

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту

(повна назва)

Кафедра прикладної математики

(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка системи AI Assistant

з можливістю структурованого виведення інформації

(тема)

Виконав:

здобувач 2 року навчання, групи САУМ-23-1

Радигін А.Є.

(прізвище, ініціали)

Спеціальність 124 Системний аналіз

(код і повна назва спеціальності)

Тип програми освітньо-професійна

(освітньо-професійна або освітньо-наукова)

Освітня програма Системний аналіз і управління

(повна назва освітньої програми)

Керівник проф. Сидоров М.В.

(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ

(підпис)

Сидоров М.В.

(прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту

Кафедра прикладної математики

Рівень вищої освіти другий (магістерський)

Спеціальність 124 Системний аналіз

(код і повна назва)

Тип програми освітньо-професійна

(освітньо-професійна або освітньо-наукова)

Освітня програма Системний аналіз і управління

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____

(підпис)

“ 25 ” листопада 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Радигіну Андрію Євгеновичу

(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи AI Assistant з можливістю структурованого виведення інформації

затверджена наказом по університету від 22 листопада 2024 р. № 1228 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 6 січня 2025 р.

3. Вихідні дані до роботи Система AI Assistant з можливістю структурованого виведення інформації

4. Перелік питань, що потрібно опрацювати в роботі _____

1. Системний аналіз предметної області

2. Вибір і обґрунтування методу розв'язання

3. Програмна реалізація

4. Результати обчислювального експерименту

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій _____

1. Актуальність теми роботи _____

2. Постановка задачі _____

3. Системний аналіз предметної області _____

4. Метод чисельного аналізу _____

5. Результати обчислювального експерименту _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	25 листопада – 1 грудня 2024 р.	виконано
2	Вибір та обґрунтування методу	2 – 8 грудня 2024 р.	виконано
3	Розробка алгоритму і програми	9 – 22 грудня 2023 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	23 – 29 грудня 2024 р.	виконано
5	Робота над текстом пояснювальної записки	30 грудня 2024 р. – 9 січня 2025 р.	виконано
6	Представлення роботи на рецензію в ЕК	10 січня 2025 р.	виконано

Дата видачі завдання 25 листопада 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Сидоров М.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 58 с., 1 табл., 5 рис., 2 дод., 15 джерел.

АНАЛІЗ ДАНИХ, ВЕЛИКА МОВНА МОДЕЛЬ, ЕМБЕДДИНГИ, ІНТЕГРАЦІЯ ДАНИХ, МАШИННЕ НАВЧАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, ОПТИМІЗАЦІЯ АЛГОРИТМІВ, ПОШУКОВА ДОПОВНЕНА СИСТЕМА, СТРУКТУРОВАНІЙ ВИВІД, ФІНАНСОВИЙ АНАЛІЗ, ШІ-АСИСТЕНТ.

Об'єкт дослідження – процес надання системами штучного інтелекту структурованої інформації та керування програмами на основі неформальних запитів користувачів у фінансовій сфері.

Мета роботи – розробка фінансово орієнтованого AI-асистента з можливістю структурованого виведення інформації та керування програмами на основі неформальних запитів користувачів.

Методи дослідження – у роботі використовуються методи системного аналізу, обробки природної мови, машинного навчання та програмування. Застосовуються алгоритми для інтерпретації неформальних запитів, перетворення їх у формальні команди, інтеграції з фінансовими програмами та сервісами, а також оцінки ефективності розробленої системи за допомогою показників точності та релевантності наданої інформації.

У кваліфікаційній роботі розроблено та впроваджено систему підтримки прийняття рішень у фінансовій сфері з використанням штучного інтелекту. Проведено детальний аналіз предметної області, досліджено сучасні підходи до обробки природної мови та автоматизації прийняття рішень. У межах роботи виконано розробку архітектури системи AI-асистента, що дозволяє обробляти складні запити користувачів, виконувати пошук інформації в реальному часі, аналізувати фінансові дані та генерувати структуровані відповіді.

Особливу увагу приділено інтеграції великих мовних моделей та механізмів структурованого виведення інформації. Розроблена система забезпечує

адаптивність до нових типів запитів, мінімізацію часу обробки та можливість розширення функціональності. Здійснено програмну реалізацію системи, протестовано її на реальних сценаріях використання та підтверджено ефективність роботи. Система здатна інтегруватися з іншими програмними продуктами та використовуватися в різних галузях.

Результати цієї роботи можуть бути застосовані для створення аналогічних рішень у сферах аналізу ринку, автоматизації бізнес-процесів, обслуговування клієнтів та інших областях, де потрібна швидка та точна обробка великих обсягів даних. Система має широкий потенціал для подальшого розвитку та вдосконалення.

ABSTRACT

Introductory note: 58 pages, 1 table, 5 figures, 2 appendixes, 15 sources.

AI ASSISTANT, ALGORITHM OPTIMIZATION, DATA ANALYSIS, DATA INTEGRATION, EMBEDDINGS, FINANCIAL ANALYSIS, LARGE LANGUAGE MODEL, MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, RETRIEVAL-AUGMENTED SYSTEM, STRUCTURED OUTPUT.

Object of research – the process of artificial intelligence systems providing structured information and program management based on informal user requests in the financial sector.

Purpose of work – development of a financially focused AI assistant with the ability to structured information output and program management based on informal user requests.

Methods of research – The work uses methods of system analysis, natural language processing, machine learning and programming. Algorithms are used to interpret informal requests, transform them into formal commands, integrate with financial programs and services, and evaluate the effectiveness of the developed system using indicators of accuracy and relevance of the information provided.

In this qualification work, a decision support system for the financial sector using artificial intelligence has been developed and implemented. A detailed analysis of the subject area was conducted, and modern approaches to natural language processing and decision-making automation were examined. The architecture of the AI assistant system was designed, enabling the processing of complex user queries, real-time information retrieval, financial data analysis, and generation of structured responses.

Particular attention was paid to integrating large language models and mechanisms for structured information output. The developed system ensures

adaptability to new query types, minimizes processing time, and allows for functionality expansion. The system was implemented programmatically, tested in real-world usage scenarios, and demonstrated effective performance. It can be integrated with other software products and applied across various industries.

The results of this work can be used to create similar solutions in market analysis, business process automation, customer service, and other fields requiring fast and accurate processing of large data volumes. The system holds significant potential for further development and improvement.

ЗМІСТ

Вступ	10
1 Системний аналіз предметної області та постановка задач дослідження	12
1.1 Системний аналіз задачі розробки системи AI assistant з можливістю структурованого виведення інформації	12
1.1.1 Велика модель мови	12
1.1.2 Ембедінг моделі	15
1.1.3 Пошукова доповнена система генерації тексту	18
1.1.4 Структурований вихід з великої моделі мови	20
1.2 Аналіз сценаріїв вирішення задачі розробки системи AI assistant з можливістю структурованого виведення інформації	22
1.2.1 Необхідність до навчання	22
1.2.2 Контекстна увага	23
1.3 Змістовна та формальна постановка задачі	23
1.4 Постановка задач дослідження	24
2 Вибір та обґрунтування методу розв’язання.....	26
2.1 Донавчання великої моделі мови	26
2.2 Пошукова доповнена система генерації тексту	27
2.2.1 Векторна база даних.....	27
2.2.2 Фінансові функції та автоматичний пошук в інтернеті	28
Висновки за розділом 2	31
3 Програмна реалізація	32
3.1 Віртуальне середовище та обчислювальні ресурси	32
3.2 Імплементация коду	35
3.2.1 Промпт	35
3.2.2 Функції	38
3.2.3 Діалог між функціями та великою мовною моделлю	38
3.2.4 Пошукова доповнена система генерації тексту	40
3.2.5 Велика мовна модель	41

	9
Висновки за розділом 3	42
4 Результати експерименту та їх аналіз	44
4.1 Експеримент для звичайного користування	44
4.2 Експеримент для користування структурованим виходом	46
Висновки за розділом 4	48
Висновки	50
Перелік джерел посилання	52
Додаток А Лістинг програми	54
Додаток Б Необхідні інсталяції для роботи асистенту.....	58

ВСТУП

Актуальність теми. Сучасний фінансовий сектор перебуває в стані постійної еволюції, обумовленої швидким розвитком технологій та глобалізацією ринків. Актуальність даної роботи зумовлена необхідністю підвищення ефективності діяльності у фінансовому секторі через автоматизацію процесів пошуку, обробки та аналізу інформації. У сучасному світі обсяг фінансових даних постійно зростає завдяки цифровізації, що створює серйозні виклики в їх оперативному аналізі та використанні для прийняття обґрунтованих і своєчасних рішень.

Зростання обсягів даних призводить до того, що традиційні методи їх обробки стають неефективними. Фінансові установи стикаються з проблемою необхідності швидкого реагування на зміни ринку, аналізу великого масиву інформації та адаптації до нових умов. У цьому контексті розробка фінансово орієнтованих AI-асистентів набуває особливого значення. Такі асистенти можуть досліджувати інформацію в інтернеті, аналізувати фінансові звіти, новини, ринкові тренди та надавати користувачам структуровану та релевантну інформацію.

Крім того, AI-асистенти здатні керувати програмами на основі неформальних запитів, що спрощує взаємодію користувача з складними системами та робить процес прийняття рішень більш інтуїтивним. Вони можуть автоматизувати рутинні завдання, зменшити ймовірність людських помилок та підвищити загальну продуктивність. Це особливо актуально в умовах високої конкуренції, де швидкість і точність прийняття рішень можуть стати вирішальними факторами успіху.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є розробка фінансово орієнтованого AI-асистента з можливістю структурованого виведення інформації та керування програмами на основі неформальних запитів користувачів. Для досягнення поставленої мети необхідно виконати наступні завдання:

- провести огляд і аналіз сучасного стану задачі розробки AI-асистентів з

можливістю структурованого виведення інформації;

- дослідити методи обробки природної мови для перетворення неформальних запитів у формальні виклики функцій;
- розробити архітектуру системи та реалізувати її основні компоненти;
- провести низку обчислювальних експериментів.

Об'єктом дослідження є процес надання системами штучного інтелекту структурованої інформації та керування програмами на основі неформальних запитів користувачів у фінансовій сфері.

Предметом дослідження є методи та засоби проектування і розробки фінансово орієнтованих AI-асистентів, здатних інтерпретувати неформальну мову користувачів, перетворювати її у формальні виклики функцій та обробляти результати для надання структурованої інформації.

Методи дослідження. У роботі використовуються методи системного аналізу, обробки природної мови (NLP), машинного навчання та програмування. Застосовуються алгоритми для інтерпретації неформальних запитів, перетворення їх у формальні команди, інтеграції з фінансовими програмами та сервісами, а також оцінки ефективності розробленої системи за допомогою показників точності та релевантності наданої інформації.

Публікації. Результати, отримані у роботі, було представлено на III International Scientific & Practical Conference “LEARNING & TEACHING: In the World after the War” November 8, 2024 (м. Харків, 8 листопада 2024 р.) [2].

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Системний аналіз задачі розробки системи AI assistant з можливістю структурованого виведення інформації

1.1.1 Велика модель мови

Велика мовна модель (LLM) – це тип штучного інтелекту, який призначений для обробки та генерування людської мови на основі машинного навчання. Ці моделі навчаються на великих обсягах текстових даних і застосовують складні архітектури нейронних мереж, щоб розуміти контекст, створювати текст і навіть вирішувати завдання, пов'язані з обробкою природної мови.

Основу великих мовних моделей складають сучасні архітектури нейронних мереж, серед яких найпопулярніша архітектура трансформера (див. рис. 1.1), представлена у 2017 році дослідниками з Google [1]. Трансформери використовують механізми уваги, що дозволяють моделі фокусуватися на різних частинах тексту залежно від завдання. Це дозволило досягти нових висот у галузі обробки природної мови, залишивши позаду попередні підходи такі, як рекурентні нейронні мережі (RNN) [2] та мережі довгої короткострокової пам'яті (LSTM) [3].

Процес навчання великих мовних моделей полягає в тому, що вони вивчають статистичні закономірності та взаємозв'язки у тексті, включаючи граматичні правила, семантичні зв'язки та навіть тонкощі контексту. Для цього модель обробляє трильйони слів та речень, взятих із книг, статей, сайтів, наукових публікацій та інших джерел.

Обсяг даних, необхідний для навчання LLM, настільки великий, що часто його вимірюють в терабайтах текстової інформації. Чим більше даних і що складніша архітектура, тим краще модель здатна генерувати зв'язковий і релевантний текст, і навіть справлятися з різними завданнями, наприклад, перекла-

дом тексту, відповідями питання, написанням коду і навіть творчими завданнями.

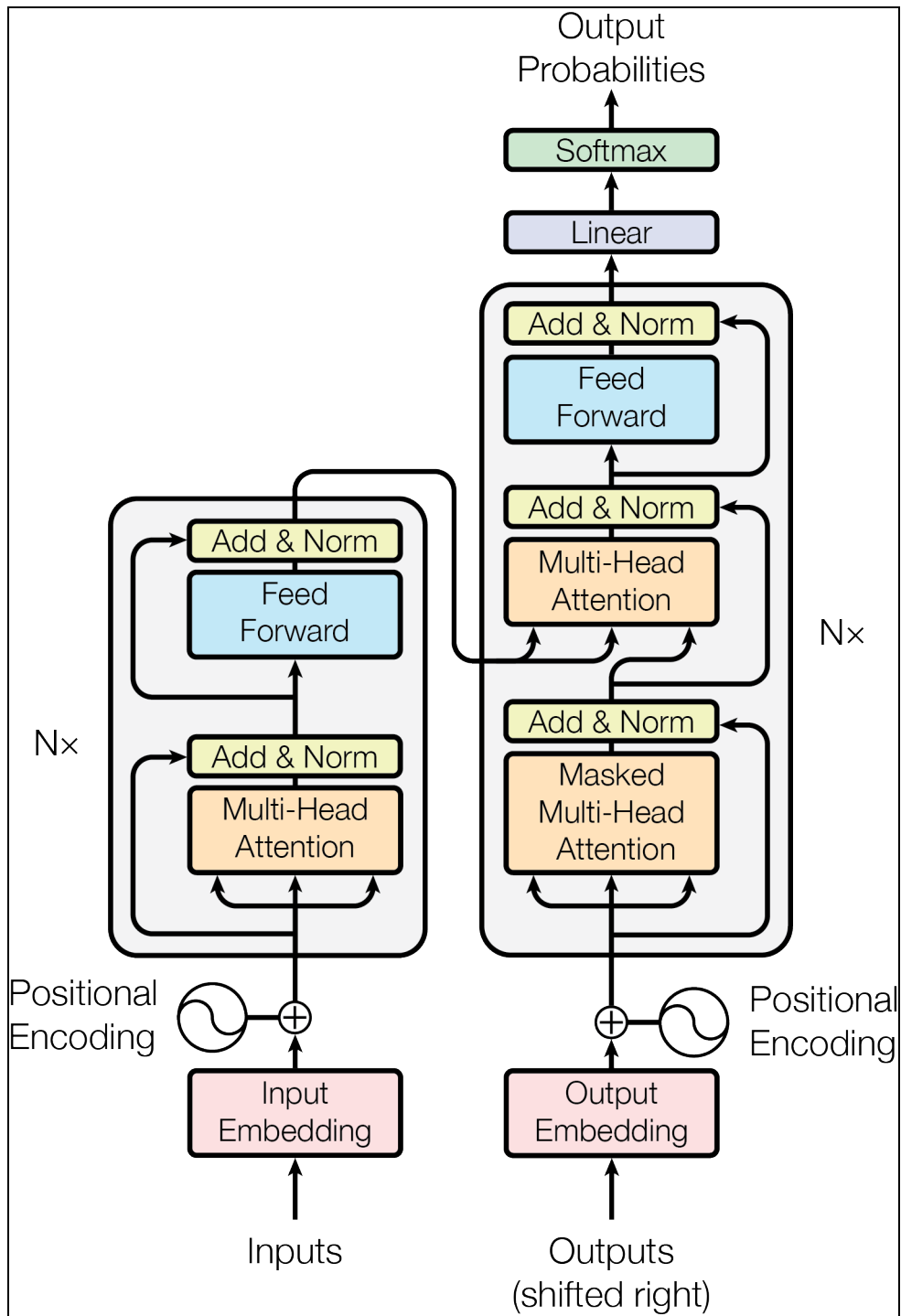


Рисунок 1.1 – Архітектура моделі трансформер

Важливо, що після базового навчання модель може бути донавчена до виконання конкретних завдань, наприклад, аналізу тональності тексту, складання

резюме документів чи вирішення конкретних питань у вузьких галузях знань [4]. Для цього застосовують техніки донавчання на спеціалізованих наборах даних, що дозволяють адаптувати загальну мовну модель до конкретних сценаріїв.

Сучасні LLM, такі як GPT-4o від OpenAI або Palm від Google, мають мільярди і навіть сотні мільярдів параметрів – ваг нейронної мережі, які коригуються в процесі навчання. Параметри можна представити як внутрішні «правила» моделі, що дозволяють їй прогнозувати, які слова чи фрази повинні йти за наявними у реченні. Чим більше параметрів має модель, тим більш детально і точно вона може аналізувати та генерувати тексти. Однак збільшення кількості параметрів також призводить до суттєвого зростання обчислювальної складності та витрат на навчання та обслуговування моделі.

Слід також розглянути проблеми, пов'язані з великими мовними моделями. По-перше, вони вимагають великих обчислювальних ресурсів, що обмежує їх доступність для широкого кола дослідників та розробників. По-друге, вони навчаються на текстах, взятих із відкритих джерел, і тому можуть успадкувати упередження, присутні у даних. Це означає, що LLM можуть генерувати тексти, що містять стереотипи або упереджені судження [5]. Для подолання цієї проблеми дослідники працюють над методами фільтрації та коригування навчальних даних, а також над розробкою механізмів контролю за результатами роботи моделі.

Крім того, важливим аспектом є інтерпретованість мовних моделей. Через величезну кількість параметрів та складність архітектури важко зрозуміти, як саме LLM приймає свої «рішення» при генеруванні тексту. Це викликає питання, пов'язані з довірою до моделі та її застосуванням у критично важливих галузях, наприклад, у медицині, праві чи фінансах.

В останні роки значну увагу приділяють створенню «інструктивних» LLM, здатних не тільки генерувати текст, а й виконувати більш складні завдання, наприклад, інтерактивну взаємодію з користувачами у форматі діалогів. Це відкриває нові можливості у створенні інтелектуальних чат-ботів, автоматизації процесів роботи з інформацією та навіть у викладанні.

1.1.2 Ембедінг моделі

Ембіддинги (або вбудовування) в машинному навчанні є векторними уявленнями даних, які дозволяють моделям обробляти і аналізувати текстову інформацію в числовій формі. Головна ідея полягає у перетворенні тексту на щільні вектори фіксованої розмірності, дозволяючи машині ефективно працювати з даними. Ці векторні уявлення захоплюють смислову та структурну інформацію про тексти, що робить їх ключовим компонентом сучасних методів глибокого навчання в обробці природної мови.

Раніше найбільш популярними методами були Word2Vec [6] і GloVe [7], які створювали ембедінги, відображаючи кожне слово в одиничний вектор, фіксуючи семантичну близькість слів через відстань між їх векторами. Однак ці методи були обмежені тим, що кожне слово мало лише одне фіксоване векторне уявлення незалежно від контексту. Це означало, що такі слова, як «банк» (у сенсі фінансової установи або річкового берега), представлялися одним і тим самим вектором, що знижувало точність аналізу залежно від контексту.

Сучасні моделі ембеддингів, такі як BERT [8], ERNIE [9] та інші, значно просунулися вперед у галузі обробки природної мови, особливо завдяки концепції «контекстуальних ембеддингів». Ці моделі розривають традицію відображення одного вектора в одне слово і створюють ембеддинги, які можуть бути як словом, так і реченням.

BERT (Bidirectional Encoder Representations from Transformers), одна з перших моделей, що використовує цю ідею. Це дозволяє моделі захоплювати двонаправлений контекст для кожного токена, генеруючи ембедінги, які відображають складні зв'язки між словами та фразами. Наприклад, слово «банк» у різних реченнях матиме різні векторні уявлення залежно від його контексту, що значно підвищує точність моделей у завданнях, пов'язаних із розумінням тексту.

Ці моделі використовують механізм самоуваги [1] для аналізу всіх токенів у вхідному тексті, дозволяючи ефективно моделювати відносини між токе-

нами різних відстанях друг від друга. Це суттєво відрізняється від старих підходів, таких як Word2Vec або GloVe, обмежених фіксованими довжинами контекстних вікон.

Крім того, відкриті моделі від дослідних інститутів, такі як ERNIE від Beijing Academy of Artificial Intelligence (BAAI), пішли ще далі. ERNIE навчається, використовуючи багатий контекст із різних джерел, включаючи знання з баз даних, що дозволяє йому створювати глибші та інформативні ембедінги. Підхід ERNIE ґрунтується на вилученні сутностей та знань з великих обсягів тексту, які потім інтегруються у процес створення контекстуальних ембедінгів. Це допомагає моделі захоплювати навіть складні та специфічні семантичні зв'язки між токенами.

Ще одним важливим завданням, у якому використовуються сучасні ембедінги, є ранжування документів. Ембедінги дозволяють ефективно порівнювати релевантність документів щодо запиту користувача. Моделі ембедінгів використовують двонаправлені контекстуальні ембедінги, щоб зрозуміти нюанси та зміст запиту та порівняти його з кожним із документів у пошуку. Цей підхід значно покращує якість ранжирування в інформаційних пошукових системах та рекомендаційних системах. На рисунку 1.2 можна побачити схему перетворення тексту у вектори.

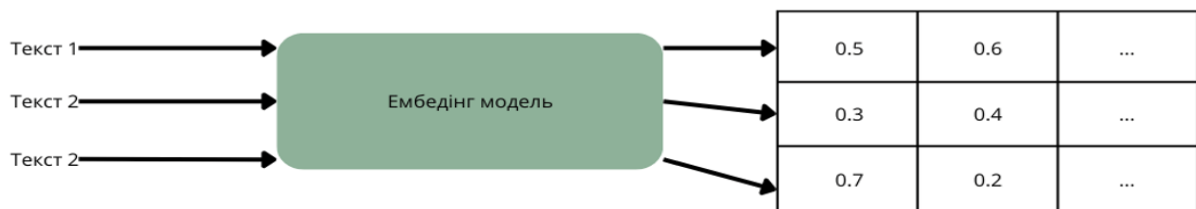


Рисунок 1.2 – Перетворення тексту у вектори

Контекстуальні ембедінги також дозволяють представляти токени (час-

тини слів) у просторі ембеддингів, що підвищує гнучкість роботи з текстом. Наприклад, використання токенизації на рівні підслів дозволяє моделі обробляти незнайомі слова чи неологізми, розбиваючи їх у складові. Таким чином, кожне слово чи підслів у реченні має своє унікальне векторне уявлення, яке враховує його контекст на відміну від старих підходів на рівні слова (див. рис. 1.3).

Зразкові дані

“This is tokenizing.”

Рівень літер

[T] [h] [i] [s] [i] [s] [t] [o] [k] [e] [n] [i] [z] [i] [n] [g] [.]

Рівень слова

[This] [is] [tokenizing] [.]

Рівень підслова

[This] [is] [token] [izing] [.]

Рисунок 1.3 – Токенизація різними способами

Ембіддинги можуть бути також використані для подання цілих речень, абзаців або документів у вигляді векторів фіксованої розмірності (див. рис. 1.3). Моделі, такі як Sentence-BERT [10], розвивають цю ідею, створюючи ембеддинги лише на рівні пропозицій. Це дозволяє порівнювати смислові значення різних пропозицій у просторі ембеддингів, застосовуючи їх у завданнях семантичного пошуку та кластеризації текстів.

Таким чином, сучасні ембеддинги захоплюють набагато складніші зв'язки між словами та фразами, ніж їхні попередники. Їх використання дозволяє моделям працювати з текстом більш гнучко і точно, що знайшло широке застосування у різних завданнях обробки природної мови, включаючи аналіз тексту, реранкінг, переклад та генерацію природної мови.

1.1.3 Пошукова доповнена система генерації тексту

Пошукова доповнена система генерації тексту (Retrieval-Augmented Generation, [11]) – це гібридний підхід у галузі обробки природної мови, який поєднує два ключові компоненти: механізми вилучення інформації та генеративні мовні моделі. Основною ідеєю пошукової доповненої системи генерації тексту є використання механізмів пошуку інформації для знаходження релевантних даних, які потім використовуються як контекст при генерації тексту за допомогою моделей глибокого навчання, таких як GPT, BERT або їх варіацій.

У традиційних генеративних моделях відповідь на запит користувача повністю генерується на основі даних, що присутні в тренувальному наборі моделі. Це створює обмеження, оскільки модель не може виходити за рамки заздалегідь навченої інформації та не може оновлюватись у реальному часі. Однак пошукова доповнена система генерації тексту вирішує цю проблему шляхом інтеграції пошукових механізмів. Таким чином, система може динамічно звертатися до актуальних та специфічних джерел даних, таких як бази знань, вебсторінки або спеціалізовані документи, що дозволяє їй ефективніше відповідати на питання, що виходять за межі початкових тренувальних даних.

Процес роботи пошукової доповненої системи генерації тексту зазвичай складається з двох етапів: пошук та генерація (див. рис. 1.4). На етапі пошуку система аналізує запит користувача та отримує релевантну інформацію із зовнішньої бази даних або пошукового індексу. Це дозволяє звужити обсяг інформації та знайти найкорисніші дані, які будуть використовуватися при генерації відповіді. На другому етапі (генерація) мовна модель, приймаючи на вхід знайдену інформацію, створює осмислену та докладну відповідь. Таким чином, пошукова доповнена система генерації тексту поєднує сильні сторони інформаційно-пошукових систем (для пошуку специфічної інформації), так і мовних моделей (для генерації зв'язкового тексту).

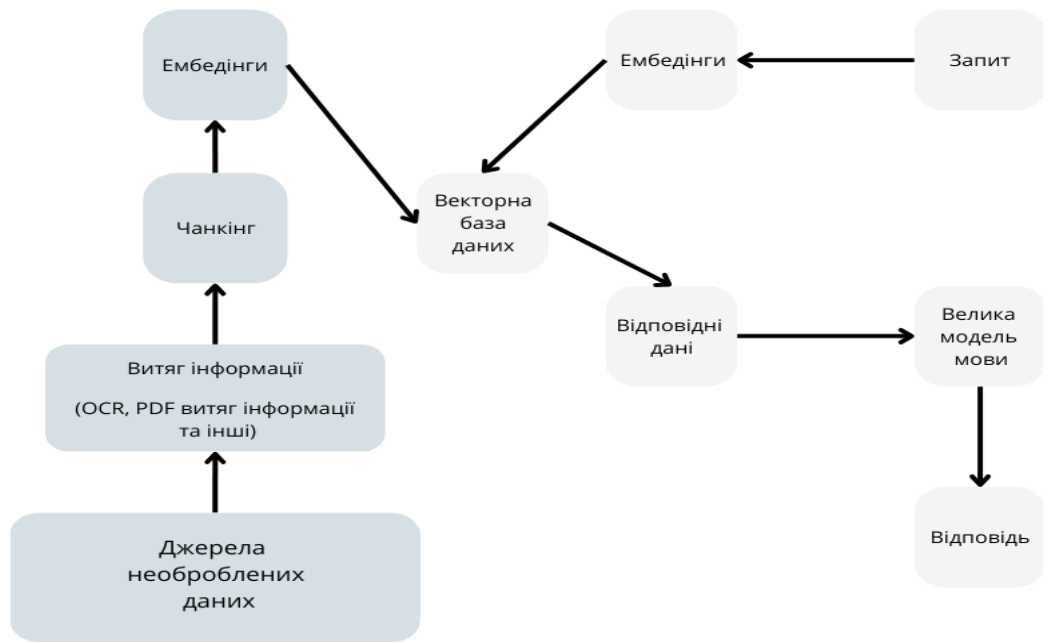


Рисунок 1.4 – Пошукова доповнена система генерації тексту

Однією з ключових переваг пошукової доповненої системи генерації тексту є її здатність надавати більш точні та актуальні відповіді на запити користувача. Наприклад, у сценарії, коли потрібне вилучення фактичної інформації (наприклад, дата події або опис процесу), система пошукової доповненої системи генерації тексту може знайти цю інформацію в реальному часі та включити її до згенерованого тексту. Крім того, оскільки пошукова доповнена система генерації тексту використовує механізм отримання інформації, обсяг знань, доступний системі, стає практично безмежним, і вона не обмежується фіксованим набором даних, як традиційні мовні моделі.

У технічному аспекті пошукова доповнена система генерації тексту використовує техніку векторного пошуку роботи з ембеддингами – чисельними уявленнями текстових даних у багатовимірному просторі. Векторні ембедінги дозволяють ефективно знаходити семантично схожу інформацію у базі даних на основі косинусу (див. рис. 1.5) або скалярного добутку.

Після того як знайдені релевантні документи, вони передаються генеративній моделі, яка створює остаточну відповідь. Цей підхід дозволяє забезпечувати високу відповідність тексту, що генерується.

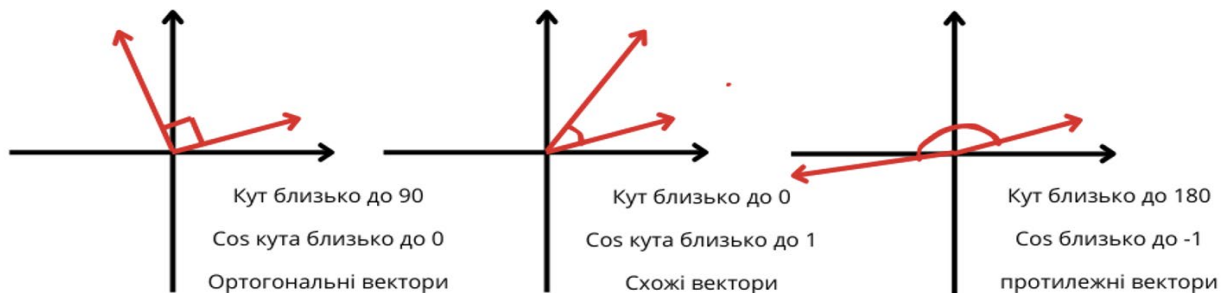


Рисунок 1.5 – Схожість за косинусом

З точки зору застосування, пошукова доповнена система генерації тексту може бути використаний в різних областях, включаючи чат-боти, системи підтримки клієнтів, пошукові механізми, а також у будь-яких сценаріях, де потрібен доступ до обширних баз даних, що постійно оновлюються. Наприклад, система підтримки клієнтів на базі пошукової доповненої системи генерації тексту здатна аналізувати запит користувача, знаходити найбільш підходящу інформацію в базі знань компанії та надати розгорнуту та точну відповідь, тим самим покращуючи якість обслуговування.

1.1.4 Структурований вихід з великої моделі мови

Структурований вихід із великої мовної моделі — це підхід, який дозволяє генерувати та використовувати дані у заданому форматі на основі текстового запиту користувача. Тобто замість того, щоб просто повертати неструктурований текст у відповідь на запит, мовна модель створює дані, що відповідають заданій структурі. Це може бути у вигляді JSON, таблиць або іншого структурованого формату, який може бути зручно використаний для подальшого аналізу, автоматизації процесів чи інтеграції з іншими системами.

Основною метою використання структурованого виходу є підвищення точності та зручності інтеграції моделей штучного інтелекту в різні програмні системи. Завдяки структурованому виходу, велика мовна модель може виконувати роль своєрідного «інтерфейсу» між користувачем та системою, обробляю-

чи текстовий запит та повертати чіткі, добре організовані дані, які легко використовувати для прийняття рішень або проведення автоматизованих операцій.

Наприклад, у разі запиту «Знайди мені найближчий ресторан», модель може не тільки видати текстовий відповідь з рекомендаціями, а й створити об'єкт JSON зі структурованими даними, який буде містити назву ресторану, його адресу, рейтинг, час роботи та контактні дані. Цей об'єкт може бути використаний іншою програмою, яка, наприклад, виведе цю інформацію у зрозумілому для користувача форматі або використовує її для навігації до ресторану.

Важливо розуміти, що для отримання структурованого виходу з великої мовної моделі зазвичай використовуються спеціальні інструкції та шаблони. Це може включати використання підказок (prompts) з конкретними вказівками щодо того, як мають бути організовані дані у відповіді. Мовні моделі мають можливість інтерпретувати ці інструкції та генерувати вихідні дані у потрібному форматі. Наприклад, запит може виглядати так: «Склади список з трьох найпопулярніших туристичних міст у Європі та подай їх у форматі JSON з ключами для назви міста, країни та кількості туристів».

Таким чином, структурований вихід дозволяє не тільки підвищити точність та корисність відповіді від моделі, а й інтегрувати її у більш складні робочі процеси. Наприклад, у розробці чат-ботів для обслуговування клієнтів або систем підтримки, де необхідно отримати від користувача певну інформацію та передати її далі до баз даних чи інших автоматизованих систем. Інтеграція структурованого виходу також робить взаємодію з великою мовною моделлю більш природною для користувача, адже він може формулювати свої запити звичайною мовою, а система вже обробляє їх і перетворює на необхідні дані.

Крім того, цей підхід значно розширює сферу використання LLM у бізнесі та індустрії в цілому. Наприклад, він може бути використаний у фінансових додатках для автоматичної обробки запитів на аналіз ринку, в медичних системах для збирання та класифікації симптомів пацієнтів або в службах доставки для автоматизації обробки замовлень.

З технічної точки зору, структурований вихід із великої мовної моделі

включає в себе можливість налаштовувати інтерфейси та методи для обміну даними з іншими програмами та сервісами. Це дозволяє створювати більш гнучкі та адаптовані системи, які можуть взаємодіяти з мовними моделями в реальному часі. Зрештою, це не лише спрощує розробку додатків та сервісів, а й покращує досвід користувача завдяки точності та чіткості відповідей.

1.2 Аналіз сценаріїв вирішення задачі розробки системи AI assistant з можливістю структурованого виведення інформації

1.2.1 Необхідність донавчання

Сьогодні існує багато сценаріїв вирішення завдань із залученням мовних моделей. Однак персональні комп'ютери все ще не мають достатньої обчислювальної потужності для створення та навчання великих мовних моделей з нуля. Навчання таких моделей потребує значних ресурсів, включаючи високоякісні графічні процесори та великі набори даних. Тому більш реалістичним підходом є пошук попередньо навченої моделі, здатної виконувати потрібні функції.

Однак цей підхід створює новий виклик: концепція структурованого виводу, коли модель повинна генерувати інформацію в певному форматі або структурі, є відносно новою, і більшість існуючих моделей не були навчені для цього конкретного завдання. У результаті ми стикаємося з подвійним завданням: з одного боку, нам потрібно вибрати модель відповідного розміру, яка може працювати на персональному комп'ютері та виконувати завдання в розумні терміни; з іншого боку, нам потрібно знайти модель, яка вже була налаштована відповідно до вимог структурованого виведення.

Безумовно, можливо використовувати будь-яку попередньо навчену модель і адаптувати її до конкретного завдання шляхом оперативного проектування. Однак цей підхід може бути менш ефективним і вимагати більше зусиль, ніж вибір моделі, спочатку навченої з урахуванням особливостей структурова-

ного виведення. Оптимальним рішенням буде пошук і використання моделі, яка вже має вбудовані механізми структурованого виведення, оскільки це значно підвищить якість і надійність результатів.

1.2.2 Контекстна увага

Якщо кількість функцій, які можна виконати за допомогою структурованого виходу великої мовної моделі, надмірно велика, модель може почати плутатися між різними функціями. Це відбувається тому, що модель намагається знайти відповідність між заданим завданням і доступними функціями, але велика кількість варіантів збільшує ймовірність помилки. Важливо також врахувати обмеження контекстного вікна – максимальної кількості тексту, який модель здатна обробити за одну ітерацію. Наприклад, якщо контекстне вікно моделі обмежене кількома тисячами токенів, а ми надаємо великий перелік інструкцій та описів функцій, це може призвести до витіснення важливих деталей із контексту. У такій ситуації модель може «не помітити» або неправильно скористатися потрібною функцією. Це особливо критично, коли інструкції займають значну частину контекстного вікна, залишаючи менше місця для корисної інформації, необхідної для коректного виконання завдання.

1.3 Змістовна та формальна постановка задачі

Змістовна постановка задачі: необхідно розробити систему AI-асистента, яка здатна генерувати відповіді на запити користувачів та забезпечувати структуроване виведення інформації у зручному форматі для подальшої обробки. Асистент має інтегруватися з різними джерелами інформації та обробляти великі обсяги даних, зокрема текстові повідомлення, документи та інші текстові джерела. Основні функції системи включають:

- розуміння природної мови для інтерпретації запитів користувачів;
- збір та аналіз інформації із зовнішніх джерел;
- генерацію структурованих відповідей у вигляді таблиць, списків, карток або інших зручних для сприйняття форматів;
- забезпечення користувача підходящою інформацією на основі введеного запиту.

Формальна постановка задачі: вхідні дані: текстові запити користувачів природною мовою, а також дані із зовнішніх джерел, таких як документи, веб-сторінки, бази знань тощо. Вихідні дані: структурована відповідь на запит у вигляді таблиць, списків, карток або інших форм, що забезпечують легке сприйняття та подальше використання отриманої інформації.

Обмеження:

- AI-асистент повинен підтримувати обробку тексту різними мовами;
- час обробки запитів повинен бути мінімізований для забезпечення швидкої відповіді користувачу;
- система повинна мати можливість адаптуватися до нових типів запитів і бути розширюваною для подальшої інтеграції додаткових джерел даних.

Цільова функція: забезпечити максимальну релевантність та структурованість відповідей на запити користувача

1.4 Постановка задач дослідження

Метою кваліфікаційної роботи є проєктування та імплементація AI асистента зі структурованим вихідом, який буде інтерпритовано для допомоги у фінансах та пошуку інформації в інтернеті в реальному часі з подальшою обробкою для виводу з інформацією у форматі діалогу.

Для досягнення поставленої мети треба виконати наступні етапи:

- провести огляд і аналіз сучасного стану задачі розробки AI-асистентів з можливістю структурованого виведення інформації;

- дослідити методи обробки природної мови для перетворення неформальних запитів у формальні виклики функцій;
- розробити архітектуру системи та реалізувати її основні компоненти;
- провести низку обчислювальних експериментів.

2 ВИБІР ТА ОБГРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ

2.1 Донавчання великої моделі мови

Ми вирішили вибрати модель Llama 3.1, оскільки вона демонструє вражаючі результати серед моделей з відкритим кодом (див. рис. 2.1). Крім того, її гнучкість та адаптивність до різних завдань роблять її перспективним кандидатом для нашого проєкту.

T ▲	Model	Average ⓘ
○	meta-llama/Meta-Llama-3.1-8B-Instruct	27.91
○	vicgalle/Configurable-Llama-3.1-8B-Instruct	27.77
○	01-ai/Yi-1.5-9B-Chat	27.71
🏆	mlabonne/NeuralDaredevil-8B-abliterated	27.01
○	Danielbrdz/Barcnas-Llama3-8b-ORPO	26.38
○	byroneverson/Yi-1.5-9B-Chat-16K-abliterated	26.37
○	princeton-nlp/Llama-3-Instruct-8B-ORPO-v0.2	25.68
○	ValiantLabs/Llama3.1-8B-Cobalt	25.56
🏆	Sao10K/L3-8B-Lunaris-v1	25.48
○	vicgalle/Humanish-RP-Llama-3.1-8B	25.17

Рисунок 2.1 – Топ великих мовних моделей

Згідно зі статтею [12], присвяченою релізу Llama 3.1, модель може працювати з багатьма типами даних, але існують деякі обмеження щодо структурованого виведення інформації. Іншими словами, ми не можемо бути впевнені, що вона завжди видаватиме дані в строгому форматі, наприклад, у вигляді JSON. Це потенційно може викликати складнощі при інтеграції моделі в робочий процес, де потрібна строго певна структура відповіді.

Щоб забезпечити стабільність та передбачуваність виведення, потрібно провести додаткове навчання моделі на прикладах, орієнтованих на генерацію строго структурованих виходів, таких як JSON. Цей процес дозволить моделі

краще адаптуватися до вимог завдання та знизить ймовірність помилок. Однак навіть після донавчання можуть виникати випадки відхилень від заданої структури. Для їх обробки необхідно буде впровадити постобробку результатів.

На щастя, ми вже маємо рішення для донавчання моделі. Ми можемо використовувати модель NousResearch Llama 3.1 [13], яка була спеціально донавчена для створення структурованого виходу на основі архітектури Llama 3.1. При цьому важливо відзначити, що сама архітектура моделі залишилася незмінною. Це означає, що багато бібліотек прискорення підтримуватимуть її, оскільки архітектура не змінилася і дуже популярна.

2.2 Пошукова доповнена система генерації тексту

2.2.1 Векторна база даних

Для вирішення проблеми контекстної уваги у великій мовній моделі ми плануємо використати пошуково-розширену систему генерації тексту. Такий підхід дозволяє ефективно знаходити та використовувати релевантні функції з великого набору інструментів, зберігаючи при цьому високу продуктивність моделі та точність відповідей.

Основна ідея методу полягає в тому, щоб скористатися ембеддингами, для пошуку функцій під конкретний запит користувача. Кожна функція буде описана за допомогою спеціального текстового документа, так званого `doc string`, який містить всю необхідну інформацію: назву функції, список та опис її аргументів, а також загальне призначення та сферу застосування. Цей підхід забезпечує ясне та зрозуміле опис функціоналу, що дозволяє автоматично підбирати потрібні інструменти для вирішення поставленого завдання.

Першим кроком реалізації цієї системи буде створення векторної бази даних, що містить закодовані описи функцій. Кожен `doc string` буде перетворено на ембеддинг за допомогою мовної моделі, що дозволить нам організувати по-

шук у багатовимірному векторному просторі. Коли користувач вводить запит або завдання, ми також кодуємо цей текст в ембеддинг. Потім виконується пошук найближчих векторів (функцій) у базі даних, що дає попередній набір відповідних функцій.

Після первинного пошуку виконується додатковий етап ранжування. На цьому етапі використовується більш точна модель ранжирування (реранкер), яка оцінює знайдені функції на відповідність запиту користувача. Це дозволяє уточнити список кандидатів, щоб гарантувати, що у підсумковому наборі будуть найбільш релевантні функції.

На заключному етапі ми відбираємо топ-3 найбільш відповідних функцій за результатами ранжування. Цей невеликий список передається як підказка (промпт) великої мовної моделі, яка потім використовує цю інформацію для генерації підсумкової відповіді. Такий метод забезпечує високу точність та контекстну релевантність, дозволяючи ефективно вирішувати завдання з використанням вибраних функцій.

2.2.2 Фінансові функції та автоматичний пошук в інтернеті

Структурований результат, створений моделлю, забезпечує суттєву користь, але для повного використання цього потенціалу потрібні додаткові кроки. Щоб ефективно використовувати цей результат, ми розробимо набір функцій, здатних отримувати фінансову інформацію в реальному часі з різних джерел і перетворювати її в текстовий формат, який може оброблятися моделлю.

Якщо попередньо визначені функції не можуть надати необхідну інформацію, буде використано динамічний резервний механізм. Цей механізм передбачає пошук в Інтернеті в режимі реального часу для отримання певних даних, недоступних у межах наших поточних функцій. Інтегруючи цю резервну систему, ми забезпечуємо комплексний пошук інформації та забезпечуємо надійність нашого загального процесу вилучення інформації.

Функція пошуку в Інтернеті слугує не тільки для пошуку відповідних даних, але й для отримання їх у простій текстовій формі. Потім цей текст можна проаналізувати та обробити великою мовною моделлю, яка синтезуватиме інформацію та генеруватиме добре структуровану, релевантну контексту відповідь. Цей процес підвищує гнучкість і точність моделі в сценаріях, де потрібні конкретні або актуальні дані.

Крім того, ми включимо додаткову специфіку в інші функції, щоб підвищити їх продуктивність і адаптивність (див. табл. 2.1).

Таблиця 2.1 – Функції та їх опис

Назва функції	Аргументи	Тип виходу	Опис
<code>get_current_stock_price</code>	<code>symbol (str)</code>	<code>float</code>	Отримує поточну ціну акцій за вказаним символом. Повертає <code>None</code> , якщо виникає помилка.
<code>get_stock_fundamentals</code>	<code>symbol (str)</code>	<code>dict</code>	Отримує фундаментальні дані для символу акцій, включаючи назву компанії, сектор, індустрію, ринкову капіталізацію, P/E коефіцієнт тощо.
<code>get_financial_statements</code>	<code>symbol (str)</code>	<code>dict</code>	Надає фінансові звіти (звіт про доходи, баланс та рух грошових коштів) для вказаного символу

Кінець таблиці 2.1

Назва функції	Аргументи	Тип виходу	Опис
get_key_financial_ratios	symbol (str)	dict	Отримує ключові фінансові коефіцієнти для вказаного символу акцій.
get_analyst_recommendations	symbol (str)	pd.DataFrame	Отримує рекомендації аналітиків для вказаного символу акцій.
get_dividend_data	symbol (str)	pd.DataFrame	Отримує дані про дивіденди для вказаного символу акцій.
get_company_news	symbol (str)	pd.DataFrame	Отримує новини та прес-релізи компанії для вказаного символу акцій.
get_technical_indicators	symbol (str)	pd.DataFrame	Надає дані про технічні індикатори для вказаного символу акцій.
get_company_profile	symbol (str)	dict	Отримує профіль компанії та огляд для вказаного символу акцій.

Ці функції будуть розроблені з урахуванням модульності, що дозволить розширити їхні можливості на основі нових вимог до пошуку даних. Цей методичний підхід гарантує, що ми підтримуємо надійну та масштабовану систему, здатну обробляти широкий спектр фінансових запитів.

Висновки за розділом 2

Було ухвалено рішення зупинитися на моделі Llama 3.1, оскільки вона продемонструвала вражаючі результати серед моделей з відкритим кодом. Її гнучкість та здатність до навчання на різноманітних наборах даних робить її перспективним вибором для багатьох завдань, зокрема там, де потрібна точна та релевантна генерація тексту. Водночас існує певне обмеження у вигляді відсутності гарантовано структурованого виходу (наприклад, у форматі JSON), що може ускладнити інтеграцію моделі в робочі процеси з жорсткими вимогами до формату даних. Для розв'язання цієї проблеми пропонується донавчати модель на спеціальних прикладах і впроваджувати постобробку результатів. З огляду на це, NousResearch Llama 3.1 видається оптимальним варіантом, оскільки зберігає архітектуру оригінальної моделі й, таким чином, сумісна з більшістю популярних бібліотек для її прискорення.

Щоб забезпечити більшу точність і контекстну релевантність відповідей, в основу рішення було закладено пошуково-розширену систему генерації тексту. Ідея полягає в тому, щоб спочатку зберігати описи доступних функцій у вигляді векторів у спеціалізованій базі даних. Під час обробки запиту користувача ці описи порівнюються з текстом запиту за допомогою пошуку найближчих сусідів у векторному просторі. Знайдені функції ранжуються вдруге за допомогою більш точної моделі (реранкера). У результаті відбирається топ-3 найбільш релевантних функції, які передаються великій мовній моделі як підказка. Це суттєво покращує якість фінальних відповідей і дозволяє розв'язувати різноманітні завдання з урахуванням наявного функціоналу.

Особлива увага приділяється можливості отримувати та обробляти фінансові дані в реальному часі. Запропоновані функції дають змогу не лише збирати цінову інформацію про акції, але й отримувати фундаментальні показники та рекомендації аналітиків. Крім того, передбачено механізм динамічного пошуку в Інтернеті, що діє як резервне рішення для тих випадків, коли внутрішні інструменти не в змозі задовольнити конкретний запит.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Віртуальне середовище та обчислювальні ресурси

Великі моделі мови потребують багато VRAM пам'яті для швидкої роботи. Звісно, є можливість користуватися ними за допомогою CPU і тільки, але це не є нормальною практикою.

В нашому розпорядженні є лише одна NVIDIA GeForce RTX 3080 Ti Laptop GPU. Є можливість користуватися й хмарними обчислюваннями, але вони усі не безкоштовні.

Квантизація – це процес відображення великого набору вхідних значень у менший набір вихідних значень, зазвичай шляхом округлення або скорочення. У цифровій обробці сигналів він використовується для апроксимації безперервних амплітуд сигналів дискретними рівнями, що дозволяє представляти аналогові сигнали в цифровій формі. У сфері машинного навчання, особливо з великими мовними моделями, квантування використовується для зниження точності числових представлень, таких як ваги моделі та активації. Це перетворення, яке часто перетворює 32-розрядні числа з плаваючою комою у формати з нижчою точністю, такі як 8-розрядні цілі числа, значно зменшує використання пам'яті та покращує обчислювальну ефективність, полегшуючи розгортання великих мовних моделей на пристроях з обмеженими ресурсами, такими як смартфони чи периферійне обладнання. Такі методи, як квантування після навчання та навчання з урахуванням квантування, широко застосовуються для оптимізації великої мовної моделі, зберігаючи при цьому точність моделі. Компроміс у квантуванні полягає в управлінні помилкою апроксимації, яку часто називають шумом квантування, яка може вплинути на продуктивність моделі. Удосконалені стратегії квантування, такі як квантування зі змішаною точністю або адаптивне квантування, спрямовані на мінімізацію цієї помилки, гарантуючи, що великі мовні моделі зберігають свою ефективність у створенні узгоджених і контекстуально релевантних виходів, використовуючи при цьому знижені обчислюва-

льні витрати.

Для того, щоб архітектура Llama 3.1 змогла запуснитися на нашій відеокарті, будемо користуватись її «популярністю». Тобто, існує багато реалізацій під цю конкретну архітектуру техніки квантизації. Якщо бути більш точним, будемо користуватись подвійною квантизацією та форматом normal float, представлених у QLoRa [15]. Подвійна квантизація дозволяє знизити точність представлення чисел у два етапи: первинна квантизація для грубого зменшення розмірності, а потім додаткове уточнення для збереження ключової інформації. Формат Normal Float (NF4) є важливим елементом цього процесу, оскільки він був спеціально розроблений для нейронних мереж і оптимізований для зберігання значень, які підпорядковуються нормальному (Гауссовому) розподілу.

На відміну від традиційних технік квантизації, таких як fixed-point або просте зменшення бітності, NF4 забезпечує ширший динамічний діапазон, дозволяючи зберігати як великі, так і дуже малі значення, які часто зустрічаються в вагах і активаціях моделей. Це особливо важливо для роботи Llama 3.1, яка використовує складні взаємодії між параметрами для генерації тексту високої якості.

Використання QLoRa дозволяє не тільки знизити апаратні вимоги, але й зберегти точність моделі завдяки інтеграції таких форматів, як NF4, і підходів, орієнтованих на специфічні властивості даних нейронних мереж. Таким чином, ми отримуємо змогу запускати великі моделі на відеокартах з обмеженими ресурсами, зберігаючи при цьому високу якість результатів.

Віртуальне середовище – це ізольований простір в системі, що дозволяє запускати програмне забезпечення або виконувати код з чітко визначеними залежностями, без впливу на глобальну конфігурацію системи. Воно створюється для забезпечення сумісності та уникнення конфліктів між різними версіями бібліотек чи інструментів, що використовуються в різних проєктах.

У контексті програмування в Python, віртуальне середовище (наприклад, створене за допомогою інструментів `venv` або `virtualenv`) дозволяє створити окрему область для збереження та використання залежностей проєкту. У цьому

середовищі можна встановлювати конкретні версії бібліотек, необхідні для певного проєкту, що не впливатиме на інші проєкти або системний інтерпретатор Python.

Віртуальні середовища широко використовуються для:

- тестування програмного забезпечення в різних конфігураціях;
- розробки, щоб уникнути конфліктів між залежностями проєктів;
- забезпечення можливості відтворюваності середовища під час розгортання додатків.

Ключова перевага віртуальних середовищ – це ізоляція, яка гарантує стабільність та передбачуваність роботи проєктів у будь-якому середовищі розробки чи розгортання. Завдяки ізоляції, кожен проєкт може використовувати свої специфічні версії залежностей, бібліотек або інструментів, не впливаючи на інші проєкти на тій самій системі. Це особливо важливо в ситуаціях, коли різні проєкти потребують різних версій однієї і тієї ж бібліотеки, що може створити конфлікти у глобально встановлених залежностях.

Для управління віртуальними середовищами та пакетами ми будемо використовувати Anaconda Navigator – зручний інструмент для швидкого та ефективного менеджменту. Це дозволить легко створювати, налаштовувати й керувати ізольованими середовищами для проєкту.

Щоб забезпечити підтримку квантизації та роботи з моделлю Llama 3.1, необхідно встановити такі бібліотеки:

- а) Bitsandbytes – для реалізації технік квантизації;
- б) Torch – для підтримки обчислень на GPU;
- в) Huggingface – для завантаження та роботи з моделлю Llama 3.1.

Більш детальну інформацію про кожну з цих бібліотек наведено в Додатку Б.

Після встановлення всіх необхідних компонентів ми будемо готові до реалізації нашого проєкту, використовуючи обрану архітектуру та оптимізаційні підходи.

3.2 Імплементация коду

3.2.1 Промпт

Промпт – це вхідний запит або текстова інструкція, яка передається моделі штучного інтелекту (наприклад, великій мовній моделі, як GPT), щоб викликати відповідь або виконання певного завдання. Промпт визначає контекст, формат і мету відповіді, дозволяючи моделі генерувати релевантний і змістовний результат.

У машинному навчанні та генеративному ШІ промпт може бути як простим (одне речення або питання), так і складним (детальний опис завдання, приклади вхідних даних і бажаного формату відповіді). Наприклад, у роботі з мовними моделями промпт може виглядати так: «Напиши коротку розповідь про подорож до космосу» або «Відповідай на питання: що таке квантизація?».

Ефективність моделі значною мірою залежить від якості та точності промпту. Добре сформульований промпт допомагає отримати більш коректну і цільову відповідь, тоді як нечіткий або занадто загальний промпт може призводити до неточних або нерелевантних результатів.

Для нашої великої мовної моделі нам також потрібно створити подібну структуру. Ми розділимо створення підказки на такі частини: роль, ціль, інструменти, схема та інструкції. Таким чином, у нас є п'ять категорій для структурування наших інструкцій.

Роль відображає особистість або «персонаж», який приймає модель. Тут ми можемо вказати назву, загальну мету та будь-які організаційні деталі, наприклад, як позначати початок і кінець списку інструментів.

Ціль надає абстрактне визначення мети моделі, описуючи, чого ми прагнемо досягти та якою має бути ідеальна відповідь для користувача. Наприклад, ми можемо включити такі інструкції, як планування відповідей, послідовне використання функцій і надання підсумків, щоб забезпечити задоволення користувачів.

Розділ «Інструменти» описує функції, які ми визначили раніше (наприклад ті, що наведені у таблиці 2.1). Щоб включити їх у модель, їх слід написати структуровано з чіткими індикаторами, що позначають початок і кінець списку інструментів. Наприклад, ви можете описати функцію, її призначення, параметри та очікуваний результат для ясності.

Схема визначає очікуваний формат виведення, коли моделі необхідно викликати функцію. Наприклад, якщо модель хоче виконати функцію, вона має вивести такий результат: `<tool_call> {"name": <function-name>, "arguments": <args-dict>} </tool_call>`.

Нарешті, розділ «Інструкції» містить усі прямі інструкції, уникаючи абстракції. Наприклад, вказівки можуть включати без вагань послідовний виклик функцій, коли це необхідно, аналіз вихідних даних функції для надання більш точної інформації тощо.

Зрештою, ми створюємо підказку з об'єднаними всіма цими елементами.

Role: |

You are a function calling AI agent with self-recursion.

You can call only one function at a time and analyse data you get from function response.

You are provided with function signatures within `<tools></tools>` XML tags.

The current date is: {date}.

Objective: |

You may use agentic frameworks for reasoning and planning to help with user query.

Please call a function and wait for function results to be provided to you in the next iteration.

Don't make assumptions about what values to plug into function arguments.

Once you have called a function, results will be fed back to you within `<tool_response></tool_response>` XML tags.

Don't make assumptions about tool results if `<tool_response>` XML tags are not present since function hasn't been executed yet.

Analyze the data once you get the results and call another function.

At each iteration please continue adding the your analysis to previous summary.

Your final response should directly answer the user query with an anlysis or summary of the results of function calls.

Tools: |

Here are the available tools:

```
<tools> {tools} </tools>
```

If the provided function signatures doesn't have the function you must call, you may write executable python code in markdown syntax and call `code_interpreter()` function as follows:

```
<tool_call>
  {{{"name": "code_interpreter", "arguments": {{"code_markdown": <python-
code>}}}}}
</tool_call>
```

Make sure that the json object above with code markdown block is parseable with `json.loads()` and the XML block with `XML ElementTree`.

Examples: |

Here are some example usage of functions:

```
{examples}
```

Schema: |

Use the following pydantic model json schema for each tool call you will make:

```
{schema}
```

Instructions: |

At the very first turn you don't have `<tool_results>` so you should not make up the results.

Please keep a running summary with analysis of previous function results and summaries from previous iterations.

Do not stop calling functions until the task has been accomplished or you've reached max iteration of 10.

Calling multiple functions at once can overload the system and increase cost so call one function at a time please.

If you plan to continue with analysis, always call another function.

For each function call return a valid json object (using double quotes) with function name and arguments within `<tool_call></tool_call>` XML tags as follows:

```
<tool_call>
  {"name": <function-name>, "arguments": <args-dict>}}
</tool_call>
```

3.2.2 Функції

Всі фінансові функції зробимо за допомогою `yfinance` (Yahoo Finance). Вона дає можливість робити запити та отримувати сьогоднішні новини, ціни акцій і так далі. Щодо функції пошуку в інтернеті:

Виконує пошук у Google за заданим запитом та витягує вміст із перших знайдених сторінок. Вона шукає текстову інформацію та дані таблиць з цих сторінок, щоб подати їх у структурованому вигляді.

Функція спочатку надсилає запит до пошукової системи, щоб отримати список посилань на найрелевантніші результати. Потім вона обробляє кожну з цих посилань паралельно, витягуючи текстові дані з параграфів та структуровані дані з таблиць на сторінці. Підсумком роботи є список із детальною інформацією, що включає URL, текстовий вміст та дані таблиць для кожної обробленої сторінки.

3.2.3 Діалог між функціями та великою мовною моделлю

Взаємодія між функціями додатку та великою мовною моделлю (LLM) базується на циклічному процесі, що забезпечує динамічне виконання функцій

та обробку їхніх результатів для користувача. Цей підхід можна формалізувати у вигляді наступного алгоритму:

а) ініціалізація виклику API: додаток формує запит до LLM, який містить текстовий запит користувача та специфікації доступних функцій (їхні назви, аргументи та інтерфейси);

б) рішення моделі: на основі отриманого запиту модель визначає, чи достатньо її власного текстового відповіді, чи потрібно викликати одну або декілька функцій для уточнення інформації чи виконання додаткових обчислень;

в) вказівка виклику функції: якщо функції необхідні, модель надсилає додатку вказівку з переліком функцій, які слід виконати, та відповідними аргументами;

г) виконання функцій: додаток локально виконує визначені функції з переданими аргументами, отримані результати зберігаються для подальшої обробки;

д) зворотній зв'язок із LLM: результати виконання функцій передаються назад до моделі разом із первинним запитом користувача для генерації остаточної текстової відповіді;

е) формування відповіді: модель інтегрує результати функцій у текстову відповідь, яка повертається користувачу.

Для підтримки безперервної взаємодії необхідно якось зберігати історію чату, яку ми подаємо великій мовній моделі. Це потребує визначення ролей у структурі даних чату. Llama 3.1 була натренована на роботу з трьома основними ролями: system (система), user (користувач) та assistant (помічник). Однак NousResearch версія моделі була донаведена на додаткову роль – tool. Це дозволяє розширити функціональність системи, додаючи підтримку інтеграції викликів функцій у структурі історії чату. Завдяки цьому система стає більш адаптивною та здатною враховувати як інструкції користувача, так і результати функцій, не порушуючи послідовності діалогу.

Базуючись на інструкціях нашого промпту, коли модель формує свою відповідь, вона може включати спеціальні маркери `<tool_call>` і `</tool_call>`. Ці

токени сигналізують про те, що вміст між ними містить інструкцію для виклику певної функції з вказаними аргументами. Цей вміст не призначений для відображення користувачу безпосередньо. Натомість, він передається на подальшу обробку.

Перед виконанням функції інструкція проходить етап валідації. Цей етап критично важливий для перевірки коректності виклику: правильність формату, відповідність аргументів очікуваному типу та іншим критеріям. Якщо валідація виявляє помилку, система генерує відповідь із роллю `tool`, яка повідомляє про проблему. Це повідомлення передається назад до моделі, яка отримує можливість виправити інструкцію або звернутися до користувача за уточненнями. У разі успішної валідації функція виконується відповідно до переданої інструкції. Результати її роботи формуються у вигляді відповіді з роллю `tool` і повертаються назад до LLM. Модель інтегрує ці результати у свою подальшу відповідь, яку, в кінцевому підсумку, отримує користувач. Такий підхід забезпечує прозорість взаємодії: користувач бачить лише релевантну інформацію, тоді як технічні деталі обробки залишаються «за кадром».

Ця архітектура дозволяє будувати гнучкі системи, які поєднують мовну інтелектуальність моделей з потужністю виконання спеціалізованих функцій. Таким чином, користувач отримує точні, вичерпні відповіді, навіть якщо їх отримання вимагає виконання складних алгоритмів чи обчислень.

3.2.4 Пошукова доповнена система генерації тексту

Великі мовні моделі мають властиві обмеження та іноді можуть неправильно інтерпретувати або плутати доступні функції. Щоб пом'якшити це, необхідно обмежити кількість функцій, які розглядаються в кожній відповіді. Точне налаштування цих моделей є потенційним рішенням для покращення їх продуктивності; однак цей підхід вимагає значних обчислювальних ресурсів.

Початковий крок передбачає кодування всіх описів функцій у векторні представлення за допомогою вбудованих моделей BGE M3 [14]. Потім ці вектори зберігаються у векторній базі даних, зокрема Milvus, яка розгортається у віртуальній підсистемі Linux, що працює в середовищі Windows. Згодом ранжування функцій виконується за допомогою метрики косинусної подібності. Цей підхід дозволяє порівнювати закодований запит користувача, перетворений у вектор ідентичної розмірності, із попередньо обчисленими векторами функцій.

Алгоритм працює наступним чином: спочатку запит користувача кодується у вектор за допомогою BGE M3. Використовуючи обчислювально ефективну операцію, система визначає три основні описи функцій, які найбільше відповідають запиту, на основі їх косинусної подібності. Потім ці вибрані функції включаються в промпт, що передається мовній моделі під кожен запит користувача.

3.2.5 Велика мовна модель

Тепер, коли ми завершили усі можливі підготовки, алгоритми та системи. Ми можемо завантажити за допомогою huggingface велику мовну модель, та її токенайзер.

Токенайзер – це компонент або алгоритм в системах обробки природної мови, який розбиває текст на менші частини, звані токенами. Токени можуть бути словами, частинами слів, символами або навіть спеціальними маркерами залежно від специфіки завдання та архітектури моделі.

Основна мета токенайзера – перетворити текст у формат, який може бути оброблений нейронною мережею. Наприклад, для мовної моделі, такої як GPT або Llama, токенайзер перетворює текст у числову послідовність, що представляє кожен токен. Ці числа згодом використовуються моделлю для обчислень.

Існують різні типи токенайзерів:

- а) токенізація за словами: текст розбивається на окремі слова;
- б) токенізація за символами: кожен символ тексту стає окремим токеном;
- в) субсловна токенізація: розбивка тексту на частини слів, наприклад, за допомогою алгоритмів Byte Pair Encoding (BPE) або WordPiece, цей підхід дозволяє ефективніше працювати з рідковживаними словами та різними мовними формами.

Токенайзери також додають спеціальні токени, наприклад <SOS> (start of sentence), <EOS> (end of sentence) або <PAD> (padding), які допомагають моделі правильно інтерпретувати текст.

Бібліотека Transformers дозволяє налаштовувати моделі під будь-які задачі, зокрема для генерації тексту, як у нашому випадку – за допомогою класу AutoModelForCausalLM. Перед запуском моделі кожен вхідний текст необхідно закодувати, використовуючи словник токенайзера, спеціально підготовлений для обраної моделі.

Щоб усі вхідні дані мали однакову розмірність, до них додаються пусті токени (padding). Це забезпечує коректну обробку послідовностей незалежно від їх початкової довжини. Після виконання цих кроків усе готово до тестування моделі.

Висновки за розділом 3

У результаті проведеної роботи реалізовано поєднання великої мовної моделі Llama 3.1 з різними спеціалізованими функціями, що виконуються поза межами самої моделі. Ключем до цього стала техніка квантизації QLoRa, яка поєднує подвійне зниження точності та формат Normal Float (NF4). Такий підхід надав змогу суттєво оптимізувати використання пам'яті на рівні GPU та водночас зберегти належний рівень продуктивності й точності моделі. Застосування QLoRa виявилось ефективним навіть за умов обмежених ресурсів, коли для об-

числень доступна лише одна NVIDIA GeForce RTX 3080 Ti Laptop GPU. Такий формат квантизації відкриває широкі можливості для використання великих мовних моделей у випадках, коли немає доступу до високопродуктивних обчислювальних кластерів або коштовних хмарних сервісів.

Поруч із методами квантизації значну роль у спрощенні процесу розробки відіграє ізоляція залежностей за допомогою віртуального середовища Python. Застосування менеджера пакетів Anaconda надало змогу зручно керувати бібліотеками й уникати конфліктів під час налаштування різних версій одних і тих самих інструментів. Критично важливими з-поміж них стали Bitsandbytes (що відповідає за специфічні методи зниження точності), Torch (для обчислень на графічному процесорі) та набори інструментів від Huggingface (зокрема Transformers), без яких завантаження й інтеграція моделі Llama 3.1 не були б можливими.

Щоб взаємодія з користувачем була максимально ефективною, було розроблено й відповідним чином структуровано промпт. У ньому передбачено декілька частин, де кожна має свою чітко визначену функцію. Розділи «роль» та «ціль» забезпечують загальний контекст і пояснюють моделі, якої мети бажано досягти. Розділ «інструменти» включає інформацію про функції, які можуть бути викликані модельними інструкціями: наприклад, це можуть бути виклики до бібліотеки `ufinance`, що дає доступ до фінансових даних, або виклики власних інструментів для пошуку в мережі Google.

Сама модель Llama 3.1 була налаштована таким чином, щоб могли реагувати на запити користувача та, за необхідності, направляти їх до додаткових функцій. Інтеграція цих функцій у структуру діалогу реалізується через ролі `system`, `user`, `assistant` та `tool`: перші дві ролі застосовуються відповідно для загальних системних інструкцій і питань користувача, третя – для відповідей моделі, а остання – для надсилання та отримання даних від зовнішніх функцій. Така чітка розмежованість уможливорює одночасне зберігання контексту спілкування й аналіз результатів викликаних функцій, аби модель могла послідовно враховувати як колишні повідомлення, так і вихідні дані від додатка.

4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

4.1 Експеримент для звичайного користування

В цьому випадку ми спробуємо використувати створеного асистента як звичайного чатбота, побудованого на базі великої мовної моделі. Нижче наведено приклад взаємодії, де AI – асистент, а U – користувач:

U – Hello, who are you?

AI – I am an AI agent designed to assist with various tasks by calling different tools and analyzing their results. I'm here to help you with any queries you might have. What specific task would you like me to perform?

U – What can you do, explain.

AI – As an AI agent, I can perform various tasks based on the available tools:

1. Perform a Google search and scrape the top search results for a given query.
2. Get the current stock price for a given symbol.
3. Retrieve company news and press releases for a given stock symbol.
4. Obtain a company's profile and overview for a given stock symbol.
5. Fetch fundamental data for a given stock symbol using the yfinance API.
6. Acquire financial statements for a given stock symbol.
7. Obtain key financial ratios for a given stock symbol.
8. Get analyst recommendations for a given stock symbol.
9. Retrieve dividend data for a given stock symbol.
10. Obtain technical indicators for a given stock symbol.

Please provide me with specific queries or tasks related to these functionalities, and I'll proceed accordingly.

U – Okay, can you say something like Shakespeare.

AI – Of course! Here's a line from Shakespeare:

«All the world's a stage, And all the men and women merely players.»

I hope this satisfies your request without needing any specific tools. If you'd like me to provide more context or analysis related to this quote, please let me know.

Бачимо що з англійською все добре, асистент розуміє команди, та відповідає структуровано на усі питання. Спробуємо все те ж саме, але українською. Ми можемо розмовляти не тільки українською, тому що великі мовні моделі майже завжди тренувались на якійсь частці інших мов, в нашому випадку, ми перевіримо українську.

U – Привіт, хто ти?

AI – Я є AI агентом, створений для виконання завдань з аналізу даних і надання інформації на основі функцій, які мені надходять. Я можу виконувати пошуки інформації за допомогою пошукових запитів, аналізувати акціонерний ринок та фінансові дані компаній, а також витягати ключові дані зі статей новин. Я можу відповідати на ваші запитання та допомагати вам у пошуку інформації. Напиши ваш запит, і я спробую допомогти.

U – Що вмієш, поясни.

AI – I am able to perform various tasks related to finance, such as:

1. Perform a Google search and scrape the top search results.
2. Get the current stock price for a given symbol.
3. Retrieve company news and press releases for a given stock symbol.
4. Obtain company profile and overview for a given stock symbol.
5. Fetch fundamental data for a given stock symbol.
6. Get financial statements for a given stock symbol.
7. Obtain key financial ratios for a given stock symbol.
8. Get analyst recommendations for a given stock symbol.
9. Retrieve dividend data for a given stock symbol.
10. Obtain technical indicators for a given stock symbol.

To begin, please provide me with a specific task or stock symbol to work with.

U – Гарзд, можеш сказати щось як Шекспір.

AI – The quote from William Shakespeare is:

«Ваше серце знає правду, коли ви говорите з ним.» – Вільям Шекспір

Summary: I have provided a quote from William Shakespeare as requested. It says, «Your heart knows the truth when you speak to it.» – William Shakespeare.

Проведене тестування демонструє, що асистент добре справляється з обробкою запитів англійською мовою. Відповіді структуровані, змістовні та повністю відповідають поставленим завданням. Асистент вільно переходить між різними темами, зокрема цитує літературні джерела, такі як твори Вільяма Шекспіра, що свідчить про високий рівень розуміння тексту та здатність адаптуватися до різних типів запитів.

Українська мова також розпізнається асистентом, і загалом він здатний підтримувати діалог, однак спостерігається певна різниця у якості відповідей. Іноді відповіді українською менш структуровані, а в окремих випадках асистент переходить на англійську, що може свідчити про неповну адаптацію до української мови в контексті специфічних завдань. Попри це, прості запити українською виконуються зрозуміло і логічно.

Загальний рівень роботи асистента виявився високим, проте при використанні української мови можуть виникати нюанси, особливо у складних технічних завданнях. Це свідчить про необхідність додаткової оптимізації для рівноцінної підтримки різних мов. Тим не менш, асистент підтвердив свою ефективність як інструмент для вирішення різноманітних завдань, особливо при роботі англійською мовою.

4.2 Експеримент для користування структурованим виходом

Тепер, спробуємо користуватись функціями, які ми написали і вони виконуються автоматично, якщо асистент видає їх. Нагадаємо, наш асистент може виконувати багато функцій послідовно, аналізуючи їх результати. Тому, для цього конкретного експерименту ми поставимо максимально 5 ітерацій (відповідей), які асистент може вивести заради відповіді на одне питання. Також, пам'ятаємо, що наш асистент орієнтований на фінанси та може шукати в інтернеті, тому наші питання будуть орієнтовані на це. Далі AI – асистент, U – користувач, FR – відповідь від виконаної функції (будь-які дані переведені в тексто-

вий формат).

U – hi, what is current price for Tesla.

AI – <tool_call>

```
{"name": "get_current_stock_price", "arguments": {"symbol": "TSLA"}}
```

</tool_call>

FR – Here’s the response from the function call: get_current_stock_price

```
{"name": "get_current_stock_price", "content": 431.66}
```

AI – The current stock price for Tesla (TSLA) is \$431.66.

Експеримент продемонстрував, що асистент ефективно справляється із запитамі, пов’язаними з отриманням актуальної інформації, наприклад, про ціни акцій. У даному випадку користувач запитав про поточну ціну акцій Tesla. Асистент правильно ініціював виклик відповідного інструменту для отримання актуальних даних, а після обробки результатів надав чітку і зрозумілу відповідь.

Цей приклад підтверджує, що асистент добре інтегрує інструменти у свої відповіді, що дозволяє йому надавати точну інформацію у реальному часі. Відповідь була короткою, структурованою та відповідала запиту. Це свідчить про високий рівень адаптації до завдань такого типу.

Спробуємо тепер все теж саме, але українською.

U – Привіт, яка зараз ціна на Tesla.

AI – <tool_call>

```
{"name": "get_current_stock_price", "arguments": {"symbol": "TSLA"}}
```

</tool_call>

FR – {"name": "get_current_stock_price", "content": 431.66}.

AI – Починаючи з поточного рівня ринку, вартість акцій компанії Tesla (TSLA) становить приблизно 431.66 доларів США. Ціна може варіюватися в залежності від різних факторів, таких як новини ринку та загальна економічна ситуація.

Тестування показує, що асистент коректно розпізнає запити українською мовою і використовує відповідні інструменти для отримання актуальної інформації.

ції. У цьому випадку він зміг точно відповісти на запит користувача про поточну ціну акцій Tesla, залучивши необхідний функціонал для отримання даних.

Однак відповідь українською містить додаткові деталі, яких не було у відповіді англійською, зокрема зауваження про можливість варіювання ціни залежно від ринкових факторів. Це може свідчити про спробу адаптувати відповідь до стилістики мови або бажання надати більше контексту. Водночас така розширена відповідь може не завжди відповідати очікуванням користувача, якщо він хоче отримати лише коротку і точну інформацію.

Висновки за розділом 4

Проведене тестування демонструє, що асистент ефективно функціонує як багатомовний інструмент, здатний обробляти запити англійською та українською мовами. Його робота виявляється висококваліфікованою у більшості випадків, особливо англійською мовою, де відповіді структуровані, чіткі й максимально адаптовані до запиту. Асистент демонструє здатність залучати зовнішні інструменти для отримання даних і правильно інтегрує їх у свої відповіді.

Українська мова розпізнається й обробляється належним чином, хоча спостерігається певна різниця в якості та стилі відповідей. В окремих випадках відповіді українською можуть бути менш структурованими або містити додаткові пояснення, які не завжди потрібні користувачеві. Це може бути наслідком меншого обсягу навчальних даних для української мови або відмінностей у мовних конструкціях.

Асистент демонструє сильні аналітичні здібності, здатність до логічних висновків і вміння відповідати на запити в контексті різних завдань: від простих діалогів до спеціалізованих запитів, наприклад, фінансових. Однак для забезпечення повної рівноцінності роботи різними мовами варто зосередитися на оптимізації відповідей українською, щоб уникати зміни стилю або зайвого розширення відповідей.

Загалом, асистент підтвердив свою ефективність як інструмент для вирішення завдань, демонструючи високу функціональність і адаптивність. Подальше вдосконалення здатності до роботи українською мовою може значно підвищити його універсальність і зручність використання.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи розроблено систему, здатну забезпечувати виведення структурованої інформації та виконувати призначені користувачем завдання за допомогою запитів природною мовою. Помічник штучного інтелекту включає в себе передові методології, включаючи великі мовні моделі, контекстні вбудовування та системи генерації з доповненим пошуком, які разом дозволяють надавати користувачам точну та контекстно релевантну інформацію. Ці досягнення тісно пов'язані з сучасними тенденціями в області штучного інтелекту, особливо в областях обробки природної мови та аналізу даних.

Результати проєкту свідчать про його відповідність сучасному рівню науково-технічного прогресу. Завдяки використанню точно налаштованих моделей, таких як Llama 3.1, і доданню в них можливостей розширеного пошуку, система відповідає найвищим світовим стандартам розробки ШІ. Помічник не тільки ефективно обробляє інформацію, але й виводить її в структурованих форматах, забезпечуючи її зручність у різних професійних областях. Ці структуровані результати особливо цінні в процесах прийняття рішень, де чітке та стисле представлення даних має першочергове значення. Крім того, здатність системи адаптувати свої відповіді як до англійської, так і до української мов покращує її доступність і зручність використання в багатомовному контексті, хоча деякі відмінності в якості відповідей українською підкреслюють можливості для подальшого вдосконалення.

Розроблений помічник має значний потенціал для розгортання, зокрема у фінансовому секторі. Він призначений для отримання фінансових даних у режимі реального часу, аналізу показників акцій та інтеграції з пошуком в Інтернеті, щоб надати найновішу інформацію. Ця функція робить його потужним інструментом для фінансових професіоналів, що дозволяє швидше та більш обґрунтовано приймати рішення. Окрім фінансів, дизайн і методологія системи пропонують багатообіцяючі застосування в інших сферах, таких як охорона

здоров'я, юридичні консультації та підтримка клієнтів, де структуровані та контекстно релевантні вихідні дані є важливими.

Наукове та соціально-економічне значення цієї роботи є величезним. З наукової точки зору проєкт просуває інтеграцію технологій штучного інтелекту, поєднуючи структуровані комунікаційні можливості з надійним функціональним виконанням. Це являє собою значний крок вперед у розробці інтелектуальних систем, призначених для виконання предметно-спеціальних завдань з високою точністю. В економічному та соціальному плані помічник має потенціал революціонізувати робочі процеси, зменшивши ручні зусилля, підвищивши ефективність і підвищивши точність рішень у критичних секторах.

Заглядаючи наперед, подальші дослідження та розробки в цій галузі є здійсненими та бажаними. Однією з ключових сфер уваги має бути покращення багатомовних можливостей помічника, особливо для української та інших недостатньо представлених мов. Це забезпечить однакову якість і зручність для різних користувачів. Іншим перспективним напрямком є інтеграція додаткових структурованих вихідних форматів і розширення функціональних можливостей системи для задоволення більш широкого діапазону предметно-специфічних вимог. Крім того, вивчення методів тонкого налаштування моделі в режимі реального часу на основі відгуків користувачів підвищить адаптивність і швидкість реагування системи, забезпечуючи її актуальність і ефективність у динамічних середовищах.

Таким чином, розроблений помічник ШІ є значним досягненням у сфері інтелектуальних систем. Він поєднує в собі найсучасніші технології штучного інтелекту з практичними застосуваннями, демонструючи свій потенціал стати незамінним інструментом у різних галузях. Ця робота забезпечує міцну основу для майбутніх досягнень, забезпечуючи постійний внесок як у наукову спільноту, так і в практичні сфери застосування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention Is All You Need. Arxiv. URL: <https://arxiv.org/abs/1706.03762> (дата звернення: 06.10.2024).
2. Schmidt R. M. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. Arxiv. URL: <https://arxiv.org/abs/1912.05911> (дата звернення: 06.10.2024).
3. Vennerød C.B., Kjærø A., Bugge E.S. Long Short-term Memory RNN. URL: <https://arxiv.org/abs/2105.06756> (дата звернення: 06.10.2024).
4. Parthasarathy V.B., Zafar A., Khan A., Shahid A. The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities. Arxiv. URL: <https://www.arxiv.org/abs/2408.13296> (дата звернення: 06.10.2024).
5. Sun L., Huang Y., Wang H., Wu S. [та ін.]. TrustLLM: Trustworthiness in Large Language Models. URL: <https://arxiv.org/abs/2401.05561v2> (дата звернення: 06.10.2024).
6. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. URL: <https://arxiv.org/abs/1301.3781> (дата звернення: 06.10.2024).
7. Pennington J., Socher R., Manning C.D. GloVe: Global Vectors for Word Representation. URL: <https://nlp.stanford.edu/pubs/glove.pdf> (дата звернення: 06.10.2024).
8. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. URL: <https://arxiv.org/abs/1810.04805> (дата звернення: 06.10.2024).
9. Zhang Z., Han X., Liu Z., Jiang X., Sun M., Liu Q. ERNIE: Enhanced Language Representation with Informative Entities. URL: <https://arxiv.org/abs/1905.07129> (дата звернення: 06.10.2024).

10. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. URL: <https://arxiv.org/abs/1908.10084> (дата звернення: 06.10.2024).
11. Gao Y., Xiong Y., Gao X., Jia K. [та ін.]. Retrieval-Augmented Generation for Large Language Models: A Survey. URL: <https://arxiv.org/abs/2312.10997> (дата звернення: 06.10.2024).
12. Dubey A., Jauhri A., Pandey A., Kadian A. [та ін.]. The Llama 3 Herd of Models. URL: <https://arxiv.org/abs/2407.21783> (дата звернення: 06.10.2024).
13. HuggingFace. URL: <https://huggingface.co/NousResearch/Hermes-3-Ллама-3.1-8B> (дата звернення: 06.10.2024).
14. HuggingFace. URL: <https://huggingface.co/BAAI/bge-m3> (дата звернення: 28.11.2024).
15. Bamoria H. QLoRA: Efficient Finetuning of Quantized LLMs. URL: <https://arxiv.org/abs/2305.14314> (дата звернення: 06.10.2024).