МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

СОЗАНСЬКИЙ АНДРІЙ БОГДАНОВИЧ

ПРОГРАМНИЙ МОДУЛЬ ПОШУКУ РЕЛЕВАНТНОЇ ІНФОРМАЦІЇ НА ОСНОВІ RAG / SOFTWARE MODULE FOR SEARCHING FOR RELEVANT INFORMATION BASED ON RAG

спеціальність: 122 - Комп'ютерні науки
Освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
А.Б. Созанський

_____

Науковий керівник
к.т.н, доцент Н.М. Васильків

_____

Кваліфікаційну роботу допущено до захисту
"___"_____20___р.

Завідувач кафедри

_____ Н.В. Дзюбановська

ТЕРНОПІЛЬ - 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
СОЗАНСЬКОМУ Андрію Богдановичу

1. Тема кваліфікаційної роботи **Програмний модуль пошуку релевантної інформації на основі RAG / Software Module for Searching for Relevant Information based on RAG**
керівник роботи к.т.н., доцент Н.М. Васильків
затверджений наказом по університету від 01 грудня 2025 року № 894.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити
– проаналізувати сучасні підходи до створення інтелектуальних систем пошуку інформації на основі методів обробки природної мови та великих мовних моделей;
– дослідити принципи функціонування архітектури RAG та підходи до інтеграції retrieval-компонента з великими мовними моделями;
– проаналізувати методи представлення текстових даних у вигляді embeddings та підходи до реалізації семантичного пошуку інформації;
– виконати аналіз сучасних програмних засобів і фреймворків для розробки систем на основі великих мовних моделей та обґрунтувати вибір інструментів для реалізації програмної системи;
– розробити архітектуру RAG-агента для обробки запитів користувача, пошуку релевантної інформації та генерації відповідей на основі зовнішніх джерел даних;
– реалізувати механізми збереження даних користувача та історії взаємодії із системою;
– розробити користувацький інтерфейс для взаємодії із системою у форматі діалогу та перегляду використаних документів;
– реалізувати модуль спостережуваності для моніторингу роботи системи, збору метрик та аналізу виконання окремих етапів обробки запитів;
– провести оцінювання ефективності retrieval-компонента та дослідити особливості роботи системи в різних сценаріях використання.
5. Перелік графічного матеріалу у роботі
- граф виконання RAG-агента, реалізований за допомогою LangGraph
- схема бази даних модуля збереження профілю користувача та історії документів

- схема початкової ініціалізації компонентів системи
- схема основних програмних компонентів RAG-агента
- архітектура модуля збереження даних користувача та історії взаємодії
- архітектура модуля спостережуваності

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Огляд літературних джерел відповідно до предметної області та складання плану роботи	до 01.01.2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02.2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05.2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06.2026 р.	

Студент _____ А.Б. Созанський

Керівник роботи _____ к.т.н., доцент Н.М.Васильків

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль пошуку релевантної інформації на основі RAG» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом 62 сторінок і містить 14 ілюстрацій, 3 додатки та 35 використаних джерел.

Метою роботи є дослідження підходів до семантичного пошуку інформації та розроблення прогнозного програмного модуля для автоматизованого відбору та генерації релевантних відповідей на основі архітектури RAG (Retrieval-Augmented Generation) та аналізу інформаційних потреб користувача.

Методами розроблення обрано метод аналізу (для дослідження існуючих підходів до інтелектуального пошуку та обробки природної мови), методи об'єктно-орієнтованого аналізу та проектування (для побудови архітектури пошукового модуля), методи математичного та лінгвістичного моделювання (для обчислення семантичної близькості векторів текстових даних) та метод тестування (для оцінювання точності пошуку й генерації відповідей).

Внаслідок виконання роботи обґрунтовано раціональний підхід до інтелектуального пошуку знань із використанням великих мовних моделей, а також розроблено прогнозний програмний модуль, який дає змогу автоматично здійснювати семантичний відбір фрагментів документів, формувати релевантний контекст та генерувати точні відповіді без ефекту галюцинацій мовної моделі.

Результати дослідження можуть бути використані в інформаційно-довідкових системах, корпоративних базах знань, аналітичних центрах, а також розробниками інтелектуальних пошукових сервісів та чат-асистентів для автоматизації роботи з великими масивами текстових даних.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, СЕМАНТИЧНИЙ ПОШУК, АРХІТЕКТУРА RAG, ВЕКТОРНА БАЗА ДАНИХ, ВЕЛИКІ МОВНІ МОДЕЛІ.

ANNOTATION

Qualification work on the topic «Software Module for Searching for Relevant Information based on RAG» for Bachelor's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 62 pages and contains 14 figures, 3 annex and 35 sources.

The purpose of the work is to research approaches to semantic information retrieval and to develop a predictive software module for automated selection and generation of relevant answers based on the RAG (Retrieval-Augmented Generation) architecture and user information needs analysis.

Research methods are analysis (to study existing approaches to intelligent retrieval and natural language processing), object-oriented analysis and design methods (to build the retrieval module architecture), mathematical and linguistic modeling methods (to calculate the semantic similarity of text data vector representations), and software testing methods (to evaluate the accuracy of information retrieval and answer generation).

As a result of the work, a rational approach to intelligent knowledge retrieval using large language models was substantiated. Furthermore, a predictive software module was developed that enables the automated semantic selection of document chunks, dynamic context construction, and generation of precise answers while mitigating language model hallucinations.

The results of the research can be used in information and reference systems, corporate knowledge bases, analytical centers, and by developers of intelligent search services and chat assistants to automate workflows with large-scale text datasets.

Keywords: SOFTWARE MODULE, SEMANTIC SEARCH, RAG ARCHITECTURE, VECTOR DATABASE, LARGE LANGUAGE MODELS.

ЗМІСТ

ВСТУП

Стрімкий розвиток інформаційних технологій та постійне зростання обсягів цифрових даних зумовлюють необхідність створення ефективних інтелектуальних систем обробки інформації. У сучасному інформаційному середовищі значна частина даних представлена у вигляді текстових документів, що потребує застосування спеціалізованих методів аналізу, пошуку та узагальнення інформації. У зв’язку з цим особливої актуальності набувають технології штучного інтелекту та обробки природної мови, які дозволяють автоматизувати процеси роботи з великими масивами текстових даних.

Одним із ключових напрямів розвитку інтелектуальних інформаційних систем є використання великих мовних моделей (LLM). Такі моделі здатні аналізувати текстову інформацію, визначати семантичні зв’язки між словами, узагальнювати дані та формувати змістовні відповіді на запити користувачів. Завдяки цьому великі мовні моделі активно використовуються у різних галузях, зокрема в системах автоматичного перекладу, інформаційного пошуку, створенні чат-ботів, рекомендаційних системах та системах підтримки прийняття рішень.

Разом з тим традиційні мовні моделі мають певні обмеження. Вони формують відповіді на основі знань, отриманих під час навчання, і не мають прямого доступу до актуальних джерел інформації. Це може призводити до ситуацій, коли модель генерує неповні або неточні відповіді. Для подолання цих обмежень у сучасних інформаційних системах використовується підхід Retrieval-Augmented Generation (RAG), який поєднує можливості генерації тексту великими мовними моделями з механізмами пошуку релевантної інформації у зовнішніх джерелах даних.

Архітектура Retrieval-Augmented Generation передбачає інтеграцію мовної моделі з системами пошуку інформації та базами знань. У межах такого підходу система спочатку виконує пошук релевантних документів у базі даних, після чого використовує мовну модель для формування відповіді на основі знайденої інформації. Це дозволяє підвищити точність отриманих результатів, зменшити

ризик генерації некоректної інформації та забезпечити використання актуальних даних під час обробки запитів.

У процесі створення систем на основі Retrieval-Augmented Generation важливу роль відіграють методи представлення текстових даних у вигляді векторних представлень, використання векторних баз даних для зберігання інформації, а також реалізація ефективних алгоритмів семантичного пошуку. Крім того, сучасні програмні фреймворки, такі як LangChain та LangGraph, дозволяють створювати складні програмні системи, що інтегрують мовні моделі з різними джерелами даних та забезпечують реалізацію агентних підходів до обробки інформації.

Метою кваліфікаційної роботи є дослідження сучасних підходів до побудови інтелектуальних систем пошуку інформації на основі технології Retrieval-Augmented Generation, а також розробка програмної системи у вигляді RAG-агента для семантичного пошуку та генерації відповідей на основі зовнішніх джерел знань.

Об’єктом дослідження є процеси пошуку та обробки текстової інформації у сучасних інформаційних системах.

Предметом дослідження є методи та програмні засоби побудови систем пошуку релевантної інформації на основі технології Retrieval-Augmented Generation та великих мовних моделей.

Практичне значення дослідження полягає у розробці концепції архітектури інтелектуальної системи, яка поєднує механізми семантичного пошуку, векторні представлення тексту та генеративні можливості мовних моделей. Отримані результати можуть бути використані під час створення систем автоматизованого аналізу документів, інформаційних чат-ботів, корпоративних систем управління знаннями та інших інтелектуальних інформаційних систем.

1 СТАН ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Розвиток інформаційних технологій та зростання обсягів цифрових даних зумовили необхідність створення ефективних інтелектуальних систем, здатних здійснювати пошук, аналіз та узагальнення інформації. Сучасні інформаційні системи дедалі частіше використовують методи штучного інтелекту, зокрема обробку природної мови (Natural Language Processing), машинне навчання та великі мовні моделі.

Особливої актуальності набувають системи, що поєднують можливості генерації тексту з механізмами пошуку релевантної інформації. Одним із сучасних підходів до вирішення цієї задачі є концепція Retrieval-Augmented Generation (RAG), яка інтегрує великі мовні моделі з системами пошуку знань.

Такий підхід дозволяє створювати інтелектуальні інформаційні системи нового покоління, які можуть працювати з великими масивами текстових даних, забезпечувати доступ до релевантної інформації та формувати узагальнені відповіді на основі знайдених документів. У цьому контексті важливими складовими є формування векторних представлень тексту (embeddings), використання векторних баз даних та розробка архітектур RAG-пайплайнів.

У межах предметної області досліджуються підходи до створення систем, що реалізують автоматизований пошук інформації, аналіз текстових джерел та генерацію змістовних відповідей на основі отриманих результатів.

1.1.1 Інтелектуальні інформаційні системи

Інтелектуальні інформаційні системи є одним із ключових напрямів розвитку сучасних інформаційних технологій. Такі системи призначені для автоматизованої обробки інформації з використанням методів штучного інтелекту, машинного навчання та аналізу даних. Основною їхньою метою є підтримка процесів прийняття рішень, аналіз великих обсягів інформації та забезпечення ефективного доступу до знань [8].

На відміну від традиційних інформаційних систем, інтелектуальні системи здатні не лише зберігати та передавати дані, але й виконувати їх інтерпретацію, аналіз та узагальнення. Це досягається завдяки використанню алгоритмів машинного навчання, моделей обробки природної мови та методів семантичного аналізу тексту.

Інтелектуальні інформаційні системи широко застосовуються в різних галузях, зокрема в інформаційному пошуку, автоматичному перекладі, рекомендаційних системах, системах підтримки прийняття рішень та чат-ботах. Однією з важливих характеристик таких систем є здатність працювати з неструктурованими даними, серед яких значну частину становлять текстові документи.

З розвитком технологій штучного інтелекту значного поширення набули системи, що використовують великі мовні моделі для аналізу та генерації тексту. Такі системи здатні виконувати складні завдання, пов’язані з розумінням змісту тексту, узагальненням інформації та формуванням відповідей на природній мові [25] .

1.1.2 Великі мовні моделі у задачах обробки тексту

Великі мовні моделі (LLM) є сучасним класом моделей машинного навчання, які призначені для обробки, аналізу та генерації текстової інформації [4]. Основою таких моделей є архітектура трансформерів, яка забезпечує ефективне врахування контексту під час аналізу послідовностей слів [3, 4].

LLM навчаються на великих корпусах текстових даних, що дозволяє їм формувати статистичні залежності між словами та фразами. Завдяки цьому моделі здатні виконувати широкий спектр завдань обробки природної мови.

Однією з ключових переваг великих мовних моделей є їх універсальність. Одна й та сама модель може використовуватися для різних задач без необхідності створення спеціалізованих алгоритмів для кожного окремого випадку.

Разом з тим, LLM мають певні обмеження. Зокрема, моделі можуть генерувати відповіді, що не ґрунтуються на фактичних даних, а також не мають

доступу до актуальної інформації, яка з’явилася після завершення процесу навчання моделі. Для вирішення цієї проблеми використовуються методи інтеграції мовних моделей із зовнішніми джерелами знань, зокрема підхід Retrieval-Augmented Generation [23].

Retrieval-Augmented Generation (RAG) є сучасною архітектурною концепцією, яка поєднує механізми інформаційного пошуку з можливостями генерації тексту великими мовними моделями. Основна ідея цього підходу полягає у використанні зовнішніх джерел знань під час формування відповіді мовною моделлю [17].

На відміну від традиційних моделей генерації тексту, які покладаються виключно на знання, отримані під час навчання, RAG-системи здійснюють попередній пошук релевантних документів у базі знань. Отримані результати передаються мовній моделі разом із запитом користувача, що дозволяє сформувати більш точну та обґрунтовану відповідь.

Застосування підходу RAG має низку переваг. Система може використовувати актуальні дані, що зберігаються у базах знань або інформаційних сховищах. Зменшується ризик генерації неправдивої або неточної інформації, а також підвищується прозорість процесу формування відповіді, оскільки можна визначити джерела інформації, використані під час генерації тексту [27].

У сучасних інтелектуальних системах RAG використовується для створення чат-ботів, систем підтримки прийняття рішень, корпоративних пошукових систем та інструментів аналізу документів.

1.1.3 Основні компоненти RAG-систем

Архітектура Retrieval-Augmented Generation складається з кількох основних компонентів, які забезпечують обробку інформації на різних етапах роботи системи.

Першим компонентом є модуль підготовки даних. На цьому етапі відбувається збір текстових документів, їх попередня обробка та формування структурованої бази знань. Документи можуть походити з різних джерел,

зокрема з баз даних, веб-ресурсів, наукових статей або внутрішніх інформаційних систем організації.

Другим компонентом є модуль створення векторних представлень тексту. Для цього використовуються спеціалізовані моделі embeddings, які перетворюють текстові фрагменти на числові вектори у багатовимірному просторі.

Третім компонентом є система пошуку релевантної інформації. На цьому етапі здійснюється порівняння векторного представлення запиту користувача з векторами документів у базі знань. Для визначення найбільш релевантних результатів використовуються метрики подібності, зокрема косинусна подібність.

Четвертим компонентом є генеративна модель, яка формує відповідь на основі запиту користувача та знайдених документів. Мовна модель аналізує отриману інформацію та генерує узагальнений текст, що містить відповідь на поставлене питання [1, 9].

1.1.4 Використання векторних представлень тексту

Векторне представлення є одним із ключових елементів сучасних систем обробки природної мови. Вони являють собою числові представлення текстових даних, які відображають семантичний зміст слів, фраз або документів.

Основна ідея використання полягає у тому, що тексти зі схожим змістом мають подібні векторні представлення у багатовимірному просторі. Завдяки цьому стає можливим виконувати ефективний пошук інформації, кластеризацію документів та аналіз семантичної подібності текстів [6].

У системах Retrieval-Augmented Generation векторні представлення використовуються для перетворення документів та запитів користувачів у векторний формат. Після цього виконується пошук найближчих векторів у базі знань, що дозволяє визначити найбільш релевантні документи для конкретного запиту.

Сучасні моделі embeddings створюються на основі нейронних мереж та навчаються на великих текстових корпусах. Вони здатні враховувати контекст

використання слів та відображати складні семантичні залежності між різними текстовими елементами.

1.1.5 Векторні бази даних у системах пошуку інформації

Векторні бази даних є спеціалізованими системами зберігання та обробки даних, призначеними для роботи з векторними представленнями інформації. Їх основною функцією є ефективний пошук найближчих векторів у багатовимірному просторі.

На відміну від традиційних реляційних баз даних, які використовують точні запити, векторні бази даних орієнтовані на виконання пошуку за семантичною подібністю [7]. Це дозволяє знаходити документи, зміст яких близький до запиту користувача, навіть якщо у тексті не використовуються однакові слова.

У сучасних системах обробки природної мови векторні бази даних широко застосовуються для зберігання embeddings документів та виконання семантичного пошуку. Серед найбільш поширених рішень можна виділити FAISS, Chroma та Pinecone [8].

Такі системи забезпечують масштабоване зберігання великих обсягів векторних даних та швидке виконання операцій пошуку. Завдяки цьому вони є важливим компонентом архітектури RAG-систем та інших інтелектуальних систем, що працюють з текстовою інформацією.

1.2 Огляд і аналіз існуючих рішень

Активний розвиток великих мовних моделей та систем обробки природної мови призвів до появи значної кількості програмних інструментів і платформ, призначених для створення інтелектуальних інформаційних систем. Особливо важливу роль у сучасних системах пошуку інформації відіграють фреймворки та програмні платформи, що дозволяють інтегрувати мовні моделі з джерелами даних, системами індексації та механізмами семантичного пошуку.

У контексті створення систем на основі Retrieval-Augmented Generation важливими є інструменти, що підтримують побудову RAG-пайплайнів, роботу з векторними представленнями тексту, а також реалізацію агентних систем. Серед найбільш поширених рішень можна виділити фреймворки LangChain, LangGraph, LlamaIndex, платформи для використання великих мовних моделей, а також спеціалізовані векторні бази даних, такі як FAISS, Chroma та Pinecone [28, 29].

1.2.1 Фреймворк LangChain для побудови LLM-додатків

LangChain є одним із найбільш популярних фреймворків для розробки додатків на основі великих мовних моделей. Основною метою цього інструменту є спрощення інтеграції мовних моделей з різними джерелами даних, сервісами та програмними компонентами.

Фреймворк забезпечує набір інструментів для створення ланцюжків обробки інформації (chains), які дозволяють комбінувати різні етапи взаємодії з мовною моделлю [20]. Завдяки цьому можна реалізовувати складні сценарії обробки запитів, що включають пошук інформації, аналіз документів та генерацію відповідей.

LangChain підтримує інтеграцію з великою кількістю мовних моделей, включаючи моделі OpenAI, Anthropic, Google та інші. Крім того, фреймворк має вбудовану підтримку роботи з embeddings, системами зберігання документів та векторними базами даних.

Однією з ключових можливостей LangChain є реалізація Retrieval-Augmented Generation. Фреймворк дозволяє створювати RAG-пайплайни, які виконують пошук релевантної інформації у базі знань та передають знайдені документи мовній моделі для генерації відповіді.

Також LangChain підтримує створення агентів, які можуть використовувати різні інструменти, виконувати послідовні дії та приймати рішення щодо подальших кроків під час виконання запиту користувача. Це значно розширює можливості систем на основі великих мовних моделей.

1.2.2 Фреймворк LangGraph для створення агентних систем

LangGraph є сучасним фреймворком, призначеним для розробки складних агентних систем на основі великих мовних моделей. Він був створений як розширення екосистеми LangChain та орієнтований на побудову багатокрокових процесів обробки інформації [10].

Основною ідеєю LangGraph є представлення логіки роботи системи у вигляді графа станів. Кожен вузол графа відповідає за виконання певної операції, наприклад виклик мовної моделі, пошук інформації або обробку даних. Переходи між вузлами визначають послідовність виконання дій у системі.

Такий підхід дозволяє створювати більш гнучкі та керовані архітектури агентів, які можуть виконувати складні сценарії обробки запитів. Зокрема, агент може аналізувати проміжні результати, приймати рішення щодо подальших дій та повторно звертатися до джерел інформації.

LangGraph також забезпечує підтримку станів та контексту, що є важливим для систем, які повинні зберігати інформацію про попередні дії під час взаємодії з користувачем. Завдяки цьому стає можливим створення складних інтелектуальних систем, що виконують багатоетапний аналіз інформації.

Використання LangGraph у поєднанні з LangChain дозволяє реалізувати архітектури RAG-агентів, у яких процес пошуку інформації та генерації відповіді керується логікою графових переходів.

1.2.3 Платформи великих мовних моделей

Важливим компонентом сучасних інтелектуальних систем є платформи, що надають доступ до великих мовних моделей. Такі платформи забезпечують можливість використання потужних моделей машинного навчання без необхідності їх самостійного навчання та підтримки.

Однією з найбільш поширених платформ є OpenAI, яка надає доступ до сімейства мовних моделей GPT [19]. Ці моделі можуть виконувати широкий спектр завдань обробки природної мови, включаючи генерацію тексту, узагальнення документів, переклад та відповіді на запитання.

Крім OpenAI, існують також інші платформи, що пропонують мовні моделі для інтеграції в програмні системи. До них належать рішення від Google, Meta та Anthropic. Багато з цих моделей можуть використовуватися у складі RAG-систем для генерації відповідей на основі знайдених документів [26].

Використання хмарних платформ для роботи з мовними моделями дозволяє значно спростити процес розробки інтелектуальних систем, оскільки розробники можуть зосередитися на логіці обробки інформації та інтеграції різних компонентів системи.

1.2.4 Фреймворк LlamaIndex

LlamaIndex є спеціалізованим фреймворком для створення систем пошуку інформації та RAG-додатків. Основною метою цього інструменту є спрощення процесу інтеграції мовних моделей із зовнішніми джерелами даних.

Фреймворк забезпечує інструменти для індексації документів, створення embeddings та виконання семантичного пошуку. LlamaIndex підтримує роботу з різними типами джерел даних, включаючи текстові файли, бази даних, веб-ресурси та хмарні сховища [16].

Однією з ключових переваг LlamaIndex є можливість створення гнучких індексів для різних типів інформації. Це дозволяє ефективно організовувати структуру бази знань та забезпечувати швидкий доступ до релевантних документів.

Фреймворк також підтримує інтеграцію з популярними векторними базами даних та мовними моделями, що робить його зручним інструментом для створення повноцінних RAG-систем.

1.2.5 Векторні бази даних

Векторні бази даних є важливим елементом сучасних систем семантичного пошуку. Вони забезпечують ефективне зберігання векторних представлень тексту та виконання операцій пошуку найближчих сусідів у багатовимірному просторі.

Однією з найбільш відомих систем є FAISS (Facebook AI Similarity Search). Ця бібліотека була розроблена компанією Meta та призначена для швидкого пошуку подібних векторів у великих наборах даних. FAISS широко використовується у дослідницьких та промислових системах, що працюють з великими обсягами текстової інформації.

Chroma є сучасною векторною базою даних, орієнтованою на використання у системах на основі великих мовних моделей. Вона забезпечує зручні інтерфейси для роботи з embeddings та легко інтегрується з такими фреймворками, як LangChain та LlamaIndex.

Pinecone є хмарною платформою для зберігання та обробки векторних даних. Основною перевагою цього рішення є висока масштабованість та можливість обробки великих обсягів даних у режимі реального часу. Pinecone часто використовується у комерційних системах, де необхідно забезпечити високу продуктивність та надійність пошуку інформації.

1.2.6 Порівняльний аналіз існуючих підходів

Розглянуті інструменти та платформи виконують різні функції у процесі створення систем на основі Retrieval-Augmented Generation. Фреймворки LangChain та LangGraph забезпечують реалізацію логіки обробки запитів та взаємодії між компонентами системи.

LlamaIndex орієнтований на організацію доступу до джерел даних та створення ефективних механізмів індексації документів. Платформи великих мовних моделей забезпечують генерацію тексту та аналіз інформації, тоді як векторні бази даних відповідають за швидкий семантичний пошук документів.

У практичних системах ці інструменти зазвичай використовуються у поєднанні. Наприклад, RAG-система може використовувати LangChain для побудови пайплайну обробки запитів, LlamaIndex для індексації документів, векторну базу даних для зберігання embeddings та мовну модель для генерації відповіді.

Таким чином, сучасні програмні інструменти забезпечують широкий набір можливостей для створення інтелектуальних систем пошуку інформації. Вибір

конкретних технологій залежить від вимог до продуктивності, масштабованості та функціональності розроблюваної системи.

1.3 Постановка задачі дослідження

Стрімке зростання обсягів текстової інформації у цифровому середовищі створює нові виклики для систем пошуку та обробки даних. Традиційні інформаційно-пошукові системи здебільшого базуються на ключових словах або статистичних методах аналізу тексту, що не завжди дозволяє ефективно визначати семантичну відповідність між запитом користувача та наявними інформаційними ресурсами.

Сучасні підходи до обробки текстової інформації дедалі частіше використовують методи штучного інтелекту, зокрема великі мовні моделі (LLM) та семантичний пошук на основі векторних представлень тексту. Такі технології дозволяють підвищити якість пошуку інформації, а також забезпечити можливість генерації змістовних відповідей на основі знайдених документів.

Одним із найбільш перспективних підходів у цьому напрямі є архітектура Retrieval-Augmented Generation, яка поєднує можливості інформаційного пошуку з генеративними моделями обробки природної мови. У межах такого підходу система спочатку виконує пошук релевантних документів у базі знань, після чого використовує мовну модель для формування узагальненої відповіді на основі отриманої інформації.

Незважаючи на значні переваги цього підходу, розробка ефективних RAG-систем потребує вирішення низки технічних і методичних завдань. Зокрема, важливими є питання організації бази знань, створення векторних представлень тексту, вибору ефективних алгоритмів пошуку інформації, а також інтеграції мовних моделей із системами обробки даних.

Тому дослідження методів створення інтелектуальних систем пошуку інформації, які використовують архітектуру Retrieval-Augmented Generation та забезпечують можливість автоматизованого аналізу текстових даних, є актуальним.

Метою даної роботи є дослідження та розробка концепції програмного модуля пошуку релевантної інформації на основі технології Retrieval-Augmented Generation.

Модуль повинен забезпечувати інтеграцію механізмів семантичного пошуку з можливостями великих мовних моделей для формування змістовних та контекстно обґрунтованих відповідей на запити користувачів.

Для досягнення поставленої мети було визначено такі завдання:

1. Провести аналіз сучасних підходів до створення інтелектуальних систем пошуку інформації на основі методів обробки природної мови та великих мовних моделей.

2. Дослідити принципи функціонування архітектури Retrieval-Augmented Generation та підходи до інтеграції retrieval-компонента з великими мовними моделями.

3. Проаналізувати методи представлення текстових даних у вигляді embeddings та підходи до реалізації семантичного пошуку інформації.

4. Виконати аналіз сучасних програмних засобів і фреймворків для розробки систем на основі великих мовних моделей та обґрунтувати вибір інструментів для реалізації програмної системи.

5. Розробити архітектуру RAG-агента для обробки запитів користувача, пошуку релевантної інформації та генерації відповідей на основі зовнішніх джерел даних.

6. Реалізувати механізми збереження даних користувача та історії взаємодії із системою.

7. Розробити користувацький інтерфейс для взаємодії із системою у форматі діалогу та перегляду використаних документів.

8. Реалізувати модуль спостережуваності для моніторингу роботи системи, збору метрик та аналізу виконання окремих етапів обробки запитів.

9. Провести оцінювання ефективності retrieval-компонента та дослідити особливості роботи системи в різних сценаріях використання.

10. Визначити основні переваги, обмеження та напрями подальшого вдосконалення запропонованої системи.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОПОНОВАНОГО РІШЕННЯ

2.1 Принципи функціонування та взаємозв'язку компонентів RAG

Розробка сучасних інтелектуальних систем пошуку інформації потребує використання архітектур, які поєднують можливості семантичного пошуку, аналізу текстових даних та генерації змістовних відповідей. Одним із найбільш ефективних підходів до побудови таких систем є використання архітектури Retrieval-Augmented Generation, що забезпечує інтеграцію механізмів пошуку інформації з генеративними можливостями великих мовних моделей.

У межах даної роботи пропонується концептуальна архітектура прогнозового модуля пошуку релевантної інформації, що реалізується на основі RAG-підходу. Основною метою такої системи є автоматизований аналіз текстових джерел, пошук найбільш релевантної інформації та формування узагальнених відповідей на запити користувача.

Запропонована архітектура передбачає використання багаторівневої структури обробки інформації, яка складається з кількох функціональних компонентів. Кожен із цих компонентів виконує окремі завдання у процесі підготовки даних, пошуку інформації та генерації відповіді.

Архітектура Retrieval-Augmented Generation базується на поєднанні двох основних підходів: інформаційного пошуку (retrieval) та генерації тексту (generation). У традиційних системах генерації тексту мовна модель формує відповіді виключно на основі знань, отриманих під час навчання. Такий підхід має певні обмеження, оскільки модель не має доступу до актуальних джерел інформації та не може використовувати нові дані, що з’явилися після завершення процесу навчання. У системах, побудованих за архітектурою Retrieval-Augmented Generation, цей процес доповнюється використанням зовнішніх інформаційних ресурсів, що дозволяє значно підвищити точність та актуальність отриманих результатів.

Загальна архітектура RAG-системи передбачає інтеграцію кількох функціональних компонентів, які спільно забезпечують повний цикл обробки запитів користувача. Одним із ключових елементів такої системи є модуль збору

та підготовки даних, який відповідає за отримання текстової інформації з різних джерел, її попередню обробку та формування структурованої бази знань. До таких джерел можуть належати текстові документи, наукові статті, веб-ресурси, внутрішні корпоративні бази даних або інші інформаційні сховища. На цьому етапі також виконується очищення тексту, нормалізація даних та розбиття документів на окремі фрагменти, що дозволяє підвищити ефективність подальшого пошуку інформації.

Після підготовки даних здійснюється формування векторних представлень тексту за допомогою спеціалізованих моделей embeddings. Ці моделі перетворюють текстові фрагменти на числові вектори у багатовимірному просторі, які відображають семантичний зміст тексту. Завдяки такому представленню стає можливим виконання семантичного пошуку, що дозволяє знаходити документи зі схожим змістом навіть у тих випадках, коли в текстах використовуються різні слова або формулювання.

Отримані векторні представлення документів зберігаються у векторній базі даних, яка забезпечує ефективне зберігання великих обсягів embeddings та швидке виконання операцій пошуку. Використання спеціалізованих алгоритмів пошуку найближчих сусідів дозволяє визначати документи, які мають найбільшу семантичну подібність до запиту користувача.

Після цього виконується етап семантичного пошуку, під час якого запит користувача також перетворюється на векторне представлення. Далі система виконує порівняння цього вектора з векторами документів, що зберігаються у базі знань, та визначає найбільш релевантні текстові фрагменти. Отримані результати формують контекст, який буде використаний під час генерації відповіді.

Наступним важливим компонентом архітектури є генеративна мовна модель, яка отримує запит користувача разом із знайденими текстовими фрагментами та формує узагальнену відповідь на основі цієї інформації. Завдяки використанню зовнішніх джерел даних мовна модель може генерувати більш точні, інформативні та контекстно обґрунтовані відповіді.

Координацію взаємодії між усіма компонентами системи забезпечує агент керування процесом обробки запиту. Такий агент виконує функції аналізу запиту користувача, запуску процедур пошуку інформації, передачі отриманих результатів мовній моделі та формування остаточної відповіді. У складніших системах агент може також приймати рішення щодо повторного пошуку інформації, використання додаткових джерел даних або зміни параметрів генерації тексту [22].

Взаємодія описаних компонентів дозволяє реалізувати повний цикл обробки запиту користувача від аналізу текстових джерел до генерації змістовної відповіді на основі знайдених документів. Така архітектура забезпечує високу гнучкість системи, можливість масштабування та ефективну роботу з великими обсягами текстових даних.

2.2 Стратегія обробки та ранжування документів

Однією з ключових характеристик сучасних систем Retrieval-Augmented Generation є використання спеціалізованого пайплайну обробки даних, який визначає послідовність перетворень інформації від моменту отримання запиту користувача до формування фінальної відповіді. Такий пайплайн забезпечує узгоджену взаємодію між компонентами системи, включаючи модулі обробки тексту, пошуку інформації та генерації відповіді. У загальному вигляді процес функціонування RAG-системи можна подати як композицію двох основних функцій - пошуку релевантних документів та генерації відповіді [1]:

$$y = G\big(q, R(q, D)\big), \tag{2.1}$$

де q - запит користувача;

D - множина документів у базі знань;

R - функція пошуку;

G - генеративна модель;

y - сформована відповідь.

Першим етапом обробки є перетворення текстових даних у векторні представлення. Кожен документ або його фрагмент відображається у вигляді вектора у багатовимірному просторі ознак.

Аналогічно формується вектор запиту користувача . Використання векторних представлень дозволяє враховувати семантичний зміст тексту, а не лише поверхневі збіги слів.

Після отримання векторного представлення запиту виконується пошук найбільш релевантних документів у базі знань. Для цього використовується міра подібності між векторами запиту та документів. Найбільш поширеною є косинусна подібність, яка визначає кут між двома векторами у просторі ознак [31]:

$$\cos(\theta) = \frac{q \cdot v_i}{||q||,||v_i||}, \tag{2.2}$$

Чим більше значення цієї метрики, тим більш релевантним вважається документ щодо запиту користувача. На основі цієї метрики виконується відбір множини найбільш релевантних документів, що формалізується як задача пошуку топ-(k) найближчих векторів.

Однак простий відбір найближчих векторів може призводити до дублювання інформації, коли відібрані документи є надто схожими між собою. Для усунення цієї проблеми використовується метод Maximal Marginal Relevance, який дозволяє збалансувати релевантність і різноманітність результатів [32]:

$$\text{MMR} = \lambda \cdot \text{sim}(q, d_i) - (1 - \lambda) \cdot \max_{d_j \in S} \text{sim}\left(d_i, d_j\right), \tag{2.3}$$

де S - множина вже вибраних документів;

λ - параметр, що визначає баланс між релевантністю та різноманітністю.

Наступним етапом є фільтрація отриманих документів за метаданими. У запропонованій системі кожен документ містить додаткові атрибути, такі як

тематика, рівень складності або вікова категорія користувача. Це дозволяє застосовувати додаткові обмеження до результатів пошуку.

Після цього виконується додаткове ранжування документів із використанням великої мовної моделі. На відміну від класичних метрик подібності, мовна модель здатна враховувати складні семантичні залежності та контекст запиту.

Завершальним етапом пайплайну є генерація відповіді. На цьому етапі мовна модель отримує запит користувача разом із відібраними документами та формує узагальнену відповідь [1]:

$$y = G(q, D_k). \tag{2.4}$$

Отриманий результат повертається користувачу, а інформація про взаємодію може бути збережена у системі для подальшого аналізу [18].

Таким чином, пайплайн RAG-системи забезпечує послідовне перетворення запиту користувача у відповідь шляхом поєднання методів семантичного пошуку, фільтрації, ранжування та генерації тексту [24].

Запропонований підхід дозволяє підвищити точність відповідей, зменшити ймовірність генерації некоректної інформації та адаптувати результати до потреб конкретного користувача.

2.3 Формування бази знань та індексація даних

Ефективність RAG-систем значною мірою залежить від якості бази знань, яка використовується для пошуку інформації. Тому важливим етапом розробки системи є формування структури зберігання документів та організація їх індексації.

На етапі підготовки даних документи розбиваються на менші текстові фрагменти.

Формально цей процес можна представити як відображення одного документа у множину фрагментів:

$$d \rightarrow c_1, c_2, \dots, c_n \ , \tag{2.5}$$

де d - вихідний документ;

c_i - окремі текстові фрагменти.

Такий підхід дозволяє підвищити точність пошуку, оскільки система працює з більш дрібними семантичними одиницями.

Після цього для кожного фрагмента створюється embedding, який відображає семантичний зміст тексту. Отримані векторні представлення зберігаються у векторній базі даних разом із метаданими. Метадані можуть містити інформацію про джерело документа, дату створення, автора або інші атрибути, що можуть бути корисними під час аналізу інформації.

Такий підхід дозволяє створити масштабовану систему зберігання знань, яка забезпечує швидкий доступ до релевантних документів.

2.4 Використання LangChain та LangGraph у реалізації системи

Для реалізації запропонованої архітектури використано сучасні програмні фреймворки, що підтримують роботу з великими мовними моделями та системами семантичного пошуку.

Одним із ключових інструментів є LangChain, який забезпечує базову інфраструктуру для побудови RAG-систем. Зокрема, він надає засоби для завантаження та обробки документів, створення embeddings, інтеграції з векторними базами даних та взаємодії з великими мовними моделями. Завдяки цьому значно спрощується реалізація основних компонентів системи.

Однак класичні можливості LangChain орієнтовані переважно на побудову лінійних пайплайнів, у яких послідовність виконання дій є заздалегідь визначеною. Такий підхід є ефективним для простих сценаріїв, проте при реалізації складніших механізмів обробки запитів виникає необхідність підтримки адаптивної логіки, умовних переходів та можливості динамічної зміни сценарію виконання.

Для вирішення цієї задачі у даній роботі використано фреймворк LangGraph, який розширює можливості LangChain та дозволяє реалізовувати більш складні процеси взаємодії між компонентами системи. Використання LangGraph дає змогу представляти процес виконання не як лінійну послідовність операцій, а як структуру взаємопов'язаних станів та переходів між ними.

Для керування процесом обробки запитів у запропонованій системі використовується агентний підхід. У межах даної архітектури RAG-агент виступає центральним компонентом системи, який координує взаємодію між усіма модулями та визначає послідовність виконання окремих етапів обробки інформації. На відміну від традиційних підходів, де сценарій виконання є жорстко визначеним, агентний підхід дозволяє адаптувати поведінку системи залежно від особливостей вхідного запиту та проміжних результатів роботи [30].

Реалізація агента базується на використанні LangGraph, який дозволяє представити процес обробки запитів у вигляді орієнтованого графа станів. У такій моделі кожен вузол відповідає окремому етапу роботи системи, наприклад аналізу запиту, пошуку інформації, оцінюванню релевантності отриманих результатів або генерації відповіді. Переходи між вузлами визначаються логікою виконання та можуть залежати від проміжних результатів роботи системи.

Даний підхід забезпечує підтримку умовних переходів, циклів, повторного виконання окремих операцій та обробки виняткових ситуацій. Крім того, графова структура дозволяє реалізувати більш гнучке керування процесом взаємодії між компонентами системи та забезпечує можливість масштабування архітектури шляхом додавання нових вузлів без необхідності суттєвої зміни існуючої логіки виконання.

Саме на основі цього підходу було побудовано граф виконання RAG-агента, представлений на рисунку 2.1, який відображає основні етапи обробки запиту користувача та можливі варіанти переходів між ними.

У початковому стані агент отримує запит користувача та додає його до контексту взаємодії. Далі виконується завантаження профілю користувача, який містить інформацію про попередні запити, вподобання та інші характеристики. Це дозволяє персоналізувати подальшу обробку запиту.

Після цього виконується етап маршрутизації (routing), на якому система визначає, який сценарій обробки буде використано. Залежно від складності запиту можливі різні варіанти: використання повного RAG-пайплайну або формування простої відповіді без звернення до бази знань.

У випадку використання повного пайплайну виконується переформулювання запиту (rewrite), що дозволяє уточнити його зміст та покращити якість пошуку. Далі відбувається аналіз контексту та перевірка достатності інформації для формування відповіді.

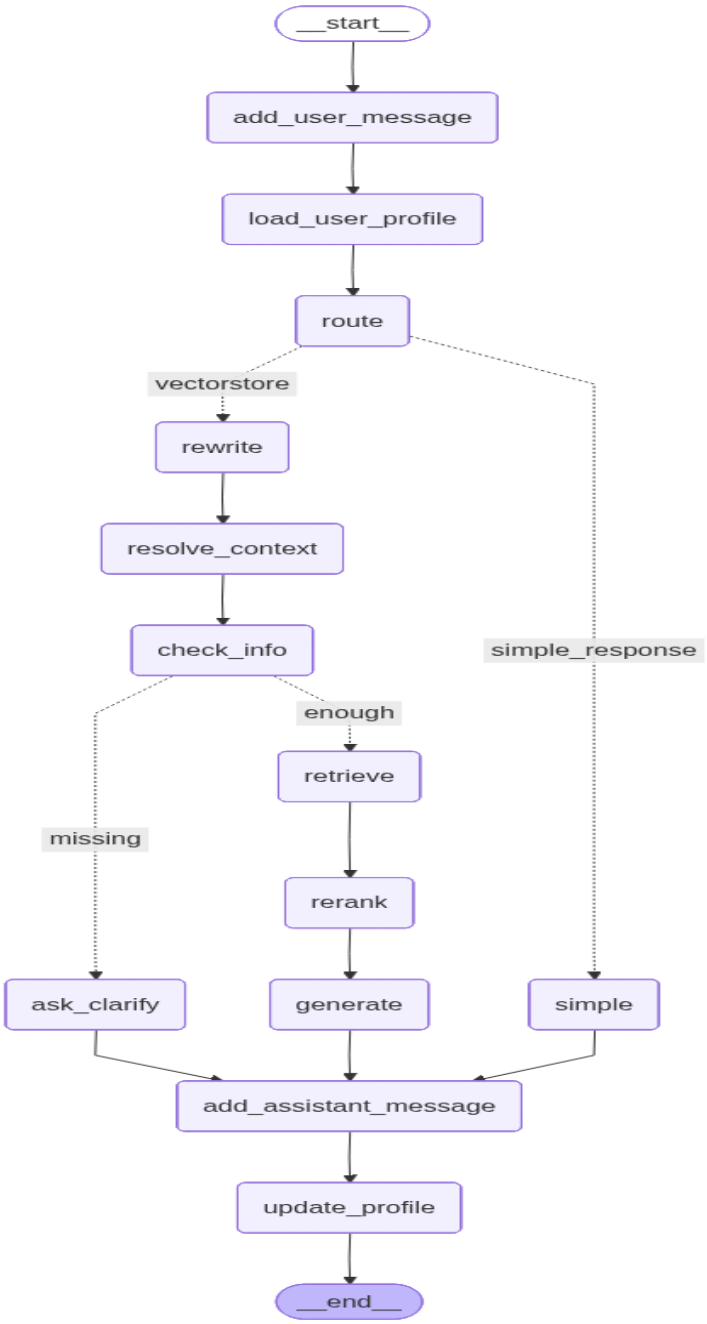


Рисунок 2.1 - Граф виконання RAG-агента, реалізований за допомогою LangGraph

Якщо інформації недостатньо, агент переходить до вузла уточнення запиту (ask_clarify), що дозволяє отримати додаткові дані від користувача. У протилежному випадку запускається процес пошуку релевантних документів у векторній базі даних (retrieve).

Отримані документи проходять етап додаткового ранжування (rerank), після чого передаються до мовної моделі для генерації відповіді (generate).

У випадку простих запитів система може обійти ці етапи та одразу сформувати відповідь (simple), що дозволяє зменшити затримки та обчислювальні витрати.

Після формування відповіді вона додається до історії взаємодії, а також виконується оновлення профілю користувача. Це дозволяє накопичувати інформацію про поведінку користувача та використовувати її для подальшої персоналізації системи.

З формальної точки зору поведінку агента можна представити як орієнтований граф [34]:

$$G = (V, E), \tag{2.6}$$

де V - множина вузлів;

E - множина переходів між ними.

Використання графової моделі дозволяє реалізувати асинхронне виконання, повторні виклики окремих вузлів та гнучке керування логікою обробки запитів. Це є ключовою перевагою порівняно з традиційними лінійними пайплайнами.

2.5 Модуль збереження даних користувача та історії взаємодії

Важливою складовою запропонованої системи є забезпечення збереження даних користувача та історії взаємодії з системою. Це дозволяє реалізувати персоналізацію, аналіз поведінки користувача та підвищення якості відповідей у

майбутніх сесіях. Для реалізації даного функціоналу використовується рівень персистентності, побудований із застосуванням підходу ORM (Object-Relational Mapping).

ORM дозволяє представити структуру бази даних у вигляді об’єктної моделі та спростити взаємодію з нею на рівні програмного коду. У межах системи реалізовано дві основні сутності: профіль користувача та історія використання документів.

Профіль користувача містить узагальнену інформацію про взаємодію із системою, зокрема вподобання, рівень підготовки та інші характеристики, що використовуються для персоналізації відповідей.

З метою забезпечення конфіденційності ці дані зберігаються у зашифрованому вигляді. Перед записом у базу даних профіль серіалізується та шифрується із використанням симетричного криптографічного алгоритму, після чого зберігається у вигляді бінарного об’єкта.

Історія взаємодії представлена у вигляді окремої сутності, яка містить інформацію про документи, що були використані під час формування відповідей. Для кожного користувача зберігається ідентифікатор документа, його назва, час використання та кількість звернень до нього.

У випадку повторного використання документа значення лічильника оновлюється відповідно до залежності:

$$count_{new} = count_{old} + 1. \quad (2.7)$$

Це забезпечує коректний облік частоти використання документів у системі.

Структура бази даних, що використовується для збереження інформації, представлена у вигляді двох взаємопов’язаних таблиць (рисунок 2.2).

Перша таблиця відповідає за збереження профілю користувача, а друга - за облік використаних документів.

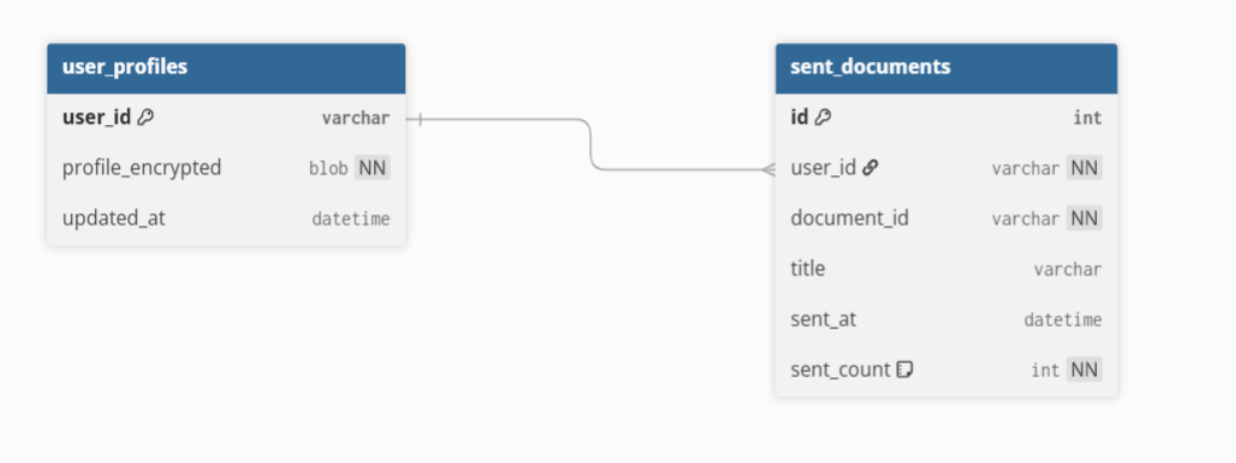


Рисунок 2.2 - Схема бази даних модуля збереження профілю користувача та історії документів

Використання ORM у поєднанні з механізмами шифрування забезпечує надійне, масштабоване та безпечне збереження даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Загальна архітектура програмного модуля

Програмна реалізація створена із використанням модульного підходу, що дозволяє розділити функціональність на окремі незалежні компоненти та спростити їх подальший розвиток. Такий підхід забезпечує можливість зміни або розширення окремих елементів без необхідності внесення змін до загальної логіки роботи. Архітектура організована таким чином, щоб компоненти могли створюватися динамічно залежно від заданих параметрів конфігурації.

Загальна схема взаємодії програмних компонентів є досить об’ємною, тому для покращення читабельності її повне представлення подано у додатку А. У даному підрозділі наведено окремі фрагменти архітектури, що демонструють структуру та взаємозв’язок основних компонентів програмної реалізації.

Процес запуску починається із завантаження конфігураційних параметрів та виконання початкової ініціалізації службових механізмів. На цьому етапі здійснюється налаштування параметрів середовища, а також підготовка компонентів моніторингу та трасування. Загальну схему початкової ініціалізації представлено на рисунку 3.1.

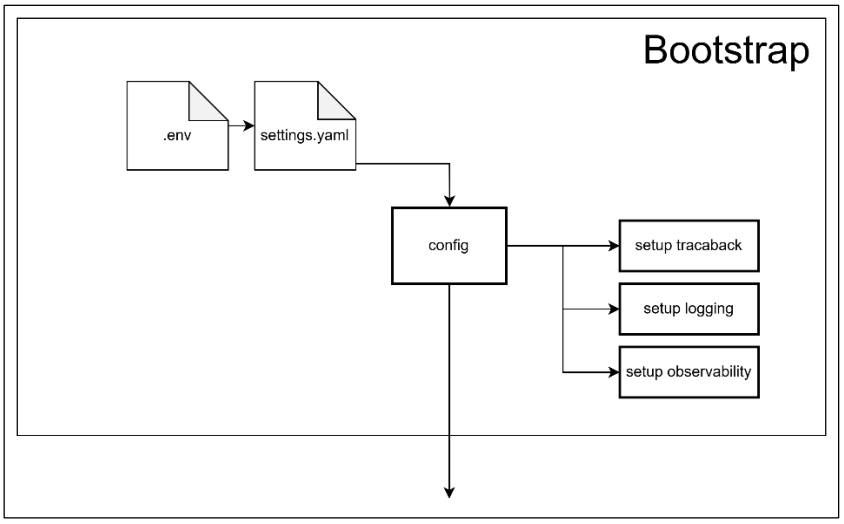


Рисунок 3.1 - Процес початкової ініціалізації компонентів системи

Після завершення початкового налаштування виконується створення основних компонентів, необхідних для роботи RAG-агента. Для цього використовується підхід, за якого кожен модуль отримує набір параметрів та створює відповідний екземпляр на основі переданих налаштувань. Це дозволяє використовувати різні реалізації моделей, сховищ даних або допоміжних сервісів без зміни загальної структури програмного забезпечення.

Основні програмні компоненти, що використовуються під час функціонування RAG-агента, наведено на рисунку 3.2.

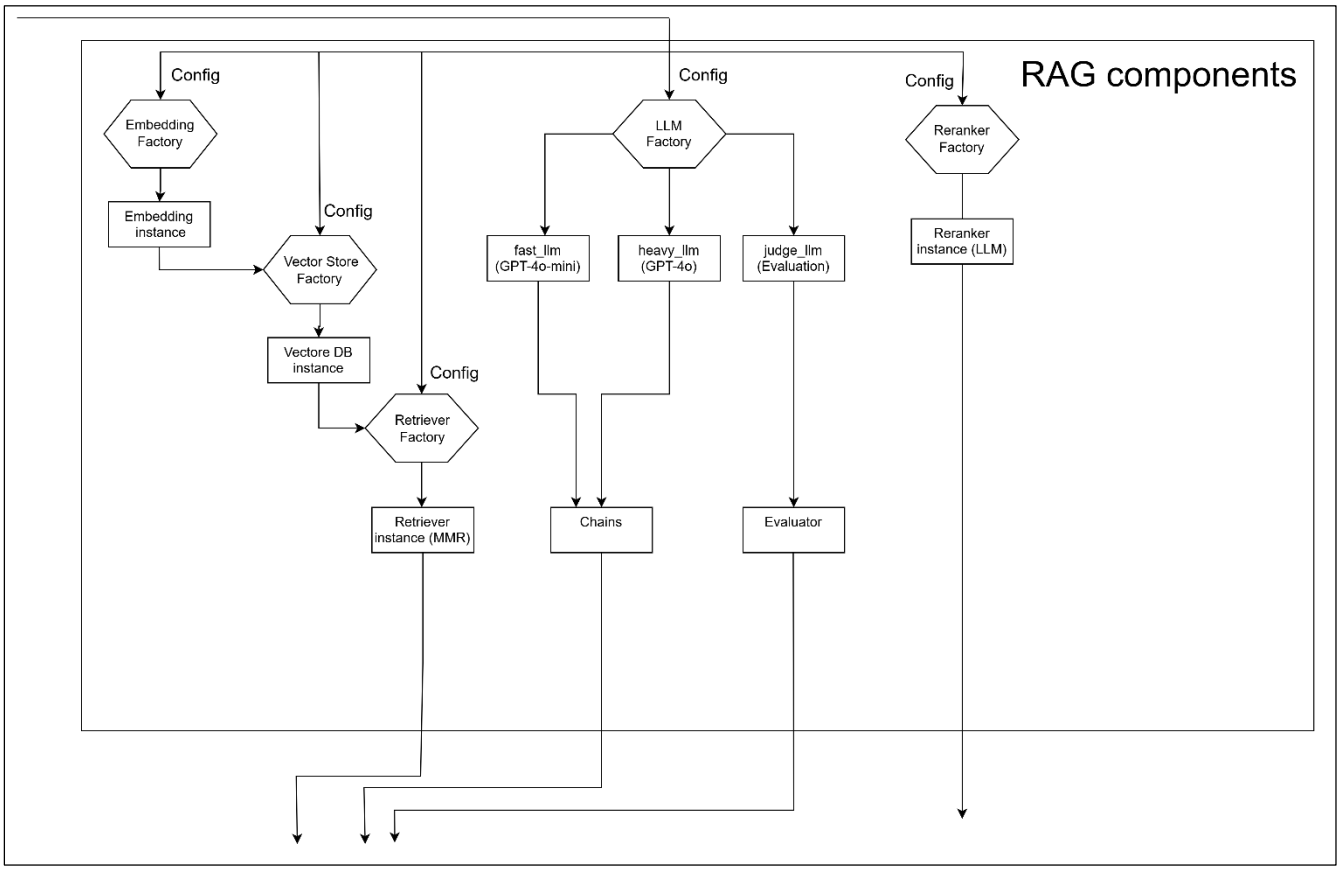


Рисунок 3.2 - Основні програмні компоненти RAG-агента

У межах реалізації створено окремі модулі для роботи з мовними моделями, генерацією векторних представлень, механізмами пошуку та ранжування результатів, а також засобами оркестрації виконання.

Одним із важливих елементів архітектури є модуль персистентності, який забезпечує збереження профілю користувача та історії взаємодії із системою. Даний компонент виступає проміжним рівнем між логікою застосунку та базою

даних і забезпечує механізми отримання, оновлення та збереження інформації. Архітектура модуля містить ORM-компоненти, репозиторії доступу до даних та механізми шифрування інформації користувача. Реалізацію даного компонента наведено на рисунку 3.3.

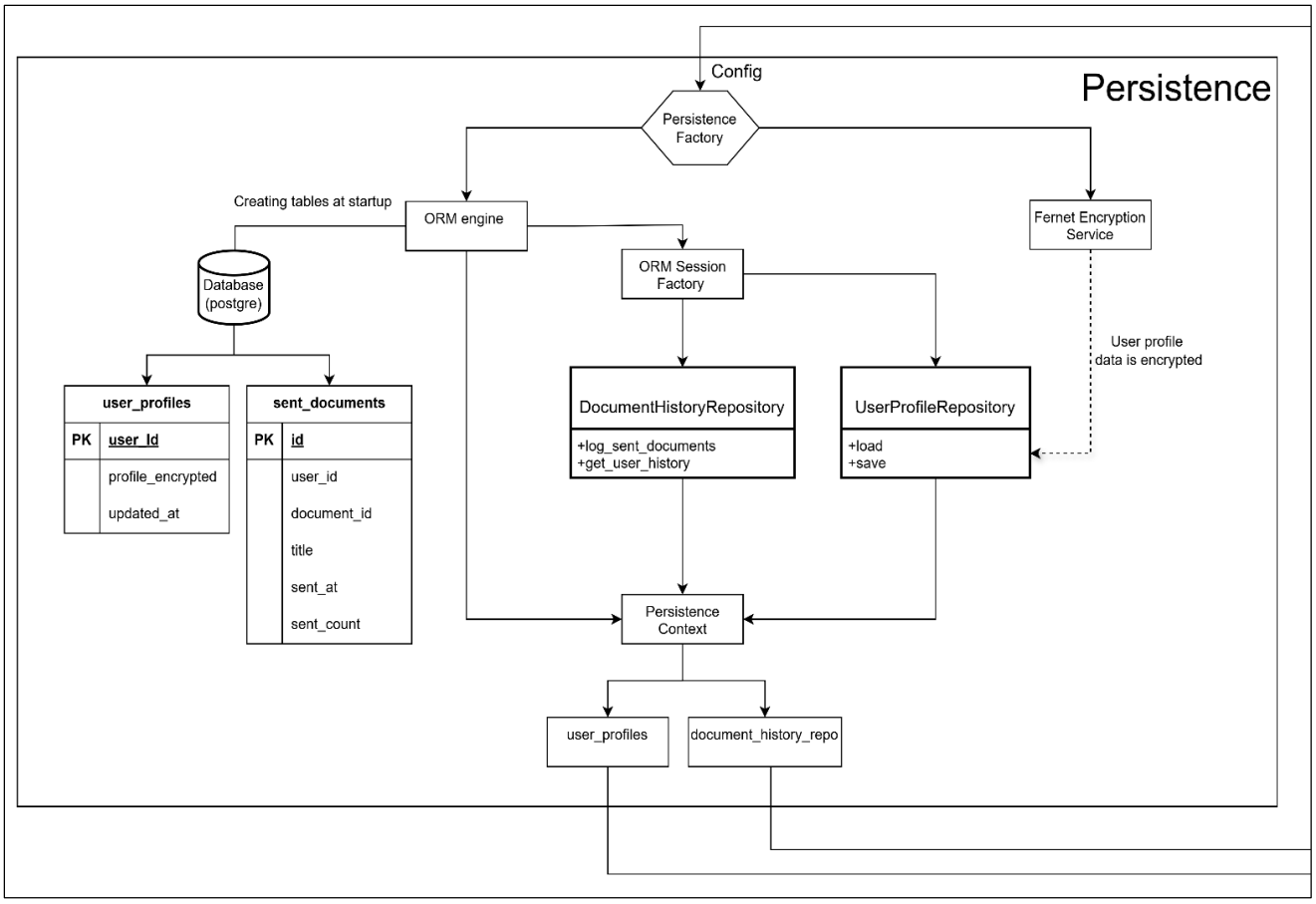


Рисунок 3.3 - Архітектура модуля збереження даних користувача та історії взаємодії

Детальний опис структури бази даних та організацію збереження інформації наведено у підрозділі 2.4.

Для аналізу продуктивності та контролю роботи окремих частин програмної реалізації використовується підсистема спостережуваності (додаток Б). Вона забезпечує збір інформації про виконання запитів, характеристики роботи окремих компонентів та внутрішні параметри функціонування RAG-агента. Архітектуру модуля спостережуваності показано на рисунку 3.4.

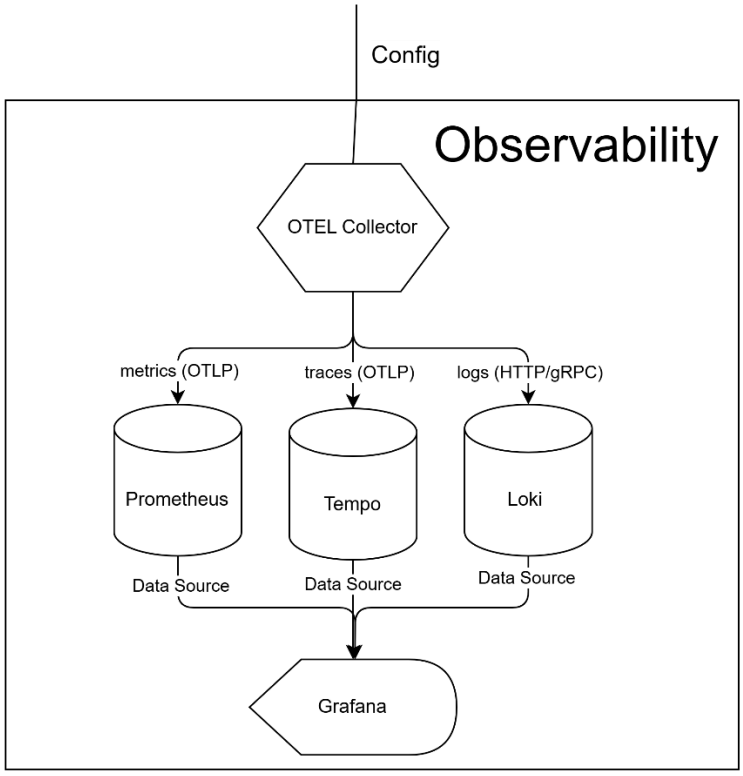


Рисунок 3.4 - Архітектура модуля спостережуваності

Після завершення процесу створення та налаштування всі ініціалізовані компоненти об'єднуються у централізований контекст застосунку, який використовується як єдина точка доступу до ресурсів програмної реалізації. Такий підхід дозволяє уникнути повторного створення об'єктів, спрощує керування залежностями між компонентами та забезпечує їх повторне використання в різних частинах програмного забезпечення. Організацію контексту застосунку наведено на рисунку 3.5.

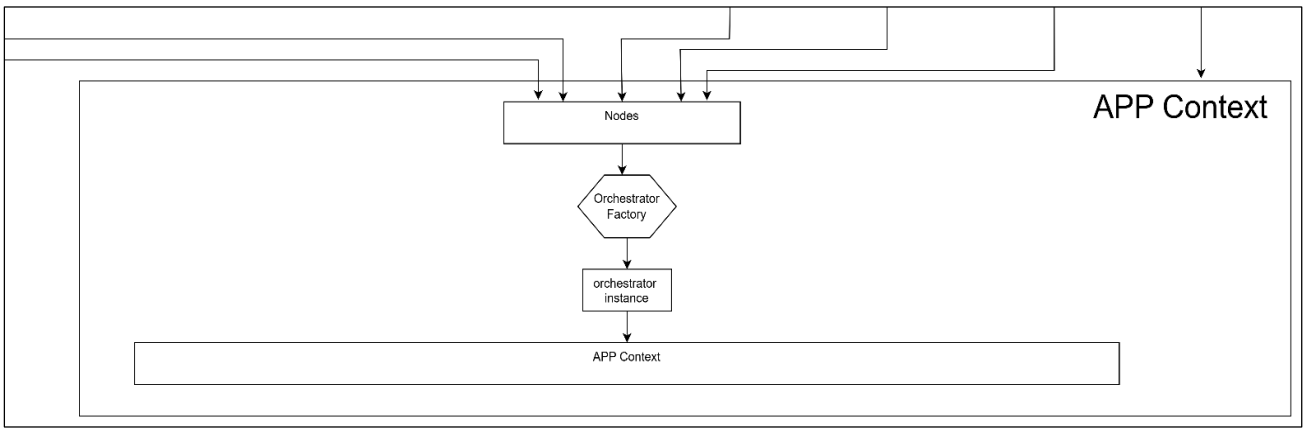


Рисунок 3.5 - Організація контексту застосунку та збереження компонентів системи

Таким чином, програмна реалізація побудована у вигляді сукупності взаємопов’язаних модулів, що забезпечують виконання окремих функціональних задач та спільно формують повний процес роботи RAG-агента

3.2 Реалізація інструментів логування та моніторингу

Для забезпечення контролю, аналізу та оптимізації роботи системи у межах даної роботи реалізовано підсистему спостережуваності (observability). Вона дозволяє відстежувати виконання запитів, аналізувати продуктивність окремих компонентів та виявляти вузькі місця у роботі RAG-агента.

Загальну архітектуру модуля спостережуваності було представлено на рисунку 3.4.

В основі модуля використано підхід, що базується на OpenTelemetry. Даний стандарт є основою для більшості сучасних систем трасування та моніторингу, що використовуються у промислових рішеннях.

Його використання дозволяє гнучко налаштовувати збір метрик, логів та трас виконання, а також інтегрувати систему з різними інструментами візуалізації.

У рамках реалізації зібрані дані передаються до системи моніторингу та відображаються за допомогою Grafana. Це дозволяє здійснювати аналіз роботи системи у реальному часі та досліджувати поведінку окремих компонентів.

На рисунку 3.6 представлено приклад дашборду, який відображає інформацію про виконання запитів у системі.

Зокрема, показано список виконаних запитів (trace), їх тривалість та часові характеристики. Також наведено графік, що дозволяє оцінити розподіл часу виконання запитів.

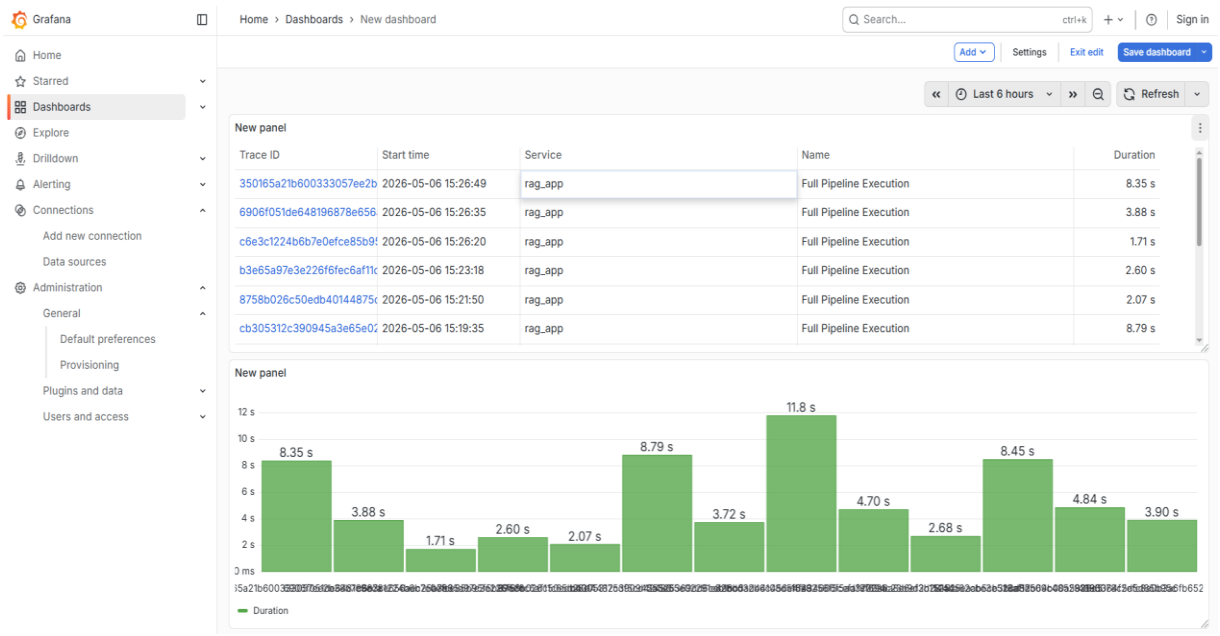


Рисунок 3.6 - Дашборд моніторингу виконання запитів у системі (Grafana)

Аналіз таких даних дозволяє оцінити навантаження на систему, визначити середній час обробки запиту та виявити аномалії у продуктивності.

На рисунку 3.7 представлено детальний трас (trace) виконання одного запиту. У трасі відображено всі етапи обробки, включаючи виклики окремих вузлів графа (наприклад, retrieve, rerank, generate), а також їх тривалість.

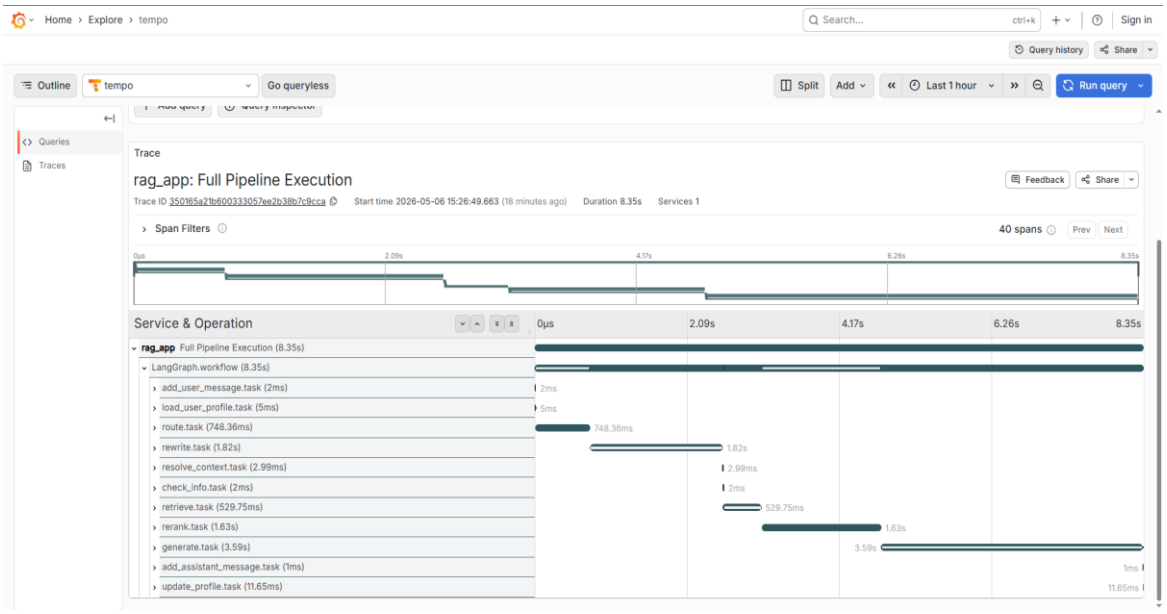


Рисунок 3.7 - Трас виконання запиту з деталізацією етапів RAG-пайплайну

Такий рівень деталізації дозволяє точно визначити, які саме компоненти системи займають найбільше часу. Наприклад, можна проаналізувати затримки, пов’язані з викликами мовної моделі або пошуком у векторній базі даних, що відкриває можливості для оптимізації системи шляхом зміни конфігурації її компонентів або алгоритмів їх роботи. Окрім аналізу часу виконання, підсистема спостережуваності забезпечує збір додаткових метрик, зокрема відстеження кількості токенів, використаних під час запитів до мовної моделі як для вхідних, так і для вихідних даних, врахування параметрів генерації, таких як температура, що впливає на варіативність відповіді, а також фіксацію кількості викликів окремих компонентів системи та результатів виконання тестових сценаріїв.

Збір таких даних дозволяє проводити benchmark-тестування системи та оцінювати її ефективність у різних умовах. Наприклад, можна дослідити вплив зміни параметрів генерації на якість та швидкість відповіді або порівняти різні конфігурації retrieval-компонента.

Окремо слід відзначити можливість контролю коректності відповідей мовної моделі. У системі реалізовано механізм перевірки структурованих відповідей, зокрема у форматі JSON. Це дозволяє забезпечити відповідність результатів заданому формату та зменшити кількість помилок при подальшій обробці даних.

Таким чином, використання підсистеми спостережуваності на основі OpenTelemetry забезпечує повний контроль над роботою системи, дозволяє аналізувати продуктивність, проводити тестування та підвищувати якість роботи RAG-агента.

3.3 Оцінка ефективності retrieval-компонента

Для забезпечення взаємодії користувача із системою було реалізовано веб-інтерфейс із використанням Streamlit. Обраний підхід дозволяє швидко створити інтерактивний прототип, який підтримує діалогову форму взаємодії та забезпечує наочне відображення результатів роботи RAG-агента.

Інтерфейс побудований у вигляді чат-системи, де користувач може вводити запити у довільній формі, а система формує відповіді з урахуванням контексту, профілю користувача та результатів пошуку у базі знань.

На рисунку 3.8 представлено приклад початкової взаємодії користувача із системою. У цьому випадку агент обробляє простий запит привітання та формує відповідь без використання повного RAG-пайплайну.

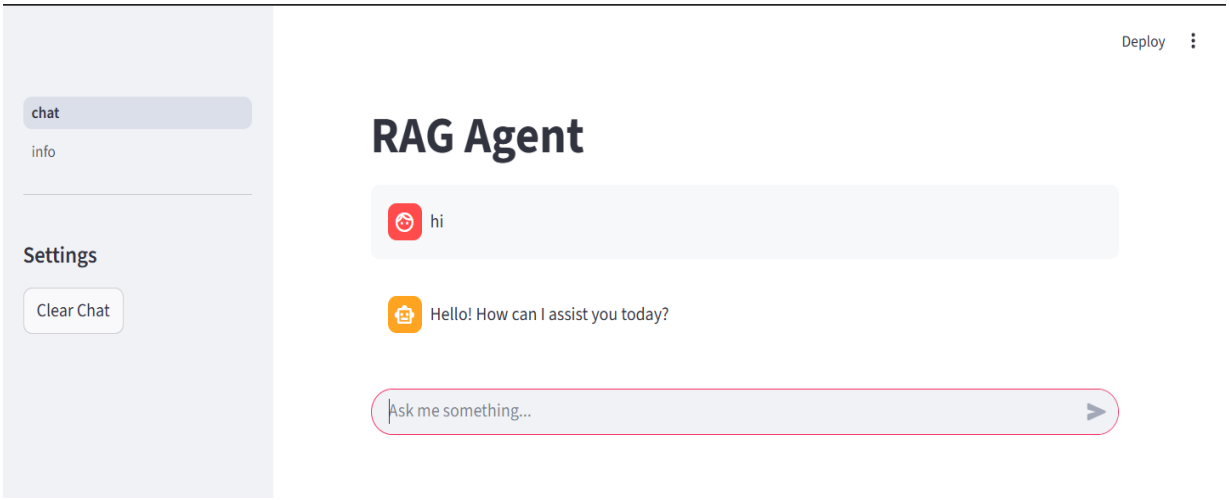


Рисунок 3.8 - Приклад початкової взаємодії користувача із системою

На рисунку 3.9 показано приклад обробки більш складного запиту, у якому система не має достатньо інформації для формування відповіді. У такому випадку агент переходить до етапу уточнення та генерує додаткове запитання до користувача.

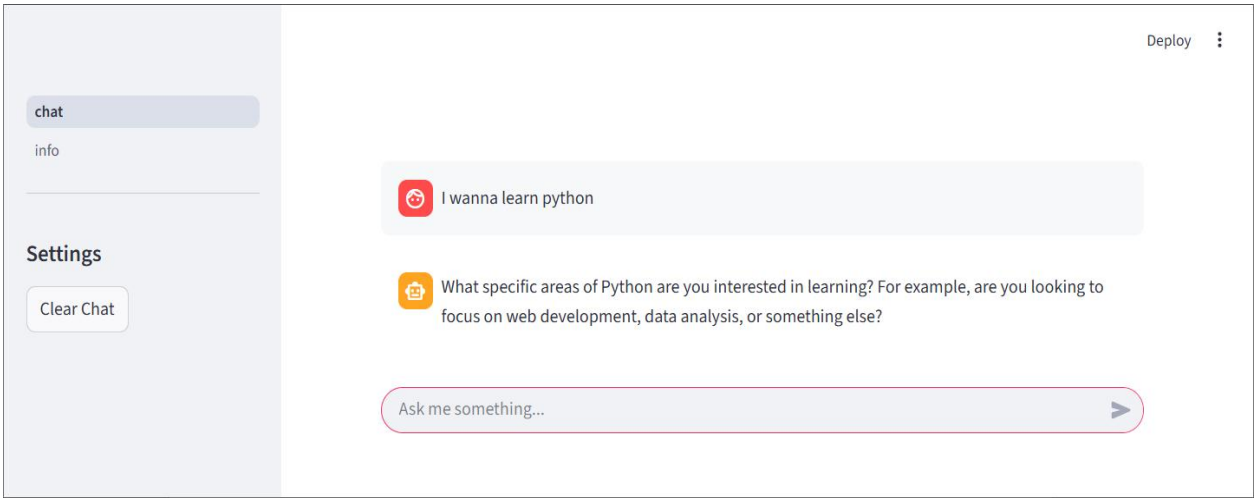


Рисунок 3.9 - Приклад уточнення запиту користувача

На рисунку 3.10 представлено приклад роботи повного RAG-пайплайну. У цьому випадку після отримання запиту система виконує пошук релевантних документів у векторній базі даних, застосовує фільтрацію та ранжування результатів, після чого формує відповідь із використанням знайденого контексту.

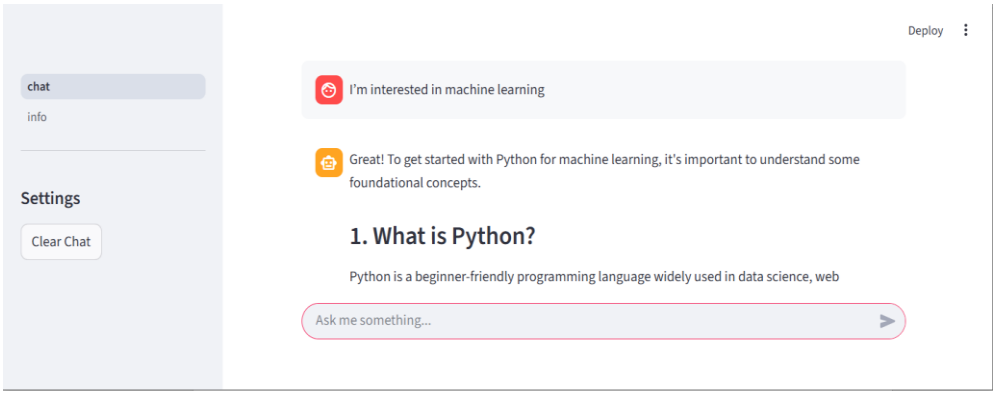


Рисунок 3.10 - Приклад відпороботи RAG-пайплайну з використанням retrieval

На рисунку 3.11 показано відображення інформації про документи, які були використані під час формування відповіді. Інтерфейс системи надає можливість переглянути список використаних джерел, їх метадані (зокрема тематику та рівень складності).

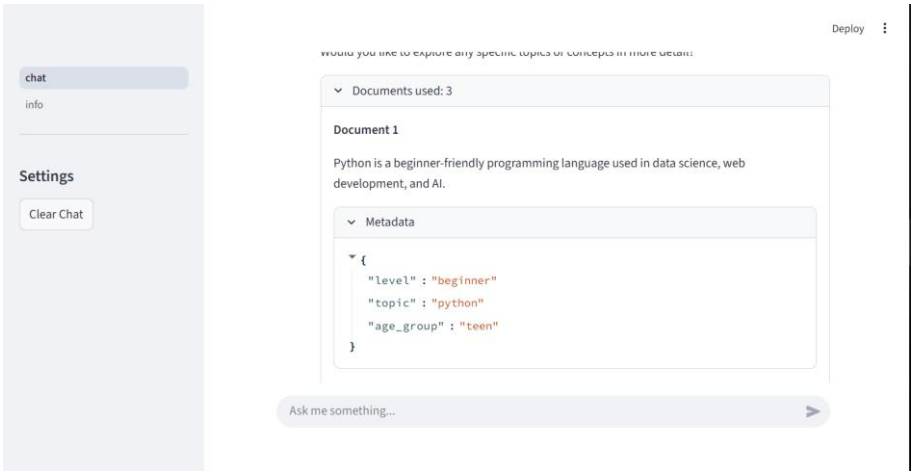


Рисунок 3.11 - Відображення використаних документів та їх метаданих у системі

Відповідний підхід підвищує прозорість роботи системи та дозволяє користувачу оцінити, на основі яких саме даних була сформована відповідь. Крім того, відображення статистики використання документів забезпечує додаткові можливості для аналізу ефективності retrieval-компонента.

Таким чином, реалізований користувацький інтерфейс не лише забезпечує зручну взаємодію із системою, але й дозволяє наочно продемонструвати роботу основних компонентів RAG-агента, включаючи адаптивну обробку запитів, уточнення контексту, використання механізмів семантичного пошуку та відображення джерел інформації.

Для оцінки ефективності роботи retrieval-компонента було реалізовано спеціалізований модуль тестування, який дозволяє кількісно оцінити якість пошуку релевантних документів у базі знань. Оцінювання зосереджене саме на retrieval-частині системи, оскільки вона є ключовим елементом RAG-архітектури та безпосередньо впливає на якість фінальної відповіді.

Реалізований evaluator працює на основі набору тестових запитань, кожне з яких описує типовий сценарій взаємодії із системою. Для кожного запиту задається його текст, очікувані ключові елементи змісту, які повинні бути присутні у релевантних документах, а також категорія, що визначає тип запиту. Додатково використовується еталонна відповідь, яка може бути задіяна для розширення оцінювання у майбутньому, зокрема для аналізу якості генерації [21].

Такий підхід дозволяє формалізувати процес тестування та забезпечити його відтворюваність. Використання ключових слів як критерію релевантності дає можливість перевірити, чи дійсно retriever знаходить документи, що містять необхідну інформацію, а не лише семантично подібний, але неповний контекст.

У процесі тестування evaluator виконує серію запитів до retriever-компонента, отримує список документів та обчислює метрики якості пошуку.

Для оцінки використовуються наступні метрики:

– Mean Reciprocal Rank (MRR) - характеризує позицію першого релевантного документа у списку результатів. Чим вище значення метрики, тим

раніше у видачі з’являється релевантний документ. Значення обчислюється як обернений ранг першого релевантного результату.

– Normalized Discounted Cumulative Gain (nDCG) - враховує не лише факт наявності релевантних документів, але й їх розташування у списку. Метрика надає більшу вагу документам, що знаходяться на верхніх позиціях, і нормалізується відносно ідеального ранжування.

– Keyword Coverage - показує відсоток ключових слів, які були знайдені у retrieved-документах. Ця метрика дозволяє оцінити повноту покриття інформації.

Для більш детального аналізу тестові запити було згруповано за типами відповідно до характеру інформаційної потреби. Зокрема, виокремлено запити у розмовній формі, наближені до природного спілкування користувача, запити, орієнтовані на отримання чіткої фактичної інформації, а також запити, що потребують інтеграції відомостей із кількох джерел.

На рисунку 3.12 представлено результати оцінювання retrieval-компонента за різними категоріями запитів.

Отримані результати свідчать, що найвищі показники досягаються для категорії direct_fact, де значення MRR та nDCG наближаються до 1.0, що вказує на ефективне ранжування релевантних документів. Для категорії spanning також спостерігається високий рівень якості, що підтверджує здатність системи працювати із запитами, які потребують комбінування інформації з кількох джерел.

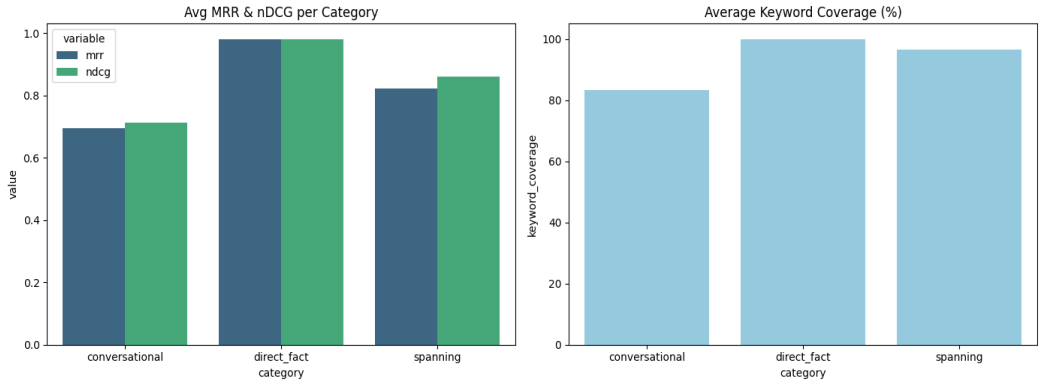


Рисунок 3.12 - Результати оцінювання retrieval-компонента за різними категоріями запитів

Водночас для conversational-запитів значення метрик знаходяться на рівні близько 0.7, що свідчить про складність обробки більш вільних формулювань і наявність потенціалу для покращення.

Покращення може бути досягнуто, зокрема, за рахунок використання семантичного чанкування документів, що дозволить формувати більш змістовні фрагменти для пошуку, а також шляхом вдосконалення механізмів reranking або застосування більш потужних моделей для створення embedding-представлень.

Таким чином, проведене оцінювання підтверджує ефективність retrieval-компонента, а також дозволяє визначити напрями для подальшого покращення системи.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи:

1. Проведено дослідження сучасних підходів до створення інтелектуальних систем пошуку інформації на основі методів обробки природної мови та технологій штучного інтелекту. Особливу увагу приділено використанню великих мовних моделей та архітектури Retrieval-Augmented Generation, яка поєднує семантичний пошук із генерацією тексту.

2. Проаналізовано принципи функціонування інтелектуальних інформаційних систем, що працюють із текстовими даними, а також досліджено підходи до представлення тексту у вигляді векторних embeddings. Встановлено, що використання семантичних представлень дозволяє значно підвищити якість пошуку інформації порівняно з класичними методами, заснованими на збігу ключових слів.

3. Досліджено концепцію Retrieval-Augmented Generation та визначено основні принципи побудови RAG-систем. Показано, що інтеграція retrieval-компонента з мовною моделлю дозволяє зменшити кількість некоректних відповідей та забезпечити формування результатів на основі зовнішніх джерел знань.

4. Виконано аналіз сучасних програмних інструментів для розробки систем на основі великих мовних моделей. Обґрунтовано вибір фреймворків LangChain та LangGraph для реалізації системи, що дозволяють будувати гнучкі RAG-пайплайни та реалізовувати агентну логіку обробки запитів.

5. Реалізовано програмну систему у вигляді RAG-агента, який виконує роль інтелектуального асистента. Система включає модулі семантичного пошуку, фільтрації, reranking та генерації відповіді, а також механізми персоналізації на основі профілю користувача. Розроблено графову оркестрацію процесу обробки запитів та збереження даних користувача за допомогою ORM.

6. Реалізовано користувацький інтерфейс, який забезпечує взаємодію із системою у форматі діалогу, а також надає можливість перегляду використаних документів і їх метаданих, що підвищує прозорість роботи системи.

7. Вдосконалено систему rag за допомогою модуля спостережуваності на основі OpenTelemetry, що дозволяє здійснювати моніторинг виконання запитів, аналіз затримок, використання ресурсів та поведінки окремих компонентів системи. Використання Grafana забезпечує наочну візуалізацію метрик і трас виконання.

8. Проведено експериментальне дослідження retrieval-компонента з використанням метрик MRR, nDCG та покриття ключових слів. Отримані результати підтвердили високу ефективність системи для запитів, орієнтованих на фактичну інформацію, та задовільні результати для розмовних сценаріїв, що вказує на можливості подальшого вдосконалення.

9. Аналіз результатів роботи системи показав, що найбільший вплив на час обробки запитів мають операції пошуку у векторній базі даних та виклики великої мовної моделі. Використання підсистеми спостережуваності дало можливість детально дослідити виконання окремих етапів RAG-пайплайну, визначити потенційні вузькі місця та сформувати напрями подальшої оптимізації системи.

10. Запропонована система демонструє значно кращу здатність знаходити семантично релевантні документи, порівняно з класичними підходами до пошуку інформації, такими як keyword-based пошук або TF-IDF. Використання embedding-представлень дозволяє враховувати зміст запиту, а не лише збіг ключових слів.

11. Визначено обмеження запропонованого підходу, пов’язані із залежністю якості retrieval-компонента від способу сегментації документів, якості embedding-представлень та структури бази знань. Встановлено, що подальше підвищення ефективності системи може бути досягнуте за рахунок використання семантичного чанкування, удосконалення методів ранжування та покращення механізмів оцінювання якості генерації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 2020, 33. http://dx.doi.org/10.48550/arXiv.2005.11401

2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017, 30. http://dx.doi.org/10.48550/arXiv.1706.03762

3. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 2020, 33. http://dx.doi.org/10.48550/arXiv.2005.14165

4. OpenAI. GPT-4 Technical Report. *arXiv preprint*, 2023. http://dx.doi.org/10.48550/arXiv.2303.08774

5. Devlin, J., Chang, M., Lee, K., & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT*, 2019. http://dx.doi.org/10.48550/arXiv.1810.04805

6. Reimers, N., & Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019. http://dx.doi.org/10.18653/v1/d19-1410

7. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020. http://dx.doi.org/10.48550/arXiv.2004.04906

8. Bommasani, R., Hudson, D., Adeli, E., Altman, R., Arora, S., et al. On the Opportunities and Risks of Foundation Models. *arXiv preprint*, 2021. http://dx.doi.org/10.48550/arXiv.2108.07258

9. LangChain. LangChain Documentation: Building Applications with LLMs. LangChain, 2024. https://docs.langchain.com

10. LangChain. LangGraph Overview. LangChain Documentation, 2024. https://docs.langchain.com/oss/python/langgraph/overview

11. Huang, Y., Huang, J. A Survey on Retrieval-Augmented Text Generation for Large Language Models. *arXiv preprint*, 2024. http://dx.doi.org/10.48550/arXiv.2404.10981

12. Ye, F., Li, S., Zhang, Y., & Chen, L. RAG: Incorporating Retrieval Information into Retrieval Augmented Generation. *arXiv preprint*, 2024. http://dx.doi.org/10.48550/arXiv.2406.13249

13. Wang, L., Chen, H., Yang, N., Huang, X., Dou, Z., & Wei, F. Chain-of-Retrieval Augmented Generation. *arXiv preprint*, 2025. http://dx.doi.org/10.48550/arXiv.2501.14342

14. Zhu, C. The Development Status of Retrieval-Augmented Generation Technology. *Journal of Computer Science Research*, 2025. http://dx.doi.org/10.54254/2755-2721/2026.TJ30058

15. Zhang, Y. A Retrieval-Augmented Generation Framework with Retriever and Generator Modules for Enhancing Factual Consistency. *Journal of Electrical and Electronic Engineering Research*, 2025. http://dx.doi.org/10.54254/2755-2721/2025.TJ24496

16. Liu, J. LlamaIndex: Data Framework for Large Language Model Applications. LlamaIndex Documentation, 2024. https://docs.llamaindex.ai

17. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint*, 2023. http://dx.doi.org/10.48550/arXiv.2312.10997

18. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys, 2023. http://dx.doi.org/10.48550/arXiv.2107.13586

19. Bubeck, S., Chandrasekaran, V., Eldan, R., et al. Sparks of Artificial General Intelligence: Early Experiments with GPT-4. *arXiv preprint*, 2023. http://dx.doi.org/10.48550/arXiv.2303.12712

20. Chase, H. LangChain Expression Language (LCEL). LangChain Documentation, 2024. https://python.langchain.com/docs/expression_language

21. Jin, J., Zhu, Y., Yang, X., Zhang, C., & Dou, Z. FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research. *arXiv preprint*, 2024. http://dx.doi.org/10.48550/arXiv.2405.13576

22. Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv preprint*, 2023. http://dx.doi.org/10.48550/arXiv.2310.11511

23. Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. REALM: Retrieval-Augmented Language Model Pre-Training. *Proceedings of ICML*, 2020. http://dx.doi.org/10.48550/arXiv.2002.08909

24. Lewis, M., Liu, Y., Goyal, N., et al. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation. *Proceedings of ACL*, 2020. http://dx.doi.org/10.48550/arXiv.1910.13461

25. Ouyang, L., Wu, J., Jiang, X., et al. Training Language Models to Follow Instructions with Human Feedback. Advances in Neural Information Processing Systems, 2022. http://dx.doi.org/10.48550/arXiv.2203.02155

26. Touvron, H., Lavril, T., Izacard, G., et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint*, 2023. http://dx.doi.org/10.48550/arXiv.2302.13971

27. Peng, B., Galley, M., He, P., et al. Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback. *arXiv preprint*, 2023. http://dx.doi.org/10.48550/arXiv.2302.12813

28. Weaviate. Weaviate Vector Database Documentation. Weaviate, 2024. https://weaviate.io/developers/weaviate

29. Pinecone. Pinecone Vector Database Documentation. Pinecone, 2024. https://docs.pinecone.io

30. Microsoft. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. Microsoft Research, 2024. https://microsoft.github.io/autogen/

31. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge : Cambridge University Press, 2008. 569 p.

32. Carbonell J., Goldstein J. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries // Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York : ACM Press, 1998. P. 335-336.

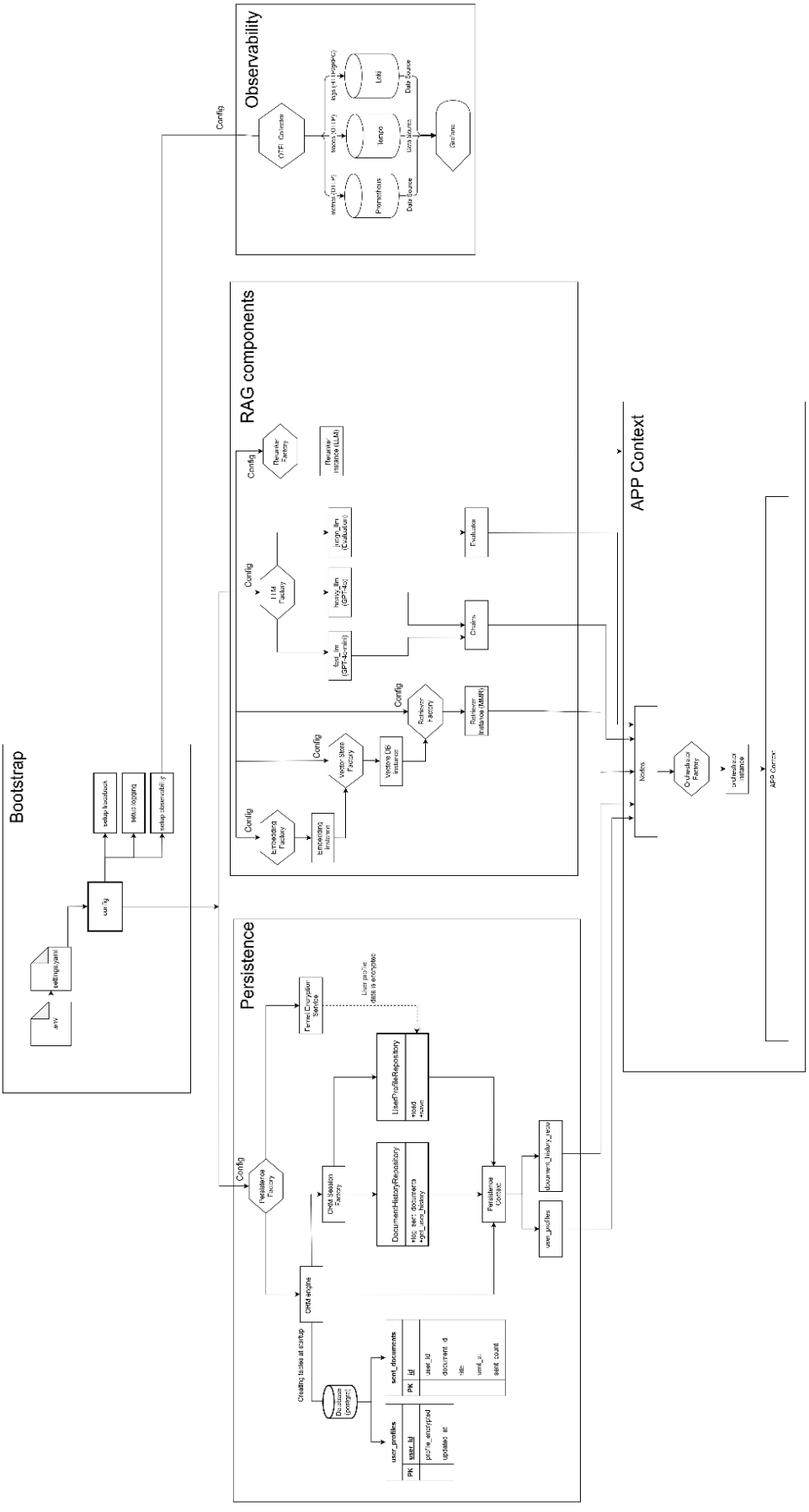
33. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. 3rd ed. Cambridge : MIT Press, 2009. 1312 p.

34. Созанський А.Б., Васильків Н.М. RAG-агент як основа програмних модулів пошуку релевантної інформації. *ICSPM 2026: Інтелектуальні обчислювальні системи та управління проєктами:* зб. тез доповідей І міжнародної науково-практичної конференції студентів і молодих вчених (21-22 травня 2026р. Тернопіль). 2026. С. 106-108.

35. Методичні рекомендації до виконання каліфікаційної роботи з освітньо-професійної програми “Комп’ютерні науки” спеціальності 122 “Комп’ютерні науки” за першим (бакалаврським) рівнем вищої освіти / Укл. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С, Ліп’яніна-Гончаренко Х.В. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Схема взаємозв’язку програмних компонентів



Додаток Б

Код модуля спостережуваності

Додаток В

Копія публікації результатів дослідження