МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ УПРАВЛІННЯ, ПСИХОЛОГІЇ ТА БЕЗПЕКИ
Кафедра інформаційних технологій

Автоматизоване семантичне реферування оперативно-службових документів із використанням великих мовних моделей

кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Владислава МАЛАЙДАХА

Науковий керівник:
Доктор філософії
Олег БАСИСТЮК

Рецензент:
_____
_____

Кваліфікаційна робота допущена до захисту
«___» _____ 2026 р., протокол № ____

Завідувач кафедри інформаційних технологій

_____ **Олег ЗАЧЕК**
(підпис)

Львів 2026

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота виконана студентом групи ІТ-42 Малайдахом Владиславом Миколайовичом. Тема “Автоматизоване семантичне реферування оперативно-службових документів із використанням великих мовних моделей”. Робота направлена на здобуття ступеня бакалавр за спеціальністю 126 «Інформаційні системи та технології» – Львівський державний університет внутрішніх справ, МВС України, Львів, 2026.

У даній роботі проведено аналіз сучасних методів автоматизованого семантичного реферування текстових документів та досліджено можливості використання великих мовних моделей (LLM) для інтелектуальної обробки текстової інформації.

У ході розробки було проведено огляд сучасних технологій Natural Language Processing (NLP), transformer-архітектур, а також методів abstractive та extractive summarization. Було досліджено принципи роботи великих мовних моделей, методи semantic analysis, chunking та Map-Reduce pipeline для обробки великих обсягів тексту.

Метою дипломної роботи є розробка інтелектуальної системи автоматизованого семантичного реферування оперативно-службових документів із використанням технологій штучного інтелекту та великих мовних моделей.

Об’єктом дослідження є процес автоматизованого аналізу, обробки та семантичного реферування текстових документів.

Предметом дослідження є методи NLP, transformer-моделі, алгоритми semantic summarization та програмні засоби обробки текстової інформації.

У результаті виконання дипломної роботи було розроблено інформаційну систему, яка дозволяє автоматично аналізувати

оперативно-службові документи, формувати структуровані та стислі реферати, виділяти ключові факти та зменшувати час опрацювання великих обсягів текстової інформації. Розроблена система може використовуватися для автоматизації обробки службових документів, аналізу текстової інформації, формування звітів та оптимізації роботи з великими масивами документів у правоохоронних та інформаційних системах.

Ключові слова: семантичне реферування, NLP, LLM, штучний інтелект, transformer-моделі, semantic summarization, оперативно-службові документи, обробка природної мови, FLAN-T5, інформаційна система.

ABSTRACT

The bachelor's qualification thesis was completed by Vladyslav Mykolaiovych Malaidakh, a student of group IT-42. The topic is "Automated Semantic Referencing of Operational and Service Documents Using Large Language Models." The work is submitted for obtaining the Bachelor's degree in specialty 126 "Information Systems and Technologies" – Lviv State University of Internal Affairs, Ministry of Internal Affairs of Ukraine, Lviv, 2026.

This work analyzes modern methods of automated semantic summarization of textual documents and explores the possibilities of using Large Language Models (LLM) for intelligent processing of textual information.

During the development, an overview of modern Natural Language Processing (NLP) technologies, transformer architectures, as well as methods of abstractive and extractive summarization was carried out. The principles of operation of large language models, methods of semantic analysis, chunking, and the Map-Reduce pipeline for processing large volumes of text were studied.

The purpose of the thesis is to develop an intelligent system for automated semantic summarization of operational and service documents using artificial intelligence technologies and large language models.

The object of the study is the process of automated analysis, processing, and semantic summarization of textual documents.

The subject of the study is NLP methods, transformer models, semantic summarization algorithms, and software tools for text information processing.

As a result of the thesis, an information system was developed that enables automatic analysis of operational and service documents, generation of structured and concise summaries, extraction of key facts, and reduction of the time required to process large volumes of textual information.

The developed system can be used for automating the processing of service documents, analyzing textual information, generating reports, and optimizing work with large collections of documents in law enforcement and information systems.

Keywords: semantic summarization, NLP, LLM, artificial intelligence, transformer models, semantic summarization, operational and service documents, natural language processing, FLAN-T5, information system.

ЗМІСТ

ВСТУП

Актуальність обраної теми зумовлена зростанням обсягу текстових даних. Щодня створюються тисячі сторінок оперативно-службових документів: звітів, протоколів, аналітичних довідок, пояснювальних записок, електронних листів та інших матеріалів, що супроводжують роботу державних установ, правоохоронних органів і бізнес-структур. Попри різноманітність цифрових інструментів, значна частина цієї документації опрацьовується вручну, що ускладнює систематизацію, сповільнює прийняття рішень і підвищує ризик помилок. У результаті бюрократичні процеси, які мали б забезпечувати порядок, часто створюють зворотний ефект – інформаційне перевантаження та хаотичне накопичення даних.

Зростаюча потреба у швидкому доступі до змісту великих текстових масивів вимагає використання сучасних інтелектуальних систем, здатних автоматично аналізувати текст, виділяти ключову інформацію та формувати стислий виклад. Розвиток технологій обробки природної мови та поява великих мовних моделей відкривають можливості для створення рішень, які можуть працювати з великими, складно структурованими документами та забезпечувати їх семантичне резюмування з високою точністю.

Метою роботи є створення автоматизованої системи для семантичного реферування оперативно-службових документів на основі сучасних підходів опрацювання великого даних.

Для досягнення поставленої мети передбачено розв'язання наступних **завдань:**

- дослідити предметну область автоматизованого резюмування текстової інформації;
- проаналізувати існуючі методи та алгоритми обробки текстових даних;
- дослідити можливості використання великих мовних моделей для семантичного аналізу документів;
- розробити архітектуру системи автоматизованого резюмування;
- реалізувати механізм обробки великих текстових документів;

- створити програмний засіб для формування структурованих реферату;
- провести тестування системи та оцінити ефективність її роботи.

Об’єктом дослідження є система автоматичного семантичного резюмування текстових документів.

Предметом дослідження є методи та засоби обробки природної мови при опрацювання оперативно-службових документів написаних українською мовою.

Методами дослідження є алгоритми машинного навчання, технології семантичного пошуку, векторні представлення тексту та сучасні моделі обробки природної мови.

Новизна роботи полягає у створенні комплексної системи, що поєднує можливості великих мовних моделей із методами розподіленої обробки даних для автоматичного резюмування складних документів.

Практична цінність полягає у зменшенні навантаження на працівників, скороченні часу опрацювання документації та підвищенні ефективності аналітичних процесів у державному секторі, правоохоронних органах та бізнес-структурах.

Особистий внесок здобувача полягає у дослідженні сучасних методів семантичного резюмування, розробці архітектури системи, створенні підходів до обробки великих текстів та реалізації програмного забезпечення для формування структурованих реферату.

Апробація результатів. Основні положення роботи були опрацьовані в межах наукового дослідження та підготовлені до представлення у вигляді наукової статті для участі в науково-практичній конференції з інформаційних технологій, штучного інтелекту та автоматизованої обробки текстових даних.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ ПІДХОДІВ ДО СЕМАНТИЧНОГО РЕЗЮМУВАННЯ ТЕКСТІВ

1.1. Аналіз проблеми обробки великих обсягів текстової інформації

Наразі значна частина роботи в установах, організаціях та на підприємствах безпосередньо пов’язана зі створенням, обробкою та аналізом текстових документів. Щодня тут формуються сотні службових записок, звітів, протоколів, пояснень, заяв, наказів та іншої внутрішньої документації. Частина цих матеріалів продовжує зберігатися у паперовому вигляді, інша – в електронному форматі, але незалежно від способу зберігання вони утворюють величезний і постійно зростаючий масив інформації. У результаті працівники стикаються зі все більшим навантаженням, а час, необхідний для аналізу та опрацювання документів, постійно збільшується.

Найбільше навантаження лягає на тих співробітників, чия діяльність пов’язана з аналізом текстової інформації. Їм доводиться регулярно переглядати великі обсяги тексту, вишукувати ключові факти, зіставляти дані з різних джерел, знаходити взаємозв’язки та формувати узагальнені висновки. У багатьох випадках потрібна інформація розпорошена по кількох документах або подана в неструктурованому вигляді, що ще більше ускладнює і затягує процес аналізу. Через це виконання навіть стандартних робочих завдань потребує значно більше часу, а швидкість прийняття управлінських рішень помітно знижується.

Проблема полягає не лише в самій кількості документів, а й у складності їхнього змісту. Більшість оперативно-службових матеріалів містять великий обсяг фактів, дат, детальних пояснень, посилань на інші документи та взаємопов’язаних подій. Щоб повністю розібратися в такому матеріалі, спеціаліст змушений уважно опрацювати кожен розділ і зіставити окремі фрагменти між собою. При великій кількості документів такий ручний аналіз стає надзвичайно трудомістким і потребує значних людських ресурсів.

Додаткові труднощі створює різноманітність структури документів. Деякі з них оформлені за чіткими шаблонами, інші являють собою суцільний текст без логічного поділу. Крім того, у документах часто трапляються таблиці, списки, примітки, численні скорочення та специфічна термінологія. Все це суттєво ускладнює як ручну, так і автоматичну обробку інформації.

Ручний аналіз текстових матеріалів має й інші суттєві недоліки. Це дуже трудомісткий процес, який сильно залежить від уважності, досвіду та навіть самопочуття людини. Працівники, які щодня працюють з великими обсягами текстів, можуть через втому пропустити важливі деталі або неправильно інтерпретувати окремі моменти. Особливо небезпечно це в тих сферах, де точність і швидкість прийняття рішень мають критичне значення – наприклад, у державному управлінні, правоохоронних органах чи аналітичній діяльності. Навіть невелика помилка в аналізі може призвести до неправильних висновків і негативно вплинути на подальшу роботу.

Ще однією серйозною проблемою є постійне накопичення архівів документів. З кожним роком обсяг інформації в організаціях стрімко зростає, а пошук потрібних даних серед цього масиву стає дедалі складнішим і тривалішим. У деяких випадках працівникам доводиться переглядати десятки, а то й сотні сторінок, щоб знайти один конкретний факт. Це суттєво знижує загальну продуктивність і створює додаткове навантаження на персонал.

У зв’язку з цим автоматизація процесів роботи з текстовими документами стає одним із найважливіших напрямів розвитку сучасних інформаційних технологій. [3] Використання систем штучного інтелекту та методів обробки природної мови дозволяє значно полегшити аналіз документів, скоротити час їх опрацювання та підвищити загальну ефективність інформаційно-аналітичних систем. Такі технології здатні автоматично виділяти ключові частини тексту, знаходити основний зміст, встановлювати логічні зв’язки між фактами та формувати зрозумілі структуровані реферату.

Великі мовні моделі відкрили абсолютно нові можливості для роботи з текстами. [1], [4] Їх головна перевага – вміння аналізувати не лише окремі речення, а й враховувати загальний контекст усього документа. Завдяки цьому сформовані реферату виходять більш логічними, зв’язними та близькими до того, як їх склала б людина.

Таким чином, сучасні методи автоматизованого семантичного резюмування дозволяють суттєво зменшити навантаження на працівників, прискорити процес аналізу документів і підвищити ефективність роботи з великими масивами текстової інформації. Саме тому розробка систем автоматизованого резюмування оперативно-службових документів є актуальним і перспективним напрямом у сфері інформаційних технологій та штучного інтелекту.

1.2. Особливості оперативно-службових документів

Оперативно-службові документи мають ряд важливих особливостей, які значно ускладнюють їх автоматизовану обробку та подальший аналіз. [5] На відміну від звичайних текстів, такі матеріали зазвичай містять великий обсяг інформації, насиченої спеціалізованою термінологією, складними мовними конструкціями, численними фактами, датами, подіями та взаємопов’язаними поясненнями. Усе це вимагає від системи не просто поверхневого розуміння тексту, а максимально глибокого і точного аналізу контексту. Адже навіть невелика неточність чи втрата важливого нюансу може суттєво вплинути на результати подальшої роботи та прийняті рішення.

Однією з головних вимог до систем автоматизованого резюмування є максимально точне збереження змісту оригінального документа. У оперативно-службових матеріалах часто немає права на помилку: тут важливі кожна дата, кожен факт і кожне посилання. Невелике перекручення інформації, пропуск ключового фрагмента чи неправильне трактування зв’язків між подіями може призвести до хибних висновків. Тому система повинна не просто скорочувати

текст, а вміти зберігати основну логіку документа, ключові факти та причинно-наслідкові зв’язки між різними частинами інформації.

Значних труднощів додає й структура таких документів. Частина матеріалів створюється за чіткими шаблонами і має добре організований вигляд. Однак чимало документів являють собою суцільний, погано структурований текст, який може включати різноманітні примітки, таблиці, нумеровані списки, службові позначення, скорочення та пояснення. Через це автоматичній системі доводиться не лише розуміти зміст, але й правильно розпізнавати структуру документа, визначати важливість різних елементів і встановлювати логічні зв’язки між ними.

Окремою проблемою є велика кількість повторюваної та другорядної інформації. Оперативно-службові документи часто містять стандартні формулювання, типові службові звороти, пояснення та фрагменти, які не несуть особливої смислової цінності для загального змісту. Через це системі стає складніше відокремлювати дійсно важливу інформацію від «води» та фонових даних. Якщо цього не робити якісно, фінальне реферату може вийти перевантаженим другорядними деталями або, навпаки, втратити ключові моменти.

Не менш важливою складністю є робота з великими обсягами тексту. Багато оперативно-службових документів складаються з десятків, а іноді й сотень сторінок. У таких умовах сучасні великі мовні моделі стикаються з природним обмеженням – контекстним вікном. Повний документ не завжди можливо обробити за один раз, тому виникає потреба у спеціальних технічних рішеннях: розбитті тексту на фрагменти (chunking), семантичному пошуку релевантних частин, багатоступеневій генерації реферату та інших методах, які дозволяють працювати з великими документами без суттєвої втрати якості.

Таким чином, оперативно-службові документи вирізняються складною структурою, високою інформаційною насиченістю, великим обсягом і підвищеними вимогами до точності. Усе це робить їх автоматизовану обробку досить непростим завданням. Для ефективної роботи з такими матеріалами

необхідне використання сучасних методів обробки природної мови, які здатні забезпечувати глибокий аналіз тексту, збереження контексту та формування дійсно якісних і корисних реферату. [2], [6]

1.3. Сучасні методи автоматизованого резюмування тексту та використання великих мовних моделей

Автоматизоване підсумовування тексту (або автоматичне резюмування) є одним із важливих напрямів обробки природної мови. Його головна задача – створити короткий, але при цьому змістовний і зрозумілий виклад документа, зберігаючи при цьому ключову інформацію та основну логіку оригіналу.

У сучасних умовах кількість текстових даних зростає в геометричній прогресії. Щодня в установах, організаціях та на підприємствах створюються тисячі різних документів – від службових записок і звітів до протоколів та аналітичних матеріалів. Ручне опрацювання такого величезного масиву інформації стає дедалі складнішим, потребує значних витрат часу та людських ресурсів. Саме тому системи автоматизованого резюмування набувають все більшої популярності й активно використовуються в бізнесі, аналітиці, державному управлінні та правоохоронній сфері.

Основна мета автоматичного резюмування полягає не просто в механічному скороченні тексту, а в тому, щоб зберегти його суть, головні факти, логічні зв’язки та ключові висновки. Це особливо важливо для оперативно-службових документів, які зазвичай мають великий обсяг, містять безліч деталей, дат, пояснень та взаємопов’язаних подій. Без якісного структурованого реферату аналіз таких матеріалів може займати багато часу і створювати серйозне навантаження на працівників, які займаються інформаційно-аналітичною роботою.

Існує два основних підходи до автоматизованого підсумовування тексту: екстрактивний та абстрактивний. [7]

Екстрактивний метод є простішим у реалізації. Він полягає у виборі найбільш важливих речень або фрагментів з оригінального документа. Система аналізує текст, визначає найбільш інформативні частини і об'єднує їх у скорочену версію. Головна перевага такого підходу – висока точність, адже вся інформація взята безпосередньо з джерела. Однак є й недоліки: готове реферату часто виглядає дещо уривчастим, а окремі речення не завжди добре поєднуються між собою.

Абстрактивний підхід працює набагато складніше. Тут система не копіює готові речення, а самостійно формує новий текст на основі глибокого розуміння змісту документа. По суті, модель діє так, як це зробила б людина – переказує прочитане своїми словами. Завдяки цьому реферату виходить більш цілісним, логічним і природним для сприйняття. Водночас реалізація абстрактивного резюмування є значно складнішою і вимагає використання сучасних алгоритмів машинного навчання та великих мовних моделей.

Справжній прорив у цій сфері стався з появою великих мовних моделей (Large Language Models – LLM). Завдяки навчанню на величезних обсягах текстових даних вони здатні не просто аналізувати окремі речення, а розуміти загальний контекст документа, встановлювати приховані зв'язки між різними частинами тексту та вловлювати його основний сенс. Це дозволяє отримувати значно якісніші та корисніші реферату, особливо при роботі зі складними оперативно-службовими матеріалами.

Сучасні LLM базуються на архітектурі трансформерів і механізмі уваги (attention), який допомагає моделі визначати, які слова та фрагменти тексту є найбільш важливими в певному контексті. Саме така архітектура лежить в основі популярних моделей, таких як GPT, Claude, Gemini та інших. [8], [16]

Звісно, великі мовні моделі мають і певні обмеження. Одним із найсуттєвіших є обмежений розмір контекстного вікна. Через це модель не може повністю обробити дуже великі документи (на десятки чи сотні сторінок)

за один раз. Для вирішення цієї проблеми використовуються різні технічні підходи. Найпоширенішим є chunking – розбиття документа на менші фрагменти, які аналізуються окремо, а потім об’єднуються в єдине реферату. [10]

Крім того, активно застосовуються методи Retrieval-Augmented Generation (RAG) та архітектура Map-Reduce. [11] Вони дозволяють системі знаходити найбільш релевантні частини документа і ефективніше працювати з великими обсягами тексту, мінімізуючи втрату контексту.

Поєднання потужних мовних моделей із сучасними методами обробки великих текстів відкриває широкі перспективи для автоматизації роботи з документами. Такі технології дозволяють суттєво скоротити час аналізу інформації, зменшити навантаження на працівників і підвищити загальну ефективність інформаційно-аналітичних систем.

Висновки до першого розділу

Постійне зростання обсягів текстової інформації та ускладнення структури оперативно-службових документів створюють серйозні труднощі для їх ручного аналізу та опрацювання. Кожен день у установах і організаціях з’являється величезна кількість різних документів, які містять значний обсяг даних. Висока інформаційна насиченість, необхідність швидкого пошуку ключових фактів і підвищені вимоги до точності аналізу суттєво збільшують навантаження на працівників. Крім того, постійне накопичення архівних документів ускладнює ситуацію: пошук потрібної інформації серед цього масиву займає дедалі більше часу і негативно позначається на загальній ефективності інформаційно-аналітичної роботи.

У таких умовах використання сучасних методів обробки природної мови та систем штучного інтелекту вже не є просто перевагою, а стає об’єктивною необхідністю. Автоматизовані системи семантичного резюмування дозволяють значно скоротити час аналізу текстових матеріалів, автоматично

виділяти ключову інформацію, встановлювати логічні зв’язки між фактами та формувати структуровані, зручні для сприйняття реферату документів. Особливо перспективним у цьому напрямку є застосування великих мовних моделей, які здатні глибоко розуміти контекст документа, бачити взаємозв’язки між різними його частинами та створювати якісні, зрозумілі підсумки.

Водночас оперативно-службові документи мають низку специфічних особливостей, які суттєво ускладнюють їх автоматизовану обробку. Це і великий обсяг тексту, і складна структура, і наявність великої кількості спеціалізованої термінології, фактів, дат та взаємопов’язаних подій. У таких умовах система повинна не просто механічно скорочувати текст, а забезпечувати точне збереження основного змісту, логіки документа та ключових взаємозв’язків між окремими частинами інформації.

Додаткові труднощі створює різноманітність форматів документів, значна кількість другорядної та повторюваної інформації, а також необхідність працювати з великими текстовими масивами. Сучасні великі мовні моделі мають певні технічні обмеження, зокрема щодо розміру контекстного вікна. Тому для ефективної роботи з великими документами виникає потреба у використанні спеціальних методів обробки тексту, таких як chunking, семантичний пошук релевантних фрагментів та багатоступенева генерація реферату. Саме ці підходи дозволяють мінімізувати втрату контексту та підтримувати високу якість аналізу навіть при значних обсягах інформації.

Автоматизоване підсумовування тексту сьогодні по праву вважається одним із найбільш перспективних і затребуваних напрямів сучасної обробки природної мови. Його основна задача – створити короткий, але змістовний виклад документа, зберігаючи при цьому ключову інформацію та основну логіку тексту. Проведений аналіз показав, що існують два базових підходи до

автоматизованого резюмування – екстрактивний та абстрактивний. Кожен з них має свої сильні та слабкі сторони, але найбільшу ефективність сучасні системи демонструють при комбінованому використанні можливостей великих мовних моделей.

Таким чином, поєднання потужних мовних моделей із сучасними методами обробки тексту відкриває широкі можливості для автоматизації роботи з документами. Такі технології дозволяють суттєво скоротити час аналізу інформації, зменшити навантаження на працівників, прискорити процес прийняття рішень і загалом підвищити ефективність інформаційно-аналітичних систем. Саме тому розробка систем автоматизованого семантичного резюмування оперативно-службових документів є актуальним і перспективним напрямом розвитку сучасних інформаційних технологій та штучного інтелекту.

РОЗДІЛ 2. ДОСЛІДНИЦЬКИЙ РОЗДІЛ

2.1. Особливості реалізації системи автоматизованого семантичного резюмування

Під час розробки системи автоматизованого семантичного резюмування основна увага приділялася роботі саме з великими текстовими документами. Оперативно-службові матеріали, як правило, мають значний обсяг і часто складаються з десятків, а іноді й сотень сторінок щільного тексту. Це суттєво ускладнює їх автоматизовану обробку. Використання великих мовних моделей для аналізу таких документів створює певні технічні труднощі, оскільки сучасні LLM мають суттєві обмеження щодо розміру контекстного вікна і не можуть обробити весь документ за один запит. У результаті виникає необхідність у застосуванні спеціальних методів обробки великих текстових масивів. [10]

Через це довелося додати підхід chunking – автоматичне розбиття документа на окремі текстові фрагменти. Кожен такий фрагмент аналізується моделлю незалежно, після чого система збирає всі отримані результати воєдино і формує єдине структуроване реферату документа. Такий метод дозволяє ефективно працювати навіть із дуже великими текстами, суттєво зменшує ризик втрати важливої інформації та знижує загальне навантаження на модель. Завдяки цьому система працює стабільніше і швидше навіть при обробці об’ємних документів.

(Приклад реалізації алгоритму наведено нижче. 1.1)

```
def chunk_text(text, chunk_size=1800):
  chunks = []
  for i in range(0, len(text), chunk_size):
    chunks.append(
      text[i:i + chunk_size]
    )
```

return chunks

Окрему увагу під час реалізації було приділено правильному вибору розміру текстових фрагментів. Як показала практика, занадто маленькі chunk-и часто призводять до втрати логічного зв’язку між різними частинами документа, через що фінальне реферату виходить фрагментованим, уривчастим і неповним. У той же час надто великі блоки тексту створюють велике навантаження на модель і можуть перевищувати допустимий розмір контекстного вікна. Тому в системі використовується збалансований розмір фрагментів у поєднанні з додатковим перекриттям (overlap) між сусідніми частинами. Такий підхід дозволяє моделі краще «бачити» контекст на стиках фрагментів і значно підвищує якість та цілісність фінального реферату.

Для безпосередньої обробки тексту в системі застосовуються великі мовні моделі. [1], [8], [17] Їхньою ключовою перевагою є здатність аналізувати не лише окремі речення, а й загальний зміст і контекст усього документа. Завдяки цьому система може формувати більш логічні, зв’язні та зрозумілі для людини реферату, які добре передають основну суть документа. На відміну від класичних алгоритмів, LLM здатні встановлювати приховані зв’язки між різними частинами тексту та точніше визначати дійсно важливу інформацію.

(Приклад реалізації алгоритму наведено нижче. 1.2)

chunks = chunk_text(text)

summaries = []

for chunk in chunks[:10]:

Перед початком основного аналізу кожен документ обов’язково проходить етап попередньої обробки тексту. На цьому етапі система очищає текст від зайвих символів, службових позначень, автоматичних переносів рядків, неправильного форматування та іншого «шуму», який може негативно впливати на якість роботи моделі. Попередня обробка є важливою складовою всієї системи, оскільки навіть

невеликі недоліки в структурі тексту можуть суттєво погіршувати результати генерації реферату. Після очищення документ розбивається на фрагменти і передається мовній моделі для подальшого аналізу.

(Приклад реалізації алгоритму наведено нижче. 1.3)

```
def clean_text(text):
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(
        r'[^\w\s.,:;!?()%/-]',
        '',
        text
    )
    return text.strip()
```

Для додаткового підвищення якості фінального реферату в системі використовується підхід Retrieval-Augmented Generation (RAG). [11] Перед генерацією підсумкового тексту система виконує семантичний пошук і виділяє найбільш релевантні та змістовні фрагменти документа. Саме ці частини потім передаються моделі для формування фінального реферату. Такий метод дозволяє суттєво зменшити кількість другорядної інформації в результаті та краще зберігати ключові факти. Особливо ефективно RAG працює при аналізі великих документів, де важлива інформація може бути розподілена по всьому тексту.

(Приклад реалізації алгоритму наведено нижче. 1.4)

```
keywords = extract_keywords(text)

score_sentences(
sentences,
keywords
```

)

chunks = chunk_text(text)
for chunk in chunks:

Окреме значення під час розробки приділялося структурі фінального реферату. Головна мета системи полягає не просто в скороченні тексту, а в створенні логічного, добре структурованого і зрозумілого викладу основних фактів і подій. Для цього модель отримує спеціально підготовлені prompt-запити, які чітко визначають бажану структуру відповіді, стиль викладу та перелік інформації, яку обов'язково потрібно включити. Завдяки цьому реферату виходить більш організованим, інформативним і зручним для подальшого використання користувачем.

(Приклад реалізації алгоритму наведено нижче. 1.5)

prompt = f"""

Створи короткий структурований витяг документа.

Виділи:

- основну суть

- ключові факти

- важливі події

- результати

Текст:

{chunk}

Реферат:

"""

2.2 Архітектура та логіка роботи програмної системи

Система розроблена за модульним принципом, що є одним із ключових архітектурних рішень. [12] Такий підхід дозволяє розділити весь процес обробки документів на окремі незалежні компоненти. Завдяки цьому кожен модуль можна вдосконалювати, змінювати чи замінювати незалежно від інших частин системи без необхідності переробки всього проєкту. Модульна архітектура суттєво спрощує подальшу розробку, тестування, відлагодження та масштабування системи відповідно до нових вимог і завдань.

У межах системи були реалізовані окремі модулі, відповідальні за різні етапи роботи: зчитування документів різних форматів, попередню обробку тексту, генерацію векторних представлень, семантичний аналіз, пошук релевантної інформації та формування фінального реферату. Така структура робить систему більш гнучкою, зрозумілою в підтримці та зручною для подальшого розвитку.

Для забезпечення роботи з найбільш поширеними типами документів у системі реалізовано підтримку TXT, DOCX та PDF форматів. [13] Це дозволяє користувачам завантажувати матеріали з різних джерел без додаткових обмежень. Підтримка кількох форматів є важливою практичною перевагою, оскільки оперативно-службові документи в установах і організаціях часто надходять у різному вигляді та мають різну внутрішню структуру.

from extract import read_txt, read_docx, read_pdf
from summarizer import generate_summary

Після завантаження документа система автоматично переходить до етапу попередньої обробки тексту. На цьому етапі відбувається очищення документа від зайвих символів, службових елементів, автоматичних переносів рядків, зайвого форматування та іншого текстового «шуму», який може негативно впливати на якість подальшого аналізу. Попередня обробка відіграє важливу роль у всій системі, адже навіть невеликі недоліки в тексті здатні суттєво погіршити результати роботи мовної моделі. Після очищення текст автоматично розбивається на окремі фрагменти для подальшої обробки.

(**Фрагмент програмного коду, який реалізує даний підхід 1.1**). **(Фрагмент програмного коду, який реалізує даний підхід 1.3).**

```
if file_path.endswith(".txt"):
    text = read_txt(file_path)
elif file_path.endswith(".docx"):
    text = read_docx(file_path)
elif file_path.endswith(".pdf"):
    text = read_pdf(file_path)
```

Далі кожен текстовий фрагмент передається великій мовній моделі для виконання семантичного аналізу та генерації проміжного резюме. На цьому етапі система визначає ключові факти, основні події та найбільш значущу інформацію в межах окремої частини документа. Отримані результати також використовуються для семантичного пошуку, що дозволяє знаходити важливі фрагменти не лише за ключовими словами, а й за їхнім реальним змістом. Такий комплексний підхід значно підвищує якість фінального резюме та зменшує кількість другорядної інформації в підсумковому тексті.

Фрагмент програмної реалізації наведено нижче 1.6

```
chunks = chunk_text(text)
for chunk in chunks[:10]:
result = summarizer(
prompt,
do_sample=False
)
```

Наступним етапом є формування проміжних резюме для кожного фрагмента. Після завершення аналізу всіх частин отримані результати об’єднуються в єдиний текст і повторно передаються мовній моделі для створення фінального резюме всього документа. Такий багатоступеневий підхід дозволяє краще зберігати

загальну логіку та структуру оригінального документа, підтримувати зв’язки між різними частинами тексту та ефективніше працювати з великими обсягами інформації. Крім того, повторний аналіз проміжних результатів допомагає зменшити ризик втрати важливих деталей.

Фрагмент програмної реалізації наведено нижче 1.7

final_summary = "\n\n".join(summaries)

Для взаємодії з мовною моделлю в системі використовуються спеціально підготовлені prompt-запити. Вони чітко визначають бажану структуру резюме, стиль викладу інформації та перелік елементів, які обов’язково мають бути присутні у фінальному тексті. Якість і продуманість промптів суттєво впливає на стабільність і корисність результатів. Під час розробки цьому аспекту приділялася окрема увага, щоб фінальні резюме були максимально логічними, структурованими та зручними для користувача.

Фрагмент програмної реалізації наведено нижче 1.8

```
prompt = f"""
Створи детальний структурований реферат документа.
Проаналізуй:
- основну тему
- ключові факти
- результати
Текст:
{chunk}
Реферат:
"""
```

У випадках, коли фінальний реферат виходить занадто коротким або не повністю відображає важливі аспекти документа, система має механізм повторної перевірки. Вона може автоматично повернутися до окремих фрагментів і виконати додатковий аналіз. Такий гнучкий підхід дозволяє суттєво підвищити повноту та точність результату, особливо при роботі зі складними та великими текстовими матеріалами, де ключова інформація може бути розподілена по всьому документу.

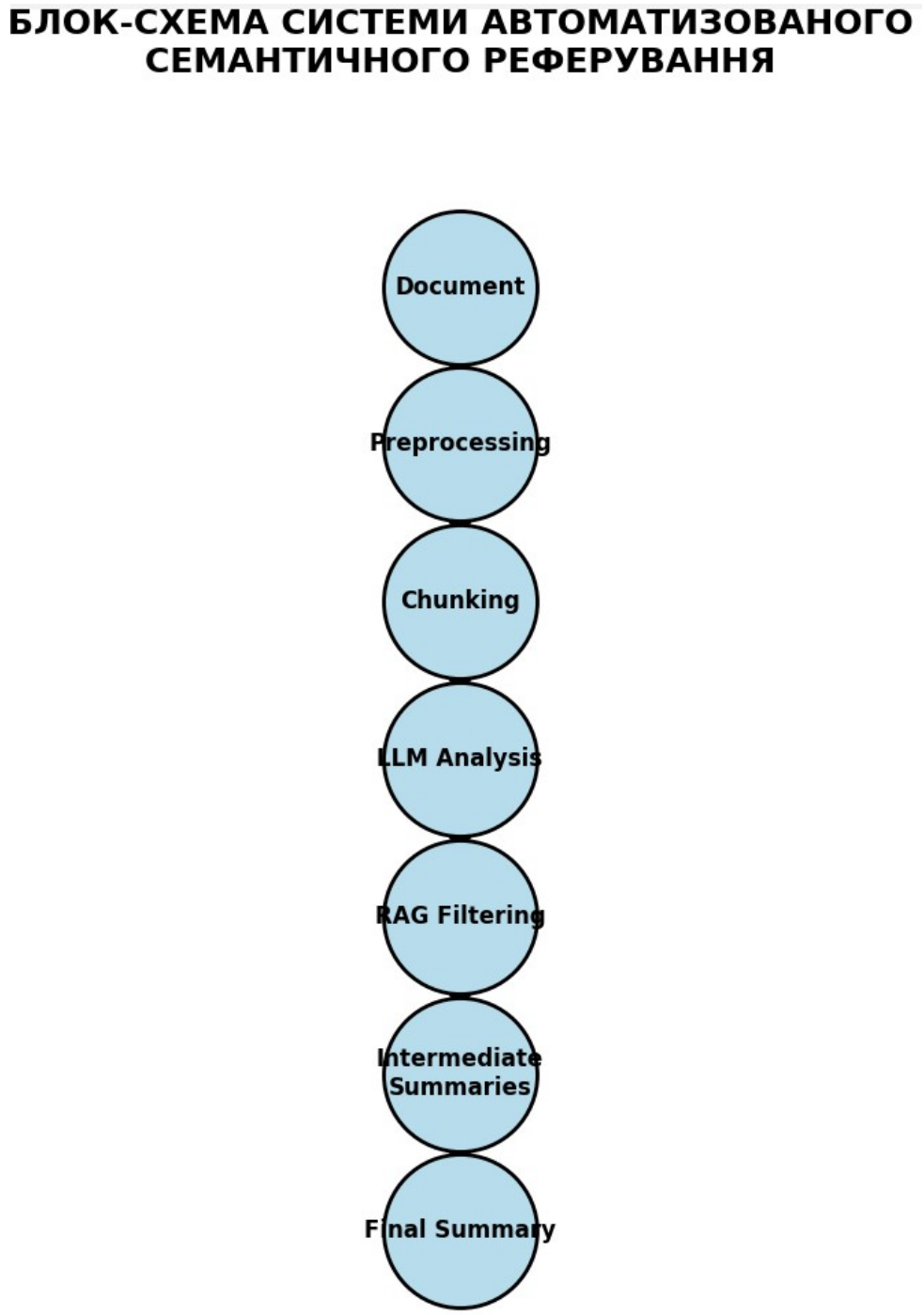


Рис. 2.1. Блок-схема роботи системи автоматизованого семантичного реферування оперативно-службових документів

На рисунку 2.1 наведено загальну блок-схему роботи системи автоматизованого семантичного реферування оперативно-службових документів**.** Представлена схема відображає основні етапи обробки текстової інформації та

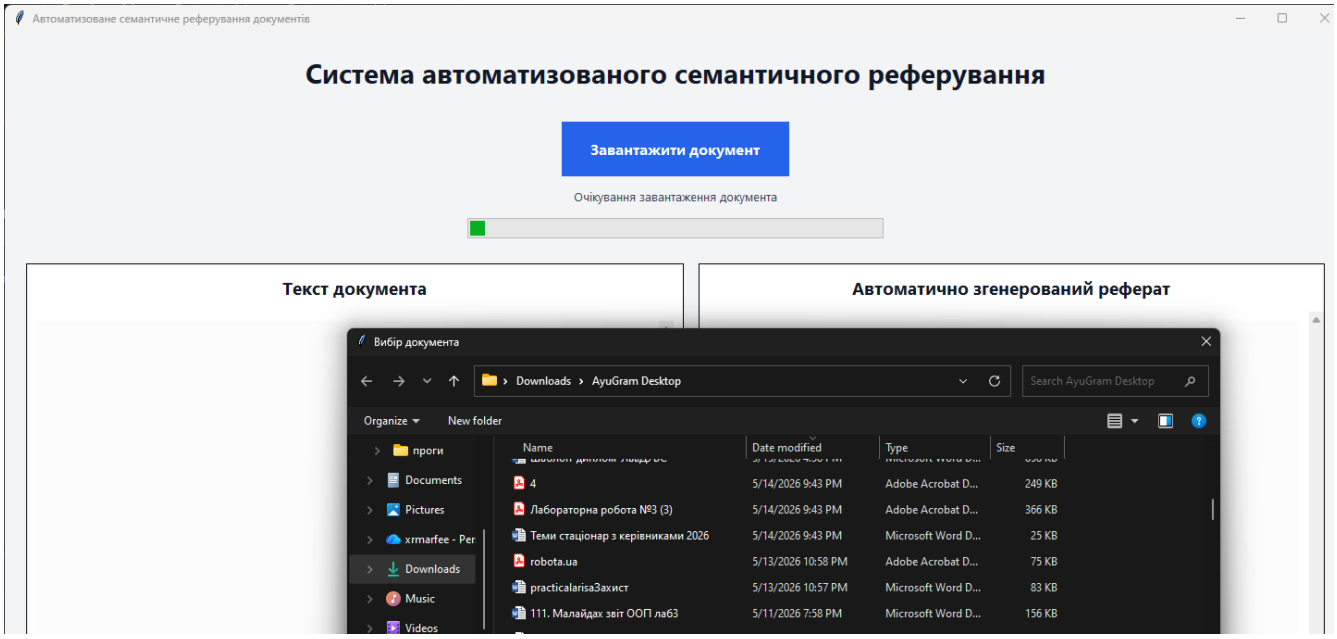
послідовність взаємодії окремих компонентів системи під час формування фінального реферату документа.

Робота системи починається із завантаження документа, після чого виконується попередня обробка тексту. На цьому етапі документ очищається від зайвих символів, службових елементів, переносів рядків та іншого текстового «шуму», який може негативно впливати на подальший аналіз. Після цього система переходить до етапу chunking, де великий документ автоматично розбивається на окремі текстові фрагменти для подальшої незалежної обробки.

Наступним етапом є semantic analysis за допомогою великих мовних моделей. Система аналізує кожен текстовий фрагмент окремо, визначає ключову інформацію, основні факти, важливі події та логічні зв’язки між окремими частинами документа. Для покращення якості аналізу додатково використовується механізм Retrieval-Augmented Generation, який дозволяє виділяти найбільш змістовні та релевантні частини тексту перед формуванням проміжного реферату.

Після завершення аналізу окремих chunk-ів система формує проміжні реферати для кожної частини документа. Отримані результати надалі об’єднуються та повторно аналізуються для генерації фінального структурованого резюме. Такий багатоступеневий підхід дозволяє значно краще зберігати загальний зміст документа, мінімізувати втрату важливої інформації та підтримувати логічний зв’язок між різними частинами тексту навіть при роботі з великими обсягами даних.

Загалом представлена блок-схема демонструє основний принцип роботи системи та дозволяє наочно показати взаємозв’язок між окремими етапами автоматизованого семантичного резюмування документів.



На рисунку 2.2 представлено графічний інтерфейс розробленої системи автоматизованого семантичного реферування документів. Через інтерфейс користувач має можливість завантажувати текстові документи різних форматів для їх подальшого аналізу та автоматичної генерації структурованого реферату. Підтримка кількох типів файлів дозволяє працювати як зі звичайними текстовими документами, так і з більш складними форматами, що робить систему універсальнішою та зручнішою для практичного використання.

Після вибору документа система автоматично запускає процес попередньої обробки тексту, очищення даних та підготовки документа до подальшого семантичного аналізу. На цьому етапі видаляються зайві символи, службові елементи, некоректні переноси рядків та інші фрагменти, які можуть негативно впливати на якість роботи мовної моделі. Далі текст проходить етап розбиття на окремі фрагменти для подальшої незалежної обробки великими мовними моделями.

У центральній частині інтерфейсу відображається індикатор виконання обробки документа, який дозволяє користувачу контролювати стан генерації реферату в режимі реального часу. Це особливо важливо при роботі з великими документами, аналіз яких може займати певний час через значний обсяг текстової

інформації та необхідність виконання кількох етапів семантичного аналізу. Наявність індикатора виконання робить процес роботи системи більш зрозумілим і зручним для користувача.

Окрему увагу під час розробки інтерфейсу було приділено простоті взаємодії користувача із системою. Основні елементи керування розташовані максимально зрозуміло та не перевантажують робочий простір зайвими компонентами. Завдяки цьому користувач може швидко завантажити документ, запустити аналіз та отримати готовий результат без необхідності додаткового налаштування системи чи виконання складних дій.

Інтерфейс системи побудований таким чином, щоб забезпечити швидке завантаження, обробку та аналіз документів незалежно від їх обсягу, структури чи рівня складності. Це дозволяє використовувати систему як для роботи з невеликими текстами, так і для аналізу великих оперативно-службових документів, які містять значний обсяг інформації та потребують детального семантичного опрацювання.

2.3. Основні проблеми під час розробки системи

Під час розробки системи однією з перших і найбільш очевидних проблем стало обмеження контекстного вікна великих мовних моделей. [10] Більшість оперативно-службових документів, з якими мала працювати система, відрізнялися значним обсягом і часто складалися з десятків, а іноді й сотень сторінок щільного тексту. Через це модель фізично не могла проаналізувати весь документ за один запит. У деяких випадках частина тексту просто відсікалася, а інколи система взагалі не могла сформувати повноцінне резюме. Саме тому одним із перших серйозних технічних рішень стало впровадження механізму chunking– автоматичного розбиття документа на окремі текстові фрагменти меншого розміру. Це дозволило організувати поетапну обробку великих текстів і зробити аналіз технічно можливим.

Однак після впровадження chunking досить швидко виявилася інша, не менш серйозна проблема–втрата контексту між окремими частинами документа. Коли

текст розбивався на незалежні фрагменти, модель переставала «бачити» зв’язки між подіями, фактами чи висновками, які знаходилися в різних розділах. У результаті сформовані резюме часто виглядали фрагментованими, неповними, а іноді навіть суперечливими в деяких моментах. Особливо гостро ця проблема проявлялася в документах, де важлива інформація була «розкидана» по всьому тексту. Для зменшення цього недоліку було реалізовано механізм overlap (перекриття) між сусідніми chunk-ами. [10], [18] Частина тексту спеціально дублювалася між фрагментами, що дозволило моделі краще зберігати логічний зв’язок і точніше розуміти загальний контекст документа.

Ще однією неприємною проблемою стало дублювання інформації у фінальному резюме. Оскільки кожен фрагмент аналізувався окремо, одна й та сама важлива інформація могла кілька разів потрапляти до проміжних результатів. Через це фінальне резюме іноді виходило надто розтягнутим, повторюваним і менш читабельним. Щоб усунути цей недолік, було додано окремий етап фінальної обробки результатів. На цьому етапі система повторно аналізувала всі проміжні резюме, виявляла дублікати, видаляла зайве та формувала більш цілісний, компактний і логічно зв’язаний варіант підсумкового тексту.

Під час активного тестування системи також яскраво проявилася одна з найбільш відомих слабкостей великих мовних моделей–генерація неточної або повністю вигаданої інформації, так звані «галюцинації». [14] Модель могла додавати факти, яких не було в оригінальному документі, або дещо змінювати зміст окремих фрагментів. Для оперативно-службових матеріалів така поведінка є абсолютно неприпустимою, оскільки навіть невелике викривлення інформації може призвести до неправильних висновків і серйозних наслідків. Тому довелося витратити чимало часу на суттєве перероблення структури prompt-запитів: були додані жорсткі інструкції використовувати тільки інформацію з наданого документа, заборона на домислювання фактів, а також чіткі обмеження стилю

генерації. Після цих змін кількість неточностей у фінальних резюме помітно зменшилася.

Не менше труднощів було через навантаження на систему під час роботи з великими документами. Аналіз об’ємних текстів вимагав великих обчислювальних ресурсів, через що час генерації фінального резюме суттєво зростав. Особливо це відчувалося при послідовній обробці кількох великих документів поспіль. Для оптимізації процесу було впроваджено поетапну обробку, кешування вже створених embeddings, а також максимально зменшено кількість зайвих звернень до мовної моделі. Завдяки цьому вдалося помітно підвищити швидкість роботи системи та знизити загальне навантаження.

Досить багато труднощів виникало також на етапі попередньої підготовки документів. Багато текстів містили велику кількість зайвих символів, неправильне форматування, автоматичні переноси рядків, таблиці, службові позначення та інші неструктуровані елементи. Усе це негативно впливало на якість аналізу та могло суттєво погіршувати результати роботи моделі. Тому в системі був реалізований окремий модуль очищення та нормалізації тексту, який видаляє зайві елементи, виправляє форматування та приводить документ до більш однорідного та «чистого» вигляду. Як показала практика, якість попередньої підготовки тексту в багатьох випадках відігравала вирішальну роль у кінцевій якості сформованого реферату.

Основні труднощі, причини їх виникнення та способи вирішення наведені в таблиці 2.1.[2].

Таблиця 2.1

Основні проблеми під час розробки системи та способи їх вирішення

Проблема	**Причина виникнення**	**Реалізоване рішення**
Обмеження контекстного вікна LLM	Великі мовні моделі не можуть обробляти дуже великі документи за один запит	Реалізовано механізм chunking – автоматичне розбиття документа на окремі фрагменти

Втрата контексту між chunk-ами	Після розбиття документа модель втрачала логічний зв’язок між окремими частинами тексту	Додано overlap (перекриття) між сусідніми фрагментами для часткового збереження контексту
Дублювання інформації у фінальному рефераті	Однакові факти могли повторюватися у проміжних резюме різних chunk-ів	Реалізовано додатковий етап фінальної агрегації та видалення повторів
Генерація неточної або вигаданої інформації (галюцинації)	LLM інколи генерували факти, яких не було в документі	Оптимізовано prompt-запити та додано жорсткі інструкції використовувати лише інформацію з документа
Повільна обробка великих документів	Велика кількість chunk-ів створювала значне навантаження на систему	Впроваджено поетапну обробку, оптимізацію запитів та кешування embeddings
Низька якість аналізу неструктурованих текстів	Документи містили таблиці, зайві символи, службові позначення та неправильне форматування	Реалізовано модуль очищення та нормалізації тексту перед аналізом
Втрата важливих деталей під час скорочення тексту	При сильному стисканні частина важливої інформації могла випадати	Додано semantic filtering та Retrieval-Augmented Generation (RAG)
Надмірно короткі або поверхневі реферати	Модель інколи генерувала занадто	Покращено структуру prompt-запитів та додано

	загальні результати	вимоги до деталізації резюме
Високе навантаження при послідовній обробці кількох документів	Велика кількість одночасних запитів до моделі уповільнювала систему	Оптимізовано чергу обробки та зменшено кількість повторних звернень до LLM
Некоректна робота з різними форматами файлів	TXT, DOCX та PDF мають різну структуру зберігання тексту	Реалізовано окремі модулі зчитування для кожного типу документів

Джерело: складено автором за матеріалами [2]

Як показали результати розробки та подальшого тестування системи, більшість виявлених проблем були тісно пов’язані між собою. Наприклад, спроба вирішити проблему обмеження контекстного вікна за допомогою chunking автоматично створювала іншу складність – втрату контексту між окремими частинами документа. У свою чергу, збільшення overlap між chunk-ами допомагало краще зберігати логіку тексту, але водночас збільшувало кількість дублювань інформації у фінальному резюме та підвищувало загальне навантаження на систему. Через це під час розробки доводилося постійно шукати баланс між якістю аналізу, швидкістю роботи та обсягом оброблюваної інформації.

Окрему складність створювали самі оперативно-службові документи, оскільки вони часто мають нестандартну структуру, містять велику кількість другорядної інформації, службових позначень, таблиць, скорочень та інших елементів, які можуть негативно впливати на якість семантичного аналізу. На практиці стало зрозуміло, що навіть сучасні великі мовні моделі не здатні стабільно працювати з «брудним» текстом без попередньої підготовки. Саме тому модуль очищення та нормалізації тексту став одним із ключових компонентів усієї системи.

Важливим етапом також стала оптимізація prompt-запитів до мовної моделі. На початкових етапах тестування результати часто були надто загальними,

перевантаженими повтореннями або містили неточності. Поступове доопрацювання структури prompt-ів, уточнення інструкцій та додавання жорстких обмежень дозволили суттєво покращити якість фінальних резюме та зробити результати більш стабільними й передбачуваними.

Крім цього, у процесі розробки стало очевидно, що ефективність системи залежить не лише від самої мовної моделі, а й від правильно організованої архітектури обробки документів. Велике значення мають спосіб розбиття тексту, порядок обробки chunk-ів, механізми семантичного пошуку, агрегація проміжних результатів та оптимізація кількості звернень до моделі. У комплексі саме ці елементи дозволили забезпечити більш стабільну роботу системи при аналізі великих текстових масивів.

На практиці, процес розробки системи показав, що створення автоматизованого семантичного резюмування для оперативно-службових документів є досить складним багаторівневим завданням, яке потребує поєднання сучасних методів обробки природної мови, оптимізації алгоритмів та правильної організації всієї архітектури системи. Реалізовані під час роботи рішення дозволили суттєво покращити якість аналізу документів, підвищити стабільність роботи системи та зробити фінальні резюме більш точними, логічними та інформативними.

2.4. Тестування системи на документах різного обсягу

Після завершення основної частини розробки було проведено комплексне тестування системи на документах різного обсягу та структури. Основною метою тестування була перевірка стабільності роботи системи, швидкості генерації реферату, а також якості обробки тексту при різному рівні навантаження. Окрему увагу приділили тому, як система поводиться з великими оперативно-службовими документами, оскільки саме вони створюють найбільше навантаження та є найбільш складними для автоматизованого аналізу.

Для тестування були відібрані реальні та синтетичні документи, які умовно поділили на три основні категорії за обсягом: невеликі документи–до 5 сторінок,

середні–від 10 до 25 сторінок та великі–понад 30–40 сторінок. Такий підхід дозволив комплексно оцінити ефективність системи в різних умовах і простежити, як змінюються швидкість роботи, стабільність та якість сформованих рефератів зі збільшенням обсягу текстових даних. Крім того, під час тестування спеціально використовувалися документи з різною структурою–від добре організованих матеріалів з чітким логічним поділом на розділи до великих неструктурованих текстів із великою кількістю другорядної інформації.

Під час роботи з невеликими документами система продемонструвала найкращі результати. Обробка тексту відбувалася швидко, практично без затримок, а сформовані реферати були логічними, добре структурованими та досить точно передавали основний зміст документа. Завдяки невеликому обсягу модель могла аналізувати текст майже повністю, без суттєвої втрати контексту, що позитивно впливало на якість кінцевого результату. Час генерації залишався мінімальним, а система працювала стабільно навіть при обробці кількох документів поспіль.

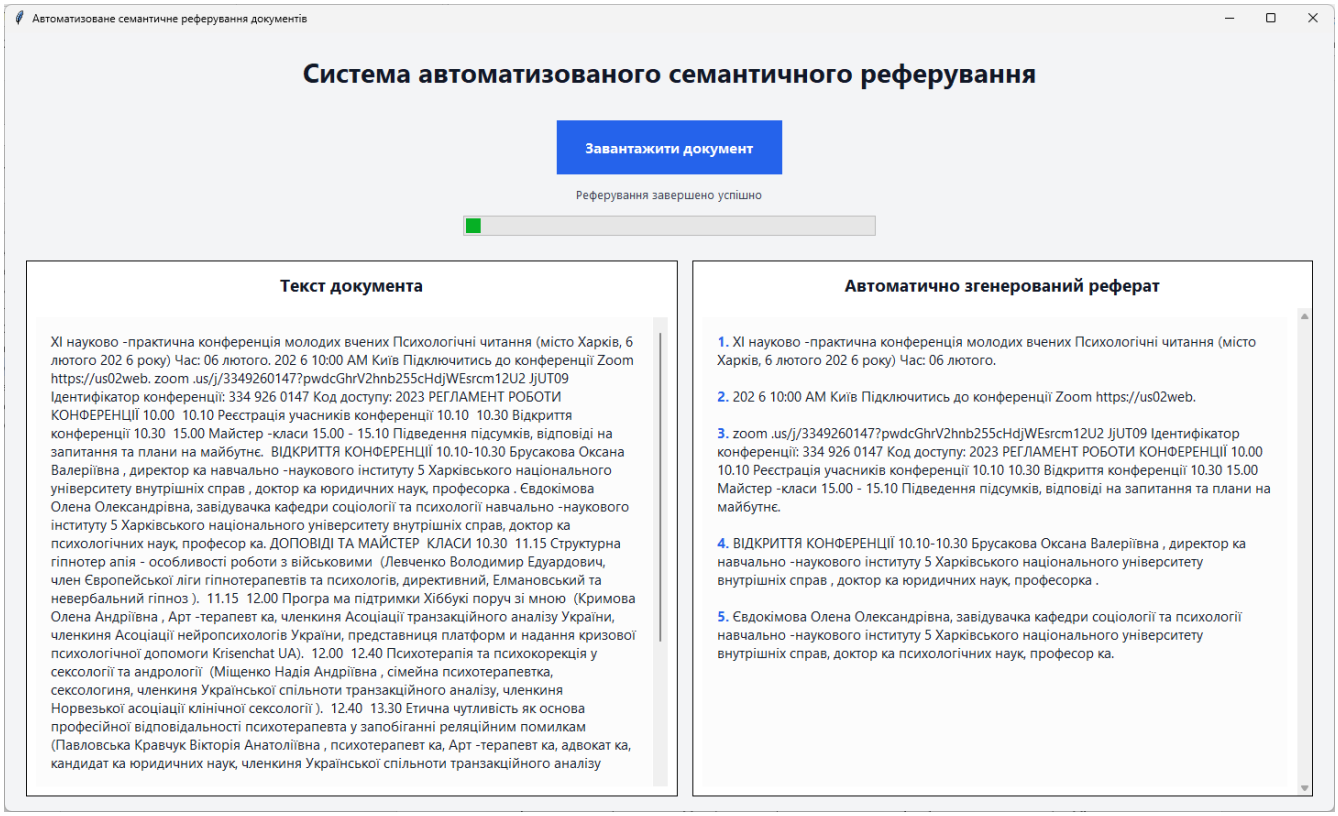
При тестуванні документів середнього обсягу навантаження на систему вже помітно зростало. Збільшувалася кількість текстових фрагментів, які потрібно було обробляти окремо, через що загальний час формування фінального реферату ставав відчутно довшим. Незважаючи на це, система продовжувала працювати стабільно і в більшості випадків формувала якісні резюме без критичних втрат інформації. Однак уже починали з’являтися незначні недоліки: окремі повтори або менш точна передача деяких деталей документа.

Найбільші труднощі, як і очікувалося, виникали під час обробки великих документів. У таких випадках система виконувала значний обсяг проміжних операцій, що суттєво збільшувало час генерації фінального реферату. Зі зростанням кількості chunk-ів зростало і загальне навантаження на мовну модель. При роботі з дуже великими текстами іноді спостерігалося часткове зниження якості: окремі важливі деталі могли втрачатися, деякі факти передавалися менш точно, а в окремих випадках до реферату потрапляла другорядна інформація.

Окремо проводилося тестування стабільності системи при тривалому навантаженні. Для цього здійснювалася послідовна обробка кількох великих документів без перезапуску системи. Було встановлено, що зі збільшенням кількості запитів до моделі загальний час генерації поступово зростає. Для пом’якшення цього ефекту були проведені додаткові оптимізації: вдосконалено проміжні етапи обробки, зменшено кількість зайвих звернень до моделі та впроваджено кешування результатів аналізу. Це дозволило зробити роботу системи стабільнішою навіть при інтенсивному навантаженні.

Додатково було детально проаналізовано вплив структури документа на якість сформованого реферату. Найкращі результати система показувала при роботі з документами, які мають чіткий логічний поділ на розділи та послідовний виклад інформації. У таких випадках модель краще зберігала контекст і точніше формувала підсумок. Натомість великі неструктуровані тексти вимагали значно більше часу на аналіз і частіше призводили до включення другорядної інформації або менш точної передачі ключових фрагментів.

Загалом проведене тестування підтвердило, що система стабільно працює з документами різного обсягу та здатна формувати досить змістовні й корисні реферати навіть при роботі з великими текстовими матеріалами. [15] Водночас результати чітко показали, що швидкість роботи та якість фінального реферату безпосередньо залежать від обсягу документа, його структури та правильно підібраних параметрів обробки тексту.



На рисунку 2.3 наведено результат роботи системи після завершення процесу автоматичного семантичного реферування документа.

У лівій частині інтерфейсу користувач бачить оригінальний текст документа в його первозданному вигляді. Праворуч же розміщено автоматично сформований структурований реферат, який система створила на основі глибокого семантичного аналізу. Таке паралельне відображення оригіналу та обробленого результату виявляється дуже зручним: людина може в будь-який момент порівняти вихідний матеріал з автоматичним узагальненням, швидко оцінити якість роботи системи та внести необхідні правки.

Під час генерації реферату система проводить комплексний аналіз усього змісту документа. Вона визначає ключові факти, основні події, найважливіші тези, а також логічні зв’язки між різними частинами інформації. Завдяки цьому формується стислий, але при цьому структурований і логічно зв’язний виклад суті документа зі збереженням найбільш значущих фрагментів оригінального тексту.

Використання сучасних великих мовних моделей дозволяє системі значно краще враховувати контекст, розуміти нюанси викладу та формувати результат, який читається природно і зрозуміло. Це суттєво відрізняє її від традиційних методів автоматичного скорочення тексту, які часто просто механічно вирізали речення, втрачаючи при цьому загальну логіку та важливі взаємозв’язки.

Такий підхід особливо цінний при роботі з оперативно-службовими документами. Вони, як правило, мають великий обсяг, містять велику кількість фактів, деталей, пояснень і взаємопов’язаних подій. Повністю прочитати й проаналізувати такий матеріал вручну вимагає багато часу та зусиль. Автоматизоване семантичне реферування дозволяє суттєво скоротити цей час, зменшити навантаження на співробітників і прискорити прийняття рішень.

Приклад на рисунку 2.3 також добре демонструє, що система ефективно справляється навіть із великими текстовими масивами. Незважаючи на значний обсяг документа, вона продовжує коректно виділяти найсуттєвішу інформацію, зберігати ключові логічні ланцюжки та формувати цілісне уявлення про зміст.

Окремо варто відзначити переваги структурованого формату подання реферату. Завдяки чіткій організації матеріалу користувач може дуже швидко пробігтися поглядом по ключових пунктах, одразу зрозуміти головне і отримати повне загальне уявлення про документ, не витрачаючи час на детальне опрацювання кожного абзацу оригіналу.

Висновки до другого розділу

У результаті виконання дослідницького розділу було детально проаналізовано процес розробки системи автоматизованого семантичного резюмування оперативно-службових документів, її архітектурні рішення, основні технічні труднощі, які виникали під час реалізації, а також результати практичного тестування. Проведена робота показала, що створення ефективної системи такого типу є досить складним і багатогранним завданням. Оперативно-службові документи характеризуються великим обсягом, складною структурою, високою

інформаційною насиченістю та підвищеними вимогами до точності обробки. Саме тому під час розробки увага приділялася не лише якості формування реферату, а й стабільності роботи системи, збереженню контексту, мінімізації втрат інформації та можливості ефективно працювати з великими текстовими масивами.

Під час дослідження було чітко встановлено, що використання великих мовних моделей «в лоб», без додаткових механізмів обробки, не дозволяє якісно працювати з об’ємними документами. Обмеження контекстного вікна та значне обчислювальне навантаження створюють серйозні перешкоди. Для подолання цих обмежень у системі були реалізовані сучасні підходи до обробки великих текстів, зокрема механізм chunking, перекриття (overlap) між фрагментами, технологія Retrieval-Augmented Generation (RAG) та багатоступенева генерація резюме. Застосування цих методів дозволило організувати поетапний аналіз документа, суттєво зменшити ризик втрати важливої інформації та підвищити логічність і цілісність фінального результату.

Значну увагу було приділено архітектурі програмної системи. Реалізований модульний підхід дав можливість розділити всю систему на окремі незалежні компоненти, кожен з яких відповідає за конкретний етап обробки: зчитування документів, попередню очистку тексту, генерацію векторних представлень, семантичний аналіз та формування фінального реферату. Така архітектура значно спростила процес розробки, тестування та подальшого вдосконалення системи, зробила її більш гнучкою та зручною в підтримці.

У ході роботи також було виявлено та вирішено низку характерних проблем, притаманних великим мовним моделям. Серед найбільш помітних – втрата контексту між фрагментами, дублювання інформації у фінальному рефераті, високе навантаження на систему при обробці великих документів, а також явище «галюцинацій» (генерація неточної або вигаданої інформації). Для мінімізації цих недоліків були впроваджені додаткові механізми обробки тексту, ретельно вдосконалені prompt-запити, оптимізована взаємодія з моделлю та посилена

попередня підготовка документів. Практика підтвердила, що якість очищення та нормалізації тексту, а також правильно сформульовані інструкції для моделі відіграють вирішальну роль у кінцевій точності та логічності резюме.

Проведене комплексне тестування системи підтвердило її працездатність і достатню ефективність при роботі з документами різного обсягу. Найкращі результати були досягнуті на невеликих та середніх документах з чіткою структурою, де система забезпечувала швидку генерацію та високу якість рефератів. При роботі з великими текстовими матеріалами навантаження помітно зростало, збільшувався час обробки, а в окремих випадках з'являлися незначні втрати деталей або повтори інформації. Проте навіть за таких умов система залишалася стабільною і продовжувала формувати змістовні, логічно зв'язані резюме, які коректно передавали основний зміст документа.

Додатково було встановлено, що структура вихідного документа має значний вплив на якість автоматизованого аналізу. Матеріали з чітким логічним поділом на розділи та послідовним викладом обробляються значно ефективніше. Натомість великі неструктуровані тексти вимагають більше часу та ресурсів і частіше створюють труднощі для моделі. Це ще раз підкреслило важливість етапу попередньої нормалізації та очищення тексту.

Загалом результати проведеного дослідження підтвердили, що поєднання великих мовних моделей із сучасними методами обробки великих текстів дозволяє ефективно автоматизувати процес семантичного резюмування оперативно-службових документів. Розроблена система продемонструвала здатність працювати з великими обсягами інформації, формувати структуровані та корисні реферати, а також суттєво скорочувати час аналізу документів. Усе це свідчить про перспективність використання сучасних технологій штучного інтелекту для автоматизації інформаційно-аналітичних процесів у державних установах та організаціях.

Список використаних джерел

1. ДСТУ ISO/IEC 18045:2015 Інформаційні технології. Методи захисту. Методологія оцінювання безпеки IT (ISO/IEC 18045:2008, IDT). Київ: «УкрНДНЦ», 2018.

2. ДСТУ ISO/IEC 33001:2016 Інформаційні технології. Оцінювання процесу. Поняття та термінологія (ISO/IEC 33001:2015, IDT). Київ: «УкрНДНЦ», 2017.

3. ДСТУ ISO 30300:2015. Інформація та документація. Системи керування документами. Основні положення і словник термінів (ISO 30300:2011, IDT). Київ: «УкрНДНЦ», 2017.

4. ДСТУ ISO/IEC 2382:2017 (ISO/IEC 2382:2015, IDT). Інформаційні технології. Словник термінів. Київ: «УкрНДНЦ», 2017.

5. ДСТУ ISO/IEC TR 10032:2012. Інформаційні технології. Еталонна модель керування даними (ISO/IEC TR 10032:2003, IDT)

6. Готування інформаційно-аналітичних документів в органах внутрішніх справ / П. П. Підюков, В. Д. Сущенко, О. А. Лупало та ін. Київ : ВПЦ МВС України, 2005. 51 с.

7. Дабіджа Д. В. Використання обліків та автоматизованих систем при розслідуванні кримінальних правопорушень : дис. ... канд. юрид. наук. спец. 12.00.09 / Національна академія внутрішніх справ. Київ, 2017. 288 с.

8. Barrios, F., López, F., Argerich, L., & Wachenchauzer, R. (2016). Variations of the similarity function of TextRank for automated summarization. arXiv preprint. [Електронний ресурс]: https://doi.org/10.48550/arXiv.1602.03606

9. Dhaini, M., Erdogan, E., Bakshi, S., & Kasneci, G. (2024). Explainability meets text summarization: A survey. In Proceedings of the 17th International Conference on Natural Language Generation (pp. 631–645). [Електронний ресурс]: https://doi.org/10.18653/v1/2024.inlg-main.49

10. Erkan, G., & Radev, D. R. (2004). LexRank: Graph-based lexical centrality AS salience in text summarization. Journal of Artificial Intelligence Research, 22, 457–479. [Електронний ресурс]: https://doi.org/10.1613/jair.10396

11. Koto, F., & Lau, J. H. Y. (2024). Large language models for meta-evaluation of summaries. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (pp. 1–15). [Електронний ресурс]: https://doi.org/10.18653/v1/2024.naacl-long.1

12. Laban, P., Hsu, C., Kottur, S., & Choi, Y. (2022). SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. Transactions of the Association for Computational Linguistics, 10, 163–177. [Електронний ресурс]: https://aclanthology.org/2022.tacl-1.10/

13. Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing order into texts. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (pp. 404–411). [Електронний ресурс]: https://doi.org/10.3115/W04-3252

14. Nenkova, A., & McKeown, K. (2011). Automatic summarization. Foundations and Trends in Information Retrieval, 5(2–3), 103–233. [Електронний ресурс]: https://doi.org/10.1561/1500000015

15. Pasichnyk, V., & Yaromych, M. (2025). Automated generation of technical documentation in IT using large language models. Studia Methodologica, 59, 250–273. [Електронний ресурс]: https://doi.org/10.32782/2307-1222.2025-59-22

16. Pasichnyk, V., & Yaromych, M. (2025). Comparative analysis of large language models in solving linguistic tasks. Current Issues of the Humanities. Linguistics. Literary [Електронний ресурс]: Studies, 89(1), 288–305. https://doi.org/10.24919/2308-4863/89-1-41

17. Pasichnyk, V., & Yaromych, M. (2025). Features of genre classification of literature using large language models. Folium, 6, 132–143. [Електронний ресурс]: https://doi.org/10.32782/folium/2025.6.19

18. Pasichnyk, V., & Yaromych, M. (2025). Large language models and ontologies in philological research: An analytical review of sources. Current Issues of the Humanities. Linguistics. Literary Studies, 83(3), 236–250. [Електронний ресурс]: https://doi.org/10.24919/2308-4863/83-3-35

19. Pasichnyk, V., & Yaromych, M. (2025). Methods and tools of large language models for conceptualizing the subject area "artificial intelligence". In Science in the Modern World: Innovations and Challenges: Proceedings of the IX International Scientific and Practical Conference (pp. 353–358). URL: [Електронний ресурс]: https://sci-conf.com.ua/wp-content/uploads/2025/05/science-in-the-modern-world-innovations-and-challenges-15-17.05.2025.pdf

20. Pasichnyk, V., Yaromych, M., Pavliv, I., & Kunanets, N. (2025). Information technology for self-analysis of large language model parameters. Scientific Bulletin of UNFU, 35(5), 130–144. [Електронний ресурс]: https://doi.org/10.36930/40350515

ДОДАТОК А

МОДУЛЬ ГРАФІЧНОГО ІНТЕРФЕЙСУ СИСТЕМИ АВТОМАТИЗОВАНОГО СЕМАНТИЧНОГО РЕФЕРУВАННЯ ДОКУМЕНТІВ (MyAPP.py)

```
import tkinter as tk
from tkinter import filedialog, scrolledtext, messagebox
from tkinter.ttk import Progressbar
from extract import read_txt, read_docx, read_pdf
from summarizer import generate_summary
def open_file():
    file_path = filedialog.askopenfilename(
        title="Вибір документа",
        filetypes=[
            ("Supported Files", "*.txt *.docx *.pdf"),
            ("Text Files", "*.txt"),
            ("Word Documents", "*.docx"),
            ("PDF Files", "*.pdf"),
            ("All Files", "*.*")
        ]
    )
    if not file_path:
        return

    try:
        status_label.config(
            text="Зчитування документа..."
        )
        progress.start()
```

```
root.update(
if file_path.endswith(".txt"):
    text = read_txt(file_path)
elif file_path.endswith(".docx"):
    text = read_docx(file_path)
elif file_path.endswith(".pdf"):
    text = read_pdf(file_path)
else:
    messagebox.showerror(
        "Помилка",
        "Непідтримуваний формат файлу"
    )
    return
document_box.delete(1.0, tk.END)
document_box.insert(
    tk.END,
    text
)
status_label.config(
    text="Виконується AI-аналіз документа..."
)
root.update()
summary, keywords = generate_summary(text)
progress.stop()
summary_box.delete(1.0, tk.END)
summary = summary.replace("•", "\n• ")
blocks = summary.split("\n")
counter = 1
```

```
for block in blocks:
    block = block.strip()
    if not block:
        continue
    if len(block) < 10:
        continue
    if (
        block.startswith("-")
        or block.startswith("•")
    ):

        summary_box.insert(
            tk.END,
            block + "\n\n"
        )
    else:
        summary_box.insert(
            tk.END,
            f"{counter}. ",
            "number"
        )
        summary_box.insert(
            tk.END,
            block + "\n\n"
        )
        counter += 1
status_label.config(
    text="Реферування завершено успішно"
```

```
        )
    except Exception as e:
        progress.stop()

        messagebox.showerror(
            "Помилка",
            str(e)
        )
        status_label.config(
            text="Помилка"
        )
root = tk.Tk()
root.title(
    "Автоматизоване семантичне реферування документів"
)
root.geometry("1450x850")
root.configure(
    bg="#f3f4f6"
)
header = tk.Label(
    root,
    text="Система автоматизованого семантичного реферування",
    font=("Segoe UI", 22, "bold"),
    bg="#f3f4f6",
    fg="#111827"
)
header.pack(pady=20)
upload_btn = tk.Button(
```

```
    root,
    text="Завантажити документ",
    command=open_file,
    font=("Segoe UI", 12, "bold"),
    bg="#2563eb",
    fg="white",
    padx=25,
    pady=12,
    relief="flat",
    cursor="hand2"
)
upload_btn.pack(pady=10)
status_label = tk.Label(
    root,
    text="Очікування завантаження документа",
    font=("Segoe UI", 10),
    bg="#f3f4f6",
    fg="#374151"
)

status_label.pack()
progress = Progressbar(
    root,
    orient="horizontal",
    length=450,
    mode="indeterminate"
)
progress.pack(pady=12)
```

```
main_frame = tk.Frame(
    root,
    bg="#f3f4f6"
)

main_frame.pack(
    fill="both",
    expand=True,
    padx=15,
    pady=15
)
doc_frame = tk.Frame(
    main_frame,
    bg="white",
    bd=1,
    relief="solid"
)
doc_frame.pack(
    side="left",
    fill="both",
    expand=True,
    padx=8
)
doc_label = tk.Label(
    doc_frame,
    text="Текст документа",
    font=("Segoe UI", 14, "bold"),
    bg="white",
```

```
    fg="#111827"
)
doc_label.pack(pady=10)
document_box = scrolledtext.ScrolledText(
    doc_frame,
    wrap=tk.WORD,
    font=("Segoe UI", 11),
    padx=15,
    pady=15,
    bg="#fcfcfc",
    fg="#111827",
    relief="flat"
)
document_box.pack(
    fill="both",
    expand=True,
    padx=10,
    pady=10
)
summary_frame = tk.Frame(
    main_frame,
    bg="white",
    bd=1,
    relief="solid"
)
summary_frame.pack(
    side="right",
    fill="both",
```

```
    expand=True,
    padx=8
)
summary_label = tk.Label(
    summary_frame,
    text="Автоматично згенерований реферат",
    font=("Segoe UI", 14, "bold"),
    bg="white",
    fg="#111827"
)
summary_label.pack(pady=10)
summary_box = tk.Text(
    summary_frame,
    wrap=tk.WORD,
    font=("Segoe UI", 11),
    padx=15,
    pady=15,
    bg="#fcfcfc",
    fg="#111827",
    relief="flat"
)
summary_scroll = tk.Scrollbar(
    summary_frame,
    command=summary_box.yview
)
summary_box.configure(
    yscrollcommand=summary_scroll.set
)
```

```
summary_scroll.pack(
    side="right",
    fill="y"
)
summary_box.pack(
    fill="both",
    expand=True,
    padx=10,
    pady=10
)
summary_box.tag_config(
    "number",
    font=("Segoe UI", 11, "bold"),
    foreground="#2563eb"
)
footer = tk.Label(
    root,
    text="LLM Semantic Summarization System | FLAN-T5 + NLP",
    font=("Segoe UI", 9),
    bg="#f3f4f6",
    fg="#6b7280"
)
footer.pack(pady=10)
root.mainloop()
```

ДОДАТОК В

МОДУЛЬ ЗЧИТУВАННЯ ТА ПОПЕРЕДНЬОЇ ОБРОБКИ ДОКУМЕНТІВ РІЗНИХ ФОРМАТІВ (EXTRACT.PY)

```
import docx
import PyPDF2
import re
def read_txt(file_path):
    with open(
        file_path,
        "r",
        encoding="utf-8",
        errors="ignore"
    ) as file:
        text = file.read()
    return clean_text(text)
def read_docx(file_path):
    doc = docx.Document(file_path)
    full_text = []
    for paragraph in doc.paragraphs:
        text = paragraph.text.strip()
        if text:
            full_text.append(text)
    return clean_text(
        "\n".join(full_text)
    )
def read_pdf(file_path):
    text = ""
    with open(file_path, "rb") as file:
```

```
        reader = PyPDF2.PdfReader(file)
        for page in reader.pages:
            try:
                page_text = page.extract_text()
                if page_text:
                    text += page_text + "\n"
            except:
                pass
    return clean_text(text)
def clean_text(text):
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'\n+', '\n', text)
    text = re.sub(
        r'[^\w\s.,:;!?()%/\-\n]',
        '',
        text
    )
    text = re.sub(
        r'МІНІСТЕРСТВО.*?Виконали:',
        '',
        text,
        flags=re.DOTALL | re.IGNORECASE
    )
    return text.strip()
```

ДОДАТОК С

МОДУЛЬ ГЕНЕРАЦІЇ СЕМАНТИЧНОГО РЕФЕРАТУ ДОКУМЕНТІВ З ВИКОРИСТАННЯМ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ (SUMMARIZER.PY)

```
import re
from transformers import pipeline
summarizer = pipeline(
    "text-generation",
    model="google/flan-t5-small",
    max_new_tokens=400
)
def clean_text(text):
    text = re.sub(r'\s+', ' ', text
    text = re.sub(
        r'[^\w\s.,:;!?()%/-]',
        '',
        text
    )
    return text.strip()
def chunk_text(text, chunk_size=1800):

    chunks = []
    for i in range(0, len(text), chunk_size):
        chunks.append(
            text[i:i + chunk_size]
        )
    return chunks
def extract_keywords(text):
```

```
words = re.findall(
    r'\b[а-яА-Яіїєґa-zA-Z]{4,}\b',
    text.lower()
)
stopwords = {
    "та", "для", "того", "після",
    "який", "була", "було",
    "цей", "ця", "вони",
    "його", "також",
    "this", "that", "with",
    "from", "have", "shall",
    "their", "there"
}
filtered = []
for word in words:
    if word not in stopwords:
        filtered.append(word)
freq = {}
for word in filtered:
    freq[word] = freq.get(word, 0) + 1
sorted_words = sorted(
    freq.items(),
    key=lambda x: x[1],
    reverse=True
)
keywords = []
for word, count in sorted_words[:10]:
    keywords.append(word)
```

```
    return keywords
def generate_summary(text):
    text = clean_text(text)
    text = text[:30000]
    chunks = chunk_text(text)
    summaries = []
    for chunk in chunks[:10]:
        prompt = f"""
        Створи короткий структурований витяг документа.
        Виділи:
        - основну суть
        - ключові факти
        - важливі події
        - результати
        Текст:
        {chunk}

        Реферат:
        """
        try:
            result = summarizer(
                prompt,
                do_sample=False
            )
            generated = result[0]["generated_text"]
            generated = generated.replace(
                prompt,
```

```
            """
        ).strip()
        if generated:
            summaries.append(generated)
    except:
        pass
if not summaries:
    sentences = re.split(
        r'(?<=[.!?])\s+',
        text
    )
    summaries = sentences[:5]
final_summary = "\n\n".join(summaries)
keywords = extract_keywords(text)
return final_summary, keywords
```