Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА TELEGRAM-БОТА ДЛЯ КОНСУЛЬТУВАННЯ АБІТУРІЄНТІВ УНІВЕРСИТЕТУ З ВИКОРИСТАННЯМ LLM ВІД OPENAI ДЛЯ ВІДПОВІДЕЙ НА ОСНОВІ ВЛАСНОГО КОНТЕНТУ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-2
Широкорад К.А.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Яковлева О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Широкорад Ксенії Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка Telegram-бота для консультування абітурієнтів університету з використанням LLM від OpenAI для відповідей на основі власного контенту

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 24 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека GPT моделей з відкритим кодом OpenAPI, бібліогтека Telegram API, мова програування Java, фреймворк Spring Boot, редактор коду IntelliJ IDEA.

4. Перелік питань, що потрібно опрацювати в роботі
1. Донавчання моделей GPT на власному контенті.
2. Використання LangChain для пошуку релевантних фрагментів у великому датасеті.
3. Використання Telegram Bot для консультування абітурієнтів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми отримання актуальної інформації про організації людською мовою, постановка задачі, тестові зображення, таблиці.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-11.04.24	
3	Аналіз літератури з досліджуваної проблеми	12.04.24-14.04.24	
4	Аналіз технічних засобів	15.04.24-17.04.24	
5	Отримання доступів до сторонніх API	18.04.24-18.04.24	
6	Програмна реалізація	19.04.24-15.05.24	
7	Оформлення пояснювальної записки	16.05.24-20.05.24	
8	Перевірка на плагіат	27.05.24	
9	Рецензування	28.05.24	
10	Підготовка презентації та доповіді	29.05.24-03.06.24	
11	Занесення роботи в електронний архів	03.06.24	
12	Попередній захист кваліфікаційної роботи	03.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Яковлева О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 66 с., 2 табл., 25 рис., 35 джерел.

МОДЕЛІ GPT, ДОНАВЧАННЯ НА ОСНОВІ ВЛАСНОГО КОНТЕНТУ, TELEGRAM-БОТ, CHATGPT, API OPENAI, LANGCHAI.

Об’єктом роботи є питання використання мовних моделей GPT для створення відповідей на основі власного контенту.

Метою роботи є розробка системи консультування абітурієнтів з використанням моделей GPT, інтеграції з телеграм-ботом, і дослідження можливостей «донавчання» мовних моделей на основі власного контенту. Робота включає розробку датасету, використання фреймворку LangChain для оптимального вибору фрагментів датасету, використання API для доступу до моделей GPT, та підготовку телеграм-бота для взаємодії.

Результати підтверджують, що використання «донавчання» на власному контенті значно підвищує точність та релевантність відповідей, що може бути використано у різних областях для поліпшення проінформованості та розуміння потреб користувачів.

GPT MODELS, FINE-TUNING ON OWN CONTENT, TELEGRAM BOT, CHATGPT, OPENAI API, LANGCHAIN.

The object of study is the use of GPT language models to generate responses based on proprietary content.

The purpose of the study is to develop a consulting system for university applicants using GPT models, integration with a Telegram bot, and exploring the possibilities of «fine-tuning» language models on proprietary content. The work includes the development of a dataset, the use of the LangChain framework for optimal selection of dataset fragments, the use of the API to access GPT models, and the preparation of a Telegram bot for interaction.

The results confirm that fine-tuning on proprietary content significantly increases the accuracy and relevance of responses, which can be used in various fields to improve informational awareness and understanding of user needs.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

NLP – Natural language processing (обробка природної мови)

GPT – Generative Pre-trained Transformer (генераторний попередньо навчений трансформер)

LLM – Large language model (велика мовна модель)

API – Application Programming Interface (інтерфейс прикладного програмування)

AI – Artificial intelligence (штучний інтелект)

MMLU – Massive Multitask Language Understanding (масштабне багатозадачне розуміння мови)

GSM8K – Grade School Math 8K (математика середньої школи 8000)

FFT – Fast Fourier Transform (швидкого перетворення Фур'є)

LSTM – Long Short-Term Memory (довготривала короткочасна пам'ять)

GRU – Gated Recurrent Units (окремі періодичні одиниці з закритим доступом)

BERT – Bidirectional Encoder Representations from Transformers (двонаправлені кодові представлення з трансформаторів)

LaMDA – Language Model for Dialogue Applications (мовна модель для діалогових застосунків)

ВСТУП

Сучасний розвиток штучного інтелекту відкриває нові можливості для використання мовних моделей у різних сферах життєдіяльності людини. Однією з найперспективніших областей є створення текстів за допомогою мовних моделей, зокрема моделей GPT від OpenAI, що можуть генерувати відповіді, які за своєю якістю схожі на людське мовлення.

Центральне місце у цьому напрямку займає використання мовних моделей для створення відповідей на основі власного контенту користувачів. Такий підхід дозволяє забезпечити високу релевантність та точність інформації, оскільки відповіді формуються на базі конкретних даних, що надходять від користувачів. Це стає невід'ємною частиною систем, які використовуються для консультацій, освіти та інших сфер де потрібна миттєва реакція на запити.

Актуальність таких систем зумовлена зростанням попиту на автоматизацію комунікаційних процесів в умовах великого обсягу інформаційних запитів, які можуть бути стандартизовані та автоматизовані за допомогою технологій штучного інтелекту. Інтеграція мовних моделей з телеграм-ботами дозволяє створювати зручні та ефективні інтерфейси для взаємодії з користувачами, що підвищує їх задоволеність і взаємодію з системою.

Робота присвячена дослідженню можливостей використання мовних моделей, зокрема моделей GPT від OpenAI, для створення відповідей на основі власного контенту. В контексті швидкого розвитку цифрових технологій та збільшення потреби у персоналізованих цифрових асистентах, значущість такого дослідження є над звичайно актуальною. Моделі GPT відомі своєю здатністю генерувати тексти, що наближені до людського рівня комунікації, що робить їх ідеальними кандидатами для розробки нейроконсультантів.

В центрі роботи – створення адаптованого нейроконсультанта для абітурієнтів, який би забезпечував достовірну та актуальну інформацію про

умови вступу, спеціальності, освітні програми та інші аспекти навчання у вищих навчальних закладах. Основна мета полягає в тому, щоб зменшити навантаження на живих консультантів та забезпечити швидкий та зручний доступ до інформації для майбутніх студентів.

Особлива увага приділяється розробці телеграм-бота, який би дозволяв абітурієнтам отримувати відповіді на свої запитання у будь-який час без необхідності чекати на відповідь від живого консультанта або відвідувати сторонні сайти. Такий бот може стати незамінним інструментом для абітурієнтів, надаючи інформацію про майбутню спеціальність, вимоги до вступу, особливості освітніх програм та інші важливі аспекти, що стосуються навчання на конкретній спеціальності в університеті.

Таким чином, ця мета роботи не тільки дослідити технологічні аспекти застосування GPT-моделей, але й визначити можливості їх практичної реалізації у сфері освіти, що може відіграти ключову роль у формуванні нових підходів до навчального процесу в університетах та інших освітніх закладах.

1 СУЧАСНИЙ СТАН РОЗВИТКУ РОЗВИТКУ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

1.1 Задачі обробки природної мови і прогрес у розвитку великих мовних моделей

Область обробки природної мови (Natural language processing, NLP) є фундаментальним напрямком у сфері штучного інтелекту, що досліджує взаємодію між людською мовою та комп'ютерами. Основна мета NLP полягає у розробці алгоритмів та систем, здатних розуміти, інтерпретувати, генерувати та відповідати на людську мову в її натуральній формі. Прогрес у цій області призвів до створення таких технологій, як машинний переклад, автоматичне розпізнавання мови, і системи автоматичного відповіді на запитання, що змінили спосіб взаємодії людей із технологічними системами.

Розвиток великих мовних моделей (Large language model, LLM), таких як GPT (Generative Pre-trained Transformer), відіграє ключову роль у цій трансформації. Завдяки їх здатності генерувати когерентний та релевантний текст на основі величезних масивів даних та навчання, ці моделі відкрили нові можливості для розширення меж існуючих систем автоматизації. LLM, як GPT, стали фундаментом для розробки застосунків, здатних проводити складні розмови, аналізувати емоції текстів, генерувати інформативні відповіді та навіть писати переконливі тексти, що знаходить застосування у сферах від маркетингу до освіти [1-3].

1.1.1 Основні завдання обробки природної мови

Основні завдання NLP включають широкий спектр методів та технологій, що дозволяють комп'ютерам розуміти, інтерпретувати та

відтворювати людську мову у її письмовій та усній формах [4]. Нижче наведені кілька ключових завдань, які входять до обробки природної мови:

– розуміння мови (одним із основних завдань NLP є розуміння природної мови, яке включає аналіз тексту для зрозуміння його смислу та контексту. Це задачі включають розуміння замислу (intent recognition) та витягування інформації (information extraction));

– генерація мови (задача генерації тексту полягає в створенні читабельного та змістовного тексту на основі вхідних даних. Це може бути автоматичне створення відповідей у чатах, генерація статей, складання резюме та інше);

– машинний переклад (включає переклад тексту або мовлення з однієї мови на іншу. Розвиток нейромереж і LLM значно покращив якість перекладів, роблячи їх більш точними та природними);

– сентимент-аналіз (визначає емоційний заряд тексту, допомагаючи аналізувати відгуки клієнтів, соціальні медіа та інші джерела).

1.1.2 Прогрес у розвитку великих мовних моделей

Розвиток LLM протягом останніх декількох років відбувається стрімко, внаслідок постійних інновацій в області машинного навчання та доступності все більших обсягів даних. Основні досягнення можна розділити на декілька ключових аспектів:

– збільшення масштабів моделей (сучасні LLM, такі як GPT-3 та його наступники, є прикладами моделей, які включають мільярди параметрів. Це збільшення масштабу дозволило значно покращити здатність моделей розуміти та генерувати людські мови, виконувати складні завдання з обробки природної мови та надавати більш точні та змістовні відповіді);

– покращення якості і розуміння контексту (зі збільшенням обсягу навчальних даних та удосконаленням алгоритмів навчання, LLM стали краще

розуміти нюанси мови та контекст, що веде до більш точного моделювання мовних відносин. Моделі тепер можуть з більшою точністю розпізнавати іронію, метафори, а також дотримуватись стилістичних особливостей тексту);

– етичні аспекти і безпека (з розвитком LLM зросла увага до етичних питань, пов'язаних з автоматизованим генеруванням тексту. Великі компанії та наукові інститути впроваджують протоколи для мінімізації упереджень в моделях та забезпечення безпечного використання штучного інтелекту);

– застосування та інтеграція (LLM стали не просто дослідницькими проєктами, але й інструментами, які інтегруються у широкий спектр прикладних програм від автоматизованого обслуговування клієнтів до підтримки рішень у бізнесі. Це відкриває нові можливості для використання технологій у багатьох сферах життя).

Ці досягнення підтверджують, що LLM продовжують бути на передовій технологічного прогресу, і їх вплив на суспільство та економіку тільки починає розкриватися [5].

1.2 Аналіз великих мовних моделей від різних виробників, їх недоліки та переваги

GPT від OpenAI, LaMDA та Gemini від Google, Sydney від Microsoft, Claude 3 (Opus, Sonnet) від Anthropic та Gemini, стали фундаментальними інструментами у сфері обробки природної мови, кожна з яких має свої унікальні переваги та обмеження. Деякі з цим моделей використовуються для задач з розумінням мови, деякі для розпізнавання зображень, створення коду, та забезпечення загальних знань. Оцінка їхньої ефективності здійснюється за допомогою спеціалізованих тестів, таких як MMLU для розуміння мови та GSM8K для шкільної математики. Ці моделі також відіграють важливу роль у визначенні нових шляхів для інновацій у технологічному світі, постійно покращуючи взаємодію між людиною та машиною [6]. Ключовою складовою

аналізу є також вартість використання цих технологій, що може значно варіюватися в залежності від обсягу та складності задач, що вони вирішують.

GPT від OpenAI. GPT є родиною моделей для обробки природної мови, розробленою компанією OpenAI. Основною характеристикою GPT є її здатність генерувати текст, що базується на попередньому навчанні з величезної бази даних текстів. Модель використовує архітектуру трансформера, яка дозволяє ефективно обробляти широкі контексти та зв'язки між словами в тексті. З часом GPT еволюціонувала від GPT-1 до останніх версій, як GPT-4, збільшуючи якість і точність текстів, що генеруються [7]. Модель широко використовується в різних системах, від автоматичного генерування тексту до більш складних задач, як машинний переклад чи автоматизація відповідей на запитання. Далі наведені переваги GPT:

– здатність до контекстуалізації (GPT може згенерувати текст, який враховує попередній контекст, що робить її відмінною для багатьох NLP задач);

– гнучкість в застосуванні (від чат-ботів до систем автоматизації, GPT виявилася корисною в різних доменах завдяки своїй адаптивності);

– велика масштабованість (моделі, як GPT-3 і GPT-4, показали вражаючі результати у збільшенні кількості параметрів та здатності обробляти великі об'єми даних);

– безперервне навчання (моделі GPT можуть покращуватися з часом, отримуючи зворотний зв'язок та нові дані).

До недоліків GPT можна віднести:

– висока вартість обчислень (навчання та використання моделей GPT великого масштабу вимагає значних обчислювальних ресурсів);

– тенденція до вироблення біасів (англ. bias) (оскільки GPT навчається на існуючих даних, вона може відтворювати існуючі упередження в тих даних);

– проблеми з безпекою та конфіденційністю (генерація текстів може випадково включати конфіденційну інформацію, якщо модель була неналежно

оброблена);

– переконливість згенерованих текстів (іноді модель може виробляти дуже переконливі, але невірні або вводячі в оману текстові відповіді).

Google LaMDA. Google LaMDA, розроблена компанією Google, є моделлю мови, оптимізованою для діалогових застосунків. Ця модель спеціалізується на створенні природних та змістовних розмов, що є ключовою характеристикою для Google Bard, розмовної служби штучного інтелекту компанії. LaMDA відрізняється від інших мовних моделей своєю здатністю вести діалог, який може змінюватися з плином розмови, реагувати на нюанси та відстежувати контекст. Модель була навчена на величезних масивах тексту з метою покращення релевантності відповідей та збереження контекстуальної згуртованості в розмовах [8]. Її архітектура базується на останніх досягненнях у технологіях трансформерів, забезпечуючи високу точність та різноманітність в генерації відповідей. Нижче наведені основні переваги моделі Google LaMDA:

– природність діалогів (LaMDA може вести розмову, що нагадує людський діалог, з підтримкою змінних тем і подій);

– здатність до контекстуалізації (модель здатна зберігати та враховувати контекст протягом тривалих діалогів);

– гнучкість у застосуванні (LaMDA є корисною у широкому спектрі застосунків, від інформаційних помічників до освітніх інструментів).

Слабкі сторони моделі Google LaMDA включають:

– точність інформації (іноді LaMDA може генерувати відповіді, які здаються правдоподібними, але містять неточності або помилки);

– обмеження на вироблення відповідей (через обережний підхід до уникнення шкідливого контенту, модель може бути обмеженою в певних темах або сценаріях);

– вимоги до ресурсів (подібно до інших великих моделей, LaMDA вимагає значних обчислювальних ресурсів для ефективної роботи).

Sydney від Microsoft. Модель Sydney, створена Microsoft на основі ChatGPT, призначена для інтеграції з пошуковою системою Bing, забезпечуючи більш зрозумілі й контекстуально відповідні відповіді на запитання користувачів. Sydney використовує архітектуру трансформера, яка дозволяє глибоко аналізувати і генерувати текст на основі навчання з великих датасетів. Важливою особливістю Сідней є його інтеграція з іншими продуктами Microsoft, що дозволяє використовувати допоміжні дані для підвищення якості відповідей. Модель орієнтована на покращення користувацького досвіду шляхом забезпечення більш релевантної, точної та швидкої взаємодії з користувачем під час пошукових запитів [9]. Далі розглянемо основні переваги Sydney:

– інтеграція з Bing (Sydney ефективно використовується для покращення пошукових результатів, вносячи більшу контекстуальну релевантність і точність);

– співпраця з іншими сервісами Microsoft (інтеграція з такими продуктами, як Office та Outlook, підвищує функціональність);

– динамічність у відповідях (модель здатна адаптуватися до різноманітних запитів, що підвищує її універсальність).

До недоліки Sydney входять:

– залежність від якості даних (як і більшість моделей на базі трансформерів, Сідней залежить від якості навчальних даних, що може включати упередження або помилки);

– приватність даних (інтеграція з різними сервісами може викликати занепокоєння стосовно обробки та зберігання персональних даних користувачів);

– високі вимоги до обчислювальних ресурсів (потреба в значних обчислювальних ресурсах для ефективної роботи моделі може обмежувати її доступність та широкомасштабне використання).

Серед розглянутих моделей, GPT від OpenAI виділяється завдяки своїй високій точності у формуванні відповідей, що базуються на розгорнутому

контексті, та ефективності в багаточисельних сферах застосування. ChatGPT, особливо його останні версії, показали вражаючі результати в швидкості та точності відповідей, що робить його ідеальним вибором для інтеграції в чат-боти та інші системи, що потребують надійного мовного інтерфейсу. Крім того, з урахуванням вартості використання, GPT виявляється одним із найбільш економічно вигідних варіантів на ринку, пропонуючи високу якість послуг при відносно низьких витратах. Така комбінація точності, швидкості відповідей та економічності робить GPT від OpenAI не лише технологічно прогресивним вибором, але й фінансово доступним рішенням для широкого спектру користувачів [10].

1.3 Сучасні бібліотеки та фреймворки для вирішення завдання обробки природної мови

Найбільшим досягненням у розвитку сучасних технологій стало машинне навчання та штучний інтелект, які невпинно трансформують багато галузей, включаючи NLP. На передньому краї цієї революції стоять деякі бібліотеки Python, що постійно розвиваються та адаптуються для задоволення високих вимог вчених та інженерів [11-14].

NumPy – це бібліотека, що походить із проєкту Numeric, є однією з основних бібліотек у Python для обробки даних та числових операцій. NumPy відома своєю високою продуктивністю у маніпуляціях з великими масивами та матрицями, що забезпечує значні переваги у швидкості та використанні пам'яті порівняно зі стандартними Python-списками. Це робить її незамінною для оптимізації продуктивності в машинному навчанні та штучному інтелекті, а також надає інструменти для інтеграції з кодом на C, C++ і Fortran.

У контексті задач NLP, NumPy відіграє критичну роль завдяки своїм функціям для обробки великих масивів даних. Бібліотека забезпечує основу для векторизації даних, що є ключовим аспектом у передпроцесингу тексту та

інших видах лінгвістичної обробки. Наприклад, у випадках, коли потрібно трансформувати слова у числові вектори або виконати матричні операції, такі як множення матриць, що часто використовується у моделях типу word embeddings та нейронних мережах.

SciPy – це розширена бібліотека Python, що використовується для виконання наукових та інженерних обчислень. Вона побудована на основі NumPy, що дозволяє їй використовувати всі переваги ефективної обробки масивів, пропонуючи при цьому більш широкий спектр наукових обчислювальних інструментів. SciPy містить модулі для оптимізації, лінійної алгебри, інтеграції, інтерполяції, спеціальних функцій, FFT, обробки сигналів та зображень, і багато інших, роблячи її ідеальним вибором для наукового аналізу та обробки великих наборів даних.

SciPy може використовуватись для різних завдань NLP.

Для завдань аналізу та оптимізації NLP:

– використання алгоритмів для мінімізації або максимізації функцій, що може бути застосовано для налаштування параметрів у моделях машинного навчання;

– рішення задач нелінійної оптимізації, що часто зустрічається при тренуванні NLP моделей.

SciPy легко інтегрується для завдань лінійної алгебри:

– обробка матриць та векторів, які часто використовуються для представлення текстових даних, таких як вектори с або матриці співвідношень між словами;

– розрахунок сингулярного розкладу або інших факторизацій, що є ключовими у зменшенні розмірності та виявленні прихованих семантичних структур у великих текстових наборах;

– обробка сигналів (застосування фільтрації та інших технік обробки сигналів до аудіо-даних для розпізнавання мовлення або аналізу мовних характеристик).

Статистичні функції: статистичний аналіз текстових даних включає визначення частотності слів, аналіз їх розподілу та інші статистичні характеристики для виявлення тенденцій та аномалій у тексті.

TensorFlow – це одна з провідних бібліотек машинного навчання, розроблена командою Google Brain. Вона відзначається своєю гнучкістю та масштабованістю, що дозволяє вирішувати завдання машинного навчання не тільки на потужних серверах, але і на мобільних пристроях, що робить її ідеальною для розробки як традиційних, так і новаторських застосунків. З випуском TensorFlow 2.0 у 2019 році, бібліотека отримала значні поліпшення у вигляді більш інтуїтивного API, покращення інтеграції з Keras, що значно спростило розробку та тренування глибоких нейронних мереж.

TensorFlow надає цілий ряд інструментів і можливостей для розв'язання складних завдань.

Обробка природної мови:

– використання предвстановлених архітектур нейронних мереж, таких як LSTM та GRU, для обробки послідовностей і текстів;

– моделювання взаємодій між словами та контекстів, необхідних для розуміння природної мови, що є основою для систем перекладу, автоматичного узагальнення текстів та чатботів.

Трансформерні моделі:

– впровадження архітектур на базі трансформерів, таких як BERT і GPT, які виявились надзвичайно ефективними для різноманітних NLP задач;

– застосування цих моделей для розуміння мовних взаємодій, аналізу сентименту, автоматичної відповіді на питання та інших завдань.

Оптимізація в реальному часі:

– використання TensorFlow Lite для стиснення та оптимізації моделей, щоби вони могли ефективно працювати на мобільних пристроях та вбудованих системах;

– можливість швидкого розгортання та використання нейронних мереж у реальних умовах з низькою затримкою, що важливо для застосунків,

які вимагають швидкої відповіді, таких як мобільні помічники та інтерактивні системи.

Масштабованість і гнучкість:

– підтримка розподіленого навчання, що дозволяє тренувати великі моделі на кластерах обчислювальних ресурсів;

– використання гнучких API для налаштування та тонкої настройки моделей, забезпечуючи необхідну адаптивність для дослідницьких проєктів та промислових застосунків.

1.4 Розвиток Telegtam ботів та програмне забезпечення для їх створення

Telegram боти стали невід'ємною частиною сучасних комунікаційних технологій, завдяки їхній простоті в розробці та великому спектру можливостей.

Telegram надає зручний API для розробки ботів, що дозволяє швидко створювати та впроваджувати їх. Для розробки ботів використовуються мови програмування, такі як Python, JavaScript, Java, а також фреймворки, що спрощують процес розробки.

Одним із основних застосувань Telegram ботів є відповіді на питання користувачів. Бот може бути налаштований для відповіді на питання на різні теми, використовуючи бази знань або аналізуючи текст питання за допомогою природної мови [15].

Telegram боти можуть бути використані для автоматизації різноманітних завдань, що значно полегшує життя користувачів. Наприклад, бот може надсилати сповіщення про події, керувати розкладом або навіть виконувати деякі дії за командами користувача.

Застосування Telegram ботів має свої переваги, такі як зручність використання та можливість швидкого розгортання. Однак, існують

обмеження, пов'язані з можливостями API та складністю деяких завдань, які не можуть бути повністю автоматизовані.

Для розробки Telegram ботів існує ряд програмних засобів, які допомагають спростити та прискорити процес розробки. Ці засоби надають різні можливості для створення та налаштування ботів залежно від потреб розробника.

BotFather є офіційним ботом Telegram для створення та керування іншими ботами. Для створення нового бота потрібно надіслати команду «/newbot» і вказати ім'я та користувацьке ім'я бота. Після цього BotFather надасть токен доступу, який потрібно буде використовувати для підключення бота до Telegram API.

Telegram API надає розробникам доступ до всіх можливостей платформи Telegram для створення ботів. З його допомогою можна налаштувати взаємодію бота з користувачами, обробляти повідомлення та виконувати різні дії відповідно до отриманих команд.

Для розробки Telegram ботів можна використовувати різні IDE, такі як IntelliJ IDEA або Eclipse. Вони надають зручні інструменти для написання, відладки та тестування коду бота.

Для спрощення розробки Telegram ботів існують різні фреймворки, які надають готові шаблони та функціонал для реалізації різних можливостей бота. Наприклад, для Java є фреймворки, такі як «telegrambots», що дозволяють швидко створювати функціональних ботів

Програмне забезпечення для створення Telegram ботів включає в себе різноманітні інструменти, які допомагають розробникам створювати та налаштовувати ботів залежно від їхніх потреб. Використання такого програмного забезпечення дозволяє створювати потужні та ефективні боти для взаємодії з користувачами та автоматизації різних завдань.

1.5 Постановка задачі

У сучасному світі технологій використання штучного інтелекту та машинного навчання стає дедалі популярнішим у різних галузях. Мовні моделі GPT відкривають нові можливості для автоматизації та оптимізації комунікаційних процесів, адаптуючись до специфічних потреб користувачів та контекстів.

Об’єктом роботи є питання використання мовних моделей GPT для створення відповідей на основі власного контенту.

Метою роботи є розробка системи консультування абітурієнтів з використанням моделей GPT, інтеграції з телеграм-ботом, і дослідження можливостей «донавчання» мовних моделей на основі власного контенту.

Основні завдання, які необхідно вирішити:

– вивчити підходи щодо донавчання мовних моделей GPT на специфічному контенті, забезпечуючи більшу релевантність та точність відповідей;

– аналізувати інструмент LanChain для ефективного вибору та формування релевантних фрагментів запитів користувачів;

– створити спеціалізований датасет із власного контенту, на базі якого GPT модель буде формувати відповіді;

– дослідити механізми формування промтів та налаштувати параметри моделі GPT для оптимізації її відповідей;

– розробити алгоритм створення відповідей на запити, базуючись на власному контенті;

– реалізувати телеграм-бота для автоматизації процесу консультування абітурієнтів;

– оцінити якість розробленого рішення на основі аналізу відгуків та оцінок експертів.

2 ПРОЄКТУВАННЯ TELEGRAM-БОТА З ВИКОРИСТАННЯМ GPT ВІД OPENAI ДЛЯ ВІДПОВІДЕЙ НА ОСНОВІ ВЛАСНОГО КОНТЕНТУ

2.1 Підходи до донавчання GPT моделей

У сфері донавчання моделей GPT існують два основні підходи.

Перший – це класичний метод, що передбачає зміну вагів нейронної мережі моделі GPT під час навчання. Другий підхід, який був обраний для використання в даній роботі, – це «донавчання» на основі передачі власного контенту під час звертання до моделі GPT.

2.1.1 Класичний метод, що передбачає зміну вагів нейронної мережі моделі GPT

Цей підхід до навчання моделей GPT передбачає зміну вагів нейронної мережі під час процесу навчання. Цей підхід базується на методі градієнтного спуску, який полягає у пошуку мінімуму функції втрат (loss function) шляхом зміни вагів моделі в напрямку, протилежному градієнту функції втрат.

Під час навчання моделі GPT використовується функція втрат, яка вимірює різницю між очікуваним вихідним значенням та фактичним вихідним значенням моделі. Ця функція втрат є диференційованою, що дозволяє обчислити градієнт функції втрат відносно кожного параметра моделі.

Після обчислення градієнта для кожного параметра моделі виконується крок градієнтного спуску, в результаті чого ваги моделі змінюються в напрямку, що зменшує значення функції втрат. Цей процес повторюється протягом багатьох ітерацій навчання, поки значення функції втрат не стабілізується або не досягне заданого рівня [16].

Формула для оновлення вагів моделі в кожній ітерації може бути виражена наступним чином:

$$W_{new} = W_{old} - \alpha \cdot \nabla L(W_{old}), \quad (2.1)$$

де W_{new} – нове значення ваги;

W_{old} – попереднє значення ваги;

α – крок навчання (learning rate), який визначає швидкість навчання;

$\nabla L(W_{old})$ – градієнт функції втрат по вазі W_{old}.

Цей процес відбувається для кожного параметра моделі, що дозволяє моделі адаптуватися до вхідних даних та покращувати свою точність у вирішенні завдань, для яких вона навчалася.

Відмінності в методах донавчання моделей GPT відображають важливі аспекти в їх застосуванні та впливають на ефективність та зручність використання [17]. Розглянемо детальніше переваги і недоліки:

Переваги класичного методу донавчання моделей GPT зі зміною вагів нейронної мережі:

– гнучкість (метод дозволяє точно налаштовувати ваги моделі, що може призводити до кращої адаптації до конкретних завдань чи даних);

– універсальність (метод може бути застосований до різних завдань і типів даних, оскільки зміна вагів може покращити виконання моделі у багатьох контекстах);

– ефективність (у випадках, коли немає доступу до великої кількості власного контенту для донавчання, класичний метод може бути ефективнішим, оскільки вимагає менших обсягів даних для досягнення певного рівня продуктивності).

Недоліки класичного методу донавчання моделей GPT зі зміною вагів нейронної мережі:

– потреба у великій кількості даних (для досягнення високої якості відповідей моделі може знадобитися значна кількість навчальних даних, що

може бути складним або дорогим процесом для деяких організацій);

– обмежена спеціалізація (класичний метод може виявитися менш ефективним для спеціалізованих завдань, де потрібна висока точність або глибока адаптація до конкретного контексту);

– висока вартість (донавчання великих мовних моделей вимагає значних обчислювальних ресурсів. Залучення потужних обчислювальних систем і серверів може бути дорогим, особливо якщо вам потрібно тренувати модель на великих обсягах даних або вносити часті оновлення).

2.1.2 Донавчання на основі передачі власного контенту

На рисунку 2.1 зображено алгоритм донавчання моделі GPT за допомогою системи пошуку та інтеграції інформації [18]:

– подача запиту (користувач вводить запит через користувацький інтерфейс);

– передача в систему пошуку (запит перенаправляється до системи пошуку, яка відповідає за пошук відповідей у базі документів);

– семантичний пошук (система пошуку використовує семантичні алгоритми для аналізу запиту та пошуку відповідних документів у базі);

– отримання відповідних документів (знайдені документи, які вважаються релевантними до запиту користувача, вибираються для подальшого використання);

– компіляція нового запиту (вибрані документи разом із оригінальним запитом формують новий розширений запит, який передається до LLM);

– генерація відповіді (модель LLM обробляє розширений запит та генерує відповідь);

– надсилання відповіді користувачеві (генерована відповідь відправляється користувачеві через користувацький інтерфейс).

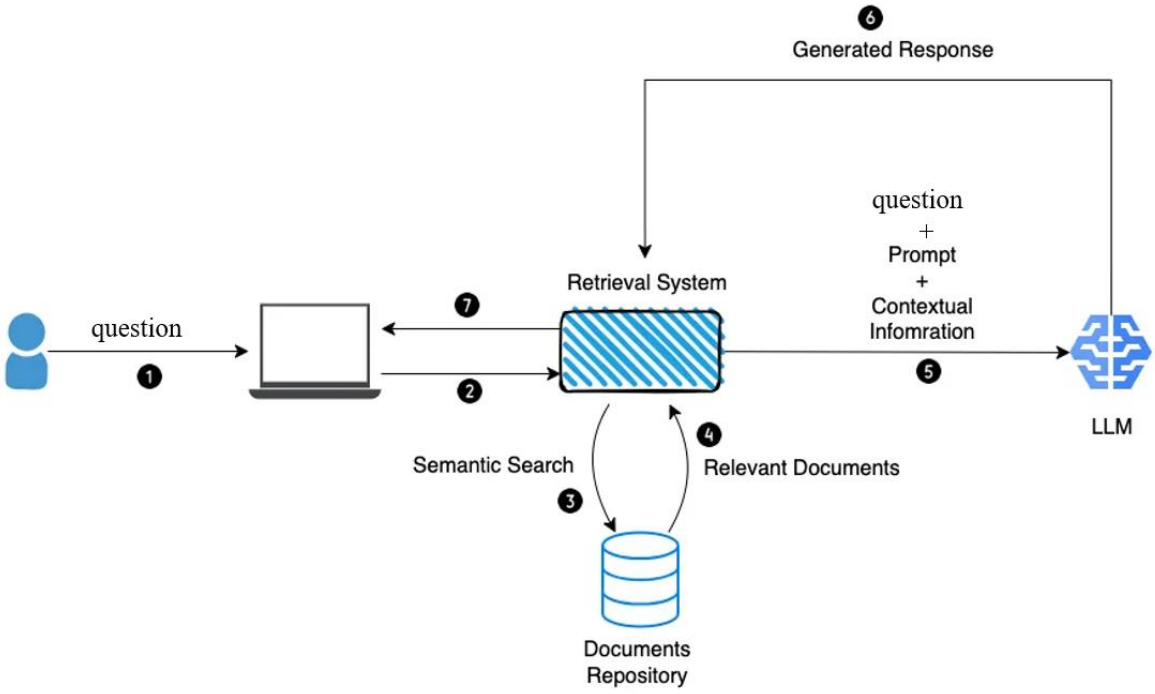


Рисунок 2.1 – Алгоритм генерування відповідей моделлю GPT на основі власного контенту

Цей процес дозволяє не тільки забезпечити більш точні та релевантні відповіді, але й постійно покращувати знання моделі, оскільки кожен такий цикл може бути використаний для її донавчання.

Переваги методу «донавчання» на основі передачі власного контенту:

– конфіденційність даних (метод дозволяє зберігати конфіденційність власних даних, оскільки не вимагає завантаження конфіденційної інформації на сервери OpenAI);

– економія часу та витрат (не потрібно витрачати час і ресурси на створення великого датасету для навчання, оскільки використовується власний контент);

– зменшення витрат (не потрібно оплачувати вартість донавчання на платформі OpenAI, оскільки завантаження датасету на сервери OpenAI та саме донавчання моделі на платформі є платними послугами);

– актуальність інформації (дозволяє використовувати найактуальнішу інформацію для генерації відповідей, що підвищує якість відповідей та задоволеність користувачів).

Недоліки методу «донавчання» на основі передачі власного контенту:

– обмежена кількість даних (для ефективного донавчання може знадобитися значна кількість власного контенту, що може бути обмеженням для деяких організацій);

– можливість перенавчання (використання власного контенту може призвести до перенавчання моделі на обмеженому обсязі даних, що може погіршити її загальну продуктивність);

– потреба у внутрішніх ресурсах (для успішного впровадження методу може знадобитися додаткові зусилля та ресурси для обробки та передачі власного контенту до моделі).

Базуючись на аналізі переваг та недоліків обох методів донавчання моделей GPT, виявлено, що метод «донавчання» на основі передачі власного контенту є більш перспективним та ефективним. Він дозволяє досягти високої якості результатів без необхідності створення великого навчального датасету, зберігаючи при цьому конфіденційність власних даних і уникненням додаткових витрат на платформі OpenAI. Такий підхід також дозволяє використовувати актуальну інформацію для генерації відповідей, що підвищує користувацьке задоволення від використання продукту.

2.2 Ролі для звернення до GPT-моделі

У розробці інтерфейсів, що взаємодіють з мовними моделями GPT від OpenAI, важливо зрозуміти ролі, які задіяні в процесі формування та отримання відповідей. Основні ролі, які беруть участь у цьому процесі, це: Система (System), Користувач (User) та Помічник (Assistant).

Система (System) відіграє ключову роль у визначенні контексту та параметрів запитів, які надсилаються до GPT-моделі. Вона може включати в себе інструкції та власний контент, який має бути врахований моделлю під час генерації відповіді. Система може також фільтрувати та підбирати релевантний контент на основі вхідних даних від користувача, що дозволяє моделі зосередитись на найбільш важливих аспектах запиту.

Користувач (User) здійснює запити до системи. Це може бути в формі питань, команд або введення даних, які потребують обробки або відповіді. Роль користувача важлива для визначення типу інформації, яку необхідно обробити, та контексту, в якому цю інформацію слід інтерпретувати.

Помічник (Assistant) – це власне GPT-модель, яка генерує відповіді на запити користувача. Ця модель використовує дані, які їй надає система, та формує відповідь, яка відображає зібрані вказівки та контент. Помічник адаптує свої відповіді залежно від контексту запиту та наданої інформації, що робить взаємодію більш природною та ефективною.

Інтеграція цих трьох ролей забезпечує плавну та ефективну взаємодію між користувачем та системою [19]. Система підготовляє контекст та направляє запити користувача до моделі помічника, яка генерує відповіді, використовуючи передані інструкції та контент. Цей процес дозволяє швидко та точно відповідати на запити користувача, оптимізуючи взаємодію для досягнення кращих результатів.

2.3 Обмеження за токенами у моделях GPT

З постійним розвитком технологій штучного інтелекту, зокрема моделей GPT, стає все більш актуальним питання обмеження за токенами, що впливає на вартість та ефективність їхнього використання. Більш детально розуміння цієї проблематики, можна переглянути у таблиці 2.1.

Таблиця 2.1 – Обмеження за токенами у моделях GPT

Модель	**Використані моделі**	**Контекстне вікно**	**Навчальні дані**	**Введення**	**Виведення**
GPT-4 Turbo	gpt-4-turbo-2024-04-09	128,000 tokens 300 pages of text	Up to Dec 2023	$10.00/1M tokens	$30.00/1M tokens
				Візуальний калькулятор цін: 150px*150px - $0.00255	
GPT-4	gpt-4 (gpt-4-0613) gpt-4-32k	8,192 tokens 32,768 tokens	Up to Sep 2021	$30.00/1M tokens $60.00/1M tokens	$60.00/1M tokens $120.00/1M tokens
GPT-3.5 Turbo	gpt-3.5-turbo-0125 gpt-3.5-turbo-instruct	16,385 tokens 4,096 tokens	Up to Sep 2021	$0.5/1M tokens $1.50/1M tokens	$1.50/1M tokens $2.00/1M tokens

GPT-4 Turbo. Модель GPT-4 Turbo, запущена 9 квітня 2024 року, має значно збільшену кількість токенів до 128,000 на контекст, що дозволяє обробляти до 300 сторінок тексту. Таке збільшення ліміту токенів відкриває нові можливості для більш глибокого аналізу текстів та комплексного вирішення завдань. Введення цін становить $10 за 1 мільйон токенів для базового використання і до $30 за 1 мільйон токенів для розширеного.

GPT-4. Стандартна версія GPT-4, представлена до вересня 2021 року, має обмеження від 8,192 до 32,768 токенів. Це ставить певні рамки для користувачів, залежно від конкретних потреб і завдань, які вони хочуть

вирішити за допомогою моделі. Ціни на використання цієї моделі варіюються від $30 до $120 за мільйон токенів.

GPT-3.5 Turbo. Нарешті, GPT-3.5 Turbo, що також була доступна до вересня 2021 року, пропонує від 4,096 до 16,385 токенів. Це робить її доступною для менших проєктів або для тих, хто не потребує великої кількості токенів для своїх задач. Вартість використання цієї моделі є найнижчою і становить від $0,5 до $2 за мільйон токенів.

Ці обмеження за токенами не тільки впливають на вартість використання моделей, але й визначають гнучкість та ефективність їх застосування у різних сценаріях [20].

2.4 Пошук релевантних фрагментів за допомогою фрейворку LangChain

LangChain – це бібліотека, яка спрощує інтеграцію мовних моделей з іншими інформаційними системами і базами даних. Однією з ключових можливостей LangChain є пошук релевантних фрагментів у великих обсягах тексту, що є особливо корисним для розробки інтелектуальних систем, які потребують ефективного витягу інформації [21, 22].

Фреймворк включає інструменти для зв'язування мовних моделей з базами даних, пошуковими системами і іншими джерелами даних. LangChain дозволяє розробникам конфігурувати власні пайплайни обробки даних, інтегрувати кастомізовані модулі для витягу і аналізу інформації.

Пошук релевантних фрагментів у тексті використовуючи LangChain можна розглянути як систематичний процес, що включає декілька ключових етапів: від інтерпретації користувацького запиту до відображення релевантних результатів [23]. Далі описано детальний процес:

– ініціалізація запиту (користувач вводить запит у природній мові, наприклад, через вебінтерфейс або мобільний застосунок. Цей запит може

бути питанням, ключовим словом або складнішим описом потрібної інформації);

– обробка та аналіз запиту (мовна модель, інтегрована через LangChain, аналізує текст запиту для визначення ключових термінів та контексту запиту. Цей етап може включати розпізнавання іменованих сутностей, визначення інтенцій запиту, і розподіл запиту на менші компоненти для детальнішого аналізу);

– визначення джерел даних (на основі аналізу запиту, система визначає, які джерела даних (бази даних, документи, вебсторінки тощо) мають потенціал містити відповіді. Це рішення може базуватись на метаданих, доступності джерел, або попередньо визначених правилах);

– витяг даних (система здійснює запити до вибраних джерел даних. Це може включати запуск SQL запитів до баз даних, API виклики до зовнішніх сервісів, або локальний пошук у файловій системі. Дані можуть бути витягнуті у вигляді цілих документів або специфічних сегментів тексту);

– вибір та ранжування релевантних фрагментів (отримані дані аналізуються для визначення релевантності до запиту користувача. Цей аналіз може включати використання технік машинного навчання для ранжування фрагментів за їхньою потенційною корисністю та відповідністю до запиту. Моделі можуть оцінювати текст на основі семантичної близькості, ключових слів, або інших характеристик тексту);

– подання результатів (релевантні фрагменти представляються користувачу у зручному і інформативному форматі. Залежно від застосування, це може бути текстова відповідь, таблиця, або навіть інтерактивний інтерфейс, що дозволяє користувачу далі взаємодіяти з отриманою інформацією).

На зображенні 2.2 ілюстровано детальний процес використання фреймворку LangChain для пошуку релевантних фрагментів тексту: від поділу документів PDF на чанки, створення вбудовань (embeddings) для кожного чанку, зберігання цих вбудовань у векторній базі даних, до семантичного

пошуку та ранжування результатів за допомогою мовної моделі для забезпечення кінцевої відповіді на запит користувача.

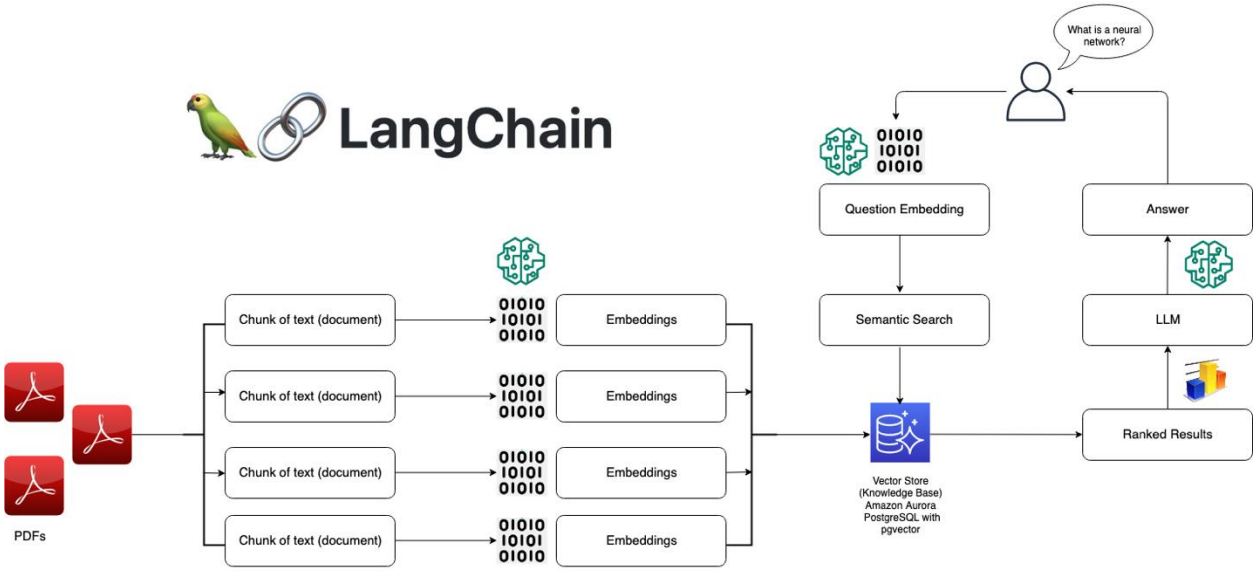


Рисунок 2.2 – Схема процесу пошуку релевантних фрагментів за допомогою фреймворку LangChain

LangChain надає API, який може бути інтегрований у різноманітні застосунки. Наприклад, можна створити систему, яка дозволяє користувачам ставити запитання в натуральній мові, і система буде автоматично знаходити і показувати найбільш релевантні відповіді. Такі системи можуть бути корисні в юридичних базах даних, наукових дослідженнях, або навіть у підтримці клієнтів.

Перевагами використання LangChain є гнучкість в інтеграції, висока точність витягу релевантних фрагментів і можливість адаптації під специфічні потреби користувача. Однак, серед обмежень можна виділити залежність від якості вхідних даних і потребу в регулярному оновленні моделей для підтримки актуальності відповідей.

Фреймворк LangChain є потужним інструментом для розробки інтелектуальних систем, здатних ефективно взаємодіяти з користувачами за

допомогою мовних запитів. Завдяки своїй гнучкості та ефективності, LangChain знаходить застосування в різних областях, від наукових досліджень до комерційних програм.

2.5 Параметри GPT моделей від OpenAI і LangChain

GPT моделі від OpenAI є одними з найпотужніших інструментів у сучасному світі NLP. Вони використовують передові технології машинного навчання для генерації тексту. Основа цих моделей лежить на архітектурі трансформера, яка дозволяє аналізувати та генерувати текст на основі контексту і семантичних відносин між словами. Різні параметри, які можна налаштувати у цих моделях, забезпечують велику гнучкість і адаптивність у їх застосуванні для різноманітних задач.

Параметри моделі GPT, такі як модель і температура, відіграють вирішальну роль у визначенні поведінки та результативності генерації тексту. Їхній вплив на якість та адаптивність відповідей моделі є критичним, особливо у складних застосуваннях, де необхідні глибоке розуміння контексту та творчий підхід. Детальніше про це можна дізнатися у таблиці 2.2, яка включає специфікації та налаштування GPT моделей [24].

Параметр «model» визначає конкретну версію моделі GPT, яка буде використовуватися для обробки запитів. Кожна модель має свої характеристики, розмір, та набір навчальних даних, на яких вона була тренована. Наприклад, моделі на кшталт «gpt-4-turbo-2024-04-09» є більш складними і потужними у порівнянні з попередніми версіями, що забезпечує краще розуміння контексту, здатність до складнішої інференції та генерацію більш релевантних відповідей. Вибір моделі впливає на якість вихідного тексту, швидкість відповіді та можливості адаптації до специфічних вимог користувача.

Таблиця 2.2 – Параметри моделей GPT від OpenAI

Назва параметра	**Можливі значення**	**Призначення**
max_tokens	Числове значення	Визначає максимальну кількість токенів у відповіді
temperature	Число від 0 до 1	Регулює рівень випадковості або творчості у відповідях
top_k	Числове значення	Вибірка токенів з фіксованого числа *k* найбільш імовірних токенів.
top_p (вибірка ядра)	Десяткове число від 0 до 1	Динамічний вибір токенів, де сумарна ймовірність перевищує заданий поріг *P*.
frequency_penalty	Число від 0 до 1	Знижує ймовірність повторення слов, що вже зустрічалися в тексті.
presence_penalty	Число від 0 до 1	Знижує ймовірність повторення слов, які вже згадувалися, незалежно від частоти.
best_of	Числове значення	Генерація кількох відповідей для вибору найкращої.
stop	Масив рядків	Зупиняє генерацію тексту при досягненні вказаної послідовності.
echo	true/false	Включає початковий промпт в відповідь.
logprobs	Числове значення	Повертає логарифмічні ймовірності для вказаної кількості найімовірніших токенів.
model	Ідентифікатор моделі (наприклад, «text-davinci-002»)	Визначає, яку модель GPT використовувати для обробки запиту.
user	Унікальний ідентифікатор користувача	Ідентифікує та відстежує запити від конкретного користувача.

Параметр температура визначає рівень випадковості або творчості у відповідях, що генеруються моделлю. Чим вища температура, тим більш різноманітні та непередбачувані відповіді модель може генерувати. Це особливо корисно у задачах, де потрібен креативний підхід, наприклад, при генерації художнього тексту, маркетингових копій або інших ситуаціях, де варіативність і новизна є цінними. З іншого боку, нижча температура сприяє більш консервативним та передбачуваним результатам, що є ідеальним при задачах з високими вимогами до точності, такими як формування ділових документів або технічних інструкцій.

Разом, параметри модель і температура дозволяють користувачам налаштувати поведінку моделі GPT для вирішення специфічних задач. Вони забезпечують необхідний баланс між інноваційністю та передбачуваністю, впливаючи на якість і релевантність відповідей, що генеруються. Тому, розуміння та правильне використання цих параметрів є ключовим для ефективного використання можливостей, які пропонують GPT-моделі.

Ось деякі з основних параметрів, які можна налаштувати у LangChain:

– model configuration (ви можете налаштувати модель, яка буде використовуватися для обробки мови. Це може бути OpenAI GPT, який можна інтегрувати через API, або будь-яка інша модель, сумісна з Hugging Face Transformers);

– chain components (LangChain дозволяє створювати ланцюги з різних компонентів, таких як розпізнавачі намірів, виконавці дій, контролери перебігу розмови тощо. Кожен компонент може мати свої налаштування, які визначають його поведінку);

– middleware configuration (можна використовувати проміжне програмне забезпечення для зміни запитів до моделі або відповідей від неї, наприклад, для кешування відповідей або додавання додаткової обробки);

– logging and monitoring (налаштування параметрів логування та моніторингу дозволяють відстежувати виконання вашого мовного агента і виявляти потенційні проблеми);

– security settings (забезпечення безпеки даних та запитів до вашого мовного агента є критично важливим, тому LangChain надає можливості для налаштування параметрів безпеки, таких як аутентифікація та шифрування.

Особливості цих інструментів демонструють великий потенціал для адаптації до різноманітних задач обробки природної мови, що дозволяє досягати високої точності та креативності у генерації тексту. GPT моделі від OpenAI з їх гнучкими налаштуваннями параметрів, такими як модель і температура надають можливість впливати на стиль та вміст генерованих відповідей, що є надзвичайно корисним у комплексних застосуваннях, де необхідно зберігати баланс між інноваційністю та передбачуваністю.

LangChain, в свою чергу, розширює можливості використання LLM шляхом інтеграції різних мовних компонентів і налаштувань безпеки, що дозволяє створювати більш комплексні та безпечні мовні аплікації. Це відкриває нові можливості для розробки інтелектуальних систем, які можуть ефективно інтерпретувати користувацькі запити, вирішувати завдання та взаємодіяти з користувачами в різноманітних контекстах.

Загалом, розуміння та правильне використання параметрів моделей GPT та можливостей LangChain можуть значно покращити якість та ефективність мовних агентів у широкому спектрі застосувань. Це важливо для розробників та дослідників, які прагнуть максимально використати потенціал сучасних технологій у галузі штучного інтелекту і машинного навчання.

2.6 Структура та алгоритм роботи Telegram-бота з використанням GPT від OpenAI для відповідей на основі власного контенту

Telegram-боти забезпечують гнучкий спосіб взаємодії з користувачами через популярний месенджер Telegram, використовуючи його бот API. Щоб розглянути структуру та алгоритм роботи такого бота, подивимося на ключові компоненти та процес обробки запитів користувачів [25].

Структура Telegram-бота:

– telegram API (це інтерфейс, через який бот взаємодіє з Telegram. Він дозволяє боту отримувати повідомлення від користувачів, відправляти їм відповіді, оновлювати повідомлення, відправляти медіафайли, керувати групами та виконувати багато інших дій);

– сервер бота (вебсервер або хмарний сервіс, де розгорнуто бота. Він обробляє логіку бота, зберігає дані, взаємодіє з зовнішніми API та базами даних. Цей сервер є місцем, де зосереджена вся «інтелектуальна» частина бота);

– база даних (система для зберігання даних, що використовується ботом для збереження інформації про користувачів, їхні запити, логи взаємодії та інші важливі дані.);

– модуль обробки команд (цей модуль відповідає за розпізнавання та обробку команд, які вводять користувачі. Він аналізує текст повідомлень та запускає відповідні дії або скрипти);

– модуль взаємодії з користувачем (забезпечує формування відповідей та їх відправлення користувачам, може містити шаблони повідомлень, клавіатури для зручної навігації користувача (рис. 2.3)).

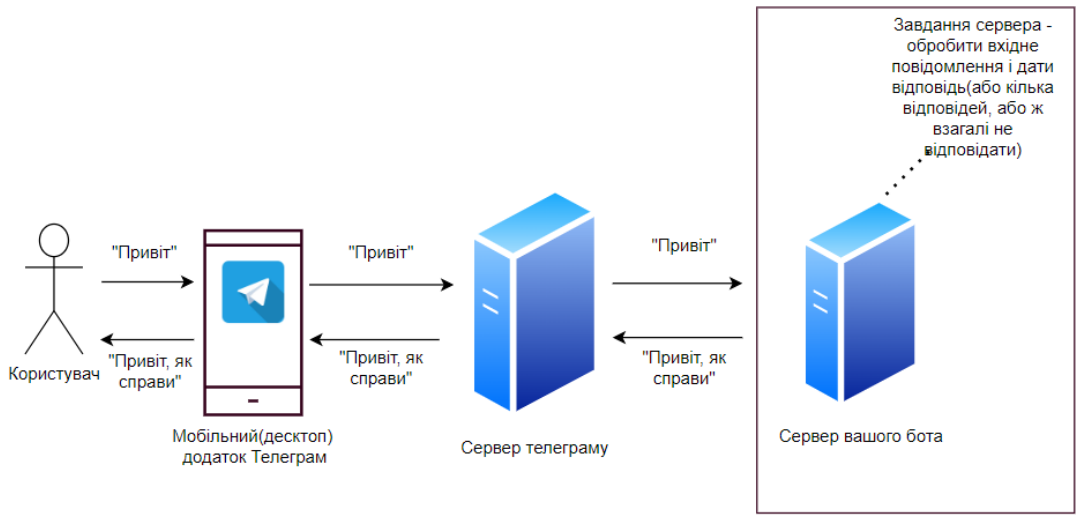


Рисунок 2.3 – Простий алгоритм роботи Telegram-бота

Алгоритм роботи Telegram-бота з використанням GPT від OpenAI для відповідей на основі власного контенту.

Збір та індексація контенту:

– збираються дані з різних джерел, наприклад, текстові файли, документацію, PDF, тощо, що формують приватну базу знань;

– перетворення контенту в ембедінги (використовуючи модель ембедінгів, бот перетворює кожен фрагмент тексту з бази даних у вектор у високорозмірному просторі (embedding space));

– індексація векторів (створені вектори зберігаються в базі даних векторів та індексуються для швидкого пошуку за допомогою методів наближеного пошуку найближчих сусідів (Approximate Nearest Neighbor, ANN)).

Обробка користувацьких запитів:

– отримання запиту від користувача (користувач надсилає запит через інтерфейс чату в Telegram);

– перетворення запиту в ембедінг (запит користувача перетворюється в вектор за допомогою тієї ж моделі ембедінгів, що й для контенту);

– пошук в індексі (вектор запиту використовується для пошуку найближчих сусідів у векторній базі даних, що дозволяє знайти найбільш релевантний контент у базі знань).

Генерація відповідей:

– формування запиту до LLM (вибрані вектори з бази даних та сам запит користувача формують запит до великої мовної моделі, такої як GPT від OpenAI (рис. 2.4));

– генерація відповіді (LLM обробляє запит та генерує відповідь на основі контексту, що було знайдено. Відповідь конструюється з урахуванням контексту без використання зовнішніх джерел, лише на основі вже індексованих даних);

– відправлення відповіді користувачу (відповідь надсилається користувачу через інтерфейс чату в Telegram).

Збір зворотного зв'язку: зворотний зв'язок від користувача (бот може запитувати зворотний зв'язок про корисність відповіді для подальшого удосконалення).

Використання великих мовних моделей, таких як GPT від OpenAI, у поєднанні з сучасними методами індексації та пошуку в ембедінговому просторі, відкриває нові можливості для розробки чат-ботів в месенджерах, зокрема в Telegram. Ця технологія дозволяє створити систему, яка не тільки здатна генерувати релевантні відповіді на запити користувачів, але й швидко адаптуватися та вчитися на основі зворотного зв'язку, підвищуючи якість взаємодії [26].

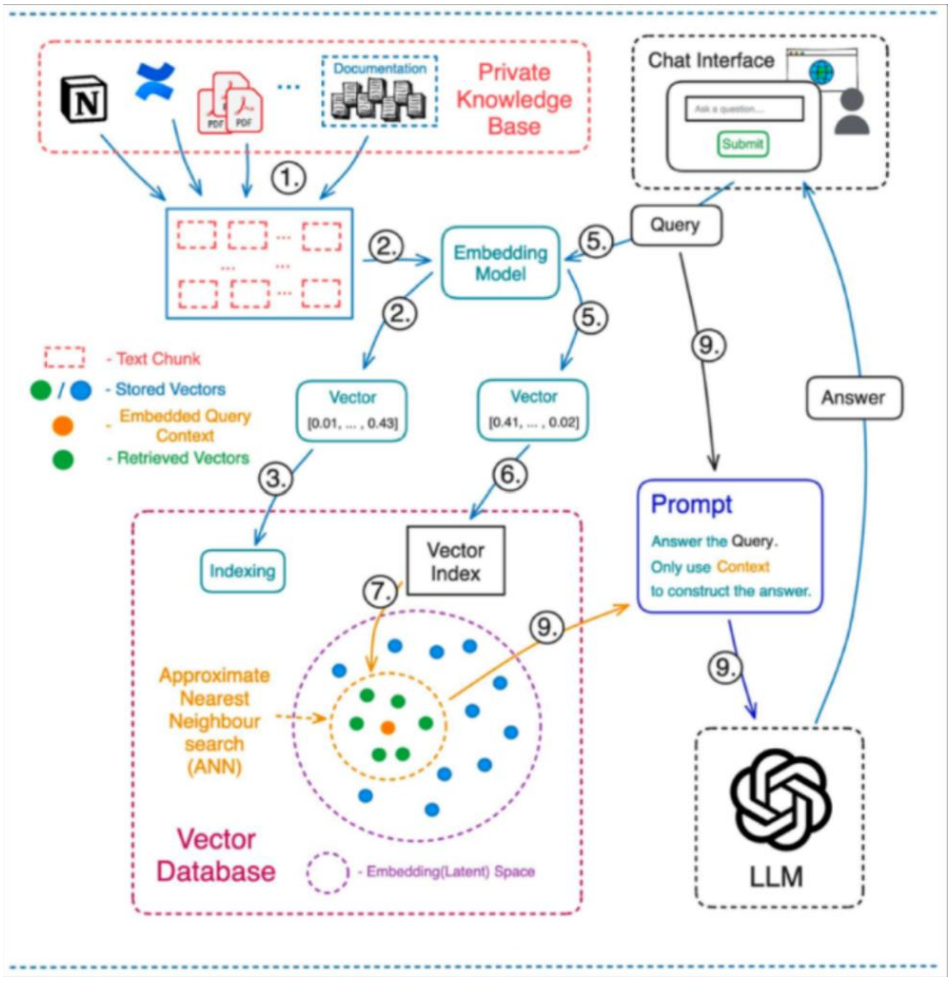


Рисунок 2.4 – Алгоритм роботи Telegram-бота з використанням GPT від OpenAI для відповідей на основі власного контенту

Завдяки такому підходу, можливості для розширення функціональності та подальшої інтеграції ботів в різноманітні бізнес-процеси та сфери обслуговування здаються майже необмеженими. Водночас, це також ставить певні вимоги до безпеки та приватності даних, оскільки обробка особистої інформації користувачів має відповідати сучасним стандартам захисту інформації.

3 РОЗРОБКА TELEGRAM-БОТА ДЛЯ КОНСУЛЬТУВАННЯ АБІТУРІЄНТІВ УНІВЕРСИТЕТУ

3.1 Налаштування програмного середовища

Для реалізації проєкту було використано низку залежностей, що додаються у файл конфігурації (POM.xml) проєкту (рис. 3.1). Кожна з цих залежностей має своє призначення, забезпечуючи необхідні можливості для розвитку та функціонування проєкту:

– «dev.langchain4j» (ця залежність використовується для інтеграції з бібліотекою LangChain, яка дозволяє використовувати можливості обробки природної мови та інтерактивної комунікації в межах Java програм. Вона забезпечує доступ до розширених функцій для розробки мовних моделей та їх впровадження у вебпроєкти);

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>dev.langchain4j</groupId>
    <artifactId>langchain4j</artifactId>
    <version>0.23.0</version>
</dependency>
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```

Рисунок 3.1 – Залежності у POM.xml проєкту

– «org.springframework.boot:spring-boot-starter-data-jpa» (ця залежність додається для підтримки роботи з базами даних через Java Persistence API (JPA). Вона включає необхідні бібліотеки для роботи з сутностями, управління транзакціями та доступом до даних, що значно спрощує розробку і підтримку баз даних у великих проєктах);

– «org.springframework.boot:spring-boot-starter-web» (ця залежність використовується для розробки вебзастосунків на платформі Spring. Вона включає підтримку створення RESTful вебсервісів, обробку HTTP запитів, серіалізацію та десеріалізацію JSON, а також інші інструменти для розробки вебінтерфейсів);

– «org.postgresql:postgresql» (ця залежність необхідна для забезпечення підключення до бази даних PostgreSQL. Вона використовується у режимі «runtime», оскільки необхідна лише під час виконання програми для з'єднання з базою даних і виконання SQL запитів).

Для належної конфігурації та адаптації проєкту до конкретних вимог розробки та середовища виконання, було налаштовано файл application.properties (рис. 3.2). Цей файл використовується для збереження різних параметрів конфігурації, які керують поведінкою програми. Встановлені налаштування включають з'єднання з зовнішніми сервісами, конфігурацію бази даних, а також деталі вебсервісу та бота. Ось детальний опис встановлених параметрів:

– «openai.model» та «openai.api.key» ці параметри використовуються для інтеграції з API OpenAI (вони визначають, яка модель машинного навчання («gpt-3.5-turbo») буде використовуватися, а також ключ API для авторизації запитів);

– «bot.name», «bot.token», і «bot.owner» є основними налаштуваннями для конфігурації Telegram бота, який в даному випадку називається «NureConsultant» («bot.token» та «bot.owner» – це токен для доступу до API Telegram та ідентифікатор власника бота відповідно, що дозволяє контролювати бота і управляти його поведінкою);

```
openai.model=gpt-3.5-turbo
openai.api.key=${API_KEY}
bot.name=NureConsultant
bot.token=${BOT_TOKEN}
bot.owner=111

spring.datasource.url=jdbc:postgresql://localhost:5432/nure
spring.datasource.username=postgres
spring.datasource.password=vivid1
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Рисунок 3.2 – Налаштування application.properties проєкту

– «spring.datasource» ці параметри задають деталі для підключення до бази даних PostgreSQL, включаючи URL, ім'я користувача («postgres»), та пароль («vivid1») (це дозволяє програмі здійснювати з'єднання з базою даних і виконувати операції з даними);

– «spring.jpa» налаштування Java Persistence API (JPA) забезпечують управління схемою бази даних, відображення запитів SQL, форматування та діалект для PostgreSQ (властивість «spring.jpa.hibernate.ddl-auto=update» автоматично оновлює схему бази даних при запуску, а «spring.jpa.show-sql=true» та «spring.jpa.properties.hibernate.format_sql=true» відповідають за виведення і форматування SQL запитів для кращого розуміння та дебагінгу).

Для налаштування бази даних в проєкті необхідно використовувати Docker, який дозволяє створювати ізольовані контейнери для запуску програмного забезпечення. Використання Docker забезпечує консистентність середовища від розробки до продакшну, спрощує розгортання та управління залежностями.

Для створення контейнера з базою даних PostgreSQL можна використовувати наступний скрипт Docker, який використовує наступні команди (рис. 3.3):

– «docker run» команда для створення і запуску нового контейнера;

```
docker run -d --name nure -p 5432:5432 -e POSTGRES_PASSWORD=vivid1 -e
POSTGRES_DB=nure postgres:latest
```

Рисунок 3.3 – Скрипт для підняття бази даних у докер контейнері

– «-d» опція, яка дозволяє запускати контейнер в фоновому режимі;

– «--name nure» призначає ім'я контейнеру «nure», що полегшує його ідентифікацію та управління;

– «-p 5432:5432» мапінг портів, який вказує, що порт 5432 всередині контейнера відповідає порту 5432 на хост-машині, дозволяючи доступ до бази даних;

– «-e POSTGRES_PASSWORD=vivid1» встановлює змінну середовища в контейнері, що визначає пароль для користувача бази даних;

– «-e POSTGRES_DB=leads» встановлює змінну середовища, яка створює базу даних із назвою «leads» при першому запуску контейнера;

– «postgres:latest» використовує останню версію офіційного образу PostgreSQL з Docker Hub.

Після успішного запуску контейнера за допомогою Docker, важливим інструментом для моніторингу та управління є Docker Desktop. Цей графічний інтерфейс користувача дозволяє візуалізувати стан активних та неактивних контейнерів. За допомогою Docker Desktop можна легко перевірити логи контейнера, що надає цінну інформацію про його роботу та можливі помилки, здійснювати рестарт контейнерів або зупиняти їх роботу без необхідності використання командної строки [27]. Ці функції роблять Docker Desktop незамінним інструментом у роботі з Docker, спрощуючи рутинні завдання і підвищуючи ефективність робочих процесів. У даному випадку, контейнер було успішно створено, і база даних піднялася (рис. 3.4).

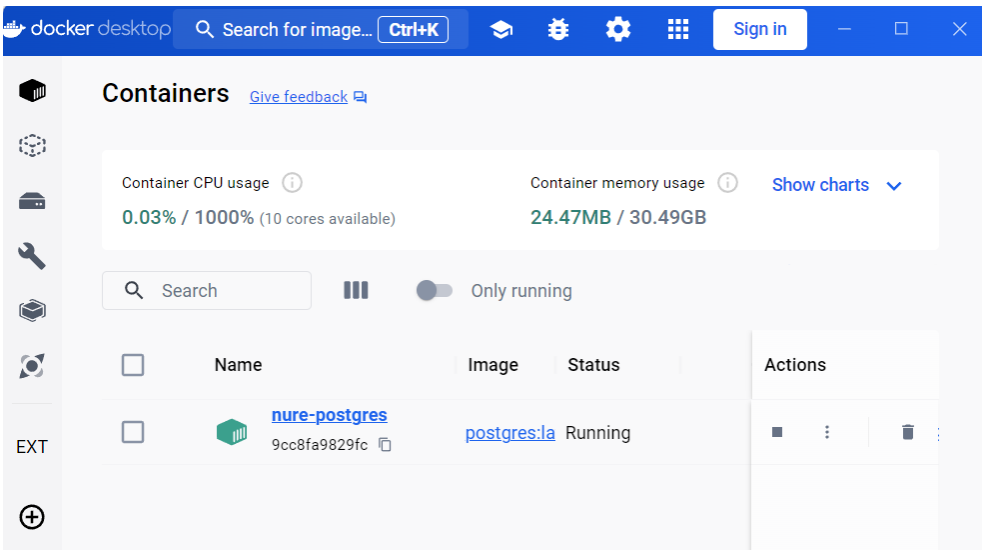


Рисунок 3.4 – Запущений контейнер для бази даних

3.2 Підготовка контенту

Контент для проєкту було взято з офіційного сайту кафедри (рис. 3.5) Інформатики Харківського Національного Університету Радіоелектроники.

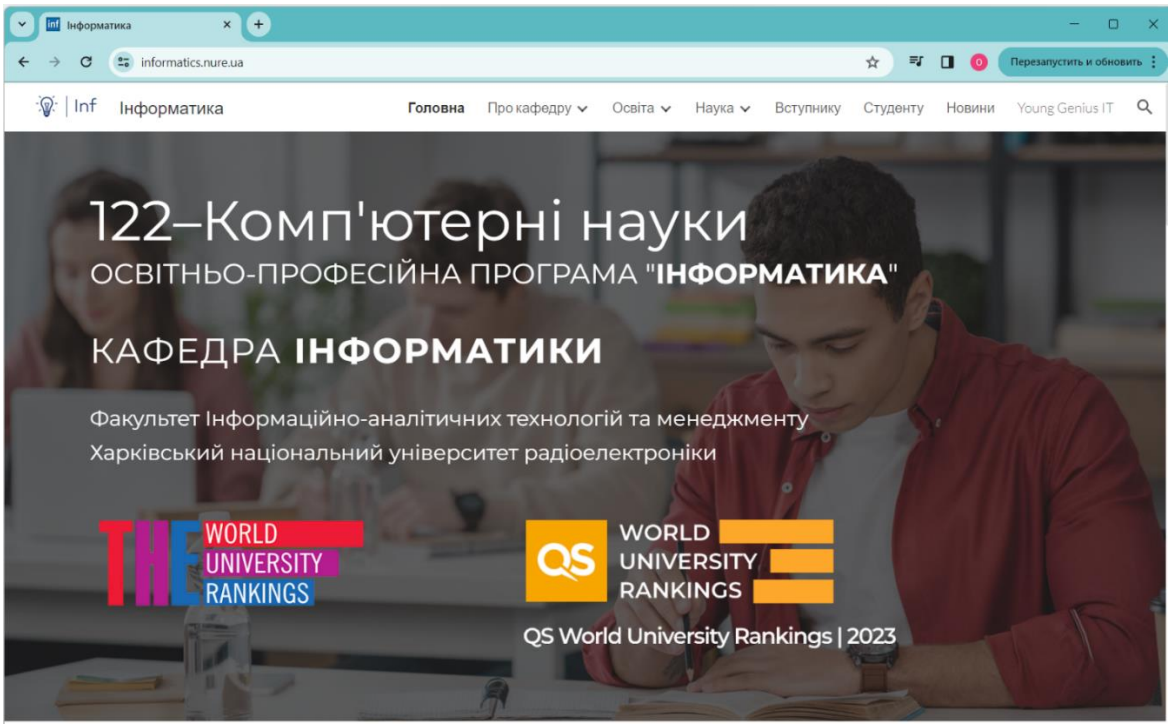


Рисунок 3.5 – Сайт кафедри Інформатики Харківського Національного Університету Радіоелектроники

Зокрема, були використані наступні сторінки вебсайту:

– «Про кафедру»;

– «Про кафедру» / «Колектив кафедри»;

– «Про кафедру» / «Міжнародна співпраця»;

– «Про кафедру» / «Компанії партнери»;

– «Наука» / «Напрями наукових досліджень»;

– «Наука» / «Науково-дослідна робота студентів»;

– «Наука» / «Навчально-наукова лабораторія»;

– «Вступнику».

Весь зібраний матеріал було переформатовано в плоский текст для легкості обробки та інтеграції у проєкт. Остаточний текстовий контент було збережено у вигляді документу, який в подальшому використовується в рамках проєкту для різноманітних цілей, включно з навчанням моделей, підготовкою матеріалів (рис. 3.6).

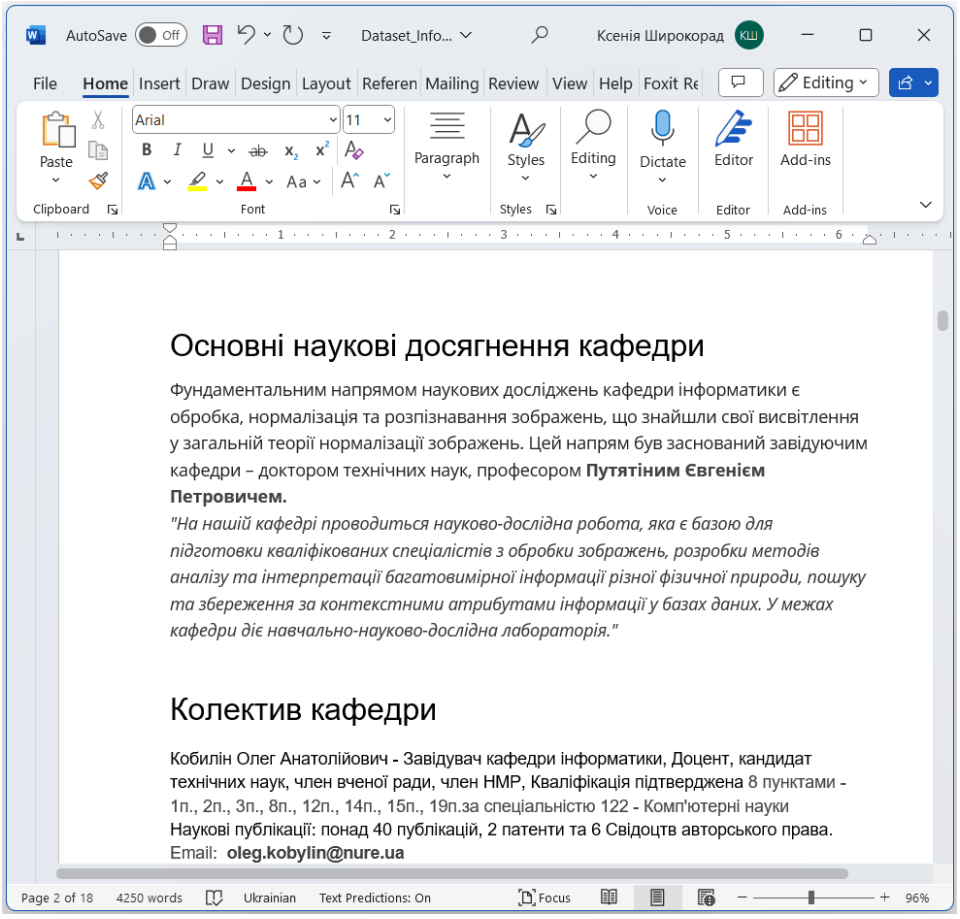


Рисунок 3.6 – Контент для проєкту

Під час підготовки контенту для використання у проєкті, особливу увагу слід приділити правильній структурації інформації. Це критично важливо для забезпечення ефективності інструментів обробки природної мови, таких як LangChain, які використовуються для створення ембедінгів документів.

Структурування інформації по темам дозволяє LangChain більш ефективно розбивати документ на ембедінги. Ембедінги є представленнями тексту у вигляді векторів, що використовуються для вимірювання семантичної близькості частин тексту та виконання різних завдань машинного навчання. Правильна структурація документу належним чином впливає на якість таких ембедінгів, а отже, і на результати їх подальшого аналізу і використання.

Для ефективної структурації інформації з сайту по темам були дотримані кількох ключових принципів:

– розділення контенту за категоріями (інформація має бути чітко поділена на логічні блоки відповідно до їх тематики);

– використання заголовків та підзаголовків (забезпечення чіткої структури документу через використання заголовків для кожної секції);

– консистентність у форматуванні (єдність у форматуванні тексту сприяє легшому розпізнаванню та обробці інформації);

– описовий змістовний текст (написання інформативного тексту з чіткими визначеннями для кращого розуміння контексту).

3.3 Створення промптів

В рамках роботи було розглянуто декілька варіантів інструкцій-промптів, кожен з яких мав на меті оптимізувати якість відповідей мовної моделі на запити абітурієнтів. Цей процес включав аналіз різних формулювань і структур промптів, а також їхнє тестування в реальних умовах для визначення, який з них найефективніше сприяє точності та релевантності відповідей.

Після ретельного вивчення та порівняння різних промптів було обрано один, який демонстрував найкращі результати (рис. 3.7). Цей промпт, деталі якого наведено на прикладному рисунку, відрізняється своєю здатністю до чіткого викладу запитань, що сприяє точнішому розумінню та обробці інформації мовною моделлю.

Для зручності використання та забезпечення стандартизації, обраний промпт зберігається у форматі документу. Це не тільки спрощує інтеграцію з різними системами та платформами, але й дозволяє здійснювати швидкі корективи або оновлення промпту без необхідності втручання в основний код програми. Використання такого формату забезпечує гнучкість та масштабованість системи, що є ключовим фактором для підтримки сучасних освітніх технологій.

Рисунок 3.7 – Промпт для звернення до чату GPT

3.4 Отримання доступу до OpenAI API

Для отримання доступу до моделей GPT через API OpenAI, розробникам необхідно володіти спеціальним ключем, який дозволяє інтегрувати та

використовувати моделі у власних застосунках. Ось алгоритм отримання доступу:

– реєстрація на сайті OpenAI (відвідайте офіційний сайт OpenAI та зареєструйтеся, вказавши необхідні особисті дані);

– отримання ключа (після реєстрації та верифікації акаунта, користувачі отримують унікальний API ключ, який використовується для аутентифікації запитів до сервісу (рис. 3.8));

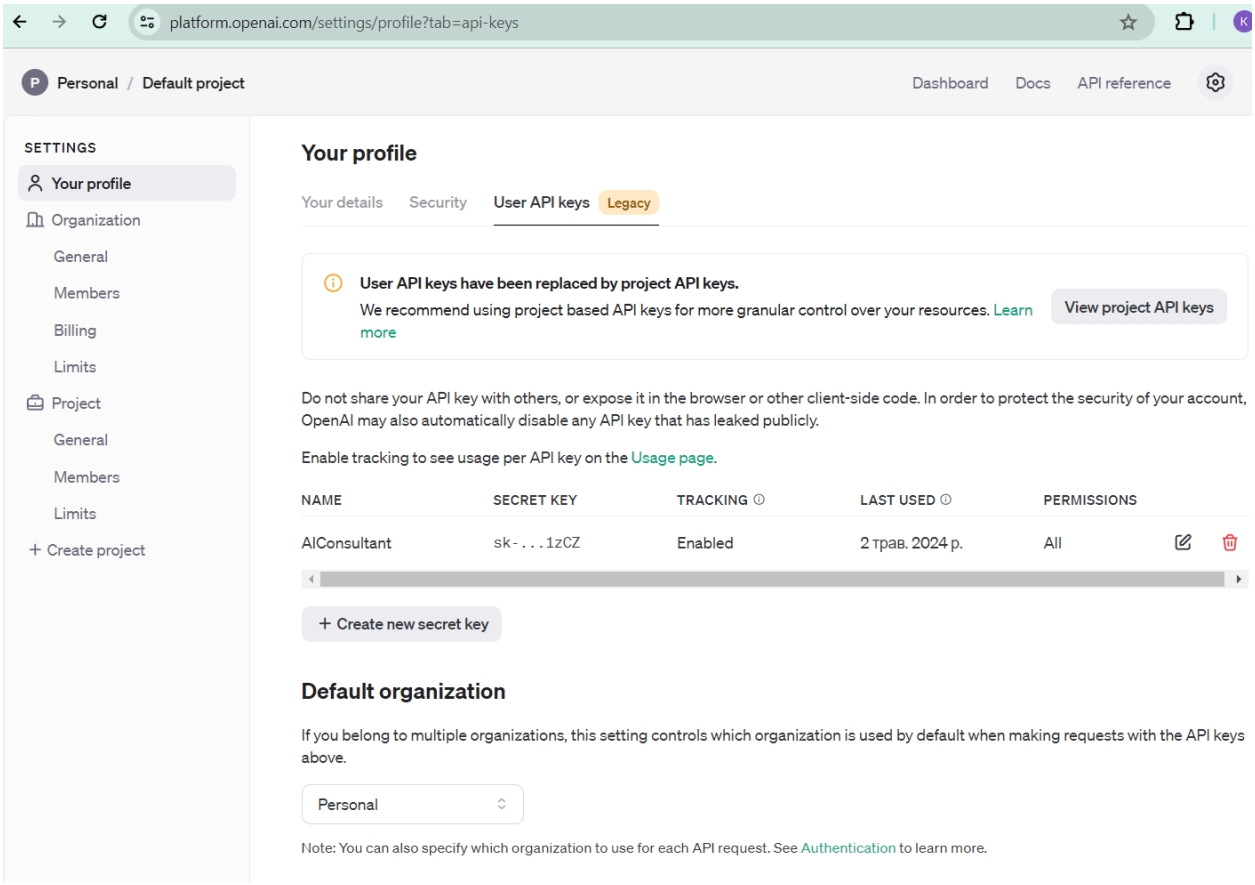


Рисунок 3.8 – Сторінка створення ключів доступу до OpenAI API

– поповнення кредитного балансу (залежно від обраного плану використання та обсягу запитів, може знадобитися поповнення кредитного балансу для покриття вартості використання API (рис. 3.9)).

Після завершення всіх кроків, отримані API ключі можна інтегрувати в проєкт та почати використовувати моделі GPT.

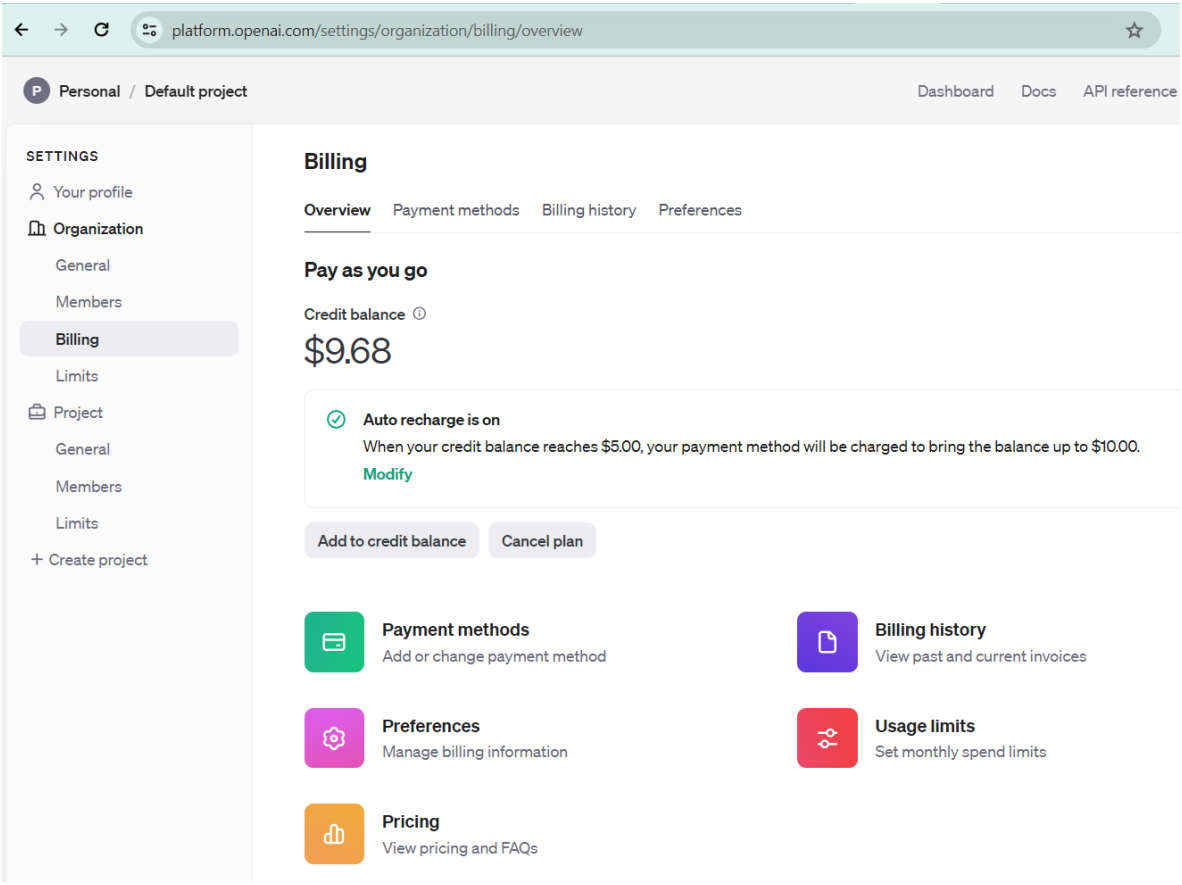


Рисунок 3.9 – Сторінка налаштування кредитного балансу аккаунта

3.5 Розробка функціоналу отримання відповідей на питання від GPT моделі на основі власного контенту

Для використання GPT моделей у проєкті необхідно правильно налаштувати її конфігурацію, забезпечуючи інтеграцію необхідних параметрів та ресурсів. Нижче наведено детальний опис конфігурації, використаної для розробки функціоналу отримання відповідей на питання від GPT моделі на основі власного контенту:

– конфігурація API ключа та моделі (openaiApiKey – ключ API, який використовується для аутентифікації запитів до OpenAI. modelName – назва моделі OpenAI, яка буде використовуватися для генерації відповідей);

– шлях до документу (docPath – вказує розташування документу на локальному ресурсі, який містить контент для відповідей);

– шаблон запиту (promptTemplate – шаблон запиту, що використовується для формування відповідей. Він містить інструкції, які допомагають моделі);

– конфігурація моделі мови («ChatLanguageModel» – створює екземпляр чат-моделі з заданими параметрами, такими як API ключ, назва моделі, температура генерації відповідей та час очікування відповіді);

– завантаження документу («Document» – завантажує документ із заданого шляху, який використовується для забезпечення контексту відповідей);

– модель ембедінгу («EmbeddingMode»l – використовується для перетворення тексту у векторні представлення, які допомагають моделі краще розуміти та обробляти запитання).

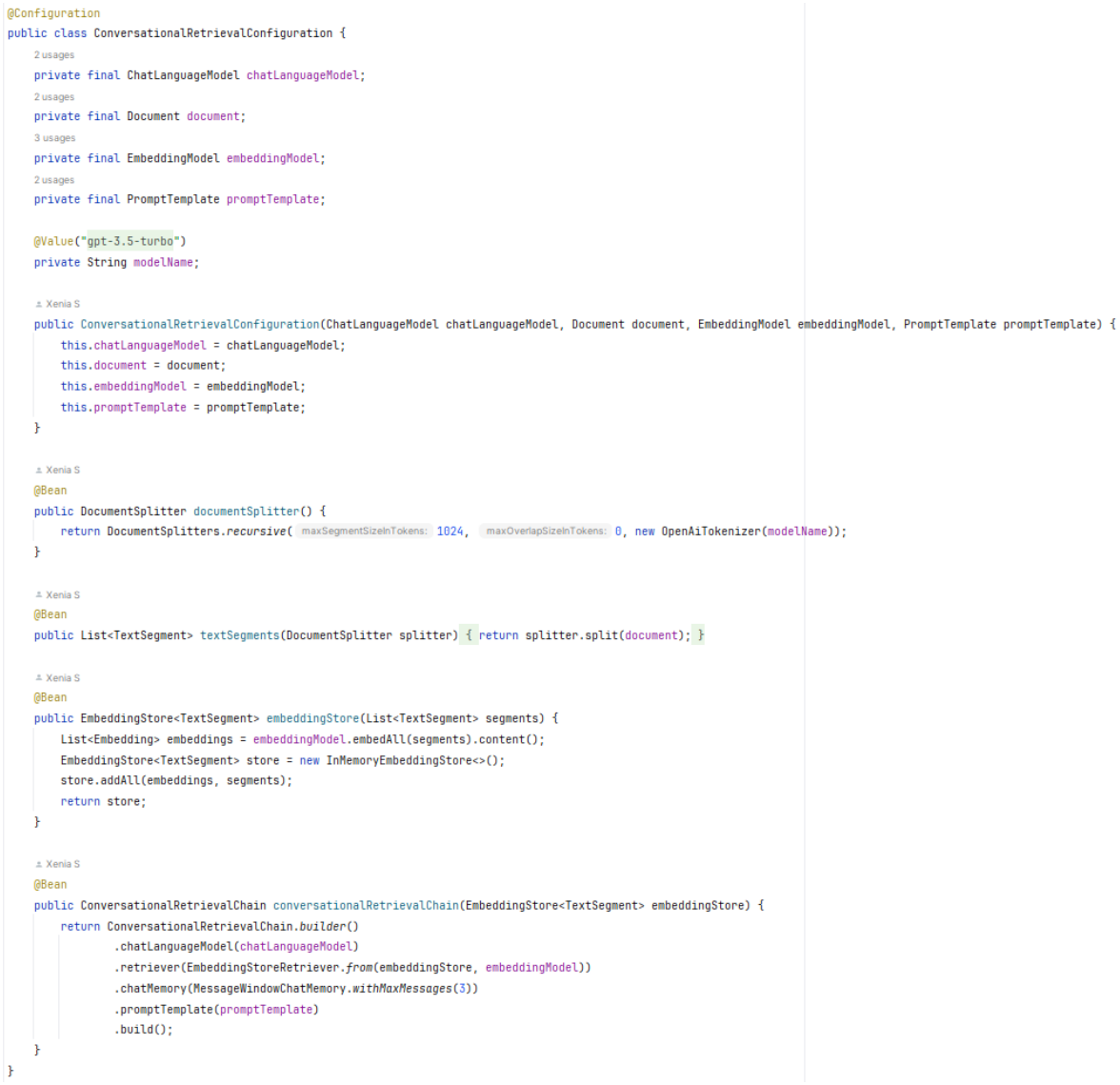
Зображення відображає конфігурацію для отримання релевантних частин документу та передачі їх до чату GPT разом з промптом (рис. 3.10). Клас «ConversationalRetrievalConfiguration» використовується для інтеграції різних компонентів, таких як модель мови, документи, модель вбудовування та шаблон запиту для ефективної роботи системи. Конфігурація забезпечує використання об'єктів, таких як «ChatLanguageModel», «Document», «EmbeddingModel» і «PromptTemplate», що використовуються для розбиття документів, створення текстових сегментів і зберігання ембедінгів, які потім використовуються для вилучення інформації на запити користувачів.

Конфігураційний файл містить методи, які визначають логіку розділення документів, вилучення текстових сегментів та створення ембедінгів для ефективної роботи з запитами (рис. 3.11). Зокрема, використовується рекурсивний метод розділення документів і методи для створення та зберігання ембедінгів, які оптимізують процес відповіді на питання на основі тексту. В коді також задіяно ланцюжок відповідей для обробки запитів із використанням шаблонів запитів і моделі чату з низькою температурою, що сприяє точності відповідей.

```java
@Configuration
public class OpenAiConfiguration {

    @Value("${API_KEY}")
    private String openaiApiKey;

    @Value("gpt-3.5-turbo")
    private String modelName;

    1 usage
    public static final String docPath = "src/main/resources/Dataset_Informatics_Department .docx";

    1 usage
    public static final String promptTemplate = """
            Ти консультант для абітурієнтів у чаті на сайті кафедри Інформатики Харківського Університету Радіоелектроніки,
            Дай відповідь на питання: {{question}}
            Документ з інформацією для відповіді клієнту: \n{{information}}
            Надай відповідь мовою, якою поставлено запитання.
            Якщо в документі з інформацією немає нічого по запитанню, пиши що не володієшь цією іформацією.
        """;

    Xenia S
    @Bean
    public ChatLanguageModel chatLanguageModel() {
        return OpenAiChatModel.builder()
                .apiKey(openaiApiKey)
                .modelName(modelName)
                .temperature(0.001)
                .timeout(Duration.ofSeconds(60))
                .build();
    }

    Xenia S
    @Bean
    public Document document() { return FileSystemDocumentLoader.loadDocument(docPath); }

    Xenia S
    @Bean
    public EmbeddingModel embeddingModel() { return new AllMiniLmL6V2QuantizedEmbeddingModel(); }

    Xenia S
    @Bean
    public PromptTemplate promptTemplate() { return PromptTemplate.from(promptTemplate); }
}
```

Рисунок 3.10 – Конфігурація чат-бота для OpenAI з Java

Останнім кроком у процесі конфігурації для обробки запитань є створення сервісного класу, який відповідає за виклик функціональності для отримання релевантних частин тесту на запитання та передачі їх моделі. Клас «LLMService» інкапсулює логіку взаємодії з об'єктом «ConversationalRetrievalChain», забезпечуючи чітку абстракцію та легкість використання (рис. 3.12).



```
@Configuration
public class ConversationalRetrievalConfiguration {
    2 usages
    private final ChatLanguageModel chatLanguageModel;
    2 usages
    private final Document document;
    3 usages
    private final EmbeddingModel embeddingModel;
    2 usages
    private final PromptTemplate promptTemplate;

    @Value("gpt-3.5-turbo")
    private String modelName;

    Xenia S
    public ConversationalRetrievalConfiguration(ChatLanguageModel chatLanguageModel, Document document, EmbeddingModel embeddingModel, PromptTemplate promptTemplate) {
        this.chatLanguageModel = chatLanguageModel;
        this.document = document;
        this.embeddingModel = embeddingModel;
        this.promptTemplate = promptTemplate;
    }

    Xenia S
    @Bean
    public DocumentSplitter documentSplitter() {
        return DocumentSplitters.recursive( maxSegmentSizeInTokens: 1024,  maxOverlapSizeInTokens: 0, new OpenAiTokenizer(modelName));
    }

    Xenia S
    @Bean
    public List<TextSegment> textSegments(DocumentSplitter splitter) { return splitter.split(document); }

    Xenia S
    @Bean
    public EmbeddingStore<TextSegment> embeddingStore(List<TextSegment> segments) {
        List<Embedding> embeddings = embeddingModel.embedAll(segments).content();
        EmbeddingStore<TextSegment> store = new InMemoryEmbeddingStore<>();
        store.addAll(embeddings, segments);
        return store;
    }

    Xenia S
    @Bean
    public ConversationalRetrievalChain conversationalRetrievalChain(EmbeddingStore<TextSegment> embeddingStore) {
        return ConversationalRetrievalChain.builder()
                .chatLanguageModel(chatLanguageModel)
                .retriever(EmbeddingStoreRetriever.from(embeddingStore, embeddingModel))
                .chatMemory(MessageWindowChatMemory.withMaxMessages(3))
                .promptTemplate(promptTemplate)
                .build();
    }
}
```

Рисунок 3.11 – Конфігурація класу для отримання релевантних частин тесту на запитання

Клас «LLMService» містить метод «getAnswer(String question)», який виконує ключову функцію: приймає вхідне запитання та ініціює процес вилучення відповідних відомостей. Викликаючи метод «execute()» ланцюжка операцій «ConversationalRetrievalChain», він активує процедуру пошуку необхідних текстових сегментів із датасету, які найбільш точно відповідають запитанню.

У результаті, «LLMService» виявляється не тільки мостом між користувачем та складними алгоритмами обробки даних, але й забезпечує гладке та ефективне взаємодію з системою, роблячи процес відповіді на

запитання швидким і точним. Така структура сервісу підвищує загальну доступність та зручність використання технології для різних кінцевих користувачів.

```
@Service
public class LLMService {

    private final ConversationalRetrievalChain conversationalRetrievalChain;

    Xenia S
    public LLMService(ConversationalRetrievalChain conversationalRetrievalChain) {
        this.conversationalRetrievalChain = conversationalRetrievalChain;
    }

    Xenia S *
    public String getAnswer(String question) { return conversationalRetrievalChain.execute(question); }

}
```

Рисунок 3.12 – Сервіс для інкапсуляції логіки отримання відповіді на основі власного контенту

3.6 Створення Telegram-бота

Для розробки та налаштування Telegram-бота, який використовується в рамках проєкту для автоматизації взаємодії з користувачами та надання інформаційних послуг, було реалізовано наступні етапи.

Реєстрація Телеграм бота. Цей процес є першим кроком для створення та використання бота в Telegram і включає наступні етапи:

– взаємодія з BotFather (BotFather – це офіційний бот від Telegram, який дозволяє створювати нові боти. BotFather'у потрібно надіслати команду «/newbot» (рис. 3.13));

– назва бота (BotFather запитає вас про назву бота, яка буде відображатися в діалогах та контакт-лісті користувачів);

– ім'я користувача бота (після назви потрібно буде вказати унікальне ім'я користувача для вашого бота, яке має закінчуватися на «bot» (наприклад, NureConsultantBot));

– отримання API токена (після вдалої реєстрації BotFather надасть токен API, який є ключем для програмного доступу до вашого бота. Цей токен є важливим і повинен бути збережений у безпеці (рис. 3.14)).

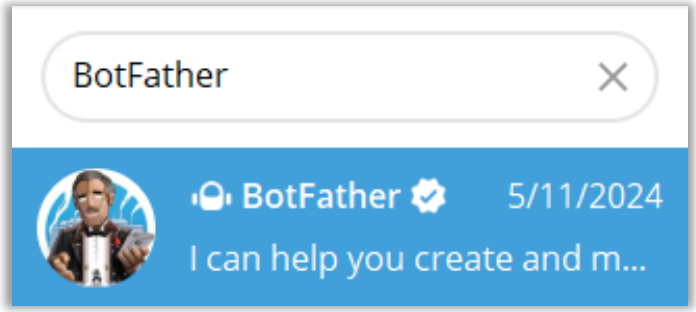


Рисунок 3.13 – Пошук BotFather у Телеграм

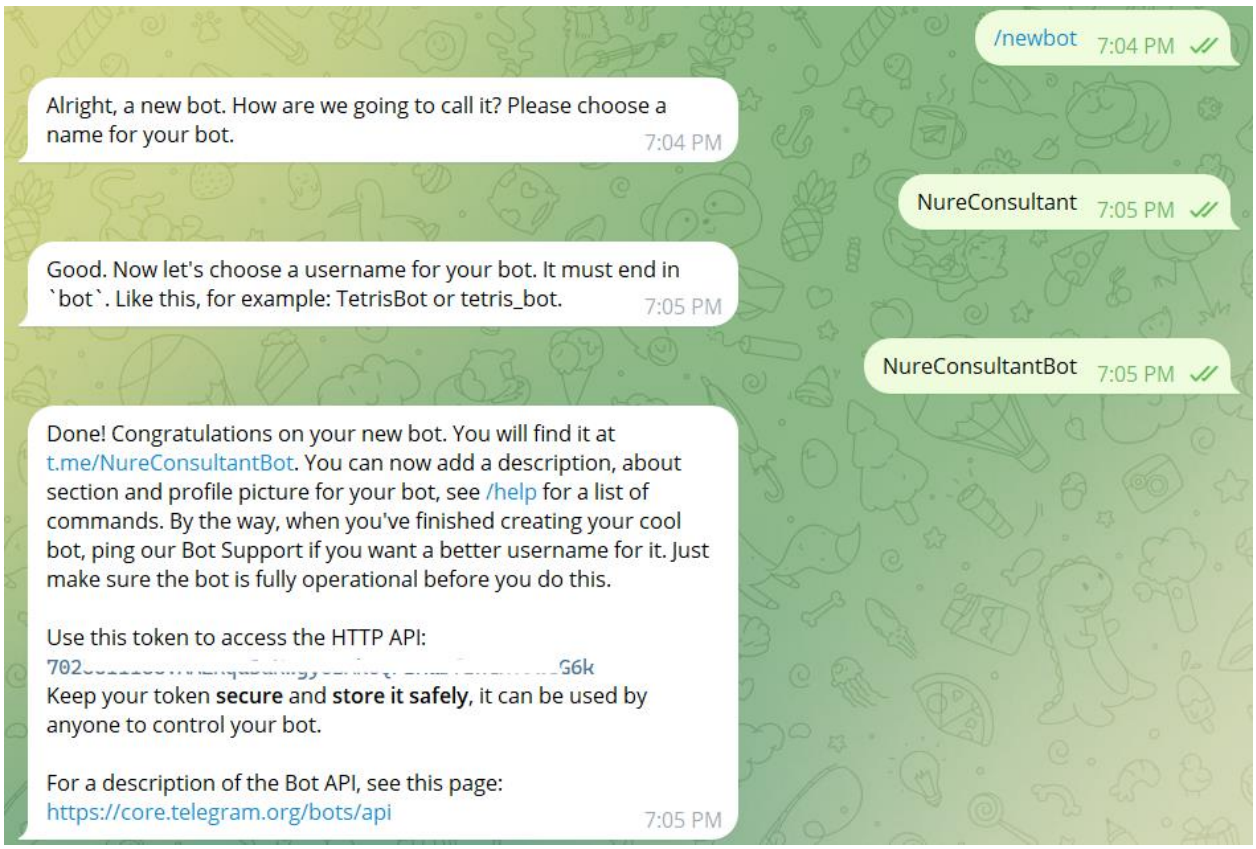


Рисунок 3.14 – Приклад реєстрації бота

Розробка логіки бота:

– аналіз запитів користувачів (розуміння та інтерпретація вхідних повідомлень для визначення потрібної реакції);

– відповідь на запити (генерація релевантних та корисних відповідей на питання або команди користувачів);

– інтеграція зі сторонніми API (залучення додаткових сервісів для розширення функціональності бота, наприклад, для отримання погоди, новин тощо);

– зберігання стану сесії (використання баз даних або інших методів збереження даних для підтримки контекстуалізованих діалогів і персоналізації).

З'єднання з API Telegram:

– реєстрація бота (створення нового бота за допомогою BotFather у Telegram та отримання токена API);

– настройка бота (конфігурація бота для використання токена для аутентифікації та викликів API);

– використання API (реалізація функціональності, що вимагає взаємодії з Telegram, наприклад, відправлення повідомлень або організація чатів).

Налаштування вебхука:

– створення вебхука (конфігурація кінцевої точки сервера, куди Telegram буде надсилати оновлення);

– реєстрація вебхука з Telegram (налаштування бота на використання вебхука шляхом відправлення HTTPS запиту до API Telegram з URL вашого сервера);

– обробка вхідних повідомлень (налаштування сервера для прийому та обробки оновлень від Telegram).

Обробка повідомлень:

– парсинг повідомлень (аналіз вхідних даних для визначення запитів користувачів);

– процесинг логіки відповідей (використання розробленої логіки для формування відповіді на кожен запит);

– підтримка стану діалогу (використання інформації з попередніх взаємодій для забезпечення послідовності в комунікації).

Відправка відповідей:

– формування відповідей (створення текстових, медійних або інших типів відповідей відповідно до запиту);

– забезпечення надійності комунікації (перевірка того, що повідомлення доставлені та коректно відображаються користувачам);

– використання API для відправки (відправлення готових відповідей через API Telegram для гарантії швидкого та ефективного спілкування з користувачами).

3.7 Ілюстрація роботи

Запуск бота ініціюється командою «/start», яка вводиться у чаті. Ця команда активує обмін повідомленнями між користувачем та ботом і часто використовується для першого знайомства з функціоналом бота (рис. 3.15).

Рисунок 3.15 – Запуск бота

На зображеннях 3.16-3.18, 3.20 продемонстровано роботу бота, який відповідає на різноманітні запитання абітурієнтів, деякі на іноземній мові. Це підкреслює здатність бота ефективно обробляти та реагувати на запитання різного типу. На зображеннях 3.19 та 3.21 показано, які релевантні фрагменти документу були використані для формування відповідей на окремі запитання.

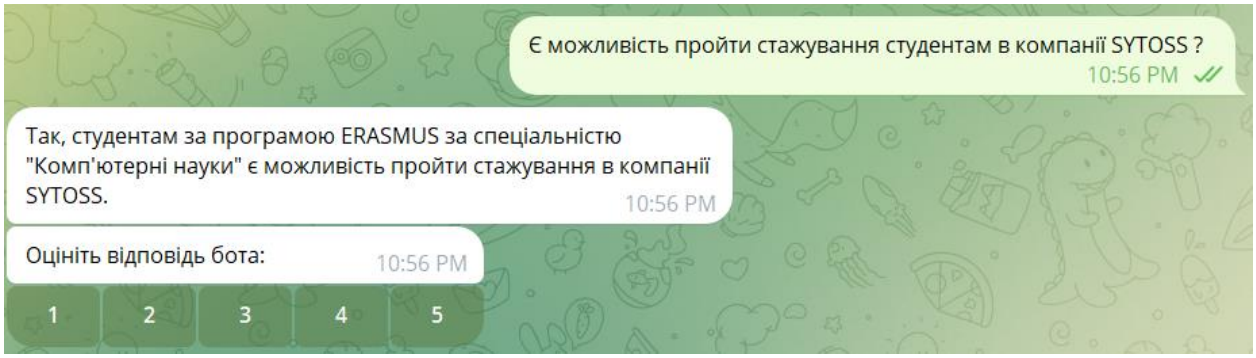


Рисунок 3.16 – Приклад питання до бота

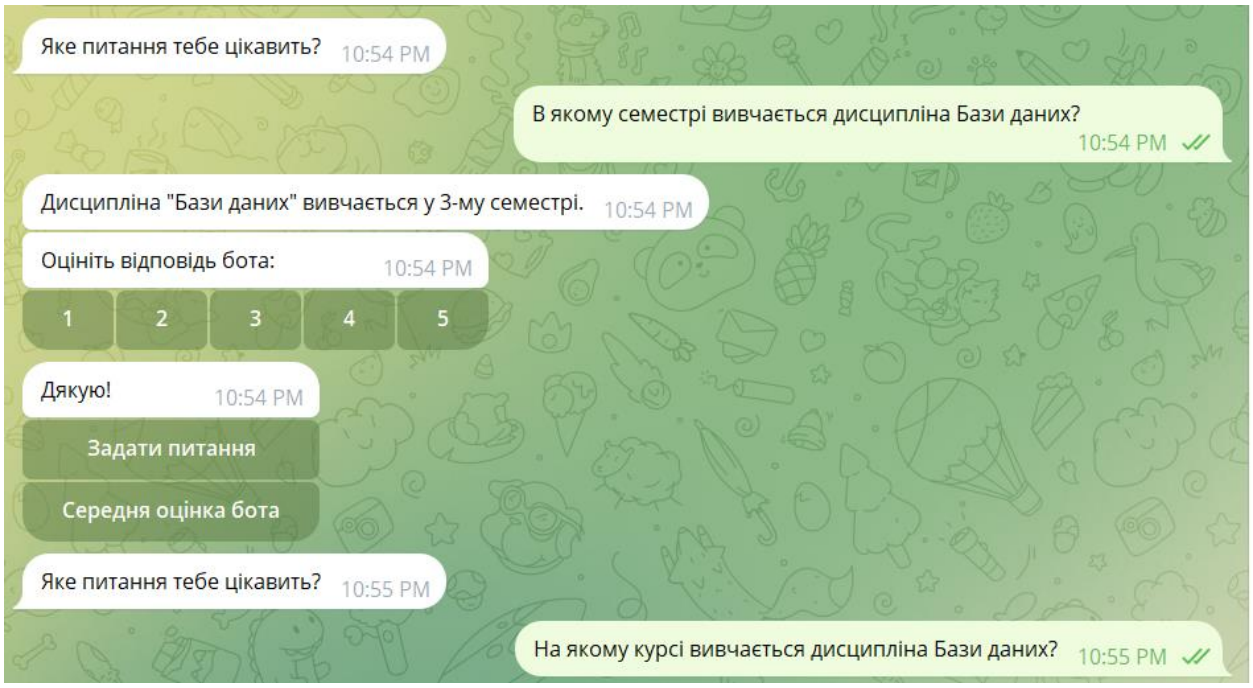


Рисунок 3.17 – Приклад питання до бота

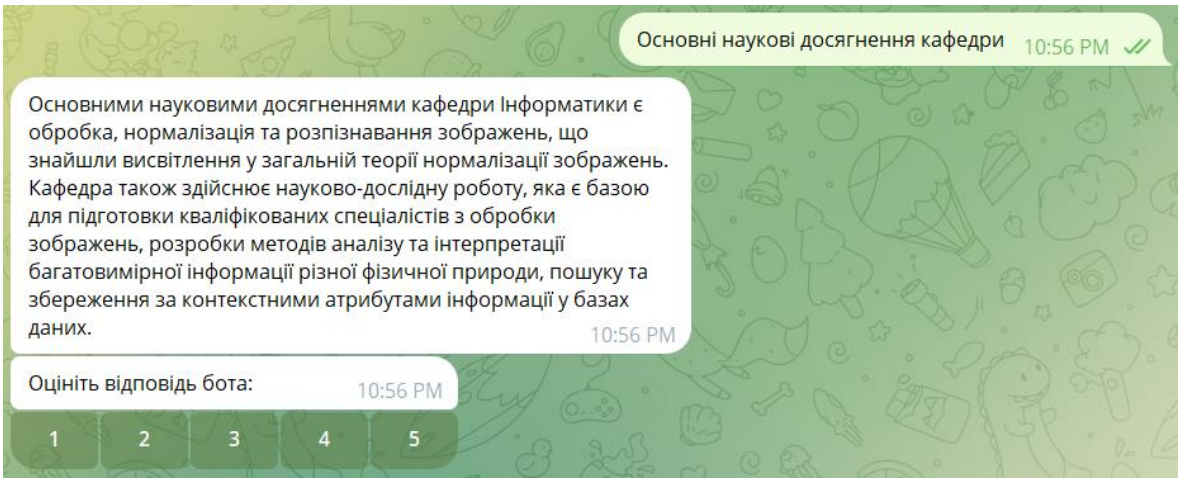


Рисунок 3.18 – Приклад питання до бота

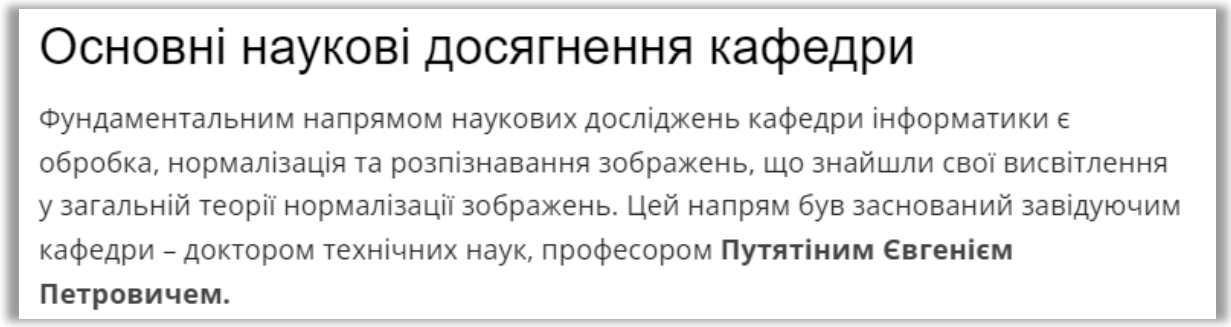


Рисунок 3.19 – Відповідний фрагмент з контексту

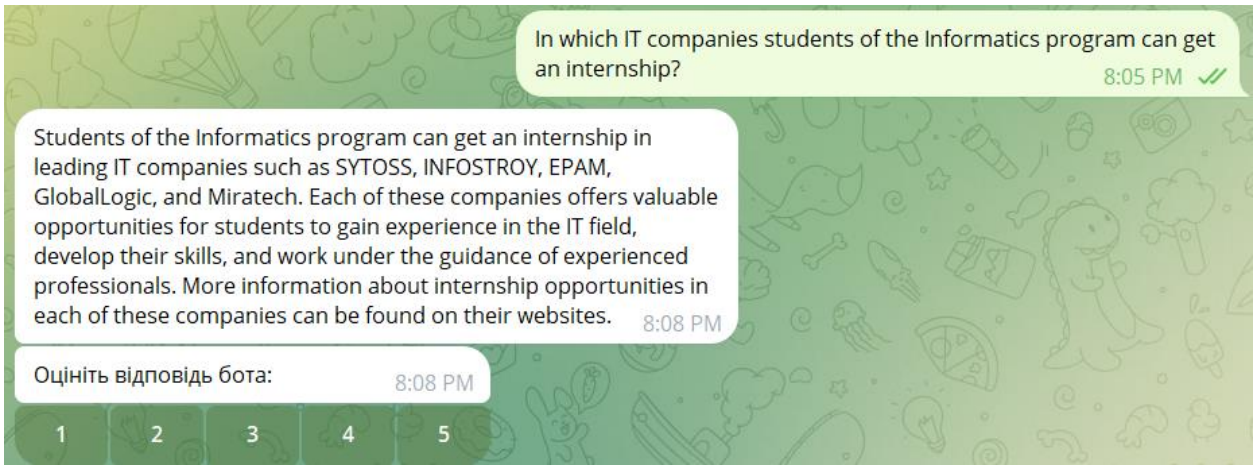


Рисунок 3.20 – Приклад питання до бота

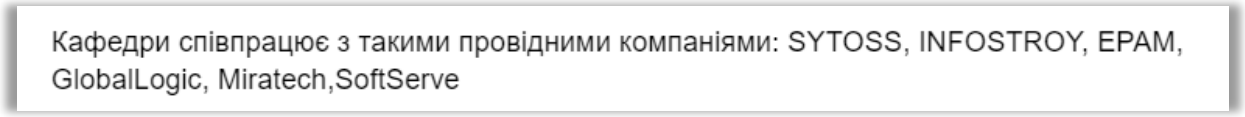


Рисунок 3.21 – Відповідний фрагмент з контексту

3.8 Експертна оцінка Telegram-бота

Для оцінки ефективності моделі у відповідях на запитання було проведено тестування за допомогою експертного підходу. Тестування включало оцінювання відповідності відповідей моделі з використанням 5-

бальної шкали, де 1 бал означає «зовсім не відповідає», а 5 балів – «ідеально відповідає запитанню».

Тестувальниками були вибрані студенти Харківського національного університету радіоелектроніки та працівники кафедри Інформатики. Ці групи мали змогу оцінити якість і релевантність відповідей моделі на основі їх фахового досвіду і знань про університет та його спеціальності.

Середній бал, отриманий від усіх тестувальників, склав 3,4.

Низькі оцінки зазвичай були зумовлені:

– ставленням питань, не пов'язаних з університетом: «Яка сьогодні погода?», «Чи продають в будівлі університету морозиво?»;

– недостатністю інформації в датасеті.

Це вказує на необхідність подальшого розширення контексту, включення більш широкого спектра інформації та форматування тексту у документі для покращення якості відповідей на різноманітні запитання. Такі заходи допоможуть підвищити загальну ефективність системи та забезпечити більш точне і корисне використання моделі у майбутньому.

У майбутньому реалізація комп'ютерного зору для автоматичного вилучення інформації з відеоматеріалів може стати значним кроком удосконалення датасетів та покращення функціональності інформаційних систем. Застосування технологій комп'ютерного зору дозволить автоматизувати процеси збору даних із відео, що може включати виявлення та розпізнавання тексту, об'єктів або дій, що відбуваються у відео, та їх подальшу обробку [28-33].

Також в результаті тестування було виявлено, що деякі часто задавані питання, які стосуються спеціальності Інформатика, не були включені в датасет, що використовувався для тренування чат-бота. Серед таких питань були:

– «Які мови програмування я буду знати?»;

– «Яка email кафедри?».

Ці питання, очевидно, є важливими для потенційних студентів, але відсутність відповідей на них у первісному контексті призводила до зниження загальної оцінки корисності бота.

Інформація для відповідей на ці питання була додана до контексту чат-бота. Це покращення виявилося важливим кроком у оптимізації релевантності відповідей. Додавання контексту дозволило моделі краще розуміти та відповідати на запити користувачів, забезпечуючи більш точні та детальні відповіді.

При повторному тестуванні з вдосконаленим контекстом, середня оцінка бота зросла до 4,1 з 5 можливих. Це свідчить про значне покращення якості відповідей та загальної корисності системи. Раніше середня оцінка бота була значно нижчою, що вказувало на необхідність додаткового навчання та налаштування.

Зростання середньої оцінки демонструє не лише покращення якості окремих відповідей, але й загальну підвищену задоволеність користувачів. Завдяки інтеграції нових даних та покращенню контексту, бот став більш адаптованим до потреб абітурієнтів, надаючи їм більш корисну та актуальну інформацію.

ВИСНОВКИ

У рамках кваліфікаційної роботи було вивчено можливості застосування мовних моделей GPT від OpenAI для генерації відповідей на запитання на основі спеціально підготовленого контенту. Основна увага була приділена консультуванню абітурієнтів, що робить дослідження особливо актуальним у контексті покращення інформаційної підтримки майбутніх студентів. Для реалізації дослідницьких завдань було створено власний датасет, який включав інформацію про умови вступу, освітні програми та академічне середовище, а також було досліджено різні налаштування промптів. Важливою частиною проєкту стала інтеграція бібліотеки LangChain, що дозволила здійснювати пошук найбільш релевантних фрагментів датасету для відповідей.

Розробка телеграм-бота для консультування абітурієнтів дозволила провести реальне тестування і отримати експертну оцінку якості відповідей, що виявило деякі важливі аспекти для подальшого розвитку. Отримані результати показали, що хоча бот давав задовільні відповіді, існувала потреба в подальшому розширенні та доповненні датасету, особливо в частині додавання недостатньо представленої інформації та оптимізації контенту.

Окрема увага в дослідженні приділялася сумаризації інформації в датасеті, що має включати агрегацію даних за тематичними блоками та подальше розділення узагальненої інформації на логічно відділені частини. Також було розглянуто можливість інтеграції відеоматеріалів шляхом конвертації аудіо в текст, що може розширити джерела даних для датасету і збагатити контекст доступною інформацією.

Робота підтвердила значний потенціал використання мовних моделей GPT для автоматизації процесу відповідей на запитання, зокрема в академічному середовищі. Застосування такого підходу в інших сферах дозволить ефективно використовувати ресурси, зменшуючи потребу в прямій участі співробітників, і таким чином заощаджуючи час та кошти.

Під час тестування було виявлено, що бот добре обробляє основні запитання абітурієнтів, такі як інформація про дисципліни та опис спеціальності. Проте, виникали складнощі з наданням детальної інформації щодо специфічних програм та додаткових можливостей. Це підкреслило необхідність більш детального наповнення датасету та регулярного оновлення інформації, щоб забезпечити її актуальність та повноту. Постійне оновлення та вдосконалення датасету є критично важливим для підтримки високої точності та актуальності відповідей. Ретельний моніторинг змін у відповідних галузях знань та регулярна інтеграція нових даних забезпечують, що нейроконсультант залишається ефективним інструментом для користувачів, пропонуючи їм найсучаснішу та найточнішу інформацію.

Результати роботи апробовано у вигляді 2 тез доповідей під час Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [34], XV-ої Міжнародної науково-практичної конференції FOSS-2024 «Free and open source software» [35].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Yakovleva O., Nebeský L., Kirichenko A. (2023). Using the GPT models for responses based on custom content to develop neural consultant for university applicants. Abstracts of V International Scientific and Practical Conference «The world of modern technologies and inventions» Madrid, Spain. Pp. 172-178. URL: https://eu-conf.com/ua/events/trends-in-science-regarding-the-creation-of-new-teaching-methods/

2. Cherednichenko, Olga, Ivashchenko, Oksana, Cibák, Ľuboš and Lincenyi, Marcel. «Item Matching Model in E-Commerce: How Users Benefit» Economics and Culture, vol.20, no.1, (2023), pp.77-90. https://doi.org/10.2478/jec-2023-0007

3. Cherednichenko, O.; Ivashchenko, O.; Lincényi, M.; Kováč, M. (2023). Information technology for intellectual analysis of item descriptions in e-commerce, Entrepreneurship and Sustainability Issues 11(1): 178-190. https://doi.org/10.9770/jesi.2023.11.1(10)

4. Обробка природної мови. URL: https://metinvest.digital/ua/page/1052 (дата звернення 28.04.2024).

5. Розвиток LLM протягом останніх декількох років. URL: https://www.it.ua/knowledge-base/technology-innovation/large-language-models-llms (дата звернення 28.04.2024).

6. Аналіз LLM від різних виробників. URL: https://mindsdb.com/blog/navigating-the-llm-landscape-a-comparative-analysis-of-leading-large-language-models (дата звернення 28.04.2024).

7. GPT від OpenAI. URL: https://platform.openai.com/docs/models/gpt-4o (дата звернення 29.04.2024).

8. LaMDA або Language Model for Dialogue Applications. URL: https://www.epravda.com.ua/news/2022/06/13/688095/ (дата звернення 29.04.2024).

9. Модель Sydney. URL: https://envision.microsoft.com/en-US/sydney (дата звернення 29.04.2024).

10. LLM Pricing. URL: https://llm-price.com/ (дата звернення 29.04.2024).

11. Cherednichenko, O., Yanholenko, O., Iakovleva, O., & Kustov, O. (2014). Models of Research Activity Measurement: Web-based monitoring implementation. Information Systems: Education, Applications, Research, 75–87. https://doi.org/10.1007/978-3-319-11373-9_7

12. Cherednichenko, O., Yanholenko, O., Liutenko, I., & Iakovleva, O. (2013). Monitoring and Evaluation Problems in Higer Education-Comprehensive Assessment Framework Development. In CSEDU (pp. 455-460).

13. Yanholenko, O., Cherednichenko, O., Yakovleva, O., & Arkatov, D. (2020). A Model for Estimating the Security Level of Mobile Applications: a Fuzzy Logic Approach. In IntelITSIS (pp. 252-266).

14. Klyuchka, Y. A., Cherednichenko, O. Y., Vasylenko, A. V., & Yakovleva, O. V. (2017). Forecasting the results of football matches on the internet based information. Bulletin of National Technical University «KhPI». Series: System Analysis, Control and Information Technologies, 0(55), 51–59. https://doi.org/10.20998/2079-0023.2017.55.09

15. Telegram LLM based bot. URL: https://diverger.medium.com/whatsapp-telegram-llm-based-bot-ac73c6e48b3e (дата звернення 30.04.2024).

16. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, IEEE Access, vol. 12, pp. 73376-73385.

17. Reasons Why You May Need a Custom-Trained ChatGPT AI Chatbot. URL: https://simplified.com/blog/ai-writing/how-to-train-chatgpt (дата звернення 30.04.2024).

18. Донавчання на основі передачі власного контенту. URL: https://medium.com/@amiraryani/a-brief-introduction-to-retrieval-augmented-generation-rag-4bd6e50da532 (дата звернення 30.04.2024).

19. Ролі для звернення до GPT-моделі. URL: https://egghead.io/lessons/node-js-training-gpt-using-roles-message-examples-for-better-chat-completions (дата звернення 01.05.2024).

20. Обмеження за токенами у моделях GPT. URL: https://platform.openai.com/docs/guides/rate-limits (дата звернення 01.05.2024).

21. Cherednichenko, O., Kanishcheva, O., Yakovleva, O., & Arkatov, D. (2020). Collection and Processing of a Medical Corpus in Ukrainian. corpus, 2(4), 7-14.

22. Cherednichenko, O., Vovk, M., Yanholenko, O., & Yakovleva, O. (2020). Towards the Technology of Employers' Requirements Collection Development. In Integrated Computer Technologies in Mechanical Engineering (pp. 228-239). Springer, Cham.

23. Пошук релевантних фрагментів у тексті використовуючи LangChain. URL: https://www.langchain.com/ (дата звернення 01.05.2024).

24. Model Parameters in OpenAI API. URL: https://medium.com/nerd-for-tech/model-parameters-in-openai-api-161a5b1f8129 (дата звернення 01.05.2024).

25. How to create a chatbot in Telegram. URL: https://sendpulse.com/knowledge-base/chatbot/telegram/create-telegram-chatbot (дата звернення 01.05.2024).

26. AI Chatbot for Custom PDF Documents with Langchain. URL: https://medium.com/@param775/build-an-ai-chatbot-for-custom-pdf-documents-with-python-and-langchain-4089fe30b30f (дата звернення 02.05.2024).

27. How to Deal With Databases in Docker. URL: https://www.baeldung.com/ops/docker-databases (дата звернення 02.05.2024).

28. А.Р. Ковтуненко, О.В. Яковлева, В.А. Любченко, & О.В. Янголенко (2020) Дослідження сумісного використання математичної морфології та згорткових нейронних мереж для вирішення задачі розпізнавання цінників. Вісник Національного технічного університету ХПІ (3). 24-31.

29. Yakovleva O., Nebeský L, Liakhov P. (2023). Research methods of texture image analysis to solve the texture search problem. Proceedings of the IV

International Scientific and Practical Conference «The world of modern technologies and inventions». Vienna, Austria. (2023) pp. 252-261 URL: https://isg-konf.com/the-world-of-modern-technologies-and-inventions/

30. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, Indonesian Journal of Electrical Engineering and Computer Science, vol. 33, no. 1, pp. 113-125. DOI: 10.11591/ijeecs.v33.i1.pp113-125.

31. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. CEUR Workshop Proceedings Vol. 3403. pp. 69-86. ISSN 1613-0073 https://ceur-ws.org/Vol-3403/paper6.pdf

32. Yakovleva, O., Kovač, M., Ardasov, V. & Yeremenko, I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities. Public Administration and Regional Development, 19(1), pp. 158–183.

33. Yakovleva, O., & Nikolaieva, K. (2020). Research Of Descriptor Based Image Normalization And Comparative Analysis Of SURF, SIFT, BRISK, ORB, KAZE, AKAZE Descriptors. Advanced Information Systems, 4(4), 89-101. doi:10.20998/2522-9052.2020.4.13

34. Широкорад К. А. (2024). 28-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті» , Квітень 16, Харків, Україна, с. 143-145.

35. Широкорад К.А., Яковлева О.В. (2024). XV-а Міжнародна науково-практична конференція «Free and open source software», Лютий 13-14, Харків, Україна, с. 103-104.