Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження моделі інтеграції соціально-зорієнтованих інформаційних моделей штучного інтелекту для сервісів роботи з клієнтами
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-1

Кіпріч І.С.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник проф. Смеляков К.С
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Кіпріч Іван Сергійович
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження моделі інтеграції соціально-зорієнтованих інформаційних моделей штучного інтелекту для сервісів роботи з клієнтами» Затверджена наказом по університету від 29.03. 2024р. № 250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14.06.2024

3. Вихідні дані до роботи опис досліджуваних великих мовних моделей, вимоги до розробки схеми бази даних для проведення досліджень за обраною предметною областю, мови програмування Python, технології fine-tuning, середовища розробки PyCharm

4. Перелік питань, що потрібно опрацювати в роботі

Аркуш завдання, аналіз предметної галузі, великі мовні моделі, методи постнавчання, архітектура інтеграції, серверна та клієнтська частини програмної системи, фрагменти коду, графічні матеріали.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	01.02 – 20.02.24	*виконано*
2	Аналіз та вибір моделей для дослідження	15.02 – 14.03.24	*виконано*
3	Аналіз та моделювання предметної області	17.02 – 28.03.24	*виконано*
4	Планування експериментів	25.02 – 28.04.24	*виконано*
5	Програмна реалізація інтеграції API	25.02 – 01.05.24	*виконано*
6	Експериментальні дослідження	02.04 – 15.05.24	*виконано*
7	Аналіз результатів експериментальних досліджень та розробка рекомендацій	20.04 – 20.05.24	*виконано*
8	Написання та оформлення статті та тез доповіді	17.04 – 23.05.24	*виконано*
9	Підготовка пояснювальної записки	01.04 – 26.05.24	*виконано*
10	Підготовка презентації та доповіді	26.05 – 2.06.24	*виконано*
11	Нормоконтроль	3.05 – 08.06.24	*виконано*
12	Рецензування	08.05 – 14.06.24	*виконано*
13	Занесення диплома в електронний архів	15.06.2024	*виконано*
14	Попередній захист	15.06.2024	*виконано*
15	Допуск до захисту у зав. кафедри	21.06.2024	*виконано*

Дата видачі завдання 01 лютого 2024р.

Студент (ка) _____ Кіпріч. І. С.
(підпис)

Керівник роботи _____ проф. Смеляков К.С.
(підпис) **(посада, прізвище, ініціали)**

РЕФЕРАТ / ABSTRACT

Пояснювальна записка: 53 с., 7 рис., 7 табл., 17 джерел.

ВЕБ-СИСТЕМА, МЕТРИКИ, AI, LLM, PYTHON, POSTGRESQL

Об’єкт дослідження – інтеграція соціально-зорієнтованих моделей підтримки прийняття рішень у системах обробки клієнтських звернень.

Мета розробки – провести аналіз методів навчання великих мовних моделей, підготувати навчальні дані та провести навчання моделі, інтегрувати модель в систему обробки клієнтських звернень.

Метод рішення – методи навчання мовних моделей, середовище розробки PyCharm, мови програмування Python, СУБД PostgreSql, контейнеризація Docker.

В результаті роботи навчена модель на власному наборі даних та інтегрована в систему обробки клієнтських запитів.

WEB-SYSTEM, METRICS, LLM, PYTHON, POSTGRESQL, AI.

The object of the research is a software system for integration artificial Intelligence models into customer support systems.

The purpose of the research is to analyze LLM training methods, prepare dataset, fine-tune the model and integrate into customer support system.

The solution method is PyCharm (development environment), Python, DBMS PostgreSql, Docker platform.

As a result of the development, a web system that integrated to customer support system.

Заява щодо самостійного виконання кваліфікаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElArKhNURE.

Я, Кіпріч Іван Сергійович, студент(ка) гр. ІПЗм-22-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя моя кваліфікаційна робота на тему «Дослідження моделі інтеграції соціально-зорієнтованих інформаційних моделей штучного інтелекту для сервісів роботи з клієнтами», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(на) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

Перелік скорочень.....7
Вступ.....8
1 Аналіз предметної галузі.....10
1.1 Аналіз предметної галузі дослідження.....10
1.2 Великі мовні моделі (LLM).....12
1.3 Постановка задачі.....17
2 Методи налаштування мовних моделей.....19
2.1 Метод налаштування LoRA.....19
2.2 Метод налаштування Q-LoRA.....21
3 Огляд метрик.....24
3.1 Параметри постнавчання.....24
3.2 Збір результатів та їх оцінка.....28
4 Практичне дослідження.....32
4.1 Підготовка даних.....32
4.2 Постнавчання моделей.....33
4.3 Збір та аналіз результатів.....35
Висновки.....40
Перелік джерел посилання.....42
Додаток А Перелік джерел посилання за науковими напрямами керівника та науковців кафедри Програмної Інженерії.....44
Додаток Б Звіт результатами перевірки на унікальність тексту в базі ХНУРЕ.....45
Додаток В.....46
ДОДАТОК Г. Презентаційний матеріал до кваліфікаційної роботи.....49

ПЕРЕЛІК СКОРОЧЕНЬ

LLM – Large Language Model

RNN – Recurrent Neural Networks

CNN – Convolutional neural network

NLP – Natural Language Processing

GPT – Generative Pre-trained Transformer

BERT – Bidirectional Encoder Representations from Transformers

LLaMa – Large Language Model Meta AI

GPC – Google Cloud Platform

PEFT – Parameter Efficient Fine Tuning

LoRA – Low Rank Adaptation

Q-LoRA – Quantized Low Rank Adaptation

API – Application Programming Interface

ВСТУП

Під час активного розвитку сфери надання послуг важливе значення набуває клієнто-орієнтованість. Отримання коректних, точних та вчасних відповідей на запити клієнтів стають одним з ключових елементів що сприяють зростанню бізнесу. Системи підтримки, віртуальні асистенти, тематичні бази знань активно використовуються та виконують декілько важливих функцій.

Системи підтримки необхідні для запису та відстеження звернень клієнтів, управління пріоритетами, зберігання історії та обробки та розподілу звернень. Під час обробки звернень для оператора постає задача класифікувати звернення, та, за можливості, надати клієнту інформацію по темі звернення.

Тематичні бази знань (Knowledge Base) надають користувачам можливість знаходити інформацію за допомогою систем навігації і пошуку, а також отримувати детальні пояснення і ілюстрації. Але пошук відповідей може бути складним для великих обсягів інформації або у випадках коли у користувача недостатньо знань в предметній галузі.

Віртуальні асистенти дозволяють створити автоматизовані сценарії для отримання відповідей на запити користувача, надавати персоналізовані рекомендації та допомагати з навігацією в базах знань, для пошуку відповідей. Опрацювання великого потоку клієнтських звернень вимагає значних людських ресурсів, що може бути фінансово недоцільно для кампанії. Для наявних баз знань в певній галузі можливо адаптувати існуючі мовні моделі систем штучного інтелекту та інтегрувати для використання в системах підтримки або віртуальних ассистентах.

Метою роботи є оцінка можливості використання постнавчених великих мовних моделей в системах підтримки користувачів, та аналіз методів постнавчаня. Задачами роботи є дослідження проблем галузі, опис ідеї, підготовка навчальних даних, тренування (налаштування) моделі, проектування та розробка інтеграції моделі в системи обробки клієнтських звернень, розробка метрик для оцінки отриманих результатів та аналіз статистичних даних після впровадження системи.

Для дослідження використовуються великі мовні моделі попередньо навчені для генерації тексту без спеціалізації в певній предметній галузі. Предметом дослідження є можливість адаптації (постнавчання) мовних моделей на тематичному наборі даних і застосування в системах підтримки клієнтів.

Впровадження постнавчених великих мовних моделей в системах підтримки користувачів зменшує складність обробки клієнтських запитів, та зменшує час відповіді, що в свою чергу підвищує рівень задоволення користувачів і лояльність до продукту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі дослідження

Системи підтримки і обробки клієнтських звернень відіграють важливу роль в життєвому циклі продукту та допомагають формувати і збільшувати рівень лояльності клієнтів до продукту [1]. Основними функціями систем підтримки користувачів є

– запис та відстеження звернень. Системи підтримки дозволяють користувачам створювати та відстежувати звернення, пов'язані з питаннями чи проблемами. Кожне звернення отримує унікальний номер (тікет), що полегшує його подальше відстеження;
– управління пріоритетами. Дозволяють призначити пріоритети різним зверненням в залежності від їхньої важливості та терміновості вирішення;
– зберігання історії звернень та вирішень питань, що дозволяє вести аналітику та вдосконалювати процеси підтримки.
– надання інформації та посилань на навчальні статті, документацію, онлайн-ресурси;
– навчання та формування документації на прикладах клієнтських звернень та вирішених питань;
– аналіз та звітність, для виявлення патернів та вдосконалення продукту чи сервісу.

Зазвичай клієнтські звернення проходять кілька етапів від постановки проблеми до її вирішення чи отримання відповіді. Загальні етапи життєвого циклу звернення це:

– постановка питання або проблеми та збір інформації;
– створення запису про звернення та реєстрація питання або проблеми в системі підтримки;
– розподіл серед відповідальних співробітників на основі типу питання, пріоритету та наявних ресурсів;

– аналіз та пошук оптимального рішення для питання чи проблеми клієнта відповідними співробітниками;

– надання відповіді клієнту разом зі зрозумілим поясненням та/або з посиланням на додаткові інформаційні ресурси;

– зворотній зв’язок від клієнта і закриття звернення, дані зберігаються в системі для подальшого аналізу.

Клієнтські звернення можуть бути різного типу, і обробка залежить від конкретного характеру питань.

Інформаційні запити: питання про продукт чи послугу, коли клієнти потребують інформацію про функції, характеристики та методи використання продукту чи послуги. У відповідь співробітники надають інформацію, з посиланням на документацію або додаткові інформаційні ресурси

Технічна підтримка, коли клієнти мають проблеми за продуктом, там потребують технічної допомоги. Звернення обробляється технічними спеціалістами, виправляються помилки в роботі продукта та надаються консультації.

Позитивні і негативні відгуки, клієнти висловлюють свої скарги або подяки щодо продукту, послуги. Співробітники компанії аналізують та вирішують скарги, а також аналізують можливі сценарії поліпшення та вдосконалення сервісу. Позитивні відгуки можуть бути використані для підвищення мотивації співробітників, та покращення позитивного іміджу компанії.

Під час обробки інформаційних запитів можуть виникати різні складнощі пов’язані з нечітким формулюванням питання, недоступністю інформації, великим обсягом бази знань, релевантністю відповіді, приватністю та безпекою даних. Пошук відповідної інформації в базі знань, та підготовка відповіді співробітником може займати багато часу і відповідно фінансових ресурсів компанії. Використання великих мовних моделей (Large Language Model LLM) можуть скоротити час обробки клієнтських запитів, та надати відповіді побудовані на основі бази знань компанії. LLM можуть ефективно функціонувати як частина рекомендаційних систем, підвищуючи їх ефективність у сценаріях з

обмеженими даними [2], використовуючи доналаштовані моделі в специфічній сфері знань [3]. Великі мовні моделі можуть бути ефективно інтегровані та використані для інноваційного менеджменту [4]

1.2 Великі мовні моделі (LLM)

Великі мовні моделі (Large Language Model LLM) це тип мовної моделі, який використовує методи глибокого навчання, має велику кількість параметрів, навчених на великій кількості немаркованого тексту за допомогою самокерованого або напівкерованого навчання і здатний генерувати та розуміти природну мову на високому рівні. LLMs працюють на основі процесу навчання, коли в модель подається велика кількість текстових даних. Ці дані можна отримати з різних джерел, таких як книги, статті, документи, веб-сайти та навіть соціальні мережі. Потім модель навчається передбачати наступне слово або наступне речення на основі контексту, у якому воно знаходиться. LLM можуть виконувати різноманітні завдання, такі як сумаризація текстів, переклад мов, відповіді на питання та навіть створення повних текстів. Ці моделі особливо корисні в програмах обробки природної мови, таких як чат-боти та віртуальні помічники, які повинні розуміти та ефективно генерувати людську мову.

Існують різні типи великих мовних моделей (LLM), кожна з яких має свої особливості та застосування. Деякі з найпоширеніших типів:

– моделі на основі трансформера: Ці моделі базуються на архітектурі Transformer, яка є технікою машинного навчання, яка дозволяє моделі обробляти вхідні послідовності паралельно. Ці моделі відомі своєю здатністю фіксувати довготривалі зв’язки між словами та реченнями, що робить їх ідеальними для таких завдань, як машинний переклад і резюмування тексту;

– моделі на основі рекурентної нейронної мережі (RNN).: Ці моделі використовують архітектуру рекурентної нейронної мережі для обробки вхідних послідовностей. Вони особливо корисні для завдань, які

включають інтелектуальний текст, наприклад автозаповнення речень у реальному часі;

– моделі на основі згорткової нейронної мережі (CNN).: ці моделі використовують архітектуру згорткової нейронної мережі для обробки вхідних послідовностей. Вони часто використовуються для завдань класифікації тексту, як-от аналіз настроїв. Смеляков [5] запропонував оцінку ефективності деяких типів згорткових нейроних мереж. В сфері пошуку зображень та розпізнавання в 2020 запропонував інноваційний підхід який частково може бути застосовано для обробки текстів [6].

Архітектура трансформера являє собою нейромережеву модель для задач обробки природної мови, що базується на кодер-декодер архітектурі яка була представлена у статті "Attention Is All You Need" Васвані та ін. у 2017 році [7]. Модель трансформер вирізняється використанням механізмів самоуваги, що дозволяє їй обробляти вхідні дані паралельно та вловлювати складні залежності в даних, значно покращуючи ефективність та результативність у широкому діапазоні завдань. Основні концепції трансформер включають токенізацію, вбудовування, увагу або самоувагу [8], попереднє навчання та трансферне навчання. У контексті LLM, трансферне навчання передбачає додаткове налаштування попередньо навченої моделі на меншому, специфічному для завдання наборі даних для досягнення високої продуктивності у вирішенні нового завдання. Перевага трансферного навчання полягає у здатності використовувати велику кількість загальних знань про мову, отриманих під час попереднього навчання, тим самим знижуючи потребу у великих анотованих наборах даних для кожного нового завдання. Оригінальна модель трансформера також використовує архітектуру кодера-декодера. Кодер складається з шарів кодування, які обробляють вхідні дані ітеративно, один шар за іншим. Декодер складається з шарів декодування, які роблять те саме з виходом кодера.

Функція кожного рівня кодувальника полягає у створенні кодувань, які містять інформацію про те, які частини вхідних даних є релевантними одна

одній . Вихідні кодування потім передаються наступному кодеру як його вхід. Кожен кодер складається з механізму самоконтролю та прямої нейронної мережі.

Крім того, кожен рівень декодера приймає всі кодування та використовує їх включену контекстну інформацію для створення вихідної послідовності. Подібно до кодерів, кожен декодер складається з механізму самоконтролю, механізму уваги до кодувань і нейронної мережі прямого зв’язку [9].

Основні частини, які дозволяють обробляти і розуміти дані, з яких складаються великі мовні моделі зображені на рисунку 1.1. Розглянемо їх детальніше

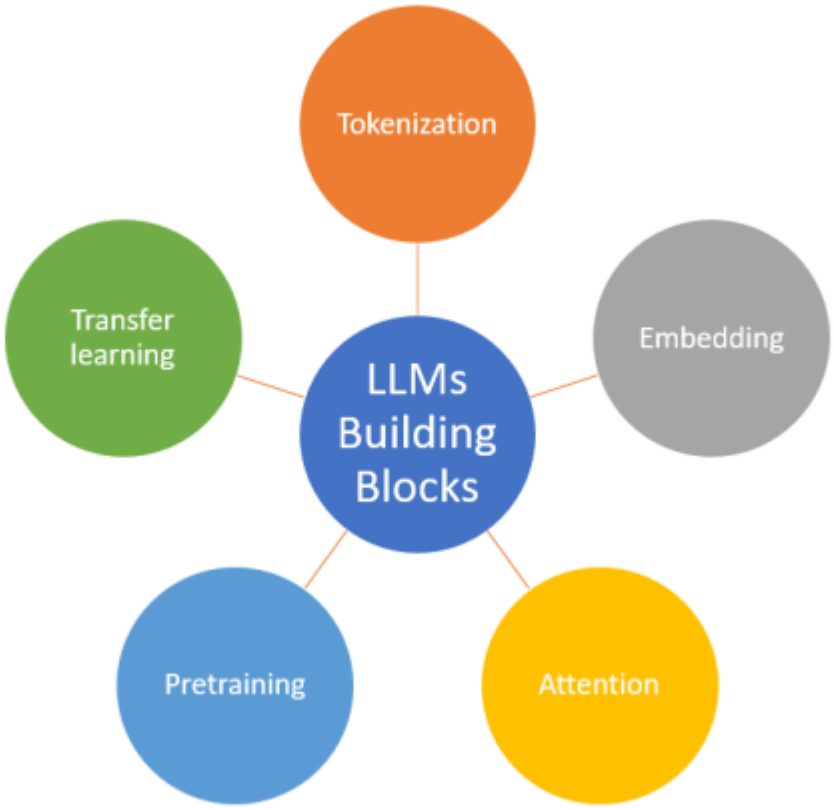

Рисунок 1.1 - Основні частини LLM [9]

Tokenization (токенізація) - процес перетворення послідовності тексту в слова, склади або токени, які модель може зрозуміти. В LLM токенізація зазвичай виконується з використанням алгоритма підслів, таких як Byte Pair Encoding (BPE) чи WordPiece, які розбивають текст на менші частини, що охоплюють строки які потрапляють часто та рідко. Цей підхід дозволяє обмежити розмір

словника моделі та зберегти можливість представити будь яку текстову послідовність [9].

Embedding. Вкладення це безперервне векторне представлення слів чи токенів яке включає семантичне значення в багатовимірному просторі. Воно дозволяє представляти токени в форматі якій може бути оброблений нейронною мережею. В LLM вкладення навчається підчас тренувального процесу, та результуюче векторне відображення може зберігати складні зв’язки між словами, такі як аналогії чи синоніми [9].

Attention. Увага або самоувага в LLM механізм який використовується в трансформерах та дозволяє призначити важливість (вагу) різним словам чи фразам в залежності від контексту. Призначаючи різну вагу токенам в вхідному тексті модель може сфокусуватись на найбільш релевантній інформації та ігнорувати менш важливі деталі. Можливість вибірково фокусуватись на певних частинах вхідних даних має вирішальне значення для виявлення довгострокових залежностей і розуміння нюансів природньої мови [9].

Pre-training. Попередня підготовка - це процес підготовки LLM на великому наборі даних, як правило, без нагляду або під власним наглядом, перед тим, як налаштувати його для виконання конкретного завдання. Під час попередньої підготовки модель вивчає загальні мовні шаблони, зв'язки між словами та інші фундаментальні знання. Результатом цього процесу є попередньо навчена модель, яку можна точно налаштувати, використовуючи менший набір даних для конкретного завдання, що значно зменшує кількість маркованих даних і час навчання, необхідний для досягнення високої продуктивності при виконанні різних завдань NLP [9]

Transfer learning Передавальне навчання це техніка передачі знань отриманих на попередньому етапі та застосування їх для нової, пов’язаної задачі. В контексті LLM передавальне навчання залучає тонкі налаштування попередньо навченої моделі на меншому специфічному для задачі наборі даних, для отримання високої швидкодії в вирішені нової задачі. Перевага передавального навчання полягає в тому що воно дозволяє використовувати великий обсяг

загальних мовних знань, отриманих на етапі попереднього навчання, та зменшенні необхідності великих маркованих наборів даних для кожної нової задачі [9].

Розглянемо різновиди моделей та їх основні характеристики.

На сьогодні існує багато різних LLM створених різними компаніями які можна використовувати в системах підтримки.

GPT (Generative Pre-trained Transformer) розроблена в OpenAI трансформерна архітектура, 3 версія має 175 мільярдів параметрів, 4 версія має 1.76 трильйона параметрів. Платна, доступна для використання та постнавчання тільки через API.

BERT (Bidirectional Encoder Representations from Transformers) від Google. Є декілько конфігурацій моделі, наприклад BERT-base має близько 110 мільйонів параметрів. Використовується для класифікації текстів та оцінки подібності текстів.

DistilBERT від Hugging Face це спрощена і швидша версія BERT, яка зберігає більшу частину точності BERT, але використовує менше обчислювальних ресурсів. Застосовується для задач класифікації тексту, де важлива швидкість обробки і обмежені обчислювальні ресурси.

Large Language Model Meta AI (LLaMa 2) Еволюція оригінальної моделі LLaMa, розроблена для покращення можливостей обробки природної мови. Хоча деталі про LLaMa-2 можуть бути обмеженими без останніх оновлень, серія LLaMa відома своєю ефективністю та здатністю виконувати широкий спектр мовних завдань, використовуючи менше обчислювальної потужності порівняно з деякими іншими моделями. Llama 2 є набором попередньо навчених і додатково налаштованих генеративних текстових моделей, які варіюються від 7 мільярдів до 70 мільярдів параметрів. Ліцензована для використання з відкритим вихідним кодом [10]

GPT-J-6B є великою мовною моделлю з відкритим кодом, розробленим EleutherAI у 2021 році. Як вказує назва, це генеративна модель трансформера, попередньо навчена для генерації тексту, подібного до людського, який

продовжується з підказки. Необов'язковий "6B" у назві вказує на те, що вона має 6 мільярдів параметрів.

GPT-NeoX Відкрита модель з параметрами до 20 мільярдів, створена як альтернатива GPT-3. Підтримує складні задачі генерації тексту та розуміння контексту.

OPT від Meta AI це відкрита мовна модель з параметрами до 175 мільярдів, створена як альтернатива GPT-3. Підтримує складні задачі генерації тексту та розуміння контексту.

Ці моделі здатні генерувати текст наближений до натурального (людського) але не мають достатньо знань для відповідей на питання з вузької предметної галузі.

1.3 Постановка задачі

Проаналізувавши результати дослідження предметної галузі можна зробити висновок що підготовка та інтеграція великих мовних моделей в системи підтримки і обробки клієнтських звернень є актуальною проблемою.

Проаналізувавши доступні великі мовні моделі та можливості постнавчання з використанням власної бази знань була обрани моделі LLaMa-2-7B та GPT-J-6B для інтеграції в систему обробки клієнтських звернень. На основі висновків що наведені вище можна сформулювати наступну задачу: інтегрувати додатково навчену велику мовну модель в систему обробки клієнтських звернень та оцінити відповіді отримані з допомогою цієї моделі. Для цього необхідно:

– дослідити методи налаштування великих мовних моделей;
– підготувати дані для постнавчання, експортувати існуючу базу знань, та історію запитів в систему обробки клієнтських запитів;
– очистити та відформатувати дані;
– провести постнавчання великої мовної моделі з використанням PyTorch або Tensorflow;
– реалізувати сервіс для інтеграції навченої моделі в систему обробки клієнтських запитів;

– зібрати та систематизувати результати застосування великої мовної моделі під час пошуку та підготовки відповідей на клієнтські запити;

– зробити висновки на підставі проведених досліджень.

Проаналізувавши обмеження які існують в роботі з різними моделями для постнавчання було обрано моделі LlaMA-2-7B та GPT-J-6B. Ці моделі доступні для завантаження, щоб проводити експерименти в локальному середовищі, а також мають декілько варіантів розгортання на популярних хмарних середовищах, таких як Amazon Web Service, Cloudflare, Google Cloud Platform (GCP) - Model Garden, Microsoft Azure.

2 МЕТОДИ НАЛАШТУВАННЯ МОВНИХ МОДЕЛЕЙ

Повне налаштування параметрів (Full parameter fine-tuning) це метод який дозволяє налаштувати всі параметри всіх шарів попередньо навченої моделі. Загалом цей метод може досягти найкращої продуктивності, але він також найбільш витратний за ресурсами та часом він потребує найбільше обчислювальних ресурсів та займає найбільше часу.

PEFT, або Parameter Efficient Fine Tuning (Ефективне налаштування параметрів), дозволяє налаштовувати моделі з мінімальними ресурсами та витратами. Існує два основних метода налаштування параметрів LoRA (Low Rank Adaptation) and QLoRA (Quantized LoRA).

2.1 Метод налаштування LoRA

LoRA, що означає Low-Rank Adaptation [11] of Large Language Models, базується на важливому розумінні: різниця між точно налаштованими ваговими коефіцієнтами для спеціалізованого завдання та початковими попередньо навченими ваговими коефіцієнтами часто демонструє «низький внутрішній ранг» — це означає, що можна добре апроксимувати матрицею низького рангу. Що означає низький ранг матриці? Матриця низького рангу має кілька лінійно незалежних стовпців, що означає, простими словами, що матриця менш «складна». Однією цікавою властивістю матриць низького рангу є те, що вони можуть бути представлені як добуток двох менших матриць. Це усвідомлення призводить до гіпотези, що ця дельта між точно налаштованими ваговими коефіцієнтами та початковими попередньо навченими ваговими коефіцієнтами може бути представлена як матричний добуток двох набагато менших матриць. Зосереджуючись на оновленні цих двох менших матриць, а не всієї початкової вагової матриці, ефективність обчислень можна суттєво підвищити.

З практичної точки зору вихідна матриця ваг залишається замороженою під час тонкого налаштування. Замість цього дві додаткові матриці, A і B, налаштовуються точно. Ці матриці діють як декомпозиція точно налаштованої вагової матриці.

LoRA має кілька ключових переваг:

– Можливість використання попередньо навченої моделі для створення багатьох невеликих модулів LoRA для різних завдань. Ми можемо заморозити загальну модель та ефективно перемикатися між завданнями, замінюючи матриці A і B на Рисунку 2.1, що значно зменшує обсяг зберігання та витрати на перемикання завдань;

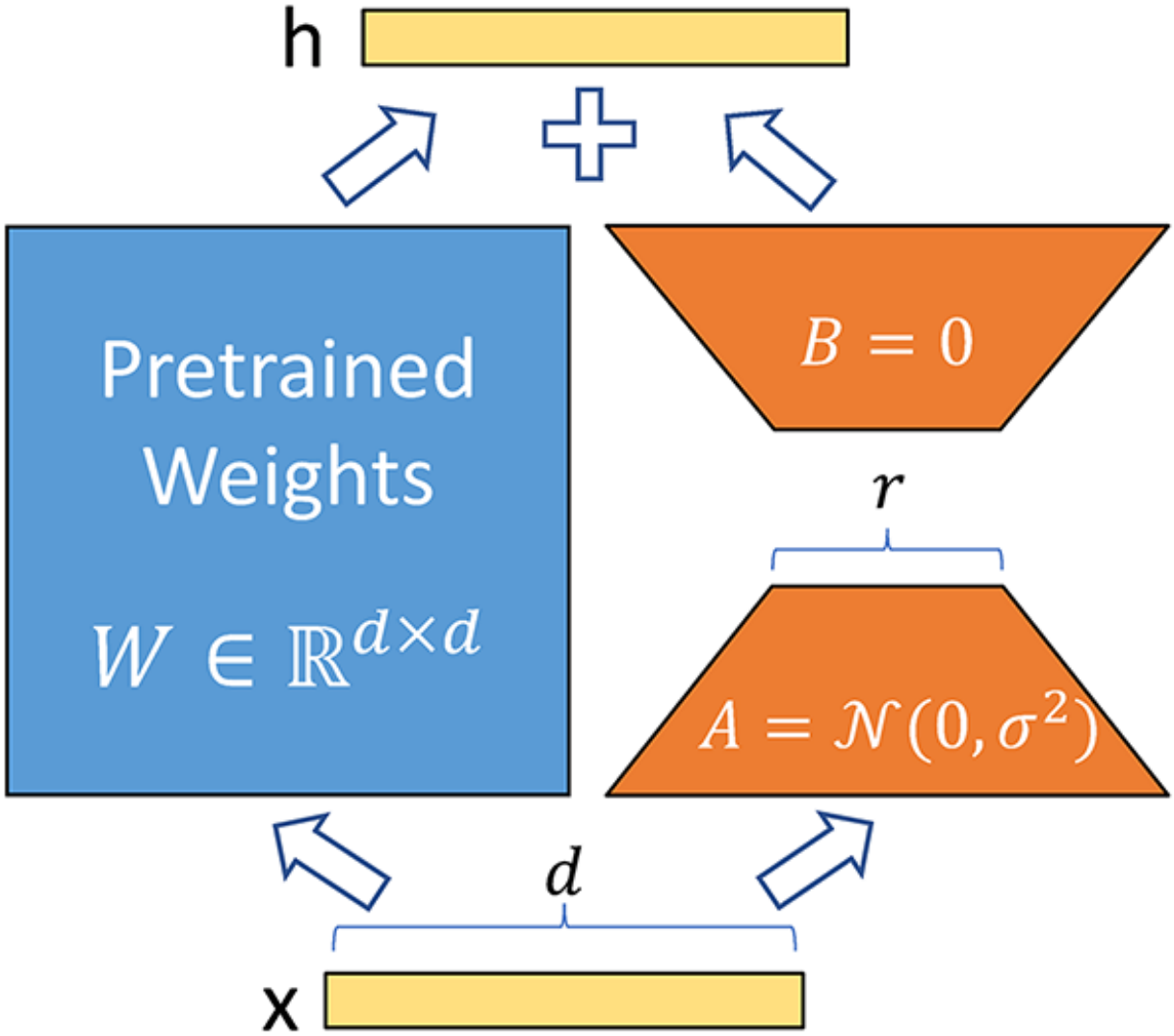

Рисунок 2.1 - Схема обробки LoRA [11]

-

– LoRA полегшує тренування та знижує бар'єр входу для обладнання в 3 рази при використанні адаптивних оптимізаторів, оскільки нам не потрібно обчислювати градієнти або зберігати стани оптимізатора для більшості параметрів. Замість цього ми оптимізуємо лише внесені набагато менші матриці з низьким рангом;

– простий лінійний дизайн дозволяє нам об'єднувати навчальні матриці зі замороженими вагами під час розгортання, що не вводить затримку в інференс порівняно з повністю налаштованою моделлю, в будові;
– LoRA є ортогональним багатьом попереднім методам і може комбінуватися з багатьма з них, такими як префікс-налаштування.

2.2 Метод налаштування Q-LoRA

Q-LoRA(Quantized Low Rank Adapters [12]): Квантовані адаптери з низьким рангом - це метод налаштування Large Language Models (LLMs), який використовує невелику кількість квантованих, оновлюваних параметрів для обмеження складності тренування. Ця техніка також дозволяє ефективно включати ці невеликі набори параметрів безпосередньо в модель, що означає можливість налаштування на великій кількості наборів даних і, потенційно, вставляти ці "адаптери" у модель за необхідності.

У QLoRA оригінальні попередньо навчені вагові коефіцієнти моделі квантуються до 4 біт та залишаються незмінними (замороженими) під час докладного налаштування. Потім під час докладного налаштування вводиться невелика кількість навчальних параметрів у формі адаптерів з низьким рангом. Ці адаптери навчаються пристосовувати попередньо навчену модель до конкретного завдання, на яке вона налаштовується, у форматі 32-бітних чисел з плаваючою комою.

Під час будь-якого обчислення в системі, будь то прямий прохід (роблення прогнозів) або зворотний прохід (оновлення ваг під час тренування), 4-бітові квантовані вагові коефіцієнти автоматично деквантуються назад у формат 32-бітних чисел з плаваючою комою. Це відбувається через те, що обчислення з числами з плаваючою комою може бути швидшими, ніж з меншою точністю через оптимізацію апаратного забезпечення.

Після процесу докладного налаштування модель складається з оригінальних 4-х бітних вагових коефіцієнтів та додаткових адаптерів з низьким рангом у їхньому форматі вищої точності. Цей комбінований підхід програмно-апаратного

забезпечення дозволяє ефективно налаштовувати великі мовні моделі обладнанні споживачів.

В QLoRA використовуються 3 метода що дозволяють знизити використання пам’яті:

– 4-Bit Normal Float;

– Double Quantization;

– Paged Optimizers.

4-Bit Normal Float (NF4) це тип даних, спеціально розроблений для застосувань у галузі штучного інтелекту, зокрема для квантування вагових коефіцієнтів в моделях для значного зменшення обсягу пам'яті при збереженні продуктивності. Це важливо для розгортання великих моделей на менш потужному обладнанні. NF4 є оптимальним інформаційно з точки зору даних з нормальним розподілом, які є звичайними для вагових коефіцієнтів моделей і може точніше представляти ці ваги, ніж стандартний 4-бітний плаваючий тип даних.

Double quantization це процес квантування константи квантування для подальшого зменшення пам'яті і збереження цих констант. Для виконання техніки деквантування нам потрібно зберігати константи квантування. Якщо ми використовуємо квантування блоками, то ми матимемо n констант квантування в їхньому оригінальному типі даних. У випадку роботи з великими, які мають значну кількість констант квантування, які повинні бути збережені, це призводить до збільшення накладних витрат пам'яті.

Порівняння повного налаштування моделі, LoRA та Q-LoRA методів схематично зображено на рис 2.2.

Full Finetuning (повне навчання) передбачає оновлення всіх параметрів попередньо натренованої моделі на новому наборі даних для адаптації до конкретного завдання.

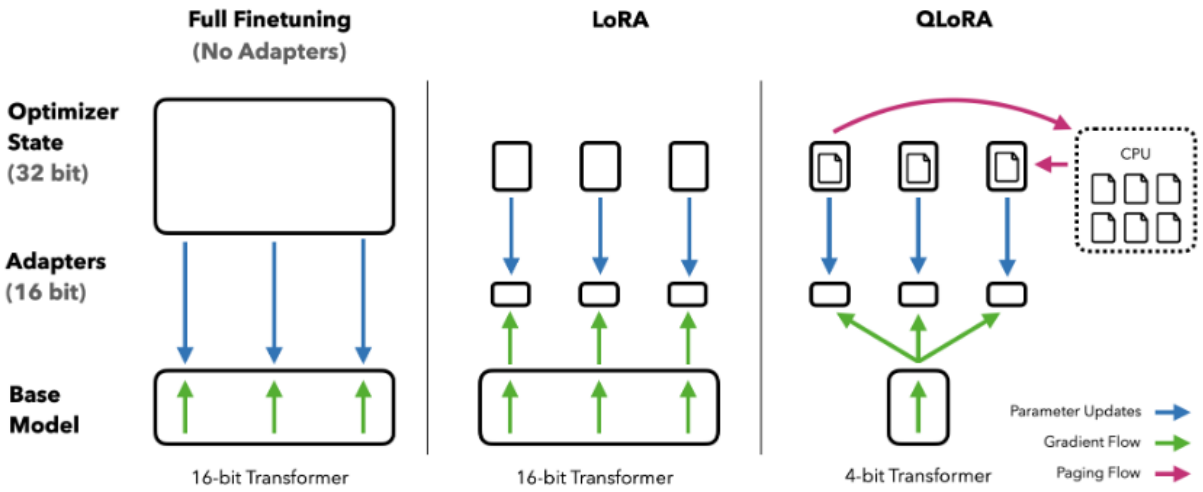

Рисунок 2.2 - Порівняння Full Finetuning, LoRA та QLoRA [13]

До переваг повного навчання можна віднести:

– максимальна адаптація, оскільки оновлюються всі параметри моделі то вона може максимально адаптуватись для нового завдання;

– висока точність для нових наборів даних, якщо вони добре структуровані;

– гнучкість та можливість застосування для широкого спектру завдань, оскільки немає обмежень на зміну параметрів.

Недоліками повного навчання є:

– великі обчислювані ресурси, потребує багато пам’яті особливо для великих моделей;

– адаптація може бути тривалим процесом, особливо для великого набору нових даних;

– ризик перенавчання на невеликому наборі даних, що може призвести до втрати узагальнюючої здатності.

3 ОГЛЯД МЕТРИК

3.1 Параметри постнавчання

Модель GPT-J з параметрами float16 вимагає близько 21 ГБ відеопам'яті, тоді як модель LLaMa-2 від Meta потребує приблизно 24 ГБ. Для зменшення використання пам'яті ми застосуємо квантування. Як вказано вище, Q-LoRA зменшує використання пам'яті без втрати точності.

8-бітний GPT-J вимагає приблизно 6 ГБ на GTX 1080 TI, LlaMA-2-7B потребує близько 7.5 ГБ. 4-бітна LLaMA-2-7B вимагає близько 4.3 ГБ, саме тому ми використали 4-бітне квантування для нашого експерименту.

LoRA фіксує та зберігає коригування в додаткових вагових параметрах, залишаючи всі ваги попередньо навченої моделі без змін. Відповідно, нова матриця ваг навчається з використанням модифікацій від попередньо навченої моделі, і ця оновлена матриця потім розкладається на дві матриці меншого рангу. Розмір цих матриць меншого рангу визначається параметром r. Чим менше це значення, тим менше параметрів потрібно навчати, отже, необхідно менше обчислень і часу, але, з іншого боку, можлива втрата інформації та продуктивності.

Для кожного шару, який буде навчений,$d \times k$ матриця оновлення ваг ΔW представлена розкладом меншого рангу BA, де B є матрицею $d \times r$, а A — матрицею $r \times k$ рис 3.1

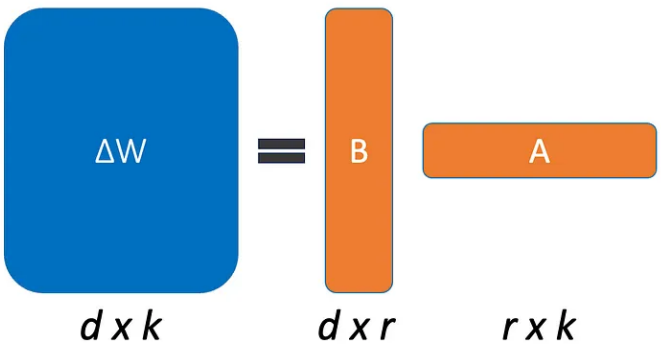

Рисунок 3.1 – Декомпозиція матриці вагових коефіцієнтів [14]

ΔW масштабується за допомогою α / r, де α є константою. При оптимізації з алгоритмом Adam, налаштування α приблизно таке саме, як налаштування швидкості навчання, якщо ініціалізація була відповідно масштабована. Причина в тому, що кількість параметрів збільшується лінійно з I.

QLoRa полягає в застосуванні квантування до методу LoRA, дозволяючи 4-бітне нормальне квантування, nf4, тип, оптимізований для нормально розподілених ваг та подвійне квантування для зменшення використання пам'яті.

Розглянемо основні параметри QloRA: рівень квантування, ранг адаптерів, коефіцієнт навчання, розмір міні-пакету, кількість епох, регуляризація, адаптивний оптимізатор.

Рівень квантування визначає ступінь квантування параметрів. Високий рівень квантування знижує потребу в пам'яті та прискорює обчислення, але може призвести до втрати точності. Зменшення рівня квантування дає кращий баланс між ефективністю та точністю, але потребує більше ресурсів.

Ранг адаптерів визначає розмір нізкорівневих адаптерів що додаються до моделі. Вищий ранг дозволяє моделі кращє адаптуватись до нових завдань, покращуючи точність, але збільшує обчислювальну складність. Зменшення рангу адаптерів знижує ресурси, але може не забезпечити достатньої гнучкості для адаптації, що призведе до нижчої точності.

Коефіцієнт навчання визначає швидкість оновлення ваг під час навчання. Високий коефіцієнт може прискорити навчання, але збільшує ризик пропуску оптимальних значень та нестабільність навчання. Низький коефіцієнт забезпечує стабільніше та детальніше навчання, але потребує більше часу.

Розмір міні-пакету задає кількість прикладів, які обробляються за один раз під час оновлення ваг. Великий розмір прискорює навчання завдяки ефективнішому використанню обчислювальних ресурсів, але може вимагати більше пам'яті. Малий розмір міні-пакету забезпечує детальніше оновлення ваг, але може уповільнити навчання.

Кількість епох задає кількість проходів через весь навчальний набір даних. Більша кількість епох може покращити точність, дозволяючи моделі краще

адаптуватися, але збільшує час навчання і ризик перенавчання. Менша кількість епох знижує час навчання, але може призвести до недостатньої адаптації.

Регуляризація задає методи для запобігання перенавчанню, такі як L2-регуляризація або dropout. Сильна регуляризація знижує ризик перенавчання, але може обмежити здатність моделі адаптуватися до специфічних особливостей даних. Слабка регуляризація дозволяє моделі більше адаптуватися до даних, але збільшує ризик перенавчання.

Адаптивний оптимізатор це алгоритм для оновлення ваг моделі, наприклад, Adam, Adagrad. Різні оптимізатори мають різні стратегії оновлення ваг, що може вплинути на швидкість збіжності і стабільність навчання.

Адаптивний оптимізатор Adam (Adaptive Moment Estimation) [15] – це оптимізаційний алгоритм, який комбінує переваги двох інших методів: Adagrad і RMSProp. Він є популярним вибором для навчання нейронних мереж через свою ефективність і простоту використання. Adam використовує адаптивні коефіцієнти навчання для кожного параметра. Це досягається шляхом оцінки перших (середнє значення градієнтів) і других (середнє значення квадратів градієнтів) моментів градієнтів. Алгоритм включає кілька ключових етапів:

– визначення градієнта функції втрат g_t по відношенню до ваг на кроці t;

– оновлення середніх значень моментів. Для першого середнього значення (першого моменту) використовується формула:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$

де β_1 коефіцієнт експоненціального зменшення для першого моменту (зазвичай 0.9), m_{t-1} значення першого моменту на попередньому кроці, g_t поточний градієнт;

– для другого середнього значення (другий момент) використовується:

$$\upsilon_t = \beta_2 * \upsilon_{t-1} + (1 - \beta_2) * g_t^2$$

де β_2 коефіцієнт експоненціального зменшення для другого моменту (зазвичай 0.999), υ_{t-1} значення другого моменту на попередньому кроці;

– корекція зміщення задається:

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \text{ та } \widehat{\upsilon}_t = \frac{\upsilon_t}{1-\beta_2^t};$$

де β_1^t ступінь коефіцієнта β_1на кроці t, β_2^t ступінь коефіцієнта β_2на кроці t

– оновлення параметрів задається:

$$\theta_t = \theta_{t-1} - \frac{\eta * \widehat{m}_t}{\sqrt{\widehat{\upsilon}_t} + \varepsilon}$$

де θ_{t-1} параметри на попередньому кроці, η початковий коефіцієнт навчання (зазвичай 0.001), ε маленьке значення для запобігання діленню на нуль (зазвичай 1e-8);

До основних переваг Adam над іншими алгоритмами належать:

– адаптивні коефіцієнти навчання що дозволяють Adam автоматично підбирати коефіцієнти навчання для кожного параметра, що особливо корисно для великих і складних моделей, де ручне налаштування може бути складним;

– стабільність завдяки корекції моментів і адаптивному навчанні, Adam забезпечує швидше збіження порівняно з алгоритмами зі сталою швидкістю навчання;

– ефективність завдяки чому Adam добре працює на великих і нерівномірних наборах даних, де інші методи можуть бути менш ефективними;

– швидка збіжність через впровадження моментів дозволяє Adam швидше зближуватися до оптимуму, що важливо для великих моделей і великих наборів даних;

Реалізація Q-LoRa в нашому експерименті вимагає визначення конфігурації BitsAndBytes та визначення змінних конфігурації LoRA:

– load_in_4bit=true; Цей прапорець використовується для включення 4-бітного квантування шляхом заміни лінійних шарів на шари FP4/NF4 від BitsAndBytes;

– bnb_4bit_quant_type=nf4; Це встановлює тип даних квантування в NF4;

– bnb_4bit_use_double_quant=true; Цей прапорець використовується для вкладеного квантування, де константи квантування з першого квантування знову квантуються;

– lora_r=16; Це встановлює параметр рангу розкладу LoRA;

– lora_alpha=8; Це встановлює константу α.

3.2 Збір результатів та їх оцінка.

Для оцінки якості постнавчання різних великих мовних моделей та збору даних була реалізована наступна система. В систему обробки клієнтських запитів було інтегровано віджет для роботи з великими мовними моделями через REST API. Постнавчені моделі були розгорнуті у віртуальному сервері. За допомогою REST API клієнтські запити з віджета спрямовувались на сервер з моделями і кожна з моделей генерувала відповідь. Менеджери роботи з клієнтами оцінювали релевантність запропонованих згенерованих відповідей за 5-ти бальною шкалою, де ‘1’ — найгірший варіант та ‘5’ — найкращий варіант. Схема інтеграції постнавчених моделей в систему підтримки коистувачів відображена на рис 3.2.

Рисунок 3.2 — Схема інтеграції постнавчених моделей (рисунок створено самостійно)

Для збереження оцінок від менеджерів був створений вебсервіс та розроблена база даних. В базі даних накопичувалась інформація про оцінку якості відповідей, запропонованих різними моделями, надана менеджерами підтримки під час обробки клієнтських запитів.

Для оцінки якості згенерованої відповіді були використані наступні метрики:

– зміст і відповідність завданню;

– граматика і орфографія;

– структура і зв’язність;

– стиль і читабельність.

Для оцінки змісту і відповідності завданню сформована наступна шкала:

– 1 бал — текст не відповідає завданню, містить багато нерелевантної інформації або зовсім не має сенсу;

– 2 бала — текст частково відповідає завданню, але містить багато нерелевантної або неправильної інформації;

– 3 бала — текст загалом відповідає завданню, але містить кількість неточностей або незначну нерелевантну інформацію;
– 4 бала — текст добре відповідає завданню з незначними неточностями або упущеннями;
– 5 балів — текст повністю відповідає завданню, чітко висвітлює всі необхідні аспекти.

Для оцінки граматики і орфографії запропонована наступна шкала:

– 1 бал — текст містить багато граматичних та орфографічних помилок, що роблять його важким для розуміння;
– 2 бала — текст містить кілька значних граматичних та орфографічних помилок;
– 3 бала — текст містить деякі граматичні та орфографічні помилки, але вони не значно впливають на розуміння;
– 4 бала — текст майже без помилок, лише з незначними помилками, які легко виправити;
– 5 балів — текст повністю без граматичних і орфографічних помилок.

Для оцінки структури і зв’язності запропонована наступна шкала:

– 1 бал — текст не має логічної структури, важко зрозуміти послідовність думок;
– 2 бала — текст має часткову структуру, але послідовність думок важко відстежити;
– 3 бала — текст має базову структуру, але місцями відсутні чіткі зв'язки між частинами;
– 4 бала — текст добре структурований, з чіткими переходами між частинами, але з невеликими недоліками;
– 5 балів — текст чітко структурований, з логічними і плавними переходами між усіма частинами.

Для оцінки стилю і читабельності запропонована наступна шкала:

– 1 бал — стиль тексту не відповідає завданню, важкий для читання і розуміння;

– 2 бала — стиль тексту частково відповідає завданню, але є важким для читання;
– 3 бала — стиль тексту загалом відповідає завданню, але місцями важкий для читання;
– 4 бала — стиль тексту добре відповідає завданню, легко читається, але є невеликі недоліки;
– 5 балів — стиль тексту повністю відповідає завданню, легко читається і розуміється.

Оскільки метрика “зміст і відповідність завданню” для застосування в системах підтримки клієнтів має важливе значення для автоматизації і прискорення обробки клієнтських запитів то для розрахунку загальної оцінки був обран метод вагового усереднення.

Наступні ваги були обрані для розрахунку:

– зміст і відповідність завданню — 0.45;
– граматика і орфографія — 0.2;
– структура і зв’язність — 0.2;
– стиль і читабельність — 0.15.

4 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ

Практичне дослідження включає постнавчання різних великих мовних моделей та демонструє різницю якості згенерованих результатів для моделей навчених на глобальних даних, та постнавчених моделей з використанням спеціалізованих даних з клієнтської бази знань.

Для реалізації поставленої задачі необхідно виконати наступні кроки

– експортувати дані з системи підтримки користувачів, та історію клієнтських звернень, відфільтрувати та відформатувати дані для постнавчання;
– провести постнавчання та зберегти великі мовні моделі;
– розробити та реалізувати веб сервіс з API який на вхід буде отримувати клієнтський запит і повертати згенерованих текст, а також розробити віджет для інтеграції API в систему підтримки клієнтів;
– розробити та реалізувати веб сервіс для збереження оцінок згенерованого тексту.

4.1 Підготовка даних

Для експорту даних з бази знань, та системи обробки клієнтських звернень був створен REST API клієнт з використанням мови Python.

Під час підготовчого процесу база знань проекту Comsend [16] була експортована разом з історією запитів та відповідей з системи підтримки клієнтів. Для експорту даних був створений REST API клієнт з використанням мови Python. Статті були збережені у форматі txt без додаткової розмітки. Тегі, зображення, посилання на зовнішні джерела були вилучені. Загалом було оброблено 139 статей, загальний обсяг яких склав 150,157 символів. Статті розділені відповідно до категорій в базі знань. Приклад форматування статті наведен нижче:

```
### Title: Orders that are not Finalized
### Category: Emailing
```

Body: At the end of your campaign, and depending on the option you choose for Finalizing member orders (see Finalize Explained) you may find that you have member selections that are not finalized, and so these will not appear in the greeting cards.To see who did not finalize go to Reports > Participation Detailed Report and click the "Is Finalized" header twice (to sort by this column). Your Dashboard will also indicate a warning of "unfinalized" orders.You can finalize for these members or send an email just to those who did not finalize (see the appropriate option within the Group By drop down when selecting recipients for your email).

Для обробки даних, отриманих з історії запитів, було обрано формат "питання-відповідь". Були відфільтровані дублікати питань та питання без відповідей. Особисті дані були вилучені. Серії “питань-відповідей” були позначені як ‘успіх’ або ‘невдача’ згідно з результатом розв’язання на основі відгуків клієнтів, або позначок менеджерів, збережених у системі підтримки клієнтів. Загальна кількість серій “питання-відповідь” склала 1638. Тегі, зображення були вилучені. У відповідях, посилання на статті з бази знань залишились з вказанням заголовку статті. Приклад вдалої серії “питання-відповідь” наведена нижче

Subject: Login problem
Question: Need help with login/username and password
Answer: Hi ,Your username is {username}. If you forgot your password please go to www.comsend.com[http://www.comsend.com/] and there you can use the "forgot password" link. The link to reset your password will go to your email. Can you let us know if this worked for you?Thanks,
Question: I’m in! Thank you!
Solved

4.2 Постнавчання моделей

LLaMA-2-7B була постнавчена з використанням власного набору даних, який містить 139 статей та 1638 пар питань-відповідей. Навчання тривало 1,5 години та було завершено за 800 ітерацій. Як можна побачити на кривій відсотка втрат (див. рис. 4.1), навчання було завершено, і значення втрат стабілізувалося на середньому рівні 1,4%. Подальше додаткове налаштування не має сенсу. Постнавчену модель можна використовувати для інтеграції в систему підтримки клієнтів. Постнавчена модель з оновленими параметрами була збережена в репозіторії Hugging Face для подальшого використання.

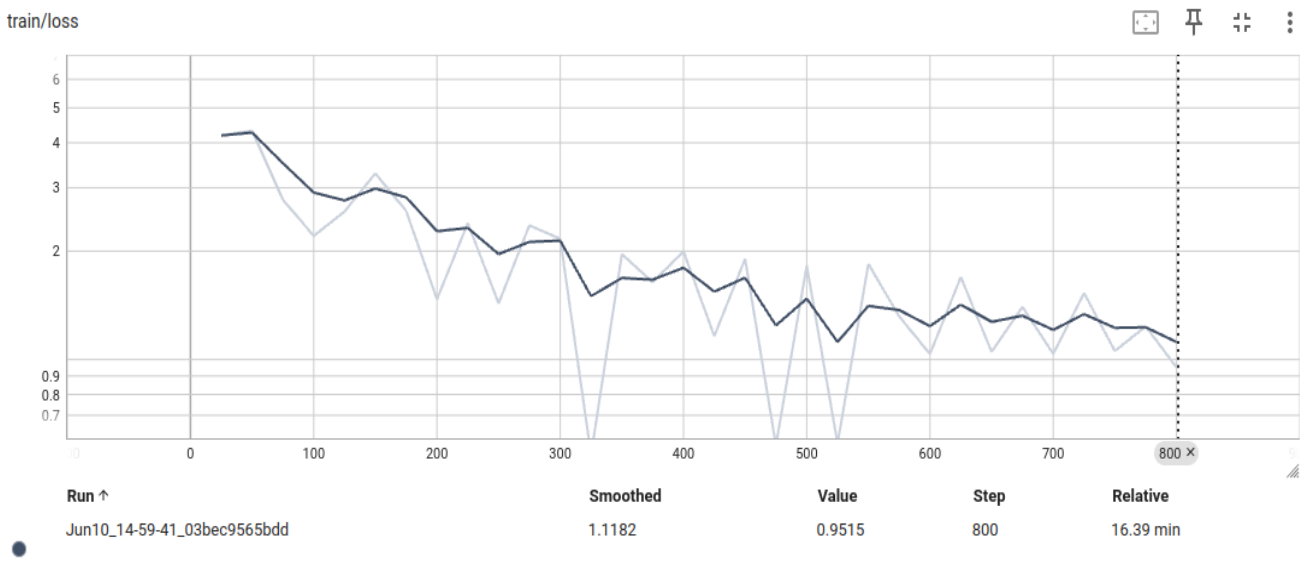

Рисунок 4.1 – Відсоток втрат під час постнавчання моделі LLaMA-2-7B (рисунок створено самостійно)

GPT-J-6B була додатково налаштована з використанням того ж набору даних. Навчання тривало близько 1,7 годин і було завершено за 950 ітерацій. Як можна побачити на кривій відсотка втрат (див. Рис. 4.2).

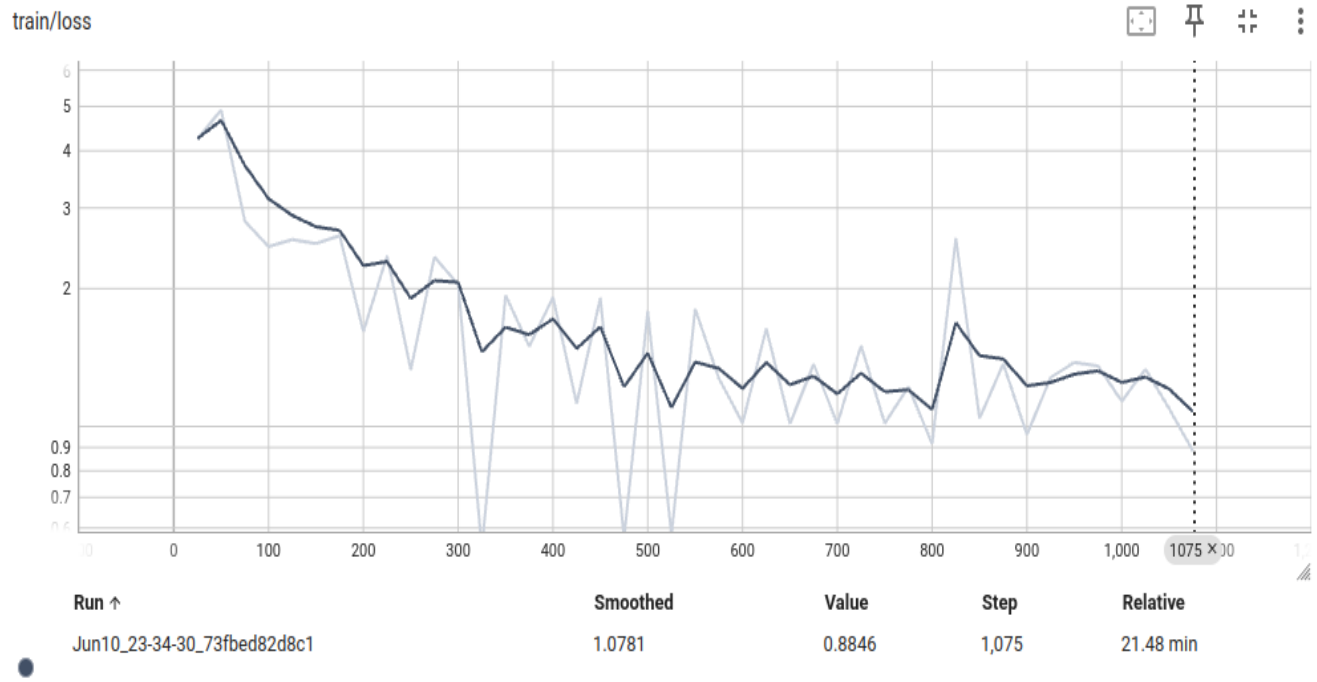

Рисунок 4.2 –Відсоток втрат під час постнавчання моделі GPT-J-6B (рисунок створено самостійно)

Навчання було завершено, і значення втрат стабілізувалося на середньому рівні 1,2%, і подальше додаткове налаштування не має сенсу. Постнавчену модель можна використовувати для інтеграції.

Для роботи з постнавченими моделями був створений веб сервіс який реалізує API інтерфейс. Вхідними даними для нього слугують клієнтські запити, а у відповідь веб сервіс надає згенеровані різними моделями тексти.

Модель розгорнута на віддаленому сервері з підтримкою GPU для швидкого оброблення запитів. Для створення серверного додатку, що приймає запити, обробляє їх за допомогою мовної моделі та повертає результати використували мову програмування Python та фреймворк FastAPI. Для забезпечення базової безпеки та контролю доступу використовується авторизація за допомогою API ключа. Використані функції роботи з моделлю з бібліотеки Transformers (Hugging Face).

Для інтеграції в систему підтримки клієнтів ZohoCRM було розроблено та реалізовано віджет (вбудований компонент в інтерфейс користувача) з використанням Node.JS та zoho-extension-toolkit [17].

4.3 Збір та аналіз результатів

Для аналізу релевантності згенерованих відповідей та збору статистики був створений віджет інтеграції в систему обробки клієнтських звернень. Віджет дозволяє спрямовувати клієнтські запити до вебсервісу в якому інтегровані постнавчені моделі. Згенеровані моделями відповіді повертаються у віджет та пропонуються менеджеру як варіант відповіді. Якість (релевантність) відповіді оцінюються менеджером за п’ятибальною шкалою. Оцінка менеджера відправляється на окремий вебсервіс для зберігання і подальшого опрацювання. Також менеджеру пропонувалось оцінити відповідь згенеровану за допомогою інтеграції з Chat CPT v3.5 моделлю яка була навчена на загальних даних, без додаткового постнавчання.

Під час експерименту було зібрано 124 оцінки щодо згенерованих відповідей на різні клієнтські запити, надані постнавченеми моделями та Chat

GPT v3.5. Результати експерименту, згруповані за значенням наданої оцінки, наведені в таблицях 4.1-4.4.

Таблиця 4.1 – Оцінки відповідей менеджерами підтримки за метрикою “зміст і відповідність завданню” (таблиця виконана самостійно)

Назва моделі	Оцінки				
	1	2	3	4	5
GPT-J-B6 fine-tuned	18	31	29	29	17
LLaMa-2-7B fine-tuned	5	23	32	35	29
Chat GPT v 3.5	17	34	25	22	24

Таблиця 4.2 – Оцінки відповідей менеджерами підтримки за метрикою “граматика і орфографія” (таблиця виконана самостійно)

Назва моделі	Оцінки				
	1	2	3	4	5
GPT-J-B6 fine-tuned	17	11	27	29	18
LLaMa-2-7B fine-tuned	5	28	31	33	27
Chat GPT v 3.5	9	30	21	25	39

Таблиця 4.3 – Оцінки відповідей менеджерами підтримки за метрикою “структура і зв’язність” (таблиця виконана самостійно)

Назва моделі	Оцінки				
	1	2	3	4	5
GPT-J-B6 fine-tuned	23	17	29	31	24
LLaMa-2-7B fine-tuned	8	18	25	34	39
Chat GPT v 3.5	23	28	18	22	32

Таблиця 4.4 – Оцінки відповідей менеджерами підтримки за метрикою “стиль і читабельність” (таблиця виконана самостійно)

Назва моделі	Оцінки				
	1	2	3	4	5
GPT-J-B6 fine-tuned	28	32	20	22	22
LLaMa-2-7B fine-tuned	9	14	30	33	37
Chat GPT v 3.5	27	29	25	21	20

Середні значення метрик для кожної моделі наведені в таблиці 4.5.

Таблиця 4.5 – Середні значення” (таблиця виконана самостійно)

Назва метрики	GPT-J-B6 fine-tuned	LLaMa-2-7B fine-tuned	Chat GPT v 3.5
Зміст і відповідність завданню	2,967741935	3,487804878	3,177419355
Граматика і орфографія	2,983870968	3,39516129	3,443548387
Структура і зв’язність	3,129032258	3,629032258	3,096774194
Стиль і читабельність	2,822580645	3,620967742	2,814516129

Усереднені значення з урахуванням вагових коефіцієнтів (0.45 для “зміст і відповідність завданню”, 0.2 для “граматика і орфографія”, 0.2 для “структура і

зв’язність” та 0.15 для “стиль і читабельність”) кожної з метрик наведені в таблиці 4.6.

Таблиця 4.6 Усереднені значення з урахуванням вагових коефіцієнтів ” (таблиця виконана самостійно)

Назва метрики	GPT-J-B6 fine-tuned	LLaMa-2-7B fine-tuned	Chat GPT v 3.5
Зміст і відповідність завданню	1,335483871	1,569512195	1,42983871
Граматика і орфографія	0,5967741935	0,6790322581	0,6887096774
Структура і зв’язність	0,6258064516	0,7258064516	0,6193548387
Стиль і читабельність	0,4233870968	0,5431451613	0,4221774194

Позитивним вважається результат оцінки від менеджерів ‘3’, ‘4’ та ‘5’ та негативним вважається значення оцінки ‘1’ та ‘2’. Відсоток позитивних і негативних результатів для кожної моделі обрахований в таблиці 4.7

Таблиця 4.7 – Відсоток позитивних і негативних відповідей (таблиця виконана самостійно)

Назва моделі	Негативний	Позитивний
GPT-J-B6 fine-tuned	37,3%	62,7%
LLaMa-2-7B fine-tuned	22,2%	77,8%
Chat GPT v 3.5	40,1%	59,9%

Після аналізу результатів ми можемо зробити висновок, що Chat GPT v 3.5 дає незадовільні результати, де приблизно 40% відповідей не можуть бути застосовані в конкретній галузі. Це вимагає додаткового налаштування з використанням даних, пов'язаними з галуззю, або подальшого навчання на основі підказок.

LLaMA-2-7B перевершує GPT-J-6B та надає релевантні результати, вимагаючи менше обчислювальних ресурсів для додаткового налаштування. Але все одно існує близько 20% неправильних пропозицій від LLaMA-2-7B, і цей показник можна зменшити, провівши додаткове навчання.

ВИСНОВКИ

На основі результатів наших досліджень можна зробити наступні основні висновки.

Дані , специфічні для вузької галузі, були експортовані з бази знань проекта. Дані питань і відповідей були експортовані з системи підтримки клієнтів. Після цього вони були очищені від конфіденційної інформації і відформатовані. До статей був доданий тематичний або категорійний позначник. Серії питань і відповідей були позначені як "успіх" або "невдача".

Під час проведення дослідження ми налаштували 2 великі мовні моделі за допомогою власного набору даних для надання рекомендацій у конкретній галузі. Під час налаштування моделі ми виявили, що LLaMA-2-7B зазнає на 7% менше втрат, ніж GPT-6B на тому ж наборі тренувальних даних. Крім того, LLaMA-2-7B може бути налаштована за меншу кількість ітерацій, приблизно на 14%.

Для налаштування GPT-J-6B краще підходить формат набору даних “запитання - відповідь” з невеликою кількістю довгих текстових відповідей. Тоді як LLaMA-2-7B працює добре як зі звичайним текстовими даними, так і з набором даних “запитання - відповідь”.

Після цього згенеровані моделями відповіді були інтегровані до системи підтримки клієнтів. Відгуки менеджерів підтримки були зібрані під час експерименту за допомогою опитування для кожного з варіантів згенерованої відповіді. Дані опитування були агреговані в межах власного веб-сервісу, розробленого для дослідження.

Після збору даних і аналізу результатів була складена порівняльна таблиця. Використання більших наборів даних, специфічних для домену, може підвищити релевантність відповідей, але вимагатиме більшої обчислювальної потужності для налаштування великих мовних моделей.

Результати дослідження і експерименту були використані для впровадження прототипу модуля рекомендацій в системі підтримки клієнтів.

Результати роботи представлені під час XXVIII міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ» (див. В.1-В.3).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. D. Gefen "Customer Loyalty in E-Commerce.", J. Assoc. Inf. Syst., 3 2002, 2. https://doi.org/10.17705/1jais.00022.

2. Z. Wang, "Empowering Few-Shot Recommender Systems With Large Language Models-Enhanced Representations," in IEEE Access, vol. 12, pp. 29144-29153, 2024, doi: 10.1109/ACCESS.2024.3368027

3. L. Chen, F. Yuan, J. Yang, X. He, C. Li and M. Yang, "User-Specific Adaptive Fine-Tuning for Cross-Domain Recommendations," in IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 3, pp.

4. V. Bilgram and F. Laarmann, "Accelerating Innovation With Generative AI: AI-Augmented Digital Prototyping and Innovation Methods," in IEEE Engineering Management Review, vol. 51, no. 2, pp. 18-25, 1 Secondquarter,june 2023, doi: 10.1109/EMR.2023.3272799

5. K. Smelyakov, A. Chupryna, D. Sandrkin and M. Kolisnyk, "Search by Image Engine for Big Data Warehouse," 2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania, 2020, pp. 1-4, doi: 10.1109/eStream50540.2020.9108782

6. K. Smelyakov, A. Chupryna, O. Bohomolov and I. Ruban, "The Neural Network Technologies Effectiveness for Face Detection," 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP), 2020, pp. 201-205, doi: 10.1109/DSMP47368.2020.9204049.

7. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. 2017. URL: https://doi.org/10.48550/arXiv.1706.03762.

8. Q. Luo, W. Zeng, M. Chen, G. Peng, X. Yuan and Q. Yin, "Self-Attention and Transformers: Driving the Evolution of Large Language Models," 2023 IEEE 6th International Conference on Electronic Information and Communication Technology (ICEICT), Qingdao, China, 2023, pp. 401-405, doi: 10.1109/

9. Introduction to Large Language Models. URL: https://www.baeldung.com/cs/large-language-models#6-encoder-decoder-architecture

10. License: https://ai.meta.com/llama/license/

11. Hu, J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. ArXiv, abs/2106.09685, 2021.

12. Xu, Y., Xie, L., Gu, X., Chen, X., Chang, H., Zhang, H., Chen, Z., Zhang, X., & Tian, Q. QA-LoRA: Quantization-Aware Low-Rank Adaptation of Large Language Models. ArXiv, abs/2309.14717, 2023

13. In-depth guide to fine-tuning LLMs with LoRA and QloRA. URL: https://www.mercity.ai/blog-post/guide-to-fine-tuning-llms-with-lora-and-qlora

14. More about LoraConfig from PEFT. URL: https://medium.com/@manyi.yim/more-about-loraconfig-from-peft-581cf54643db

15. Kingma, D. P., & Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.,2014

16. Knowledge base: https://purimproject.zohodesk.com/portal/en/kb/purim-project

17. Widgets in Zoho CRM. URL: https://www.zoho.com/crm/developer/docs/widgets/