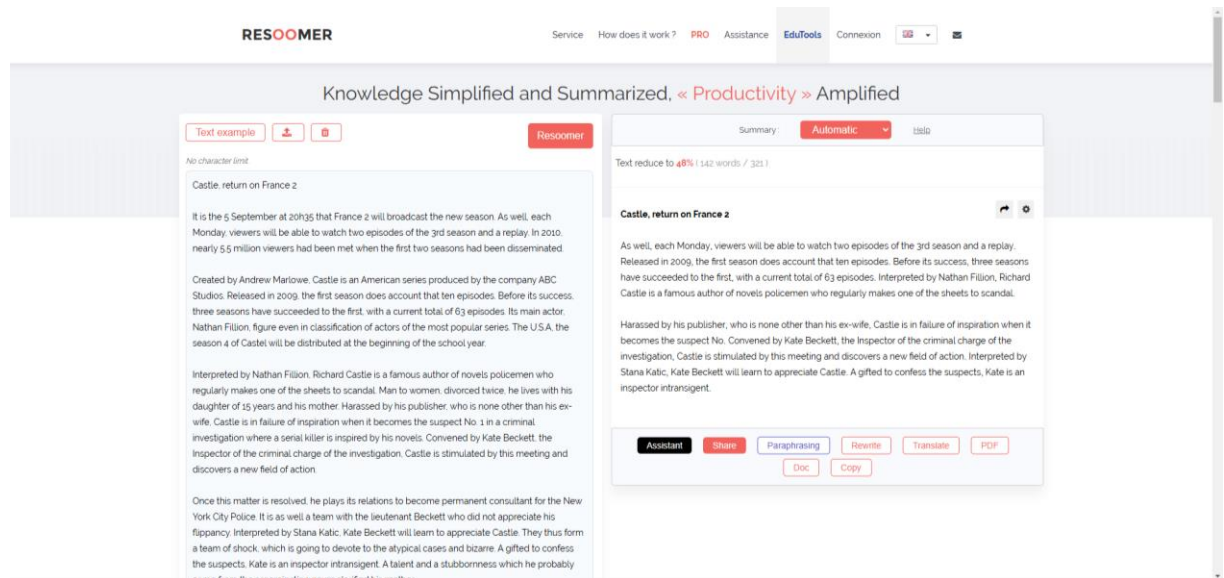


Рисунок 1.4 - Resoomer

До переваг програмного продукту належить здатність миттєво формувати короткий зміст довгих текстових матеріалів, що значно економить час користувача під час опрацювання великих обсягів інформації. Базовий функціонал системи є доступним без необхідності реєстрації або оформлення платних підписок, що спрощує початок роботи з сервісом. У процесі аналізу текстів використовується штучний інтелект, який забезпечує виділення ключових ідей та компонентів, які містять сенс, підвищуючи ефективність опрацювання інформації. Крім того, система підтримує як завантаження текстових файлів, так і роботу з матеріалами, отриманими з різних вебресурсів за посиланням.

Разом із тим програмний продукт має низку обмежень. У деяких випадках алгоритми аналізу можуть некоректно визначати найбільш важливі фрагменти тексту, що впливає на якість скороченого результату. Поточна реалізація не передбачає можливості вибору рівня детальності підсумкового тексту або налаштування способу виділення інформації. За умови погано структурованого вхідного тексту результат скорочення може бути неточним або недостатньо інформативним. Додатково інтерфейс системи може виявитися складним для користувачів без попереднього досвіду, а також відсутні готові інструменти

інтеграції з іншими сервісами. Окрім цього якість результатів значною мірою залежить від характеристик вхідних даних і може змінюватися залежно від тематики та стилю тексту.

Ще одним прикладом є QuillBot [29] (рис. 1.5). Цей інструмент допомагає користувачам працювати з текстами, скорочуючи їх та перефразовуючи для кращого розуміння. Завдяки використанню штучного інтелекту, сервіс автоматично виділяє основні моменти тексту та створює його спрощену версію. Це зручно для тих, хто хоче швидко отримати головну інформацію без детального читання всього матеріалу. Крім того, QuillBot пропонує кілька режимів обробки тексту – від більш формального до максимально стислого, що дозволяє адаптувати результат під різні потреби. Інструмент також інтегрується з популярними платформами, такими як Google Docs чи браузерні розширення, роблячи процес роботи з текстом більш зручним та доступним.

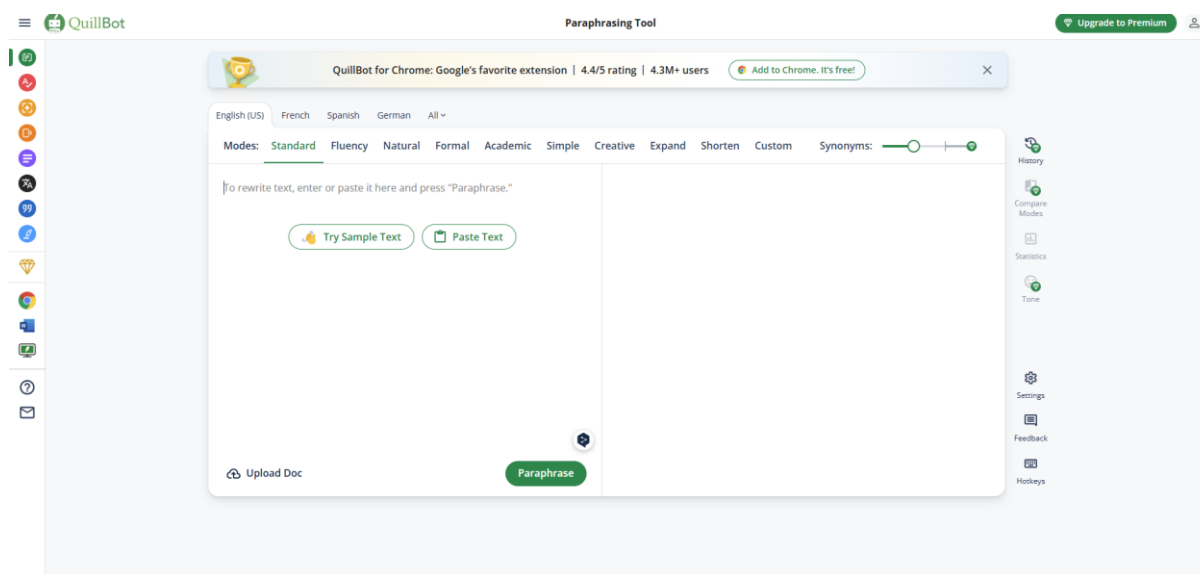


Рисунок 1.5 - QuillBot

До переваг системи належить те, що окрім скорочення текстів вона надає можливість їх перефразовування, перевірки граматики та формування цитат, що розширює спектр її практичного застосування. Інтерфейс програмного продукту є інтуїтивно зрозумілим і не потребує від користувача спеціальних технічних навичок, що полегшує взаємодію з системою. Система здатна миттєво обробляти навіть великі обсяги текстової інформації, забезпечуючи швидке отримання

результатів. Користувач може як вводити тексту вручну, так і завантажувати файли, що підвищує зручність використання сервісу. Базові функції доступні без обов'язкової реєстрації, а застосування алгоритмів штучного інтелекту дає змогу більш точно визначити ключові моменти тексту та формувати зрозумілі й логічно структуровані підсумки.

Водночас, система має низку недоліків. У деяких випадках алгоритми можуть надмірно скорочувати текст, пропускаючи важливі деталі, що негативно впливає інформативність результату. Якщо вхідний текст має слабку або хаотичну структуру, отримані результати можуть бути некоректними або недостатньо точними. Частина розширених можливостей, зокрема поглиблений рівень обробки текстів, доступна лише у преміум версії, що обмежує функціональність безкоштовного використання. Крім того, система не має вбудованих інструментів для інтеграції з іншими сервісами, а в окремих випадках згенерований текст може виглядати штучно або складним для сприйняття. Також, варто зазначити, що сервіс не завжди коректно працює зі складними технічними або текстами, які стосуються лише певної сфери, що може призвести до неправильного розуміння термінів і ключових понять.

Аналіз таких платформ, як SummarizeBot, Resoomer та QuillBot, показав (табл. 1.1), що всі вони пропонують ефективні інструменти для скорочення текстів та виділення ключових моментів, використовуючи ШІ. Кожен з цих сервісів має свої переваги, такі як швидкість обробки інформації, простота використання і підтримка різних форматів тексту. Однак, незважаючи на всі ці позитивні моменти, вони також мають свої недоліки, такі як обмежена точність скорочення, залежність від структури тексту та обмежені можливості налаштування. Для розробки власної системи важливо врахувати ці аспекти, зокрема забезпечити більш точне виділення важливих частин тексту, покращити налаштування для користувачів і надавати гнучкість в управлінні результатом. Розробка таких функцій дозволить створити більш ефективний та зручний інструмент для користувачів, орієнтуючи продукт на більш зручний підхід до засвоєння матеріалу.

Порівняльна таблиця

Параметр	Інтелектуальна система	SummarizeBot	Resoomer	QuillBot
Тип	Документи (.doc, .docx, .pdf, .pptx), посилання	Посилання	Текст, документи (.pdf, .epub, .docx, .odt, .txt)	Текст, документи (.docx)
Мови	Українська, Англійська	Англійська	Багатомовна	Багатомовна
Аналіз даних	Терміни, визначення, ключові слова	Виділення ключових слів	Виділення ключових слів	Перефразування тексту, перевірка на правильність написання слів, перекладач
Метод обробки	Аналіз структурованої інформації	ШІ та обробка природної мови	Алгоритмічна обробка	Використовуються нейронні мережі
Аудиторія	Школярі, студенти, звичайні користувачі	Школярі, студенти, звичайні користувачі	Школярі, студенти, звичайні користувачі, читачі	Школярі, студенти, звичайні користувачі, автори
Додаткові функції	Можлива інтеграція з Telegram	Підтримка аудіофайлів	Плагін для браузера	Граматична перевірка, перекладач

1.4. Обґрунтування вибору інструментальних засобів

Вибір інструментальних засобів для розробки вебдодатка є важливим кроком, який визначає ефективність створення системи, її подальшу масштабованість, продуктивність та якість кінцевого результату. У межах проєкту потрібно забезпечити гнучку й надійну обробку текстових даних, інтеграцію алгоритмів аналізу та генерації змісту, а також стабільну роботу сервісу в умовах різних навантажень. Саме тому добір інструментів здійснювався з урахуванням їхньої продуктивності, сумісності між собою, простоти підтримки та можливості подальшого розширення функціональності. Обрані технології дозволяють реалізувати всі необхідні етапи роботи системи – від попередньої обробки тексту до формування підсумкового конспекту. Забезпечуючи високу швидкість, точність, зручність використання та надійність роботи застосунку.

Перш за все, для реалізації даної системи було обрано мікросервісну архітектуру [40]. Такий підхід дозволяє розподілити функціональність додатку на окремі, незалежні сервіси, що забезпечить більшу гнучкість, масштабованість і можливість легкої заміни компонентів. Мікросервісна архітектура дозволяє зосередити увагу на окремих аспектах додатку, таких як обробка текстів, зберігання даних чи взаємодія з користувачем, що значно спрощує управління проектом та знижує ризики появи помилок в коді.

Для клієнтської частини (frontend) (рис. 1.6) було обрано AngularJS [33] з використанням TypeScript [38], оскільки цей фреймворк дозволяє створювати масштабовані, модульні та інтерактивні вебінтерфейси. Для реалізації сучасного та зручного дизайну інтерфейсу використовується бібліотека Ant Design [34], що забезпечує готові компоненти, підтримку адаптивності та гармонійний вигляд додатку. Такий стек дозволяє швидко реалізовувати функціонал, зокрема завантаження матеріалів, перегляд ключових слів та результат генерованого конспекту.



Рисунок 1.6 - Стек технологій клієнтської частини

Для реалізації основної логіки інтелектуальної системи було обрано мову програмування Python, яка є одним із найпотужніших інструментів для роботи з текстовими даними, машинним навчанням та мовою природної обробки (NLP). Розробка серверної частини здійснюється за допомогою фреймворку FastAPI [35], що забезпечує високу продуктивність, асинхронність обробки запитів. Основний сервіс системи базується на наборі інструментів, які дозволяють комплексно виконувати аналіз тексту, генерувати резюме та забезпечувати загальну інтерактивність роботи мікросервісу.

У головному сервісі обробки текстів використовується набір інструментів, що забезпечує якісну роботу на кожному етапі опрацювання даних. Бібліотека spaCy застосовується для попередньої обробки тексту та токенизації. Оскільки підтримка української мови в бібліотеці spaCy є обмеженою, у системі використовується мовна модель uk_core_news_md, яка забезпечує базовий морфологічний аналіз та лематизацію, доповнені власними правилами очищення і нормалізації ключових слів. Такий комбінований підхід дозволяє отримати коректні частини мови, леми та морфологічні характеристики навіть у випадках, коли spaCy не може опрацювати текст повністю. Для векторизації та обчислень застосовуються numpy та sklearn, що забезпечують швидку роботу з масивами та допоміжними статичними операціями. Витягування ключових слів реалізується за допомогою KeyBERT [36], який генерує найбільш кращі терміни на основі семантичної близькості тексту та обчислених векторних представлень. Асинхронна обробка запитів забезпечується модулем asuncio, що дозволяє сервісу ефективно працювати з декількома задачами паралельно.

Для генерації підсумкових текстів використовується інтеграція з локальним рушієм Ollama [8], який дозволяє працювати з мовною моделлю Mistral 7B [39]. Це дає змогу отримувати узгоджені, логічні та стислі конспекти, які залишаються близькими до змісту оригінального матеріалу.

У сукупності, даний вибір технологій (рис. 1.7) забезпечує комплексну, багаторівневу обробку текстів – від морфологічного аналізу до генерації структурованих підсумків, що робить систему ефективною та придатною для інтеграції у реальні навчальні чи робочі процеси.



Рисунок 1.7 - Стек технологій серверної частини

Для роботи з базою даних у проєкті використовується реляційна база даних PostgreSQL, оскільки дана система керування забезпечує високу надійність, підтримку складних структур та гарну продуктивність при обробці великих текстових матеріалів. Вона ідеально підходить для зберігання результатів аналізу тексту, таких як ключові слова, сформовані конспекти та інших проміжних даних. Для взаємодій з PostgreSQL застосовується SQLAlchemy [37] – ORM, яка дозволяє зручно працювати з таблицями й моделями, що спрощує створення, оновлення та запити до БД без необхідності вручну формувати SQL-запит. Для опису структур даних використовується Pydantic, який забезпечує валідацію, типізацію та перетворення даних між різними модулями системи. Завдяки цьому мінімізуються помилки під час обробки великих обсягів даних при обміні між сервісами. Керування змінами в БД виконується за допомогою Alembic [32], який дозволяє впроваджувати та відстежувати зміни у структурі таблиць без порушення роботи системи. Даний стек можна переглянути на рис. 1.8.



Рисунок 1.8 - Стек технологій для роботи з БД

Для забезпечення взаємодії між мікросервісами [29] буде використовуватись REST API [31] (рис. 1.9), оскільки це стандартний та простий у використанні спосіб для створення запитів та отримання відповідей між різними частинами системи. Це дозволить ефективно організувати обмін даними між сервісами, які будуть відповідати за передачу даних.



Рисунок 1.9 - Схема роботи REST API

Для розробки використовуватиметься Visual Studio Code (рис. 1.10, зліва) – популярне середовище розробки з підтримкою багатьох розширень, що забезпечить роботу з будь-якою мовою програмування.

Git (рис. 1.10, справа) буде використовуватись для контролю версій, що дозволить відслідковувати зміни в коді, зберігати історію та організовувати співпрацю з репозиторієм.



Рисунок 1.10 - Додаткові інструменти

Загалом, вибір цих технологій та інструментів був зроблений з урахуванням їхньої популярності, стабільності та відповідності вимогам проєкту. Всі компоненти технологічного стеку добре інтегруються між собою і дозволяють створити потужну, масштабовану та ефективну систему для роботи з обробкою тексту.

Висновки до розділу 1

У результаті аналізу предметної області та існуючих проблем автоматичного формування конспектів було встановлено, що процес скорочення та структурування текстової інформації є складним завданням через надлишковість даних, складність синтаксичних конструкцій, багатозначність термінів і нерівномірний розподіл важливої інформації. Традиційні методи ручного створення конспектів вимагають значних часових витрат, що є дуже важливим, і не завжди забезпечують достатню точність, а сучасні автоматизовані інструменти часто не враховують контекст та семантичне навантаження тексту, що призводить до формування неповних або некоректних конспектів.

Аналіз існуючих платформ, таких як SummarizeBot, Resoomer та QuillBot, показав, що сучасні рішення використовують штучний інтелект для виділення ключових моментів тексту та генерації стислих підсумків. Проте, мають низку

обмежень, зокрема обмежену точність, залежність від структури тексту та обмежені можливості налаштування для користувача. Виявлені переваги та недоліки наявних сервісів стали основою для визначення вимог до розробки власної інтелектуальної системи, яка має поєднувати різні підходи обробки тексту та забезпечувати високу точність і зручність у використанні.

Для реалізації поставленої мети було обґрунтовано вибір інструментальних засобів, які забезпечують комплексну обробку текстових даних. Використання Python і FastAPI дозволять створити продуктивний серверний компонент, а бібліотеки spaCy, KeyBERT та інтеграція з локальним рушієм Ollama забезпечують якісну морфологічну, семантичну та контекстну обробку тексту. Для клієнтської частини обрано AngularJS з TypeScript та Ant Design, що гарантує зручний та адаптивний інтерфейс, а робота з PostgreSQL, SQLAlchemy та Alembic дозволять надійно зберігати та опрацьовувати великі обсяги даних.

Таким чином, проведений аналіз підтвердив доцільність розробки інтелектуальної системи для автоматичного формування конспектів, яка комбінує різні алгоритми обробки тексту та забезпечує генерацію структурованих, логічно узгоджених і точних конспектів. Обґрунтований вибір технологій та інструментальних засобів створює основу для побудови ефективного та масштабованого рішення, здатного покращити процес засвоєння інформації та зменшити час на підготовку навчальних і наукових матеріалів.

РОЗДІЛ 2. РОЗРОБКА ТА ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ФОРМУВАННЯ КОНСПЕКТІВ

2.1. Визначення варіантів використання та опис бази даних

Розробка інтелектуальної системи автоматичного формування конспектів вимагає чіткого визначення функціональних можливостей даного застосунку, сценаріїв взаємодії користувача із системою та структури даних, необхідних для реалізації основних алгоритмів. На цьому етапі важливо окреслити ключові варіанти використання (use cases), які визначають поведінку системи, її реакцію на дії користувача та порядок виконання операцій. Формалізація таких сценаріїв дозволяє забезпечити логічну узгодженість між функціональними вимогами та архітектурою програмного забезпечення.

Крім того, для стабільної роботи системи необхідно розробити структуру бази даних, яка забезпечить ефективне збереження необхідних даних, обробку та доступ до текстових матеріалів, метаданих, результатів аналізу, ключових слів та сформованих конспектів. Побудова моделі даних є фундаментом для реалізації модулів обробки інформації, взаємодії мікросервісів між собою та подальшого масштабування продукту.

У цьому підрозділі буде визначено основні варіанти використання системи, описано їхню взаємодію з користувачем та внутрішніми компонентами, а також сформовано структуру бази даних із поясненням її ключових таблиць, атрибутів та взаємозв'язків.

На основі визначених вимог до функціонування даної системи було сформовано перелік основних прецедентів використання, які відображають типові сценарії взаємодії користувача із системою та внутрішні процеси обробки текстових даних. Дані прецеденти охоплюють повний життєвий цикл роботи з документом – від додавання матеріалу до отримання готового конспекту та його експорту у зручному форматі.

Основним актором системи є користувач, який ініціює процес обробки тексту шляхом завантаження документа, введення тексту вручну або надання посилання на зовнішній ресурс. Після цього система автоматично виконує низку внутрішніх операцій, зокрема витягування тексту, визначення ключових слів та генерацію структурованого конспекту за допомогою мовної моделі. Таке розмежування дій користувача та внутрішніх процесів система дозволяє чітко визначити відповідальність кожного компонента та спростити подальше проєктування архітектури.

Для наочного відображення взаємодії між актором та функціональними можливостями застосунку була побудована діаграма варіантів використання (рис. 2.1), яка ілюструє основні прецеденти роботи системи та їх взаємозв'язки. Дана діаграма дозволяє узагальнено представити логіку функціонування системи та слугує основою для подальшого проєктування системи структури бази даних і програмних модулів.

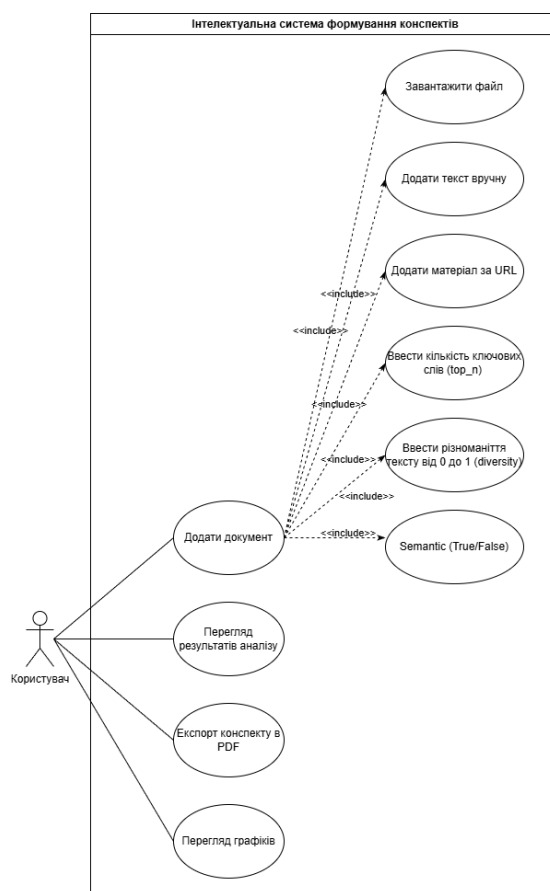


Рисунок 2.1 - Діаграма варіантів використання

Прецедент «Додавання документа до системи» передбачає, що користувач додає матеріал до системи одним із доступних способів для подальшого аналізу та задає параметри аналізу тексту для визначення ключових слів та формування конспекту. Користувач обирає дію, наприклад, завантаження файлу, вставлення посилання або введення тексту вручну. Після цього він задає параметри аналізу тексту, включаючи кількість ключових слів (top_n), рівень різноманіття ключових слів (diversity) у діапазоні від 0 до 1 або режим семантичного аналізу (semantic). Залежно від обраного способу додавання матеріалу система приймає файл формату PDF, DOC, DOCX, PPTX, CSV або TXT, обробляє його та витягує текст; або користувач вводить посилання на статтю чи інший текстовий ресурс, після чого система завантажує сторінку та витягує текстовий вміст; або вводить текст вручну, і система перевіряє формат та коректність введених даних. Отриманий текст разом із заданими параметрами аналізу зберігаються у базі даних. В результаті документ та параметри його аналізу успішно додано до системи і підготовлено для подальшого витягування ключових слів та генерації конспекту.

Прецедент «Витягування ключових слів» передбачає, що мікросервіс аналізу тексту отримує текст документа, після чого, алгоритм KeyBERT визначає первинний набір ключових слів. Далі отриманні ключові слова уточнюються та фільтруються за допомогою мовної моделі LLM, що забезпечує їхню семантичну релевантність і точність. Після цього сформований перелік ключових слів зберігається у базі даних, забезпечуючи подальше використання для генерації конспекту. В результаті користувач отримує готовий перелік ключових слів документа.

Прецедент «Генерація конспекту (summary)» передбачає, мікросервіс аналізу тексту через LLM (Mistral 7B, інтегрована через Ollama) отримує заголовок документа, оброблений текст та ключові слова. На основі цих даних мовна модель генерує узагальнений текст, який логічно відображає основні ідеї документа. Система форматує отриманий результат у вигляді структурованого конспекту,

після чого конспект зберігається у базі даних. В результаті користувач отримує готовий конспект, доступний для перегляду та подальшого використання.

Прецедент «Перегляд результатів аналізу документа» передбачає, що користувач переходить у розділ «Збережене» та обирає документ зі списку. Після цього система відображає ключові слова, сформований конспект та вихідний текст документа. В результаті користувач отримує повний огляд результатів обробки конкретного тексту, що дозволяє оцінити як зміст, так і якість аналізу.

Прецедент «Експорт конспекту в форматі PDF» передбачає, що користувач переглядає результат обробки документа та натискає кнопку «Завантажити PDF». Система формує файл на основі збереженого конспекту та ключових слів, а потім автоматично завантажує його на пристрій користувача. В результаті користувач отримує конспект та ключові слова у зручному для використання форматі PDF.

Прецедент «Перегляд аналітичних графіків за обробленими текстами» передбачає, що користувач переходить у розділ «Dashboard», де система відображає інтерактивні графіки. Користувач може переглянути порівняння кількості згенерованих ключових слів для різних текстів, залежність швидкості аналізу текстів від параметра diversity та час виконання аналізу для кожного документа. В результаті користувач отримує візуалізовану статистику, що дозволяє оцінити ефективність роботи системи та вплив параметрів аналізу на результати обробки текстів.

Для забезпечення коректного збереження текстових матеріалів, результатів їх аналізу та параметрів обробки у розроблюваній інтелектуальній системі було спроектовано реляційну базу даних. Її структура орієнтована на збереження вихідних текстів, метаданих, а також результатів автоматичного витягування ключових слів і генерації конспектів. Запропонована модель даних забезпечує цілісність інформації, зручний доступ до результатів аналізу та можливість подальшого розширення функціональності системи.

База даних системи складається з двох основних таблиць: texts та analysis_results, які пов'язані між собою відношенням один до одного.

Таблиця texts (табл. 2.1) призначена для збереження вихідних текстових матеріалів, які додаються користувачем до системи для подальшого аналізу.

Таблиця 2.1

Таблиця texts

Поле	Тип даних	ПК	ЗК	Призначення
id	INTEGER	+	-	Унікальний ідентифікатор текстового документа
title	STRING	-	-	Назва документа
content	STRING	-	-	Повний текст документа
language	STRING	-	-	Мова тексту (за замовчуванням - українська)
source_type	STRING	-	-	Тип джерела документа (файл, URL або введення вручну)
createdAt	DATE	-	-	Дата та час створення

Таблиця texts є базовою сутністю системи, на основі якої здійснюється подальший аналіз текстової інформації.

Таблиця analysis_results (табл. 2.2) призначена для збереження результатів інтелектуального аналізу тексту, що включає витягування ключових слів, генерацію конспекту та фіксацію параметрів і характеристик процесу обробки.

Таблиця 2.2

Таблиця analysis_results

Поле	Тип даних	ПК	ЗК	Призначення
id	INTEGER	+	-	Унікальний ідентифікатор результатів аналізу
text_id	INTEGER	-	+	Зовнішній ключ, що посилається на таблицю texts
keywords	JSON	-	-	Список ключових слів
summary_text	TEXT	-	-	Згенерований конспект документа
embedding_model	STRING	-	-	Модель ембедингів, використана для семантичного аналізу
text_word_count	INTEGER	-	-	Кількість слів у вихідному тексті
n_user	INTEGER	-	-	Кількість ключових слів, задана користувачем
n_semantic_recommended	INTEGER	-	-	Рекомендована кількість ключових слів на основі семантичного аналізу
n_final	INTEGER	-	-	Фінальна кількість ключових слів, використана в аналізі

Продовження таблиці 2.2

use_semantic_n	BOOLEAN	-	-	Ознака використання семантично рекомендованої кількості ключових слів
diversity	DOUBLE PRECISION	-	-	Коефіцієнт різноманіття ключових слів
keybert_duration_ms	INTEGER	-	-	Час виконання алгоритму KeyBERT
llm_duration_ms	INTEGER	-	-	Час генерації конспекту мовною моделлю
createdAt	DATE	-	-	Дата та час створення

Дана таблиця дозволяє зберігати як кінцеві результати аналізу, так і допоміжні метрики, що використані для оцінювання ефективності роботи алгоритмів.

Між таблицями `texts` та `analysis_results` реалізовано зв'язок типу один до одного, при якому кожному текстовому документу відповідає один результат аналізу. При видаленні запису з таблиці `texts` пов'язані результати аналізу автоматично видаляються, що забезпечує цілісність даних у системі.

Таблиця `alembic_version` (табл. 2.3) використовується механізмом міграцій Alembic для зберігання інформації про поточну версію схеми бази даних. Вона містить ідентифікатор останньої застосованої міграції, що дозволяє системі відстежувати стан структури БД та забезпечувати коректне виконання подальших змін. Наявність даної таблиці гарантує узгодженість схеми БД між різними середовищами розгортання та спрощує процес керування версіями.

Таблиця 2.3

Таблиця `alembic_version`

Поле	version_num	Призначення
1	9b716da8e3dd	Ідентифікатор поточної версії міграції

Структура бази даних інтелектуальної системи автоматичного формування конспектів охоплює основні сутності, необхідні для збереження текстових матеріалів, результатів їх аналізу та параметрів обробки. Запропонована модель забезпечує ефективне зберігання, обробку та доступ до вихідних даних, ключових слів і згенерованих конспектів, а також підтримує цілісність інформації в процесі

взаємодії компонентів системи. Нижче наведено ER-діаграму бази даних (рис. 2.2), яка ілюструє структуру та взаємозв'язки між основними таблицями системи. Модель даних побудована з урахуванням модульної архітектури програмного продукту та можливості подальшого розширення функціоналу без порушення існуючих зв'язків. Кожна таблиця виконує чітко визначену роль у процесі обробки текстів, що сприяє спрощенню логіки взаємодії між сервісами та підвищує масштабованість системи. Такий підхід забезпечує надійну основу для реалізації алгоритмів аналізу та генерації конспектів.

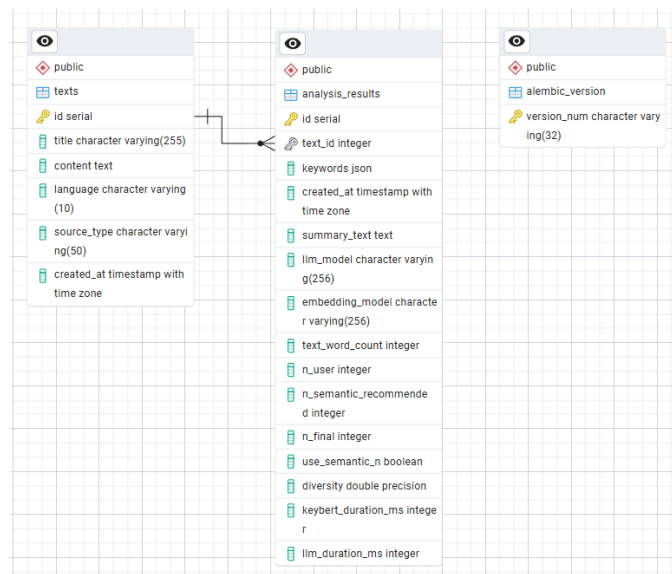


Рисунок 2.2 – ER-діаграма зв'язків бази даних

2.2. Опис реалізації функціоналу формування конспектів на основі комбінованих методів

Формування стислих та інформативних конспектів у межах розроблюваної інтелектуальної системи реалізовано на основі поєднання класичної обробки природної мови та сучасних моделей машинного навчання [4]. Такий підхід дозволяє враховувати як статичні та семантичні характеристики тексту [27], так і контекстні зв'язки між його фрагментами.

Для реалізації функціоналу аналізу текстових матеріалів було використано мову програмування Python, яка забезпечує широку екосистему бібліотек для природної мови та інтеграції з моделями машинного навчання. Архітектура

обробки тексту побудована у вигляді окремого мікросервісу, що дозволяє ізолювати ресурсоемні обчислення та забезпечити масштабованість системи.

Ключовим інструментом для витягування ключових слів у системі є бібліотека KeyBERT [36], яка використовує семантичні представлення тексту у вигляді векторів (ембедингів) [28] (рис. 2.3) для визначення найбільш релевантних термінів. Для формування ембедингів застосовано попередньо навчену мовну модель, що дозволяє враховувати семантичну близькість між словами та фразами.

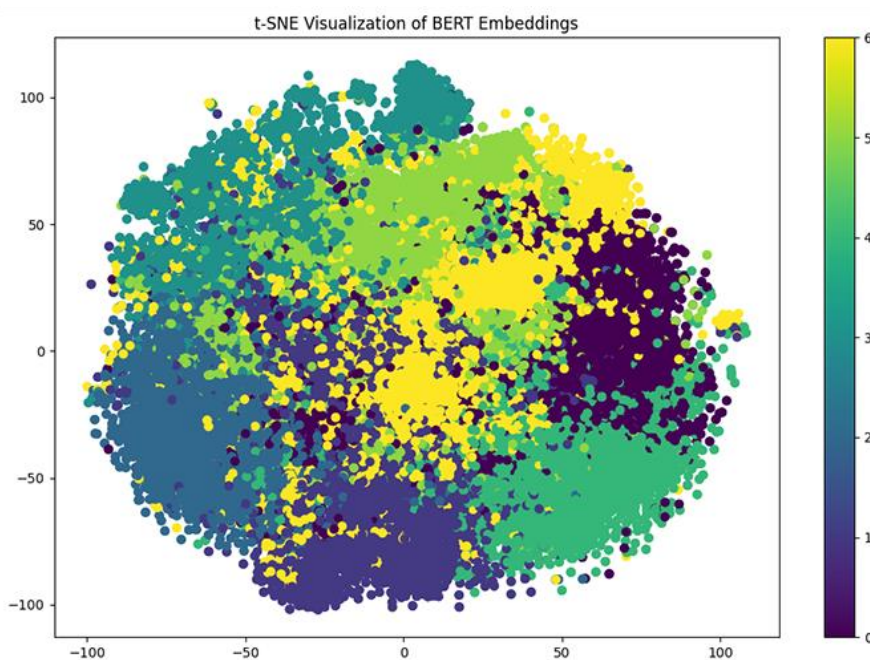


Рисунок 2.3 - Приклад векторного простору ембедингів текстових термінів

З метою підвищення якості отриманих ключових слів у процесі обробки використано алгоритм Maximal Marginal Relevance (MMR) [20], який забезпечує баланс між релевантністю ключових слів та їх різноманіттям.

$$MMR = \arg \max_{d_i \in D \setminus S} [\lambda \times Sim_1(d_i, Q) - (1 - \lambda) \times \max_{d_j \in S} Sim_2(d_i, d_j)],$$

де:

- D – множина кандидатів ключових слів;
- S – множина вже вибраних слів;
- Q – вихідний текст;
- $\lambda \in [0,1]$ – коефіцієнт різноманіття (*diversity*);

- Sim – функція семантичної подібності.

```
raw_keywords = kw_model.extract_keywords(
    text,
    keyphrase_ngram_range=(1, 1),
    stop_words=list(UK_STOPWORDS),
    top_n=top_n * 3,
    use_mmr=True,
    diversity=diversity,
)
```

У реалізації алгоритму MMR значення параметра `diversity` передається безпосередньо до функції витягування ключових слів, що дозволяє динамічно адаптувати результати залежно від типу та структури тексту.

Для обробки українськомовних текстів застосовано список стоп-слів української мови, морфологічний аналіз і лематизацію на основі бібліотеки `SpaCy` з мовною моделлю `uk_core_news_md`, а також додаткові правила очищення та нормалізації ключових слів.

Використання лематизації [16] дозволяє звести слова до їх початкової форми, що зменшує кількість дублювань та підвищує семантичну узгодженість ключових термінів.

Процес визначення ключових слів реалізовано у вигляді асинхронного механізму, що дозволяє виконувати обчислення у фоновому режимі без блокування основного пункту виконання застосунку. Для цього використано пул потоків, у межах якого здійснюється виклик синхронної функції аналізу тексту.

```
async def extract_keywords(
    text: str, top_n: int, diversity: float
) -> List[Tuple[str, float]]:
    loop = asyncio.get_running_loop()
    return await loop.run_in_executor(
        executor, lambda: _extract_keywords_sync(text, top_n, diversity)
    )
```

Кількість ключових слів, що підлягають витягуванню, визначаються динамічно на основі обсягу тексту. Це дозволяє уникнути як недостатньої, так і надмірної кількості ключових термінів при аналізі документів різного розміру.

$$N = \min \left(\max \left(N_{base}, \left\lceil \frac{w}{50} \right\rceil \right), N_{max} \right),$$

де:

- W – кількість слів у тексті;
- N_{base} – базова кількість ключових слів;
- N_{max} – максимальна допустима кількість ключових слів;
- N – кінцева кількість ключових слів.

```
def determine_top_n(text: str, base_n: int = 10, max_n: int = 20) -> int:  
    word_count = len(text.split())  
    return min(max(base_n, word_count // 50), max_n)
```

Запропонований механізм забезпечує масштабованість алгоритму витягування ключових слів та робить систему стійкою до змін обсягу вхідних даних. Завдяки цьому результати аналізу залишаються збалансованими незалежно від розміру документа.

Додатково реалізовано етап очищення результатів, який включає усунення повторів, видалення некоректних або нерелевантних слів, а також приведення ключових слів до лематизованої форми.

```
cleaned = []  
seen = set()  
for phrase, score in raw_keywords:  
    cleaned_phrase = clean_keyword(phrase)  
    if cleaned_phrase and cleaned_phrase not in seen:  
        cleaned.append((cleaned_phrase, score))  
        seen.add(cleaned_phrase)  
    if len(cleaned) >= top_n:  
        break
```

Такий підхід забезпечує формування більш інформативного та однорідного набору ключових слів, які надалі використовуються як вхідні дані для генерації конспекту.

У межах розробленої інтелектуальної системи формування конспектів запропоновано метод адаптивного визначення кількості ключових слів на основі семантичної структури тексту (Semantic Top-N). На відміну від традиційних підходів, у яких кількість слів задається фіксованим параметром або залежить виключно від довжини тексту, запропонований метод використовує семантичні ембединги речень та кластеризацію, що дозволяє враховувати внутрішню тематичну різноманітність документа.

Сутність полягає в тому, що кількість ключових слів визначається динамічно, залежно від кількості семантично окремих груп за змістом у тексті. Це забезпечує більш репрезентативне покриття змісту документа та зменшує ризик втрати важливих понять у великих або структурно складних текстах.

Нехай маємо текст, який після попередньої обробки представлений у вигляді множини векторів ембедингів речень:

$$E = \{\vec{s_1}, \vec{s_2}, \dots, \vec{s_N}\},$$

де N – кількість речень у тексті.

Кількість семантичних кластерів визначається як:

$$K = \max(3, \lfloor \sqrt{N} \rfloor)$$

Для кожного кластера виділяється фіксована кількість ключових слів k_b , після чого рекомендована кількість ключових слів обчислюється як:

$$N_{semantic} = K \times k_b$$

Остаточне значення обмежується мінімальним та максимальним порогами:

$$N_{final} = \min(\max(N_{semantic}, N_{min}), N_{max})$$

Реалізацію даного алгоритму винесено в Додаток А.

Реалізований метод інтегровано у сервіс тексту з можливістю перемикання між фіксованими та семантично адаптивним режимами. Це дозволяє порівнювати результати та забезпечує гнучкість налаштувань системи.

```
N_recommended = determine_top_n_semantic(sentence_embeddings)
```

```
if use_semantic_n:
    final_n = N_recommended
else:
    final_n = top_n
```

Таким чином, користувач або система можуть обрати режим, у якому кількість ключових слів визначається автоматично на основі семантичної структури тексту.

Наступний підхід, який запропоновано для даної системи – це каскадний підхід до фільтрації та уточнення ключових слів, який поєднує статично-семантичні методи аналізу тексту з можливостями великих мовних моделей (LLM) [30].

На відміну від класичних систем витягування ключових слів, де результат формується виключно на основі векторних подань або частотних характеристик, у даній роботі реалізовано двоетапну модель обробки, у якій LLM виступає як інтелектуальний семантичний фільтр.

Запропонований підхід складається з двох послідовних етапів: автоматичного витягування кандидатів ключових слів за допомогою алгоритму KeyBERT [36] з використанням Maximal Marginal Relevance (MMR) [20], а також, контекстно-орієнтованого уточнення та ранжування ключових слів із застосуванням LLM, яка враховує тему тексту та семантичні зв'язки між термінами.

На першому етапі формується початковий список ключових слів, який містить семантично релевантні, але потенційно шумні або надлишкові терміни.

Алгоритм MMR забезпечує баланс між релевантністю та різноманіттям ключових слів, однак не гарантує їхньої повної відповідності загальній темі документа.

Таким чином, результат першого етапу розглядається як множина кандидатів:

$$K_{init} = \{k_1, k_2, \dots, k_n\}$$

Другий етап є основним джерелом наукової новизни. Він полягає у використанні LLM для семантичного відбору перерозподілу та нормалізації ключових слів без зміни їх кількості.

Мовна модель отримує тему тексту, початковий список ключових слів та точну кількість ключових термінів, яка має бути збережена.

На основі цього LLM відбирає найбільш змістовні та тематично релевантні ключові слова, виводить їх першими та зберігає загальну кількість ключових слів без додавання нових.

Реалізацію винесено в Додаток Б.

LLM використовується в детермінованому режимі ($temperature = 0,0$), що забезпечує відтворюваність результатів та стабільність системи.

Каскадна фільтрація органічно інтегрована у загальний блок обробки тексту після етапу витягування ключових слів:

```
validated_keywords = await llm_keyword_refinement(  
    title=text_entity.title,  
    initial_keywords=keywords_only,  
    final_n=final_n,  
)
```

Отриманий список ключових слів надалі використовується як вхідні дані для генерації конспекту та як семантична основа для формування стислого й структурованого підсумкового тексту.

Останній підхід, який було запропоновано – це метод інтелектуальної адаптації параметрів генерації великих мовних моделей (LLM), який дозволяє динамічно керувати обсягом, структурою та деталізацією сформованого конспекту залежно від характеристик вхідного тексту.

На відміну від традиційних підходів, у яких параметри генерації (довжина тексту, кількість токенів, кількість речень) задаються статично або емпірично, у розробленій системі вони обчислюються автоматично на основі кількості слів у вихідному документі.

Це дозволяє забезпечити стабільну якість конспектів для текстів різного обсягу – від коротких статей до повноцінних лекційних матеріалів.

Нехай:

- W – кількість слів у вхідному тексті;
- L_{opt} – оптимальна довжина summary (у словах);

- R_{opt} – оптимальна кількість речень;
- T_{opt} – оптимальна кількість токенів для LLM.

Оптимальна довжина конспекту визначається за допомогою коефіцієнта стиснення:

$$L_{opt} = \min (\max (\lfloor W \times \alpha \rfloor, L_{min}), L_{max}),$$

де:

- $\alpha = 0.08$ – коефіцієнт стиснення;
- $L_{min} = 60$;
- $L_{max} = 180$.

Кількість речень визначається пропорційно до довжини summary:

$$R_{opt} = \min (\max (\left\lceil \frac{L_{opt}}{15} \right\rceil, R_{min}), R_{max}),$$

де:

- $R_{min} = 5$;
- $R_{max} = 12$.

Для уникнення передчасного завершення генерації або втрати контексту вводиться оцінка кількості токенів:

$$T_{opt} = \max (\lfloor L_{opt} \times \beta \rfloor, T_{min}),$$

де:

- $\beta = 1.3$ – середнє співвідношення слів до токенів;
- $T_{min} = 700$.

Реалізацію даного алгоритму винесено в Додаток В.

Обчислені параметри використовуються безпосередньо при формуванні prompt та налаштуванні параметрів LLM:

```
R_opt, L_opt, T_opt = calculate_optimal_summary_length(text_word_count)
```

та передаються у текстову інструкцію для LLM і в параметр num_predict, що обмежує кількість токенів під час генерації результату.

```
options={"temperature": 0.1, "num_predict": T_opt},
```

Інтелектуальна адаптація параметрів LLM є фінальним етапом обробки тексту та логічно поєднується з адаптивним вибором кількості ключових слів і каскадною фільтрацією ключових термінів за допомогою LLM.

Це дозволяє сформувати структурований, збалансований та контекстно-узгоджений конспект, який не перевантажений інформацією, не є поверхневим і відповідає обсягу вихідного матеріалу.

Після завершення етапів витягування та уточнення ключових слів здійснюється фінальний етап формування конспекту, який реалізовано із використанням великої мовної моделі. Генерація підсумку (summary) відбувається на основі поєднання теми документа, відібраних ключових слів та адаптивно обчислених параметрів множини.

Ключові слова виступають як семантичні орієнтири, які визначають основні ключову ідею майбутнього конспекту. На відміну від прямого скорочення тексту, запропонований підхід ґрунтується на контрольованій генерації, де мовна модель відновлює логічні зв'язки між ключовими поняттями та формує узагальнений виклад змісту.

Для підвищення якості результату в prompt передаються обмеження щодо кількості речень та орієнтованої довжини тексту. Це дозволяє уникнути надмірно коротких або, навпаки, перевантажених конспектів. Генерація здійснюється у вигляді кількох логічно пов'язаних абзаців, що імітують підсумковий виклад навчального або наукового матеріалу.

Такий підхід забезпечує формування зв'язного, інформативного та тематично узгодженого конспекту, який може бути використаний як підсумковий матеріал для навчання або повторення.

Повну реалізацію та прописаний prompt можна переглянути в Додатку Г.

2.3. Розробка алгоритму завантаження та обробки тексту із різних джерел

Для забезпечення універсальності та зручності використання системи було розроблено алгоритм завантаження та первинної обробки текстових даних із різних джерел. Користувач може передавати інформацію у вигляді файлів різних форматів, вебпосилань або безпосередньо текстового вводу. Такий підхід дозволяє використовувати систему для аналізу навчальних матеріалів, наукових статей, презентацій та інших типів документів незалежно від способу їх подання.

Загальна логіка роботи алгоритму завантаження та обробки тексту наведена у фрагменті коду нижче:

```
@router.post("/file", response_model=ExtractTextResponse, summary="Отримати текст з файлу")
async def process_file(
    file: UploadFile,
    session: AsyncSession = Depends(get_session),
):
    text = await extract_text_from_file(file)
    text_id = await get_or_save_text(session, text, "upload")

    return ExtractTextResponse(text_id)
```

Повний код можна переглянути в Додатку Г.

Розроблений алгоритм обробки тексту реалізовано у вигляді окремого сервісу, який відповідає за прийом вхідних даних, визначення їх типу, витягування текстового вмісту та збереження результатів у базі даних. На першому етапі система визначає джерело тексту (файл, URL або прямий ввід), після чого обирає відповідний механізм обробки. Отриманий текст нормалізується та передається на подальші етапи аналізу, зокрема для витягування ключових слів і генерації конспекту, які були описані в попередньому підрозділі.

Для роботи з файлами реалізовано механізм динамічного визначення формату на основі розширення файлу. Система підтримує обробку документів форматів DOC, DOCX, PDF, PPTX, TXT та CSV. Залежно від типу файлу використовується відповідна бібліотека для коректного витягування тексту. Зокрема, для форматів DOC і PPTX застосовується бібліотека Textract, для DOCX

– python-docx, для PDF – PyPDF2, для CSV – pandas. Такий підхід забезпечує коректне опрацювання структурованих і напівструктурованих даних та дозволяє уніфікувати текстовий результат незалежно від початкового формату документа. Вибір відповідного методу витягування тексту здійснюється на основі розширення файлу, який реалізовано у вигляді словника обробників.

```
FILE_EXTRACTORS: Dict[str, FileExtractor] = {
    ".doc": FileExtractor(extract_text_from_doc, "DOC"),
    ".docx": FileExtractor(extract_text_from_docx, "DOCX"),
    ".pptx": FileExtractor(extract_text_from_pptx, "PPTX"),
    ".pdf": FileExtractor(extract_text_from_pdf, "PDF"),
    ".txt": FileExtractor(extract_text_from_txt, "TXT"),
    ".csv": FileExtractor(extract_text_from_csv, "CSV"),
}
```

Повний код можна переглянути в Додатку Д.

Для аналізу тексту за URL адресою система виконує попереднє отримання вмісту сторінки та визначає тип ресурсу. У випадку, якщо посилання веде на PDF документ, здійснюється пряме витягування тексту з файлу. Якщо ж ресурс є HTML сторінкою, виконується очищення контенту від службових елементів, таких як скрипт, стилі та навігаційні блоки. Фрагмент коду нижче ілюструє логіку визначення типу ресурсу та вибору відповідного способу обробки.

```
if "application/pdf" in content_type or url.lower().endswith(".pdf"):
    return extract_text_from_url_pdf_content(response.content)

return extract_text_from_url_content(response.content)
```

Повний код можна переглянути в Додатку Д.

Додатково реалізовано механізм виявлення PDF документів, вбудованих у вебсторінки через JavaScript переглядачі, які дозволяють коректно обробляти навіть непрямі посилання на файли. Це підвищує надійність алгоритму та розширює спектр підтримуваних джерел.

Реалізація даного механізму наведена у фрагменті коду нижче:

```
def extract_text_from_pdf_js_viewer(raw_text: str) -> str:
    match = re.search(r"\.getDocument\(['\"](.*)['\"]\)", raw_text)

    if match:
        pdf_url = match.group(1)
        response = fetch_url(pdf_url)
```

```
return extract_text_from_url_pdf_content(response.content)

return raw_text
```

Повний код можна переглянути в Додатку Ж.

Окрім файлів та вебресурсів, система підтримує обробку тексту, введеного користувачем безпосередньо через інтерфейс. У цьому випадку текст одразу передається до модуля збереження без етапу витягування, що дозволяє мінімізувати затримки та спростити обробку коротких фрагментів інформації.

Після отримання текстового вмісту система перевіряє наявність ідентичного тексту в БД. У разі його відсутності створюється новий запис із автоматично згенерованою назвою документа, визначенням мови та типу джерела. Такий підхід дозволяє уникнути дублювання даних і забезпечує ефективне повторне використання вже оброблених текстів у подальших етапах аналізу.

Додатково перед збереженням тексту в базі даних реалізовано автоматичну генерацію заголовка документа із використанням мовної моделі (LLM). Заголовок формується на основі семантичного аналізу вмісту тексту та використовується для подальшої ідентифікації документа, а також як контекстна інформація на етапах аналізу та генерації конспекту.

Фрагмент коду, який реалізує генерацію заголовка тексту можна переглянути в Додатку К.

Запропонований алгоритм завантаження та обробки тексту забезпечує гнучку та масштабовану взаємодію з різними джерелами даних. Його модульна структура дозволяє легко розширювати перелік підтримуваних форматів та інтегрувати нові способи отримання інформації, що є важливим для подальшого розвитку системи.

Висновки до розділу 2

У другому розділі визначено функціональні можливості інтелектуальної системи автоматичного формування конспектів та основні сценарії взаємодії користувача із системою. Побудовано діаграму варіантів використання, які відображає ключові прецеденти роботи застосунку, а також спроектовано

структуру реляційної бази даних для збереження текстових матеріалів, результатів аналізу та параметрів обробки. Запропонована модель даних забезпечує цілісність інформації, зручний доступ до результатів і створює основу для масштабування системи.

У межах розділу описано реалізацію алгоритмів формування конспектів на основі комбінування методів обробки природної мови та мовних моделей. Розглянуто підхід до витягування ключових слів із використанням KeyBERT та алгоритму MMR, адаптивне визначення їх кількості на основі семантичної структури тексту, а також каскадний механізм уточнення ключових термінів та допомогою LLM. Запропоновано метод інтелектуальної адаптації параметрів генерації, що дозволяє формувати структуровані та збалансовані конспекти незалежно від обсягу вхідного документа.

Також, було розроблено алгоритм завантаження та первинної обробки тексту з різних джерел, зокрема файлів різних форматів, вебресурсів і прямого введення користувачем. Реалізовані механізми витягування тексту, виявлення PDF документів, у тому числі вбудованих у вебсторінки, та автоматичної генерації заголовків підвищують універсальність і надійність системи. Отримані результати забезпечують практичну готовність розробленої системи.

РОЗДІЛ 3. ОПИС ПРОГРАМНОГО ПРОДУКТУ

3.1. Графічний інтерфейс

Графічний інтерфейс програмного продукту розроблено з урахуванням принципів зручності використання, мінімалізму та орієнтації на кінцевого користувача. Основною метою інтерфейсу є забезпечення інтуїтивного доступу до функціоналу завантаження текстів, перегляду результатів аналізу та взаємодії з інтелектуальними сервісами системи без необхідності глибоких технічних знань. Інтерфейс реалізує логічну послідовність дій користувача та відображає ключові етапи роботи системи у зрозумілому візуальному вигляді.

Головний екран інтерфейсу (рис. 3.1) містить меню вибору (tabs) для завантаження тексту до системи. Користувач може обрати один із доступних способів введення: завантаження файлу, введення посилання на вебресурс або безпосереднє введення тексту у текстове поле. Інтерфейс забезпечує візуальну індикацію обраного способу та повідомляє користувача про підтримувані формати файлів. Після завантаження, користувач отримує повідомлення про успішне прийняття тексту до обробки (рис. 3.2). В разі помилки, виводить повідомлення про те, що не вдалося обробити файл, вебресурс або текст введений вручну (рис. 3.3).

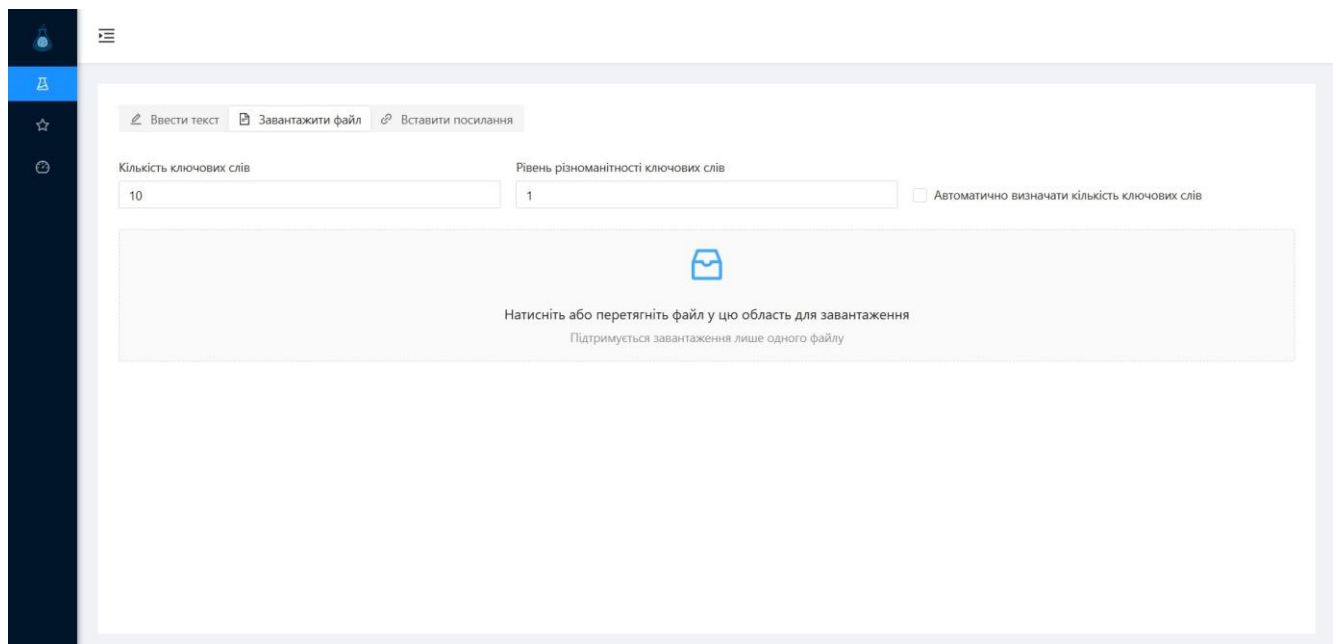


Рисунок 3.1 - Головний екран

✓ Файл "Груницький Д.С. - Теза_на_тему_Інтеграція_чат-ботів.docx" успішно завантажено.

Рисунок 3.2 - Повідомлення про успішне завантаження файлу

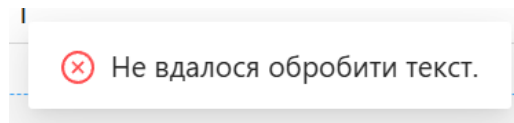


Рисунок 3.3 - Повідомлення про невдале оброблення тексту

Перед завантаженням тексту користувачеві потрібно ввести параметри аналізу. У графічному інтерфейсі це реалізовано у вигляді звичайних полів (рис. 3.4), які дозволяють обрати кількість ключових слів та рівень різноманітності результатів. Також, користувач може активувати інтелектуальний режим автоматичного визначення кількості ключових слів (див. рис. 3.4). Інтерфейс чітко відображає поточні значення параметрів перед запуском аналізу.

Кількість ключових слів	Рівень різноманітності ключових слів	☐ Автоматично визначати кількість ключових слів
10	1	



Натисніть або перетягніть файл у цю область для завантаження
Підтримується завантаження лише одного файлу

Рисунок 3.4 - Поля для введення параметрів

Після завершення обробки тексту користувач переходить до іншої частини екрану результатів (рис. 3.5), де відображаються згенеровані ключові слова та підсумковий конспект. Ключові слова подаються у структурованому вигляді, що дозволяє швидко оцінити основні теми документа. Текст підсумку відображається у зручному для читання форматі з логічним поділом на абзаци, що полегшує сприйняття результатів аналізу. Користувач має можливість взаємодіяти з отриманими результатами, зокрема виконувати повторний аналіз або зберегти сформований конспект для подальшого використання. Такий підхід забезпечує зручність роботи з системою та підвищує її практичну цінність у навчальній і професійній діяльності.

Ключові слова

ISO/IEC 27001:2022

Інформаційна безпека

Управління

Сертифікація

Кибербезпека

Цифровізація

Вразливості

Контекст

Довіра

Вимоги

Короткий зміст

Управління інформаційною безпекою є ключовим аспектом сучасних бізнесових операцій у цифровій епохи. ISO/IEC 27001:2022 визначає міжнародні стандарти для управління інформаційною безпекою, які дозволяють організаціям забезпечити контроль над вразливостями та збереження довіри клієнтів. Цей стандарт надає рекомендації щодо управління інформаційною безпекою, включаючи оцінку ризиків, визначення поточних вразливостей та розробку стратегій для їхнього зниження. Він також передбачає процедури сертифікації, які дозволяють організаціям продемонструвати свою прихильність до вимог інформаційної безпеки та отримати міжнародне визнання. Кибербезпека є одним із головних викликів для сучасних організацій, і ISO/IEC 27001:2022 надає корисну базу для її захисту. Він дозволяє організаціям врахувати контекст своєї діяльності та визначити поточні ризики, щоб створити ефективну систему управління інформаційною безпекою. Управління інформаційною безпекою є невід'ємним компонентом цифрової трансформації та дозволяє організаціям зберегти контроль над своїми даними, захистити їх від вторгнення та зберігати довіру клієнтів. ISO/IEC 27001:2022 є важливим інструментом для цього, і його рекомендації можуть бути корисними для організацій будь-якого розміру та сектору економіки.

Рисунок 3.5 - Результат обробки тексту

Інтерфейс системи передбачає можливість перегляду списку раніше оброблених документів (рис. 3.6). Користувач може обрати будь-який елемент зі списку та перейти до детального перегляду результатів аналізу для відповідного тексту (рис. 3.7). Для зручності навігації відображається коротка інформація про текст, зокрема його заголовок, дата обробки та обрані параметри (див. рис. 3.6).

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА» Кафедра інженерії програмного за...	14.12.2025 21:12
Методи та технології персоналізованого навчання	
Кількість: 10 Рівноманітність: 0.35 Автопідбір: ні	
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА» Кафедра інженерії програмного за...	13.12.2025 20:09
Методи та технології персоналізованого навчання	
Кількість: 13 Рівноманітність: 0.6 Автопідбір: ні	
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ Державний університет «Житомирська політехніка» Кафедра інженерії програмного забезпечення ...	13.12.2025 18:59
Управління інформаційною безпекою згідно ISO/IEC 27001:2022	
Кількість: 10 Рівноманітність: 1 Автопідбір: ні	
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА» Кафедра інженерії програмного за...	13.12.2025 18:47
Методи та технології персоналізованого навчання	
Кількість: 10 Рівноманітність: 1 Автопідбір: ні	

Рисунок 3.6 - Вкладка «Збережене»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА» Кафедра інженерії програмного забезпечення РЕФЕРАТ на тему: «ISO/IEC 27001:2022 – Система управління інформаційною безпекою» Житомир – 2025 ВСТУП В умовах цифровізації суспільства та активного використання інформаційних технологій питання безпеки даних набуває особливої актуальності. Кибератаки, витоки конфіденційної інформації та несанкціонований доступ до даних заважають значних фінансових та репутаційних...

Показати більше

Управління інформаційною безпекою згідно ISO/IEC 27001:2022

13.12.2025 18:59

Кількість ключових слів: 10
Рівень різноманітності ключових слів: 1
Автоматично визначати кількість ключових слів: ні

Ключові слова

ISO/IEC 27001:2022 Інформаційна безпека Управління Сертифікація Кибербезпека Цифровізація Вразливості Контекст Довіра Вимоги

Короткий зміст

Управління інформаційною безпекою є ключовим аспектом сучасних бізнесових операцій у цифровій епохи. ISO/IEC 27001:2022 визначає міжнародні стандарти для управління інформаційною безпекою, які дозволяють організаціям забезпечити контроль над вразливостями та збереження довіри клієнтів. Цей стандарт надає рекомендації щодо управління інформаційною безпекою, включаючи оцінку ризиків, визначення поточних вразливостей та розробку стратегій для їхнього зниження. Він також передбачає процедури сертифікації, які дозволяють організаціям продемонструвати свою прихильність до вимог інформаційної безпеки та отримати міжнародне визнання. Кибербезпека є одним із головних викликів для сучасних організацій, і ISO/IEC 27001:2022 надає корисну базу для її захисту. Він дозволяє організаціям врахувати контекст своєї діяльності та визначити поточні ризики, щоб створити ефективну систему управління інформаційною безпекою. Управління інформаційною безпекою є невід'ємним компонентом цифрової трансформації та дозволяє організаціям зберегти контроль над своїми даними, захистити їх від вторгнення та зберігати довіру клієнтів. ISO/IEC 27001:2022 є важливим інструментом для цього, і його рекомендації можуть бути корисними для організацій будь-якого розміру та сектору економіки.

Текст

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ Державний університет «Житомирська політехніка» Кафедра інженерії програмного забезпечення РЕФЕРАТ на тему: «ISO/IEC 27001:2022 – Система управління інформаційною безпекою» Житомир – 2025 ВСТУП В умовах цифровізації суспільства та активного використання інформаційних технологій питання безпеки даних набуває особливої актуальності. Кибератаки, витоки конфіденційної інформації та несанкціонований доступ до даних заважають значних фінансових та репутаційних...

Показати більше

Рисунок 3.7 - Детальний перегляд одного документа

Для підвищення гнучкості роботи з системою графічний інтерфейс дозволяє повторно запускати аналіз одного й того самого тексту з іншими параметрами. Користувач може змінити налаштування та ініціювати нову обробку без необхідності повторного завантаження документа. Це забезпечує зручний експериментальний режим використання системи.

Графічний інтерфейс програмного продукту передбачає можливість експорту сформованого конспекту у форматі PDF. Після завершення аналізу тексту та відображення результатів користувач має змогу в «Збережених» (див. рис. 3.7) відкрити текст, який оброблявся та завантажити його натиснувши на кнопку «Завантажити pdf», яка ініціює процес формування файлу (рис. 3.8). Система автоматично створює документ формату PDF на основі збереженого конспекту, зберігаючи логічну структуру тексту та зручний для читання формат. Після завершення генерації, файл завантажується на пристрій користувача (рис. 3.9), що дозволяє використовувати результат у автономному режимі або для подальшого поширення.

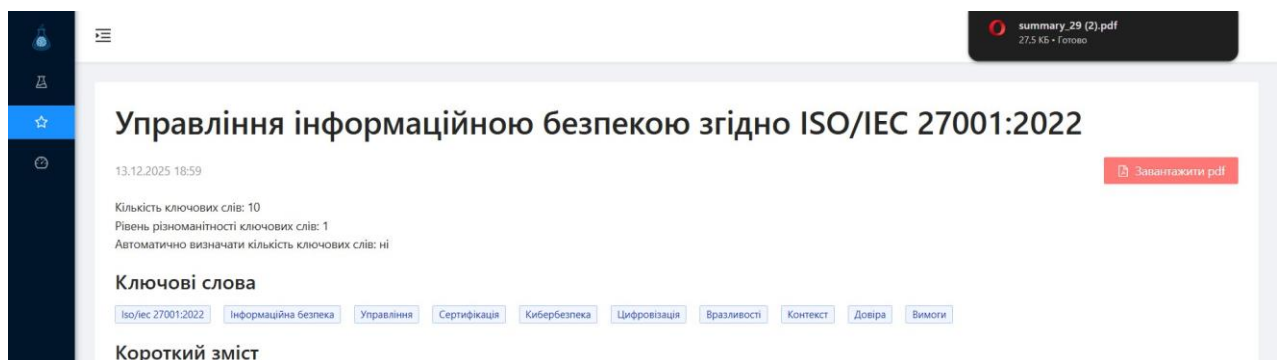


Рисунок 3.8 - Детальний перегляд документа

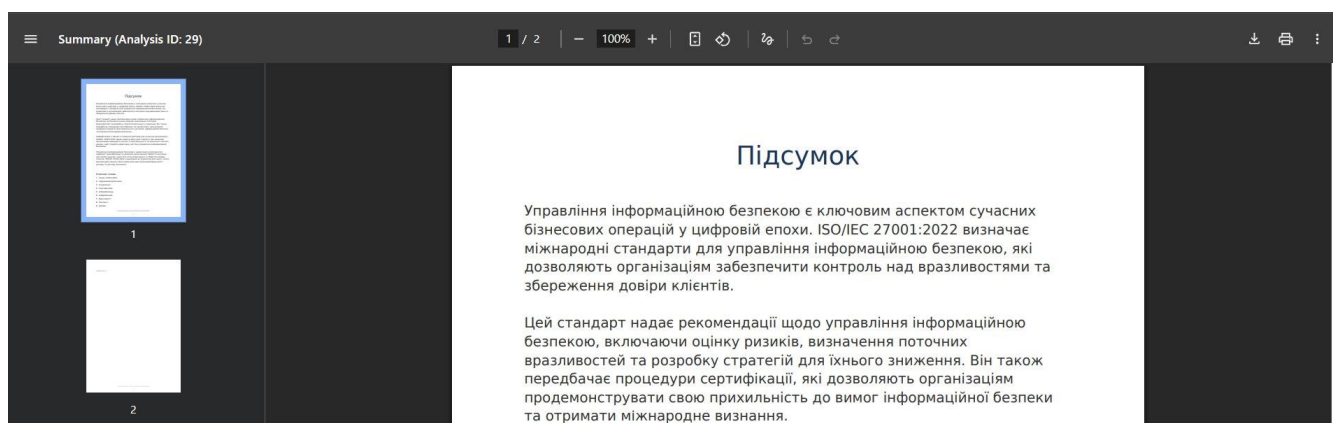


Рисунок 3.9 - Завантажений файл

Також, крім перегляду результатів окремих документів, графічний інтерфейс системи надає користувачеві доступ до узагальненої аналітики у вигляді графіків (рис. 3.10). У даному розділі відображаються статичні дані за всіма раніше обробленими текстами, що дозволяє оцінити ефективність роботи системи та проаналізувати вплив обраних параметрів на результати обробки. Зокрема, інтерфейс забезпечує візуалізацію порівняння кількості згенерованих ключових слів для різних документів, залежності швидкості аналізу від значення параметра різноманіття (diversity), а також часу виконання аналізу кожного тексту. Представлення даних у графічному вигляді спрощує інтерпретацію результатів та сприяє більш усвідомленому налаштуванню параметрів аналізу при подальшому використанні системи.



Рисунок 3.10 - Вкладка «Dashboard»

3.2. Налаштування параметрів програмного продукту та відображення результатів аналізу

У межах даного підрозділу розглядається функціональність програмного продукту, пов'язана з керуванням параметрами аналізу тексту та візуальним представленням отриманих результатів. Інтерфейс системи орієнтований на гнучке

налаштування процесу формування ключових слів і конспекту, що дозволяє адаптувати результати під різні типи текстів та індивідуальні потреби користувача.

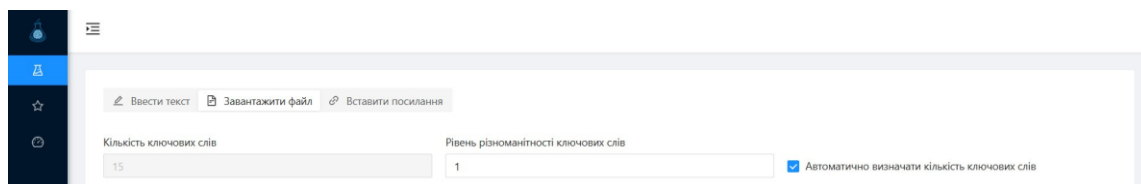
Перед запуском аналізу користувач має можливість задати основні параметри обробки тексту. До таких параметрів належить кількість ключових слів, коефіцієнт різноманіття (diversity), а також режим використання адаптивного семантичного підбору кількості ключових слів (Semantic Top-N) (рис. 3.11).



The image shows two input fields for configuring the analysis parameters. The first field is labeled 'Кількість ключових слів' (Number of key words) and contains the value '15'. The second field is labeled 'Рівень різноманітності ключових слів' (Diversity level of key words) and contains the value '0.7'.

Рисунок 3.11 - Налаштування параметрів (1)

Вибір останнього параметра дозволяє системі автоматично визначити оптимальну кількість ключових термінів на основі семантичної структури тексту, зменшуючи ризик отримання надмірно узагальнених або, навпаки, розрізнених результатів (рис. 3.12).



The image shows a screenshot of the application interface. On the left is a dark sidebar with navigation icons. The main area has a light blue header with three buttons: 'Ввести текст' (Enter text), 'Завантажити файл' (Upload file), and 'Вставити посилання' (Paste link). Below the header, there are two input fields: 'Кількість ключових слів' (Number of key words) with the value '15' and 'Рівень різноманітності ключових слів' (Diversity level of key words) with the value '1'. To the right of these fields is a checkbox labeled 'Автоматично визначати кількість ключових слів' (Automatically determine the number of key words), which is currently checked.

Рисунок 3.12 - Налаштування параметрів (2)

Після підтвердження параметрів система запускає процес аналізу, який виконується у фоновому режимі. Це забезпечує безперервну роботу інтерфейсу та дозволяє користувачу відстежувати статус обробки без блокування основних елементів керування. Після завершення аналізу результати автоматично відображаються у відповідному блоці.

Результати аналізу подаються у структурованому вигляді та включають сформований список ключових слів і згенерований конспект тексту (рис. 3.13 та рис. 3.14). Ключові слова відображаються у впорядкованому вигляді відповідно до їхньої семантичної важливості, що полегшує швидке ознайомлення з основною тематикою документа. Згенерований конспект подається у вигляді зв'язного тексту, одним абзацом, що забезпечує зручність читання та сприйняття інформації.

Ключові слова

Система Персоналізовані Конспекти Мікросервісна Архітектура Викладач Користувач Стаття Новий Політехніка Адаптивний Процес Успішність Час Технологія

Короткий зміст

У цій статті описується нова система створення персоналізованих конспектів на основі мікросервісної архітектури. Ця технологія дозволяє викладачам та користувачам адаптивно формувати конспекти, які відповідають їхнім потребам і стилю навчання. Мікросервісна архітектура дозволяє розділити систему на окремі служби, кожна з яких виконує певну функцію. Це дає можливість швидко реагувати на зміни та покращувати процес створення конспектів. У результаті користувач отримує персоналізований конспект, який допомагає успішно запам'ятати ключові моменти лекції або статті. Це може значно зменшити час, необхідний для підготовки до екзамену чи написання роботи. Використання мікросервісної архітектури в системах створення конспектів є новою та перспективною технологією, яка може значно покращити процес навчання та допомогти користувачам ефективніше запам'ятати ключові моменти.

Рисунок 3.13 - Результат з заданими параметрами (1)

Ключові слова

Система Персоналізовані конспекти Мікросервісна архітектура Створення Користувач Адаптувати Алгоритм Час Стаття Житомирський Безкоштовний Залежність

Короткий зміст

У цій статті описана система створення персоналізованих конспектів на основі мікросервісної архітектури. Ця безкоштовна платформа дозволяє користувачам адаптувати алгоритми для оптимізації процесу створення конспектів, зменшуючи час на запис та обробку інформації. Житомирський розробник представив цю систему як варіант покращення ефективності навчання та роботи. Мікросервісна архітектура дозволяє розділити систему на окремі частини, кожна з яких відповідає за певну функцію. Це дозволяє зменшити залежність між компонентами та підвищити швидкість роботи системи в цілому. У результаті, користувач може створювати персоналізовані конспекти швидко і ефективно, адаптуючи алгоритми до своїх потреб та стилю навчання. Це дозволяє зменшити час на підготовку до занять та зробити процес навчання більш привабливим і ефективним.

Рисунок 3.14 - Результат з заданими параметрами (2)

Є можливість їх подальшого використання, зокрема завантаження сформованого конспекту у форматі PDF. Таким чином, підрозділ налаштувань і відображення результатів забезпечує повний цикл взаємодії користувача з програмним продуктом – від керування параметрами аналізу до отримання готового, зручного для використання результату.

Крім налаштування параметрів аналізу, інтерфейс системи підтримує обробку текстових даних із різних джерел та у різних форматах. Нижче додатково продемонстровано загрузку таких форматів як PDF та PPTX їх обробку та результат (рис. 3.15 та рис. 3.16).

Ввести текст

Завантажити файл

Вставити посилання

Кількість ключових слів

Рівень різноманітності ключових слів

15

1

☒ Автоматично визначати кількість ключових слів

Натисніть або перетягніть файл у цю область для завантаження

Підтримується завантаження лише одного файлу

Ключові слова

Мікросервісна система Персоналізовані конспекти Технологія Викладач Користувач Адаптувати Архітектура Новий Безкоштовний Залежність Стаття Алгоритм Час

Короткий зміст

У цій статті описується нова мікросервісна система для створення персоналізованих конспектів. Ця технологія призначена для адаптації до потреб користувача та викладача, надаючи можливість створювати унікальні записи навчального матеріалу. Мікросервісна архітектура системи дозволяє безкоштовно використовувати її та зменшує залежність від окремих серверів, що робить її ефективною та легкою для впровадження. Алгоритм створення конспектів базується на аналізі часу витраченого користувачем на різні секції матеріалу, щоб забезпечити оптимальне представлення інформації. Це дозволяє зменшити час, який потрібно для підготовки конспектів та зробити їх більш ефективними для навчання. У результаті, користувач має можливість створювати персоналізовані конспекти швидко і безкоштовно, а викладач може забезпечити оптимальне представлення інформації для своїх студентів. Ця система є новою тандемом у сфері навчання та допомагає зменшити час, який потрібно для підготовки конспектів та зробити їх більш ефективними для навчання.

Рисунок 3.15 - Результат обробки тексту з формату файлу PDF

Вести текст
Завантажити файл
Вставити посилання

Кількість ключових слів
17

Рівень різноманітності ключових слів
0.9

☐ Автоматично визначати кількість ключових слів

Натисніть або перетягніть файл у цю область для завантаження
Підтримується завантаження лише одного файлу

Ключові слова

Управління навчальним процесом (викладацька діяльність)
Дидактика
Педагогіка
Компетентність
Структурний метод
Теорія
Відмінність
Читання
Соціальний
Етап
Самореалізація

Новатор
Ютеризація
Система
Фізіолог
Мозковий
Традиційний (вихідні дані: 17 ключові слова та фрази)

Короткий зміст

Управління навчальним процесом (викладацька діяльність) є ключовою складовою педагогіки та дидактики, які визначають методи і стратегії, що застосовуються для ефективного навчання. Компетентність викладача є основним фактором успіху навчального процесу, тому необхідно розвивати його знання та навички. Структурний метод є однією з найпопулярніших форм управління навчальним процесом. Він передбачає планування уроків, визначення цілей і задач, а також контролю результатів. Теорія управління навчальним процесом допомагає викладачеві розуміти принципи та методи, що використовуються для ефективного керівництва навчальною діяльністю. Відмінність між традиційним і сучасним підходом до управління навчальним процесом полягає в застосуванні інноваційних методів та технологій, які сприяють активному участі учнів у навчанні. Це включає використання комп'ютерів, інтерактивних дощок і онлайнів-платформ для створення динамічних уроків та заохочення самореалізації учнів. Читання є однією з основних складових навчального процесу, тому необхідно розвивати компетентність викладача в цьому напрямку. Це включає знання теорії читання та методик його навчання, а також здатність створювати інтерес до читання учнів. Самореалізація є ключовим елементом сучасного підходу до управління навчальним процесом. Вона передбачає активне участі учнів у плануванні та виконанні навчального процесу, а також розвиток їхньої самостійності та ініціативи. Новаторство є важливою складовою педагогіки та дидактики, тому необхідно стимулювати викладачів до пошуку нових методів та технологій для ефективного навчання. Це включає використання інноваційних підходів до планування уроків, розробки нових методик навчання та застосування сучасних технологій. Ютеризація є однією з найпопулярніших

Рисунок 3.16 - Результат обробки тексту з формату файлу RPTX

Окрім завантаження файлів, програмний продукт забезпечує можливість безпосередньо введення тексту у спеціальне текстове поле, що є зручним для швидкого аналізу коротких фрагментів або чернеткових матеріалів (рис. 3.17 та рис. 3.18).

Вести текст
Завантажити файл
Вставити посилання

Кількість ключових слів
9

Рівень різноманітності ключових слів
1

☐ Автоматично визначати кількість ключових слів

Ярослав Мудрий роки правління – 1019-1054
Ярослав народився приблизно 982 року, був другим сином князя Володимира Святославича з династії Рюриковичів від полоцької княгині **Рогнеди**. Наприкінці правління Володимира Великого 1014 року, Ярослав відмовився сплачувати щорічну данину Києву, що становила 2000 гривень, і виступив проти свого батька походом. Під час приготування до походу, Володимир помер.
Після смерті свого батька Володимира 26 листопада (день **св. Георгія**) 1015 року Ярослав Мудрий в жорстокій боротьбі за київський престол розбив війська Святополка біля Любеча і посів київський престол, але 1018 року під натиском польських військ короля Болеслава I, яких взяв собі на допомогу Святополк, мусив покинути Київ і утік до Новгорода. Після остаточної перемоги в битві над річкою Альтою 1019 року Ярослав Мудрий став київським Великим князем.
Прагнучи об'єднати всі руські землі під своєю владою, Ярослав Володимирович вів боротьбу проти свого брата Мстислава Володимировича, князя **тмутороканського** і чернігівського. Після битви, яку Ярослав програв, під Лиственом біля Чернігова 1024 року, Ярослав мусив відступити Мстиславові Чернігівщину і всі землі на схід від Дніпра, крім Переяславщини. Згодом, після укладеного 1026 року у Городку під Києвом миру, почалося порозуміння і співпраця між братами. Ярослав Мудрий допомагав Мстиславові у боротьбі **засогоми** і **ясами** у 1029 році, поширивши свої володіння до Кавказьких гір; а Мстислав — у скріпленні й поширенні держави Ярослава Мудрого на захід від Дніпра.
1030 року на півночі Ярослав зайняв землі між Чудським озером і Балтикою і там заснував м. Юр'їв (нині — Тарту), у 1030 — 1031 роках війська Ярослава і Мстислава відвоювали **Черевенські** городи, які 1018 року захопив Болеслав I. Тоді ж Ярослав здобув від королівства Польського смугу землі між ріками Сяном і Бугом. За відомостями, що містяться у "Повісті **врем'яних** літ", він в 1030 році відвоював місто Белз, а в 1031 заклав місто свого імені Ярослав.
1036 — після смерті Мстислава Ярослав стає єдиним володарем Київської держави, окрім Полоцького князівства, яке виділено Володимиром Святим в уділ роду Ізяслава. 1036 року, за літописом, Ярослав Мудрий розгромив біля Києва печенігів і нібито, за літописом, на місці перемоги над ними почав будувати у 1037 році Софійський собор. Хоча останні дослідження Софії Київської спростували це твердження літопису.
У 1038–1042 роках Ярослав вів успішні походи проти литовських племен — ятвягів, проти Мазовії, проти **прибалтицько-фінських племен** **ямь** і чудь. 1043 року він підготував під проводом свого сина Володимира і воєводи Вишати похід на Візантію, який закінчився поразкою, багато воїнів потрапило в полон чи загинуло. Щоб охороняти свою державу проти нападів кочовиків, Ярослав укріплював південний кордон, будуючи міста над ріками Россою і Трубежем: Корсунь, Канів, Переяслав; як також другу фортифікаційну лінію над Сулою: Лубни, **Лукомль**, **Войнь**.

Обробити текст

Рисунок 3.17 - Поданий текст для обробки

Ключові слова

Ярослав мудрий Об'єднання руської землі Князь Володимирович Смерть Собор Битва Руська земля

Короткий зміст

Ярослав Мудрий, відомий як князь Володимирович, є однією з найвизначніших фігур у історії Руської землі. Він став першим, хто спробував об'єднати всю Русь під своїм правлінням. Ярослав Мудрий народився у 978 році та почав правити у 1019 році після смерті батька Володимира Великого. Він був князем Новгород-Сіверським, а пізніше став великим князем Київським. Об'єднання Руської землі було складним процесом, який включав багато битв та соборів. Ярослав Мудрий успішно боровся проти половців на річці Стугні, а також завоював Переяславець та Чернігів. Він також організував Собор у Любечі, де було прийнято важливі релігійні та адміністративні постанови, які сприяли об'єднанню Руської землі. Під керівництвом Ярослава Мудрого Руська земля досягла свого найбільшого розквіту. Він був відомий як мудрий правитель, який надавав велику увагу освіті та культурі. Після смерті Ярослава Мудрого в 1054 році Руська земля почала розпадатися на окремі князівства, але його внесок у об'єднання країни залишився важливим етапом у її історії.

Рисунок 3.18 - Результат обробки поданого тексту

Також реалізовано режим обробки тексту за посиланням на вебресурс, у межах якого система здійснює автоматичне отримання та очищення текстового контенту зі сторінки перед виконанням аналізу (рис. 3.19 та 3.20). Такий підхід розширює спектр підтримуваних сценаріїв використання та підвищує універсальність програмного продукту.

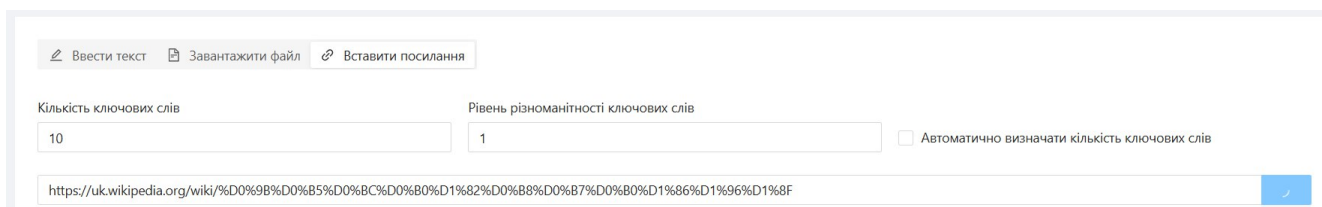


Рисунок 3.19 - Посилання на вебресурс

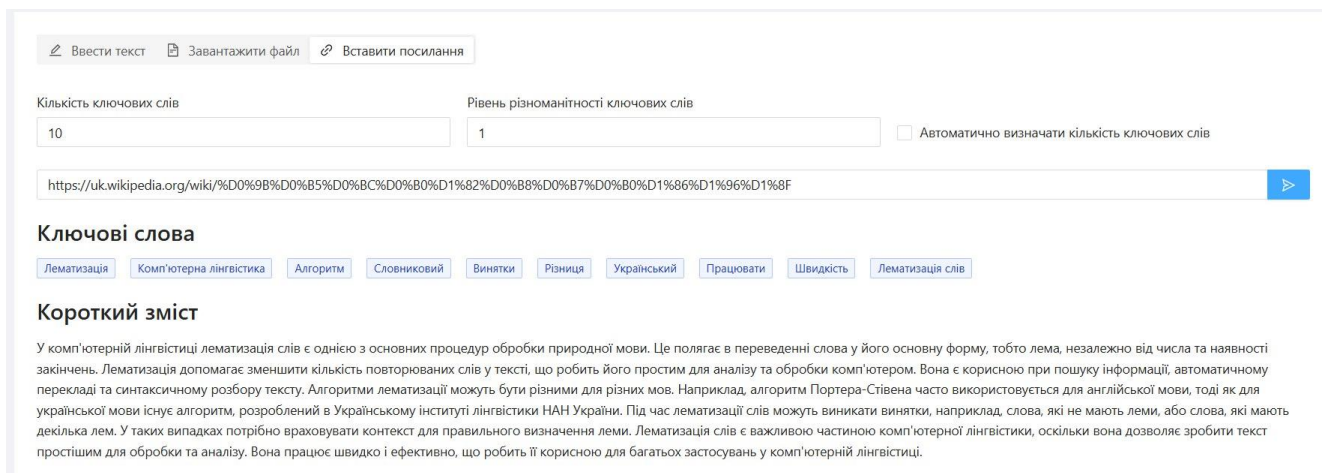


Рисунок 3.20 - Результат обробки даного посилання

Висновки до розділу 3

У третьому розділі детально описано програмний продукт з точки зору взаємодії користувача з системою та практичного використання її функціональних

можливостей. Особливу увагу приділено графічному інтерфейсу, який розроблено з урахуванням принципів зручності, логічної послідовності дії та мінімізації когнітивного навантаження на користувача. Інтерфейс забезпечує інтуїтивно зрозумілий процес завантаження текстів різними способами, налаштування параметрів аналізу, перегляд результатів та роботи з раніше обробленими документами. Реалізовані механізми повідомлень про успішні та помилкові дії, можливість повторного аналізу, збереження і їх експорту у форматі PDF підвищують зручність та ефективність використання системи у навчальній і професійній діяльності.

Також, у розділі розкрито підхід до налаштування параметрів аналізу та відображення результатів обробки тексту. Продемонстровано гнучкість системи щодо вибору кількості ключових слів, рівня різноманіття та використання адаптивного семантичного режиму, що дозволяє отримувати релевантні та збалансовані результати для текстів різної структури й обсягу. Структуроване подання ключових слів і конспекту, підтримка різних форматів вхідних даних, обробка текстів за посиланнями та наявність узагальненої аналітики у вигляді графіків підтверджують універсальність і практичну цінність розробленого програмного продукту. В цілому, описані рішення забезпечують повний цикл роботи з текстовими даними – від введення та налаштування параметрів до отримання зручного для подальшого використання результату.

ВИСНОВКИ

У результаті виконання роботи було проведено комплексне дослідження проблематики автоматичного формування конспектів та обґрунтовано доцільність створення інтелектуальної системи для скорочення й структурування текстової інформації. Аналіз предметної області та існуючих рішень показав, що наявні інструменти не завжди забезпечують достатню точність і гнучкість через обмежене врахування семантичного контексту та індивідуальних потреб користувача. Це підтвердило актуальність запропонованого підходу, який поєднує класичні методи обробки природної мови з можливостями сучасних мовних моделей, а також обґрунтований вибір технологій для реалізації масштабованого й продуктивного програмного продукту.

У межах проєктування та реалізації системи було визначено її функціональні можливості, основні сценарії використання та архітектурні рішення. Розроблено модель бази даних і алгоритмічне забезпечення для завантаження, попередньої обробки та аналізу текстів з різних джерел. Запропоновані алгоритми витягування ключових слів, адаптивного визначення їх кількості та генерації конспектів на основі комбінованих підходів забезпечують формування логічно узгоджених і змістовних результатів незалежно від структури й обсягу вхідних даних. Це створює надійну основу для практичного використання системи та подальшого розвитку її інтелектуальних можливостей.

Практична реалізація програмного продукту підтвердила ефективність обраних підходів і рішень. Розроблений графічний інтерфейс забезпечує зручну та інтуїтивну взаємодію користувача із системою, підтримує гнучке налаштування параметрів аналізу, перегляд і збереження результатів, а також їх експорту у зручному форматі. У сукупності реалізований функціонал забезпечує повний цикл роботи з текстовими даними – від введення та аналізу до отримання готового конспекту. Отримані результати свідчать про практичну цінність розробленої системи та її потенціал для застосування в різних сферах діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hrunytskyi Dmytro, Chyzhmotria Oleksii, Dmytrenko Iryna. Identification and structuring of key concepts in textual teaching materials. Тези VIII Всеукраїнської науково-технічної конференції "Комп'ютерні технології: інновації, проблеми, рішення", 02-03 грудня 2025 року. Житомир: "Житомирська політехніка", 2025. (подано до друку)
2. Hrunytskyi Dmytro, Chyzhmotria Oleksii, Dmytrenko Iryna. Chatbot Integration. Тези VIII Всеукраїнської науково-технічної конференції "Комп'ютерні технології: інновації, проблеми, рішення", 02-03 грудня 2025 року. Житомир: "Житомирська політехніка", 2025. (подано до друку)
3. Hrunytskyi Dmytro, Chyzhmotria Oleksii, Dmytrenko Iryna. System for creating personalized notes. Тези VIII Всеукраїнської науково-технічної конференції "Комп'ютерні технології: інновації, проблеми, рішення", 02-03 грудня 2025 року. Житомир: "Житомирська політехніка", 2025. (подано до друку)
4. 8 Machine Learning Models Explained in 20 Minutes. datacamp. Machine Learning. URL: <https://www.datacamp.com/blog/machine-learning-models-explained> (дата звернення: 15.12.2025).
5. Alatas B. Comparative analysis of BERT and FastText representations on crowdfunding campaign success prediction. peerj. URL: <https://peerj.com/articles/cs-2316/> (дата звернення: 27.11.2025).
6. BERT Model - NLP. geeksforgeeks.org. URL: <https://www.geeksforgeeks.org/nlp/explanation-of-bert-model-nlp/> (дата звернення: 27.11.2025).
7. BERT or FastText? A Comparative Analysis of Contextual as well as Non-Contextual Embeddings. arxiv.org. URL: <https://arxiv.org/html/2411.17661v2> (дата звернення: 25.11.2025).
8. Cloud models are now available in Ollama Chat & build with open models. Ollama. URL: <https://ollama.com/> (дата звернення: 04.12.2025).

9. Empowering Businesses with Extractive & Generative AI, and Intelligent AI Agents. SummarizeBot. URL: <https://www.summarizebot.com/> (дата звернення: 14.12.2025).
10. Graph-Based Ranking Algorithms in Text Mining. geeksforgeeks.org. URL: <https://www.geeksforgeeks.org/nlp/graph-based-ranking-algorithms-in-text-mining/> (дата звернення: 30.11.2025).
11. Grishina A., Kyrychenko R. GPT-3 vs. BERT: Ending The Controversy. softteco. URL: <https://softteco.com/blog/bert-vs-chatgpt> (дата звернення: 01.12.2025).
12. Keyword Extraction Methods in NLP. geeksforgeeks.org. URL: <https://www.geeksforgeeks.org/nlp/keyword-extraction-methods-in-nlp/> (дата звернення: 10.10.2025).
13. Knowledge Simplified and Summarized, « Productivity » Amplified. Resoomer. URL: <https://resoomer.com/en/> (дата звернення: 14.12.2025).
14. Kumar A. Maximal Marginal Relevance to Re-rank results in Unsupervised KeyPhrase Extraction. Medium. URL: <https://medium.com/tech-that-works/maximal-marginal-relevance-to-rerank-results-in-unsupervised-keyphrase-extraction-22d95015c7c5> (дата звернення: 06.12.2025).
15. Lacy L., Chedraoui K. LLMs and AI Aren't the Same. Everything You Should Know About What's Behind Chatbots. CNET. AI and Machine Learning. URL: <https://www.cnet.com/tech/services-and-software/llms-and-ai-arent-the-same-everything-you-should-know-about-whats-behind-chatbots/> (дата звернення: 09.12.2025).
16. Lemmatization vs. Stemming: A Deep Dive into NLP's Text Normalization Techniques. geeksforgeeks.org. URL: <https://www.geeksforgeeks.org/nlp/lemmatization-vs-stemming-a-deep-dive-into-nlps-text-normalization-techniques/> (дата звернення: 05.09.2025).
17. Machine Learning Models. geeksforgeeks.org. URL: <https://www.geeksforgeeks.org/machine-learning/machine-learning-models/> (дата звернення: 15.12.2025).

18. N-gram in NLP. geeksforgeeks.org. URL: <https://www.geeksforgeeks.org/nlp/n-gram-in-nlp/> (дата звернення: 09.09.2025).
19. NLP Techniques. geeksforgeeks.org. URL: <https://www.geeksforgeeks.org/nlp/nlp-techniques/>. (дата звернення: 01.10.2025)
20. Simple Vector Stores - Maximum Marginal Relevance Retrieval. Llamaindex. URL: https://developers.llamaindex.ai/python/examples/vector_stores/simpleindexdemo/mmr/ (дата звернення: 10.10.2025).
21. Staff C. BERT vs. GPT: What's the Difference? Coursera. AI and Machine Learning. URL: <https://www.coursera.org/articles/bert-vs-gpt?msoclid=08bf5804f00b639f33e64e68f1bd6271> (дата звернення: 03.12.2025).
22. Stryker C., Holdsworth J. What is NLP (natural language processing)? IBM. Machine learning. URL: <https://www.ibm.com/think/topics/natural-language-processing> (дата звернення: 01.09.2025).
23. SummarizeBot. Empowering Businesses with Extractive & Generative AI, and Intelligent AI Agents. URL: <https://www.summarizebot.com/> (дата звернення: 01.09.2025).
24. Two minutes NLP — Keyword and keyphrase extraction with KeyBERT. Medium. URL: <https://medium.com/nlplanet/two-minutes-nlp-keyword-and-keyphrase-extraction-with-keybert-a9994b06a83> (дата звернення: 01.09.2025).
25. Types of Machine Learning Models. MathWorks. What Is a Machine Learning Model? URL: <https://www.mathworks.com/discovery/machine-learning-models.html> (дата звернення: 15.12.2025).
26. Understanding Hit Rate, MRR, and MMR Metrics. Analytics Vidhya. URL: <https://www.analyticsvidhya.com/blog/2024/07/hit-rate-mrr-and-mmr-metrics/> (дата звернення: 06.10.2025).

27. Unlocking Semantic Features in NLP. Numberanalytics. URL: <https://www.numberanalytics.com/blog/ultimate-guide-semantic-features-nlp> (дата звернення: 15.12.2025).
28. Word Embeddings in NLP. geeksforgeeks.org. URL: <https://www.geeksforgeeks.org/nlp/word-embeddings-in-nlp/> (дата звернення: 11.11.2025).
29. Your ideas, better writing QuillBot selected underline lg We use AI to strengthen writing and boost productivity—without sacrificing authenticity. QuillBot. URL: <https://quillbot.com/> (дата звернення: 14.12.2025).
30. Великі мовні моделі (LLM): Повний посібник у 2025 році. shaip. URL: <https://uk.shaip.com/blog/a-guide-large-language-model-llm/> (дата звернення: 05.12.2025).
31. Вівчарик В. Повний огляд REST: нюанси, поради, приклади. dou. URL: <https://dou.ua/forums/topic/50364/> (дата звернення: 01.09.2025).
32. Документація по Alembic. alembic.sqlalchemy.org. URL: <https://alembic.sqlalchemy.org/en/latest/> (дата звернення: 07.11.2025).
33. Документація по AngularJS. angular.dev. URL: <https://angular.dev/> (дата звернення: 01.09.2025).
34. Документація по Ant Design of Angular. ng.ant.design. URL: <https://ng.ant.design/docs/introduce/en> (дата звернення: 01.09.2025).
35. Документація по FastAPI. fastapi.tiangolo. URL: <https://fastapi.tiangolo.com/> (дата звернення: 06.10.2025).
36. Документація по KeyBERT. maartengr.github.io. URL: <https://maartengr.github.io/KeyBERT/index.html> (дата звернення: 23.11.2025).
37. Документація по SQLAlchemy. SQLAlchemy. URL: <https://www.sqlalchemy.org/> (дата звернення: 05.11.2025).
38. Документація по TypeScript. typescriptlang.org. URL: <https://www.typescriptlang.org/> (дата звернення: 01.09.2025).

39. Модель Mistral 7B. Mistral. URL: <https://mistral.ai/news/announcing-mistral-7b> (дата звернення: 03.12.2025).
40. Шейка М. М., Вінницький національний технічний університет. Мікросервісна архітектура. МІКРОСЕРВІСНА АРХІТЕКТУРА: ПЕРЕВАГИ, ВИКЛИКИ ТА НАЙКРАЩІ ПРАКТИКИ. URL: <https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/48497/24968.pdf?sequence=3> (дата звернення: 01.09.2025).
41. Що таке НЛП? Як це працює, переваги, проблеми, приклади. shaip. Вступ до обробки природної мови (NLP). URL: <https://uk.shaip.com/blog/what-is-nlp-how-it-works-benefits-challenges-examples/> (дата звернення: 01.09.2025).

ДОДАТКИ

Код адаптивного визначення кількості ключових слів на основі семантичної
структури тексту (Semantic Top-N):

```
from typing import List

import numpy as np
from sklearn.cluster import KMeans

def determine_top_n_semantic(
    sentence_embeddings: List[List[float]], min_n: int = 10, max_n: int = 25
) -> int:
    if not sentence_embeddings:
        return min_n

    vectors = np.array(sentence_embeddings)
    N_sentences = len(vectors)
    K = max(3, int(np.sqrt(N_sentences)))

    try:
        kmeans = KMeans(n_clusters=K, random_state=42, n_init=10)
        kmeans.fit(vectors)

        base_keywords_per_cluster = 2
        N_recommended = K * base_keywords_per_cluster

        return min(max(N_recommended, min_n), max_n)

    except ValueError:
        return min_n
```

Код каскадного підходу фільтрації та уточнення ключових слів за допомогою LLM:

```

async def llm_keyword_refinement(
    title: str, initial_keywords: List[str], final_n: int
) -> List[str]:
    initial_keywords_str = ", ".join(initial_keywords)

    prompt = f"""
    Ти – експерт з аналізу текстів і ключових слів. Твоє завдання - відібрати ключові
    слова та фрази, які найбільш точно описують тему тексту, і вивести їх першими, а
    потім вже всі інші. Але кількість повинна бути та сама.

    Вхідні дані:
    - Тема поданого тексту: "{title}"
    - Початковий список ключових слів: {initial_keywords_str}
    - Точна кількість ключових слів: {final_n}

    Завдання:
    3. Не роби буквальну нормалізацію усіх слів у леми, залиш змістовні
    словосполучення (наприклад, "футбольний клуб", "чемпіонат Іспанії").
    4. Всі слова мають бути повернуті по темі, але обов'язково в тій самій кількості.
    5. Не додавай зайвого тексту, коментарів або пояснень.
    6. Поверни результат **через кому**, без лапок, пояснень нумерації, додаткового
    тексту та у тій самій кількості, в якій вони надійшли.

    """

    loop = asyncio.get_running_loop()
    try:
        response = await loop.run_in_executor(
            executor,
            lambda: ollama_client.chat(
                model=settings.LLM_SUMMARY_MODEL,
                messages=[{"role": "user", "content": prompt}],
                options={"temperature": 0.0, "num_predict": 300},
            ),
        )

        raw_output = response["message"]["content"].strip()

        filtered_keywords = [
            kw.strip().replace("'", "").capitalize()
            for kw in raw_output.split(",")
            if kw.strip()
        ]

```

```
    return filtered_keywords[:final_n]

except Exception as e:
    return f"Помилка LLM-фільтрації та нормалізації ключових слів: {e}"
```

Код інтелектуальної адаптації параметрів генерації великих мовних моделей
(LLM):

```
import math
from typing import Tuple

def calculate_optimal_summary_length(text_word_count: int) -> Tuple[int, int, int]:
    COMPRESSION_RATIO = 0.08
    MIN_WORDS = 60
    MAX_WORDS = 180
    MIN_SENTENCES = 5
    MAX_SENTENCES = 12
    MIN_TOKENS = 700
    TOKEN_PER_WORD_RATIO = 1.3

    L_opt = int(text_word_count * COMPRESSION_RATIO)
    L_opt = max(MIN_WORDS, min(L_opt, MAX_WORDS))

    R_opt = math.ceil(L_opt / 15)
    R_opt = max(MIN_SENTENCES, min(R_opt, MAX_SENTENCES))

    T_opt = int(L_opt * TOKEN_PER_WORD_RATIO)
    T_opt = max(T_opt, MIN_TOKENS)

    return R_opt, L_opt, T_opt
```

Код, який здійснює генерацію конспекту з переданими параметрами та прописаним відповідно prompt:

```

async def generate_summary_with_keywords(
    title: str, keywords: list[str], text_word_count: int
) -> str:
    keywords_str = ", ".join([f'"{k}"' for k in keywords])

    R_opt, L_opt, T_opt = calculate_optimal_summary_length(text_word_count)

    prompt = f"""
    Ти – помічник, який генерує максимально детальний, глибокий, стислий, логічний та
    зв'язний висновок (summary) на основі теми та поданих ключових слів.

    Мета роботи – побудувати якісний summary, який пояснює основну ідею тексту та
    демонструє взаємозв'язок між ключовими словами.

    Тема роботи (title): "{title}"

    Ключові слова: {keywords_str}

    Оптимальні параметри викладу:
    - Кількість речень: {R_opt}
    - Орієнтована довжина: {L_opt} слів

    Створи 2-3 логічно пов'язані абзаци українською мовою, де:
    1. Тема документа (title) має задавати основний контекст викладу.
    2. Ключові слова повинні бути природно інтегровані у текст, без їхнього
    перенасичення.
    3. Результат тексту має розкривати зміст таким чином, ніби це фінальний висновок
    лекції або розділу навчального матеріалу.
    4. Мова повинна бути чіткою та без формальних маркерів (списків, заголовків).
    5. Уникай поверхневих фраз.
    6. Дотримуйся заданої довжини: {R_opt} речень і приблизно {L_opt} слів.

    Якщо ключові слова частково суперечать темі або виходять за її межі - інтегруй їх
    так, щоб вони підтримували розкриття теми, але не спотворювали її.
    """

    loop = asyncio.get_running_loop()
    try:
        response = await loop.run_in_executor(
            executor,
            lambda: ollama_client.chat(
                model=settings.LLM_SUMMARY_MODEL,
                messages=[
                    {"role": "user", "content": prompt},
                ]
            )
        )

```

```
        ],
        options={"temperature": 0.1, "num_predict": T_opt},
    ),

    return response["message"]["content"].strip()

except Exception as e:
    return f"Помилка при генерації summary: {e}"
```

Код отримання тексту з файлу:

```
async def extract_text_from_file(file: UploadFile) -> str:
    try:
        content = await file.read()

        extension = get_file_extension(file.filename)
        extractor = FILE_EXTRACTORS.get(extension)

        if extractor:
            text = extractor.fn(content, extension)
        else:
            raise HTTPException(
                status_code=400,
                detail=f"Непідтримуваний формат файлу. Підтримуються:
{SUPPORTED_FILE_TYPES}",
            )

        return text

    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Помилка обробки файлу:
{str(e)}")
```

Повний код сервісу:

```

from typing import Dict, Optional

from fastapi import HTTPException, UploadFile
from sqlalchemy import select
from sqlalchemy.ext.asyncio import AsyncSession

from app.extractors.file_extractors import (
    extract_text_from_csv,
    extract_text_from_doc,
    extract_text_from_docx,
    extract_text_from_pdf,
    extract_text_from_pptx,
    extract_text_from_txt,
)
from app.extractors.generate_text_title import generate_text_title
from app.extractors.models import TextOrm
from app.extractors.schemas import FileExtractor
from app.extractors.url_extractors import (
    extract_text_from_url_content,
    extract_text_from_url_pdf_content,
)
from app.extractors.utils import fetch_url, get_file_extension

FILE_EXTRACTORS: Dict[str, FileExtractor] = {
    ".doc": FileExtractor(extract_text_from_doc, "DOC"),
    ".docx": FileExtractor(extract_text_from_docx, "DOCX"),
    ".pptx": FileExtractor(extract_text_from_pptx, "PPTX"),
    ".pdf": FileExtractor(extract_text_from_pdf, "PDF"),
    ".txt": FileExtractor(extract_text_from_txt, "TXT"),
    ".csv": FileExtractor(extract_text_from_csv, "CSV"),
}

SUPPORTED_FILE_TYPES = ", ".join(
    extractor.file_type for extractor in FILE_EXTRACTORS.values()
)

async def extract_text_from_file(file: UploadFile) -> str:
    try:
        content = await file.read()

        extension = get_file_extension(file.filename)
        extractor = FILE_EXTRACTORS.get(extension)

        if extractor:

```

```

        text = extractor.fn(content, extension)
    else:
        raise HTTPException(
            status_code=400,
            detail=f"Непідтримуваний формат файлу. Підтримуються:
{SUPPORTED_FILE_TYPES}",
        )

    return text

except Exception as e:
    raise HTTPException(status_code=500, detail=f"Помилка обробки файлу:
{str(e)}")

async def extract_text_from_url(url: str) -> str:
    response = fetch_url(url)
    content_type = response.headers.get("Content-Type", "").lower()

    if "application/pdf" in content_type or url.lower().endswith(".pdf"):
        return extract_text_from_url_pdf_content(response.content)

    return extract_text_from_url_content(response.content)

async def get_or_save_text(
    session: AsyncSession, content: str, source_type: str
) -> int:
    text_id = await get_text_id_by_content(session, content)

    if text_id is None:
        text_id = await save_text(session, content, source_type)

    return text_id

async def get_text_id_by_content(session: AsyncSession, content: str) ->
Optional[int]:
    text_result = await session.execute(
        select(TextOrm).where(TextOrm.content == content)
    )
    text_entity = text_result.scalar_one_or_none()

    if text_entity:
        return text_entity.id

async def save_text(session: AsyncSession, text: str, source_type: str) -> int:
    title = await generate_text_title(text)

```

```
text_entity = TextOrm(  
    title=title, content=text, language="uk", source_type=source_type  
)  
  
session.add(text_entity)  
await session.commit()  
  
return text_entity.id
```

Повний код роботи з URL:

```

import io
import re
from typing import List

import PyPDF2
from bs4 import BeautifulSoup

from app.extractors.utils import fetch_url


def extract_text_from_url_content(content: bytes) -> str:
    text = extract_text_from_html(content)
    text = extract_text_from_pdf_js_viewer(text)

    return text


def extract_text_from_url_pdf_content(content: bytes) -> str:
    pdf_reader = PyPDF2.PdfReader(io.BytesIO(content))
    text = []

    for page in pdf_reader.pages:
        text.append(page.extract_text())

    return "\n".join(text)


def extract_text_from_html(content: bytes) -> str:
    soup = BeautifulSoup(content, "html.parser")

    decompose_tags(soup, ["style", "noscript", "header", "nav", "footer"])
    text = soup.get_text(separator="\n", strip=True)

    decompose_tags(soup, "script")
    main_text = "\n".join([main.get_text(separator="\n", strip=True) for main in
        soup.find_all("main")])

    if main_text.strip():
        return main_text

    return text


def decompose_tags(soup: BeautifulSoup, name: str | List[str]) -> None:
    for tag in soup.find_all(name):
        tag.decompose()

```

```
def extract_text_from_pdf_js_viewer(raw_text: str) -> str:
    match = re.search(r"\.getDocument\(['\"](.*)['\"]\)", raw_text)

    if match:
        pdf_url = match.group(1)
        response = fetch_url(pdf_url)

        return extract_text_from_url_pdf_content(response.content)

    return raw_text
```

Код генерації заголовку до отриманих текстів:

```

import asyncio
from concurrent.futures import ThreadPoolExecutor

import ollama

from app.core.config import settings

ollama_client = ollama.Client(host=settings.OLLAMA_HOST)
executor = ThreadPoolExecutor()

async def generate_text_title(text: str) -> str:
    prompt = f"""
    Текст:
    {text}

    Відповідай виключно українською мовою.

    Проаналізуй текст і створи максимально короткий та точний заголовок з 3, максимум
    8 слів (це дуже ВАЖЛИВО), який передає головну тему даного тексту.

    Важливі вимоги:
    - Не копіюй перше речення тексту.
    - Не створюй художніх або емоційних назв.
    - Формуй чіткий, нейтральний, по суті короткий заголовок.
    - Не використовуй лапки.

    Якщо текст занадто розмитий, то просто виведи абстрактний заголовок. Але, НЕ
    РЕЧЕННЯ!
    """

    loop = asyncio.get_running_loop()
    try:
        response = await loop.run_in_executor(
            executor,
            lambda: ollama_client.chat(
                model=settings.LLM_SUMMARY_MODEL,
                messages=[{"role": "user", "content": prompt}],
                options={"temperature": 0.3, "num_predict": 100},
            ),
        )

        return response["message"]["content"].strip().strip("'\"")

    except Exception as e:
        return f"Помилка при генерації заголовку (title): {e}"

```