

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для генерації персоналізованих
AI-ботів на основі донавчання мовної моделі
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШ-21-4

Денис Храмцов
(власне ім'я, прізвище)

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

Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)

Керівник ас. Максим Політ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____
(підпис)

Олег ЗОЛОТУХІН
(власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____

Кафедра _____ Штучного інтелекту _____

Рівень вищої освіти _____ перший (бакалаврський) _____

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

Тип програми _____ освітньо-професійна _____

Освітня програма _____ Штучний інтелект _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«_____» _____ 20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Храмцову Денису Ігоровичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Програмна система для генерації персоналізованих AI-ботів на основі донавчання мовної моделі _____

затверджена наказом університету від 19 травня 2025 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 25 червня 2025 р.

3. Вихідні дані до роботи _____ Наукові публікації, фахова література, дані Інтернет-джерел, набір даних PIPPA, набір даних SODA, відкрите інтернет джерело Reddit, технічна документація. _____

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі та теоретичні дослідження _____

2) Аналіз конкурентів та постановка завдання _____

3) Практична реалізація системи _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	21.05.2025	виконано
3	Теоретичні дослідження	26.05.2025	виконано
4	Аналіз конкурентів	28.05.2025	виконано
5	Постановка задачі	30.05.2025	виконано
6	Проектування системи	02.06.2025	виконано
7	Збір та очистка наборів даних	04.06.2025	виконано
8	Донавчання великих мовних моделей	06.06.2025	виконано
9	Розробка веб-застосунку	11.06.2025	виконано
10	Написання пояснювальної записки	17.06.2025	виконано
11	Перевірка на академічний плагіат	18.06.2025	виконано
12	Нормоконтроль	19.06.2025	виконано
13	Підготовка презентації та доповіді	20.06.2025	виконано
14	Попередній захист	23.06.2025	виконано
15	Рецензування	24.06.2025	виконано
16	Захист перед ЕК	25.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____
(підпис)

ас. Максим Політ _____
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 107 с., 37 рис., 5 табл., 4 дод., 36 джерел.

ВЕЛИКА МОВНА МОДЕЛЬ, ГЕНЕРАЦІЯ ТЕКСТУ, ДОНАВЧАННЯ, ПОЗИЦІЙНЕ КОДУВАННЯ, ШАР УВАГИ, KV-КЕШУВАННЯ, LLM, LORA, NEURAL NETWORK, PEFT, QLORA, QUANTIZATION, ROLEPLAY DIALOGUE, TRANSFORMERS.

Об'єкт дослідження – онлайн-сервіси для знайомств та великі мовні моделі для генерації тексту

Предмет дослідження – сайт для знайомств з інтегрованою великою мовною моделлю, яка імітує стиль спілкування користувача та генерує повідомлення від його імені на початку діалогу.

Мета роботи – дослідити можливості великих мовних моделей у сфері персоналізованої комунікації та розробити прототип сервісу для знайомств, який використовує LLM для генерації перших повідомлень з урахуванням стилю користувача, з метою підвищення ефективності онлайн-взаємодії.

Методи дослідження – аналіз предметної галузі, огляд і порівняння сучасних застосунків для знайомств, вивчення архітектури трансформерів і методів донавчання, експериментальне донавчання моделей на спеціалізованих датасетах, технічна реалізація та тестування прототипу.

ABSTRACT

Bachelor's thesis contains: 107 pp., 37 fig., 5 tabl., 4 ann., 36 references.

ATTENTION LAYER, FINE-TUNING, KV-CACHING, LARGE LANGUAGE MODEL, LLM, LORA, NEURAL NETWORK, PEFT, POSITIONAL ENCODING, QLORA, QUANTIZATION, ROLEPLAY DIALOGUE, TEXT GENERATION, TRANSFORMERS.

Object of the research – online dating services and large language models used for text generation.

Subject of the research – a dating website integrated with a large language model that imitates the user's communication style and generates messages on their behalf at the beginning of a dialogue.

Purpose of the work – to explore the potential of large language models in the field of personalized communication and to develop a prototype of a dating service that uses an LLM to generate initial messages based on the user's style, in order to increase the effectiveness of online interactions.

Research methods – domain analysis, review and comparison of modern dating applications, study of transformer architecture and fine-tuning methods, experimental fine-tuning of models on specialized datasets, technical implementation, and prototype testing.

ЗМІСТ

Вступ.....	7
1 Аналіз предметної галузі та теоретичні дослідження.....	8
1.1 Сервіси для знайомств.....	8
1.2 Великі мовні моделі.....	11
1.3 Класичні архітектури для роботи з текстовою інформацією.....	14
1.4 Архітектура Transformer.....	19
1.5 Архітектура GPT.....	22
1.6 Шар уваги.....	25
1.7 Feed-forward neural network.....	28
1.8 Методи оптимізації роботи шару уваги.....	28
1.9 Методи та стратегії донавчання великих мовних моделей.....	32
1.10 Метрики оцінки якості великих мовних моделей.....	35
2 Аналіз конкурентів та постановка завдання.....	38
2.1 Аналіз конкурентів у галузі сервісів для знайомств.....	38
2.2 Аналіз конкурентів у галузі персоналізованих чат-ботів.....	42
2.3 Постановка задачі.....	46
3 Практична реалізація системи.....	52
3.1 Використанні інструменти.....	52
3.2 Структура даних.....	54
3.3 Використанні набори даних.....	58
3.4 Доновчання великих мовних моделей.....	64
3.5 Опис та огляд реалізації системи.....	75
Висновки.....	89
Перелік джерел посилання.....	91
Додаток А Програмний опис таблиць бази даних.....	95
Додаток Б Програмний опис використання великої мовної моделі.....	100
Додаток В Програмний опис фільтрування анкет.....	105
Додаток Г Відомість кваліфікаційної роботи.....	107

ВСТУП

Сервіси для знайомств стали важливою частиною життя багатьох людей. Завдяки цим платформам користувачі можуть знаходити друзів, партнерів або просто спілкуватися з новими людьми. Онлайн-знайомства дозволяють легко долати географічні бар'єри, розширювати коло спілкування та знаходити людей зі схожими інтересами.

У сучасних застосунках для знайомств активно використовуються нові технології, такі як штучний інтелект. Це дає змогу зробити підбір анкет більш точним, створити безпечне середовище та навіть вести діалоги з чат-ботами. В цій роботі буде розглядатись використання великих мовних моделей, які можуть імітувати стиль спілкування користувача. Це допоможе краще зрозуміти потенційного партнера ще до початку реального діалогу.

Актуальність теми полягає в тому, що багато людей стикаються з труднощами в онлайн-комунікації – не отримують відповідей, не почуваються впевнено або просто не знають, як краще почати розмову. Тому впровадження ботів, які говорять у стилі користувача, може стати зручним і ефективним рішенням.

Мета цієї роботи – проаналізувати існуючі сервіси для знайомств, та розробити прототип системи онлайн-знайомств, яка використовує велику мовну модель для імітації стилю спілкування користувача, а також впровадити механізм інтелектуального підбору анкет із урахуванням фільтрів, відстані, активності користувачів та історії взаємодій.

Особливу увагу приділено методам донавчання моделей, зокрема підходам PEFT, LoRa, QLoRa.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

1.1 Сервіси для знайомств

Роль сервісів для знайомств стає все більш впливовішою з кожним роком, починаючи з 1995 року, коли відкрився перший подібний сервіс – Match.com. Близько 30% американців віком від 18 до 29 років користуються сайтами знайомств або мобільними додатками для знайомств. Особливо після періоду COVID-19 онлайн-знайомства надають можливості для тих, хто ізольований від особистих соціальних контактів удома. Такий підхід звільняє людей від географічної відстані через інтернет-платформи, дозволяючи їм зустрічатися, спілкуватися та встановлювати емоційні зв'язки один з одним у віртуальному просторі [1].

Динаміку розвитку індустрії сервісів онлайн-знайомств добре ілюструє статистика (рисунок 1.1).

Revenue Of The Online Dating Industry In Europe 2018-2028

(in million U.S. dollars)

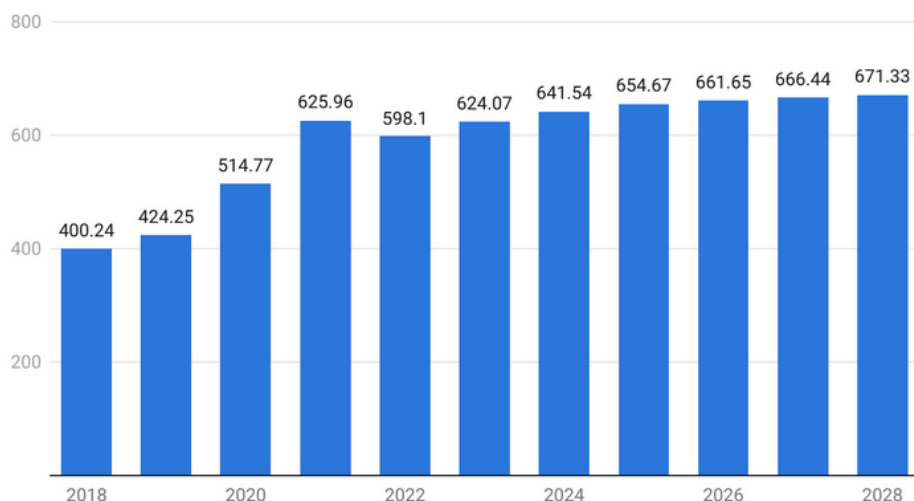


Рисунок 1.1 – Дохід індустрії онлайн-знайомств в Європі між 2018–2028 роками [2]

Так, у період між 2018 та 2019 роками сукупний дохід ринку онлайн-знайомств у Європі зріс приблизно на 6%. У подальші роки, зокрема між 2019 та 2020 роками, а також між 2020 та 2021 роками – після запровадження локдаунів – темп приросту дорівнював 21%. Надалі рівень доходів стабілізувався, зберігаючись на позначці близько 600 мільйонів доларів США щороку. Зазначена динаміка свідчить про сталий інтерес до онлайн-платформ для знайомств, що може демонструвати потенціал цієї індустрії для подальшого розвитку з перспективою на інтеграцію сучасних технологій, зокрема технологій штучного інтелекту.

Системи ШІ вже активно використовуються у сучасних сервісах для знайомств, часто працюючи разом. Один з підходів інтеграції ШІ – інтелектуальне заповнення профілю, для збільшення шансу взаємних вподобань, наприклад як генерування опису, згідно інтересам користувача або розумний вибір фото, який допомагає визначити, які фотографії з галереї користувача можуть підійти як фото для профілю. Відбір базується на багатьох факторах, таких як освітлення, композиція тощо.

Інший підхід – підбір профілів за допомогою ШІ, що дозволяє знайти персоналізовані знайомства на основі унікальних вподобань і інтересів користувачів.

Не менш важливим в подібних сервісах є питання безпеки – загрози, дискримінація та несправжні профілі. Майже половина жінок, які взяли участь в опитуванні від сервісу Bumble, (46%) заявили, що їхня головна проблема у спілкуванні в мережі – це невпевненість, чи є співрозмовник тим, за кого себе видає [3]. Через що команда застосунку ввела декілька система, одна з них це Deception Detector – інструмент, що використовує штучний інтелект для виявлення фейкових, спамних або шахрайських профілів. Інша модель це Private Detector, інструмент на базі, який автоматично розмиває потенційно ненормативне зображення, що надсилаються в чаті.

Однією з ключових проблем сучасних сервісів онлайн-знайомств є домінування візуального підходу до оцінювання потенційних партнерів, що сприяє формуванню поверхневого сприйняття особистості. Це призводить до великої кількості пар, які залишаються неактивними, користувачі або зовсім не починають спілкування, або не отримують відповіді на перше повідомлення (рисунок 1.2). Це свідчить про знижену ефективність таких платформ у контексті формування реальних соціальних зв'язків.

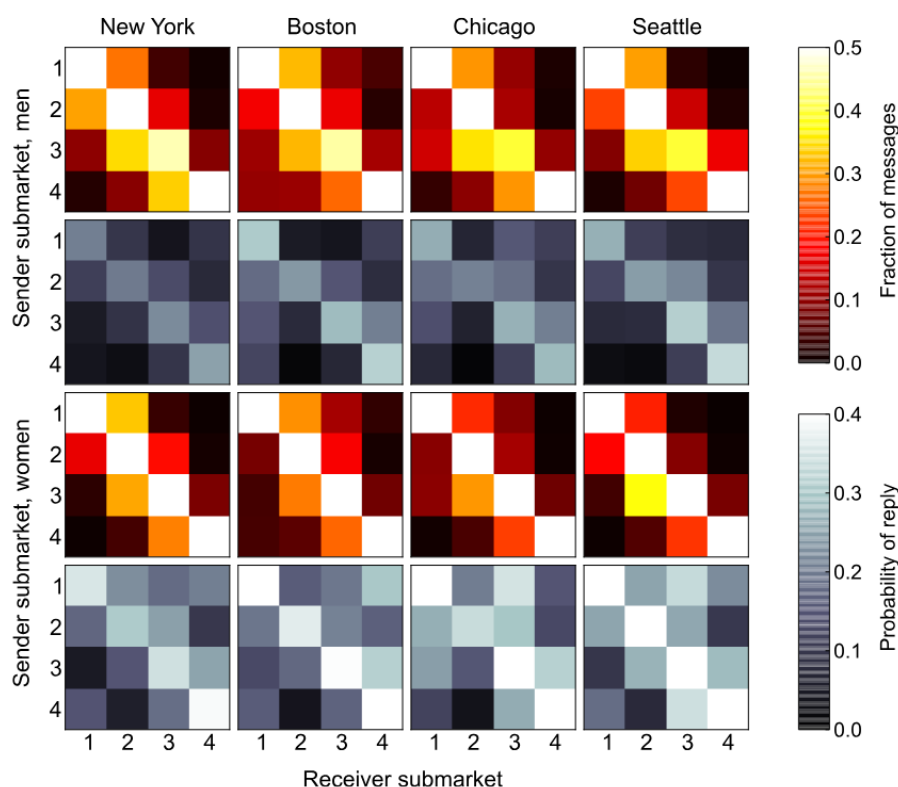


Рисунок 1.2 – Частка надісланих повідомлень між схожими групами користувачів та відсоток відповідей на повідомлення від чоловіків та жінок

Другий і четвертий рядки на рисунку показують частку перших повідомлень, що отримують відповідь, що свідчить про потенційну взаємну зацікавленість між користувачами. Відповіді жінок на повідомлення від чоловіків (другий рядок) надходять суттєво рідше, ніж відповіді чоловіків на повідомлення від жінок (четвертий рядок), імовірно через ефект обсягу:

жінки отримують у чотири рази більше перших повідомлень, ніж чоловіки, тому можуть дозволити собі бути більш вибірковими у відповідях [4].

1.2 Великі мовні моделі

Великі мовні моделі (LLM) – це клас штучного інтелекту, який здатен розуміти, інтерпретувати та генерувати тексти. Завдяки широкому навчанню на великих наборах даних, LLM-моделі здатні розуміти й створювати тексти на рівні, що практично не відрізняється від людського.

LLM-моделі стали надзвичайно популярними завдяки своїй здатності виконувати широкий спектр завдань обробки природної мови (NLP) та розуміння природної мови (NLU), таких як переклад і узагальнення тексту.

Приблизно в однаковий час з бурхливим зростом користувачів у сервісах для знайомств, відбувся стрімкий розвиток галузі моделей ШІ для генерації текстової інформації (рисунок 1.3).

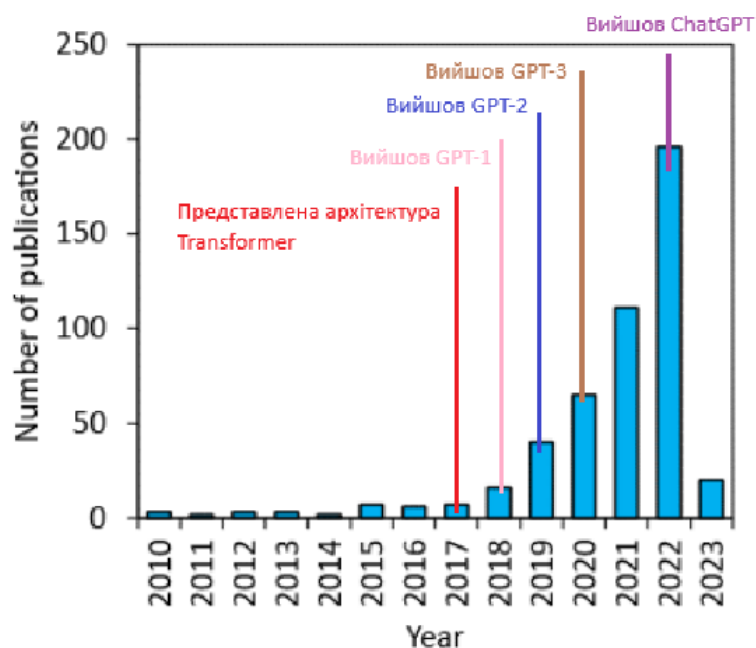


Рисунок 1.3 – Кількість публікацій за роками на тему моделей штучного інтелекту для генерації тексту [5]

Помітне зростання кількості наукових публікацій з цієї теми сталося після представлення архітектури Transformer у 2017 році, яка суттєво перевершила класичні підходи на основі рекурентних нейронних мереж (RNN) та довгої короткочасної пам'яті (LSTM). Це стало основою для стрімкого розвитку великих мовних моделей.

Вже через рік, в 2018 році, була презентована перша версія GPT (Generative Pretrained Transformer) – прорив у сфері обробки природньої мови. Ця модель містить 117 мільйонів параметрів та підтримує контекст довжиною у 512 токенів. В рамках NLP – токен це основна одиниця тексту, яка може відповідати слову, частині слова або навіть цілим словосполученням. В середньому один токен становить приблизно чотири символи.

Ще через рік, в 2019, була розроблена модель GPT-2, що мала вже 1.5 мільярда параметрів, що майже в 13 разів більше за попередню версію, та розміром контексту 1024 токени. GPT-3 став ще більш масштабним з кількістю параметрів у 175 мільярдів та розміром контексту 2048 токенів.

Вихід GPT-3.5, який адаптовано для спілкування у форматі діалогу, розпочав нову хвилю великої кількості публікацій, пов'язаних з моделями ШІ для генерації тексту. Модель має ще більший контекст, що дорівнює 4096 токенів.

Найбільш поширені моделі (GPT, DeepSeek, LLaMA, Mistral) вже можна використовувати «з коробки», бо вони вже попередньо навчені з повним набором ваг. Але їх можливості можуть не підходити для всіх задач. Часто потрібно або адаптувати стиль відповідей моделі, або додати якісь специфічні данні до бази знань моделі. Для цих задач існують способи їх налаштування, зокрема через prompt tuning (налаштування запитів) та fine-tuning (донавчання), що дозволяє покращити результати моделі без необхідності навчати її з нуля.

Prompt tuning – це простий метод налаштування, що передбачає створення спеціальних вхідних запитів для керування відповідями моделі.

Він не вимагає зміни параметрів моделі та використовує вже наявні знання й можливості моделі, лише змінюючи форму запиту.

Fine-tuning – це більш інтенсивний процес, у якому модель додатково навчається після первинного тренування на спеціалізованому наборі даних. У процесі оновлюються параметри моделі, щоб краще відповідати на цільові завдання. Fine-tuning забезпечує більш точні та релевантні відповіді, особливо у спеціалізованих випадках. Водночас, він потребує додаткових обчислювальних ресурсів і ретельно підготовленого набору даних. Цей метод особливо ефективний для адаптації результатів під певний стиль, тон, формат.

Зазвичай набір даних для донавчання складається з трійок ролей: «system», «user» та «assistant», кожна з яких має відповідний вміст [6].

Окрім наукових та професійних завдань, LLM можуть виконувати й розважальні функції. Одним з найпоширенішим прикладів є RP-боти. RP (roleplay або roleplaying) означає рольову гру, під час якої ШІ вживається в роль і світ зазначеного персонажа, виконує дії та веде діалоги від імені цього персонажа, а не від свого власного.

Типовою формою взаємодії в межах сервісів для онлайн-знайомств є обмін текстовими повідомленнями, що відкриває можливості для інтеграції LLM у систему комунікації між користувачами.

Метою проекту є донавчання вже існуючих LLM під задачу відіграшу ролей, де персонажами виступатимуть самі користувачі сервісу для знайомств. Такий підхід дозволить зберегти стиль розмови користувача, що сприятиме кращому розумінню, чи збігаються інтереси та світогляд ще до початку реального спілкування. Інший користувач, взаємодіючи з ШІ-ботом, зможе зробити висновки про потенційного партнера на основі стилю його віртуальної поведінки. Це, у свою чергу, дозволить зменшити витрати часу на спілкування з тими, з ким ймовірна несумісність.

1.3 Класичні архітектури для роботи з текстовою інформацією

Текст можна представити як послідовність символів або токенів. Символи в одному місці дають корисну інформацію про символи в іншому місці. Звичайні багатосарові перцептрони чудово підходять для задач, де вхідні данні незалежні, але вони не розрізняють послідовності і не мають явного механізму для врахування контексту в таких даних, які пов'язані в часі або просторі.

У 1986 році М. Джордан запропонував мережу, яка є штучною нейронною мережею прямого поширення, доповненою шаром контекстних нейронів, кількість яких збігається з кількістю виходів штучної нейронної мережі. Вхідні сигнали мережі разом із сигналами контекстного шару надходять на входи нейронів прихованого шару, вихідні сигнали яких передаються на вихідний шар. Виходи нейронів цього шару є виходами штучної нейронної мережі. Крім того, вихідні сигнали у вигляді сигналів зворотного зв'язку передаються з вагами на контекстний шар. Нейрони контекстного шару мають власні зворотні зв'язки з іншою вагою, що звичайно не змінюється.

Архітектура, запропонована Дж. Елмана у 1990 році, була простішою за попередню архітектуру Джордана. Мережа Елмана є модифікацією мережі Джордана, у якій сигнали зворотних зв'язків надходять не з вихідного шару, а з виходів нейронів прихованого шару, тому кількість нейронів контекстного й прихованого шарів збігається. Крім того, тут немає власних зворотних зв'язків нейронів контекстного шару.

Основним завданням нейронів прихованого шару є генерація бажаного вихідного сигналу, який запам'ятовує й порівнює знову вхідний образ, що запам'ятовується в контекстному шарі (рисунок 1.4) [7].

Ця архітектура є еквівалентною простій RNN, у якій кожен прихований шар має одне рекурентне ребро (рисунок 1.5).

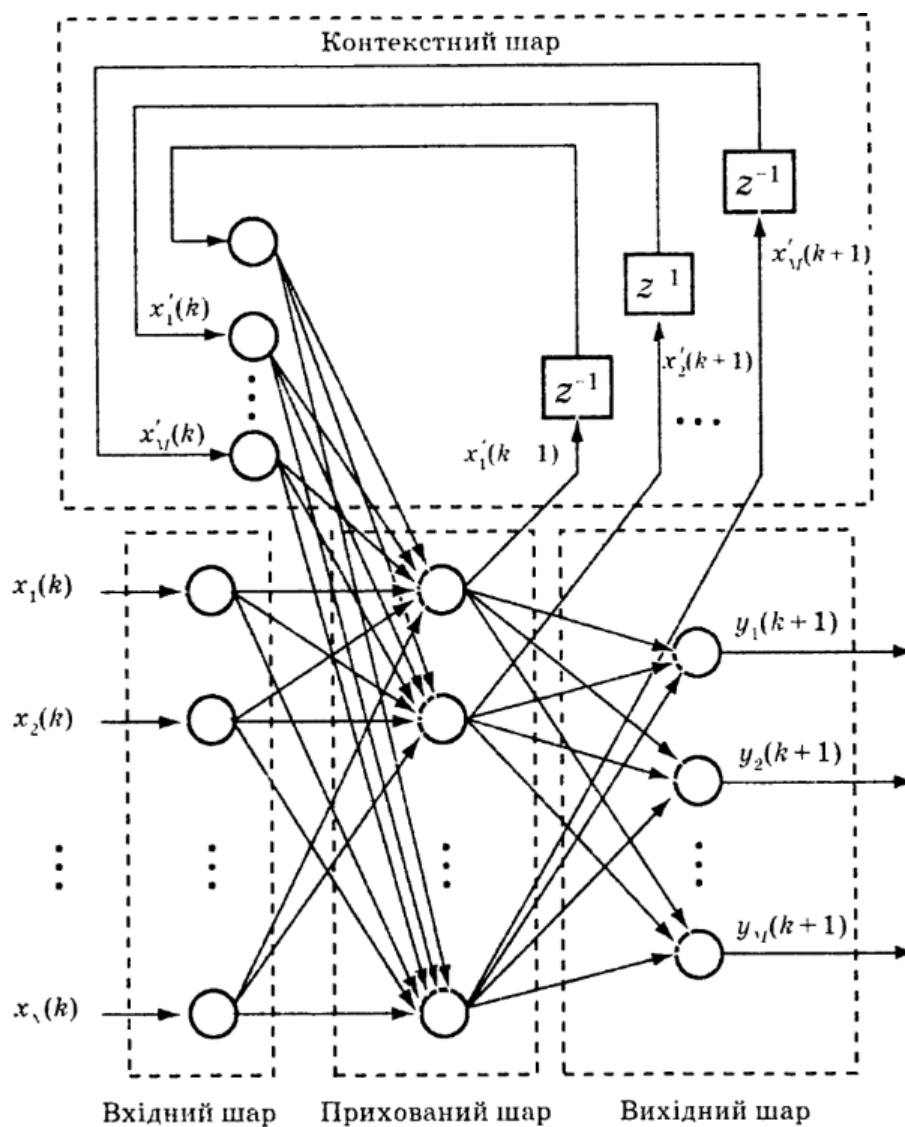


Рисунок 1.4 – Мережа Елмана

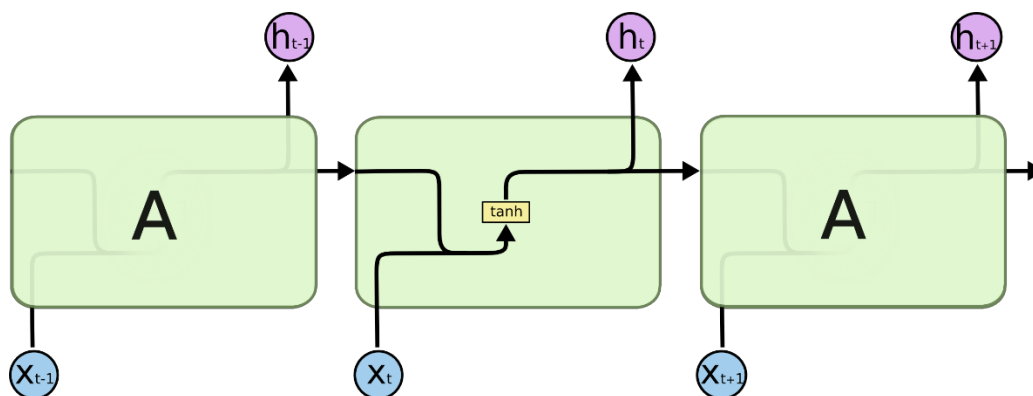


Рисунок 1.5 – Архітектура простої рекурентної нейронної мережі

На рисунку представлено одну з можливих архітектур RNN, розгорнутої в часі. На кожному кроці модель формує вихідний сигнал h_t . Цей підхід називається many-to-many і підходить для аналізу послідовності ДНК, або машинного перекладу.

Підхід many-to-one на кожному кроці приймає вхідний сигнал, але генерує єдиний вихідний результат після завершення обробки всієї послідовності. Така архітектура підходить для генерації та класифікації тексту.

Підхід one-to-many приймає єдиний вхідний сигнал, а на кожному наступному кроці генерує вихідний сигнал h_t , який надалі подається на вхід моделі. Така архітектура може підходити для задачі генерації підписів до зображень.

Long Short-Term Memory (LSTM) – це вдосконалений варіант рекурентних нейронних мереж, який вирішує проблему захоплення довготривалих залежностей та проблему зникнення градієнтів. LSTM складається з двох основних компонентів – внутрішній стан та шлязи (рисунок 1.6).

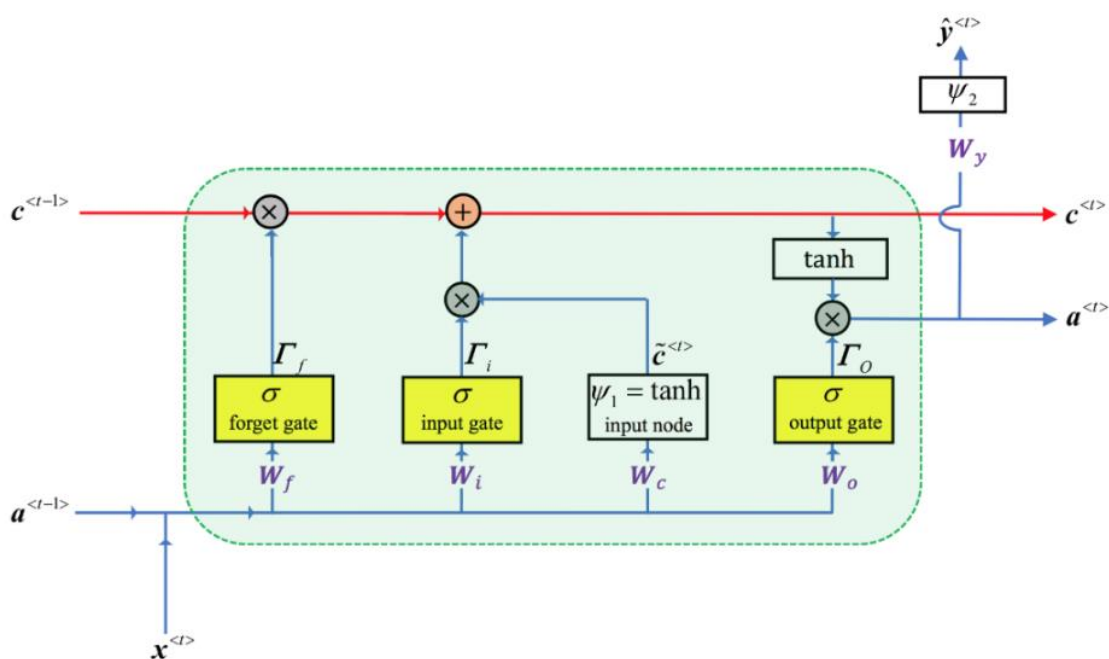


Рисунок 1.6 – Архітектура LSTM

Внутрішній стан можна уявити як пам'ять LSTM і він пов'язаний із ребром, що з'єднує суміжні часові кроки з фіксованою вагою. Оскільки вага залишається постійною, похибка може передаватися через кілька часових кроків без зникнення чи вибуху градієнту. Шлюзи можна поділити на три типи – вхідного, забуття та вихідного. Кожен шлюз виконує конкретну функцію. Вхідний шлюз визначає, як оновити внутрішній стан на основі поточного входу та попереднього внутрішнього стану. Шлюз забуття визначає, скільки з попереднього внутрішнього стану слід забути. Вихідний шлюз регулює вплив внутрішнього стану на систему [8].

Двонапрямна довготривала короткочасна пам'ять (Bi-LSTM) є розширенням архітектури LSTM, яке усуває обмеження стандартних моделей LSTM шляхом врахування як минулого, так і майбутнього контексту в задачах моделювання послідовностей. У той час як традиційні моделі LSTM обробляють вхідні дані лише у прямому напрямку, Bi-LSTM долає це обмеження, навчаючи модель у двох напрямках: вперед і назад

Bi-LSTM складається з двох паралельних шарів LSTM: один обробляє вхідну послідовність у прямому напрямку, тоді як інший обробляє її у зворотному напрямку. Така двонапрямна обробка дозволяє моделі захоплювати інформацію як з минулого, так і з майбутнього контексту, забезпечуючи більш повне розуміння часових залежностей у послідовності [9].

Gated Recurrent Unit (GRU) є спрощеним варіантом LSTM. Внутрішня структура GRU є простішою порівняно з LSTM і легше піддається навчанню, оскільки потребує менше обчислень.

Ці спрощення досягаються головним чином двома способами. По-перше вхідний шлюз і шлюз забування об'єднано в один шлюз, який називається шлюзом оновлення. По-друге внутрішній стан і прихований стан об'єднані в одне (рисунок 1.7). Ці шлюзи дозволяють GRU вибірково оновлювати та використовувати інформацію з попередніх часових кроків, що дає змогу захоплювати довгострокові залежності в послідовностях

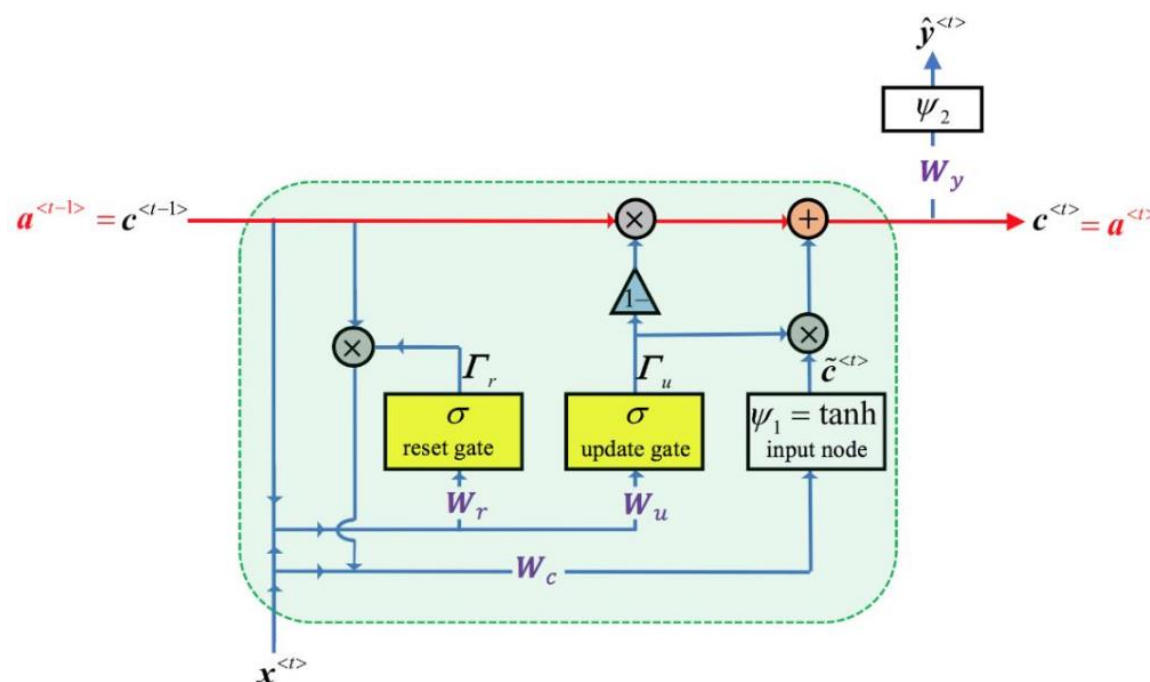


Рисунок 1.7 – Архітектура GRU

Однією з основних проблем послідовних моделей, таких як RNN, є затухання або вибух градієнтів. Хоча в LSTM вийшло частково вирішити цю проблему, при занадто великих послідовностях градієнти можуть ставати надто малими або великими. Крім того, послідовні моделі мають обмежену здатність до збереження довготривалої залежності, що знижує їхню ефективність при обробці тривалих послідовностей. Важливим для таких моделей є лише найближчий контекст.

Ще однією суттєвою проблемою є послідовна обробка. Майбутні кроки залежать від попередніх. Це ускладнює паралелізацію процесу навчання. Сучасна архітектура Transformer змогла вирішити всі ці проблеми.

Варто зазначити, що використання механізму уваги не є унікальним для архітектури Transformer. Він застосовувався і раніше в поєднанні з іншими моделями, такими як RNN, LSTM та CNN [10].

1.4 Архітектура Transformer

Всі сучасні великі мовні моделі, тією чи іншою мірою базуються на архітектурі Transformer – першій моделі, яка повністю покладається на механізм самоуваги (self-attention) без використання рекурентних нейронних мереж (RNN) або згорток.

Початково модель Transformer була розроблена для вирішення задачі машинного перекладу. Її навчали на наборах даних пар англійсько-німецьких та англійсько-французьких речень. Тому модель має структуру енкодера-декодера (рисунок 1.8).

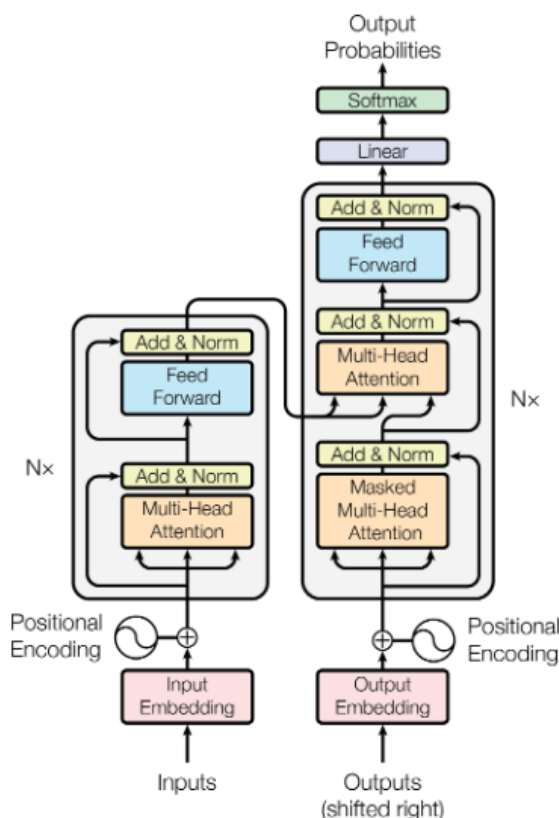


Рисунок 1.8 – Архітектура Transformer

Оригінально, енкодер відповідав за стиснення вхідних даних, а декодер за їх відновлення у вихідні. В моделі Transformer енкодер складається з шести однакових шарів, кожен шар містить два підшари –

механізм multi-head attention та нейронну мережу прямого поширення (feed-forward neural network). Вихід кожного підшару поелементно підсумовується з його вхідним значенням, після чого нормалізується. Розмір вектору ембедінгу, як і вихід із усіх шарів моделі, становить 512, щоб була можливість проводити коректне сумування на кожному етапі.

Декодер також складається з шести однакових шарів, але на відміну від енкодера, кожен шар містить не два, а три підшари – masked multi-head attention, multi-head attention та feed forward neural network. Аналогічно до енкодера, після кожного підшару виконується сумування виходів та входів та їх нормалізація. Masked multi-head attention запобігає тому, щоб токени з попередніх позицій звертались до токенів на наступних позиціях. У поєднанні з тим, що вихідні ембедінги зміщені на одну позицію, гарантується, що передбачення наступного токена базується лише на попередніх згенерованих токенах [11].

Першим етапом роботи моделі Transformer є перетворення вхідних та вихідних токенів в ембедінги – вектори розмірності d_{model} . Шар ембедінгу навчається і оптимізується під час тренування моделі. Цей метод представлення токенів суттєво перевершує класичний підхід кодування токенів за допомогою one-hot векторів, де довжина вектору дорівнює довжині словника, з одиницею у позиції відповідного токена, і нулями у всіх інших. Властивістю ембедінгів є те, що схожі токени, як правило, мають схожі вектори. Вектори токенів пов'язані з їх контекстом, це лежить в основі властивості схожості – природно, що схожі слова з'являються в схожому контексті [12].

Наступним етапом до всіх ембедінгів застосовується позиційне кодування (positional encoding). Це необхідно для того, щоб модель мала можливість використовувати інформацію про порядок токенів у послідовності, так як в моделі відсутні рекурентні або згорткові компоненти. Позиційні ембедінги мають ту ж саму довжину, що і ембедінги, щоб їх можна було підсумувати. Існує багато варіантів позиційного

кодування, найпопулярніші з них – з використанням навчених позиційних ембедингів та фіксованого за допомогою синусоїдальної та косинусоїдальної функції (формули 1.1 та 1.2).

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad (1.1)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad (1.2)$$

де pos – позиція токена в послідовності;

i – індекс у векторі ембедінгу;

d_{model} – розмір ембедінгу.

Довжини синусоїдальних хвиль формують геометричну прогресію, що дозволяє моделі розрізняти як близькі, так і далекі позиції в послідовності.

Далі позиційні ембедінги подаються на вхід енкодера та декодера відповідно. Вихід енкодера використовується у підшарі multi-head attention декодера, разом з результатом підшару masked multi-head attention.

Модель генерує по одному елементу за раз. Останнім етапом роботи моделі є передбачення вірогідності наступного токена. Для цієї задачі використовується функція активації softmax. У моделі Transformer використовується одна і та ж сама матриця ваг як для двох шарів ембедінгу, так і для лінійної трансформації перед softmax. Вихідний вектор масштабується так, щоб всі елементи вектору лежали в діапазоні від 0 до одиниці і в сумі дорівнювали 1. Значення кожного елементу дорівнює вірогідності, вибір наступного токена здійснюється випадково, відповідно до цих ймовірностей.

Для керування рівнем випадковості результату застосовується параметр температури, який контролює, наскільки несподіваним або передбачуваним буде наступне слово. Масштабування виконується шляхом

ділення вектора логітів на значення T , яке позначає температуру, після чого застосовується softmax (формула 1.3) [13]. Логіти – вектор, який поступає на вхід функції softmax.

$$\text{Softmax}(x_i) = \frac{\exp(x_i/T)}{\sum_j \exp(x_j/T)}, \quad (1.3)$$

де x – вектор логітів;

T – температура.

Чим значення параметру температури ближче до нуля, тим більш передбачуваними будуть результати, тобто модель зазвичай обирає найбільш імовірний токен. Чим більше параметр, тим більше розподіл наближається до рівномірного, надаючи кожному токену однакову ймовірність бути вибраним.

Модель є авто-регресивною, тобто використовує раніше згенеровані токени як додатковий вхід на декодер при генерації наступного. Отже алгоритм повторюється до моменту, доки не буде досягнуто максимального ліміту довжини нових токенів або не буде згенеровано спеціальний токен, що сигналізує про кінець послідовності.

1.5 Архітектура GPT

Оригінальна архітектура Transformer продемонструвала високу ефективність на таких задачах як машинний переклад, синтаксичний аналіз, але не достатньо добре на задачі генерації тексту. Для вирішення цієї задачі команда OpenAI представила архітектуру GPT (Generative pre-trained transformer), яка базується на оригінальному Transformer. Головна відмінність, що модель GPT складається лише з декодера, з 12 шарами з лише підшарами masked multi-head attention та нейронної мережі прямого поширення (рисунок 1.9).

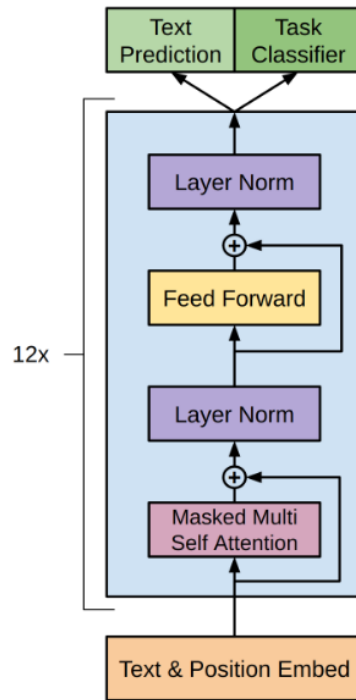


Рисунок 1.9 – Архітектура GPT

Процедура навчання складається з двох етапів. Перший етап без вчителя (unsupervised pre-training) полягає у навчанні високопродуктивної мовної моделі на великому корпусі тексту. За ним іде етап донавчання з вчителем (supervised fine-tuning), на якому модель адаптується до специфічних завдань з використанням анотованих даних [14].

Для деяких завдань, таких як класифікація тексту, можна безпосередньо донавчити модель, але такі завдання, як відповіді на запитання або встановлення текстового слідування, мають структуровані входи, такі як впорядковані пари речень – запитання та відповідей. Перша модель GPT була навчена на суцільних послідовностях тексту, тому без змін, її не можна було застосувати до таких завдань. Реалізувати завдання відповіді на запитання вийшло лише з виходом моделі GPT-3.5.

Під час донавчання моделі GPT-3 додатково використовувалось навчання з підкріпленням із зворотним зв'язком від людини. Навчання з підкріпленням (reinforcement learning) – це галузь машинного навчання,

присвячена навчанню моделей діяти в середовищі з метою максимізації їхньої корисності у досягненні певних цілей. Стратегія, якою модель користується для досягнення цілей називається політикою.

Техніка навчання з підкріпленням із зворотним зв'язком використовує людські вподобання як сигнал винагороди для донавчання моделей. Алгоритм можна розділити на декілька кроків:

- крок 1 – збір демонстраційних даних і навчання керованої політики. Модель донавчають на анотованих даних за допомогою навчання з учителем, щоб досягти бажаної поведінки;

- крок 2 – збір порівняльних даних і навчання моделі винагороди. Збирається набір даних, що містить порівняння між виходами політики, у яких розмітники даних вказують, якому з варіантів вони віддають перевагу для заданого запиту. Далі навчається модель винагороди (reward model), яка передбачає, який варіант обере людина;

- крок 3 – оптимізація політики відносно моделі винагороди за допомогою PPO. Вихід моделі винагороди сприймається як скалярна винагорода. Далі політику, натреновану з учителем, донавчають для оптимізації цієї винагороди за допомогою алгоритму PPO.

Кроки 2 і 3 можна повторювати по колу: збір нових порівняльних даних проводиться на основі поточної найкращої політики, які потім використовуються для навчання нової моделі винагороди, і потім – нової політики [15].

PPO – сімейство методів градієнтної оптимізації політики для навчання з підкріпленням, які чергують між вибіркою даних шляхом взаємодії з середовищем та оптимізацією «сурогатної» цільової функції за допомогою стохастичного градієнтного підйому. Якщо стандартні методи градієнта політики виконують лише одне оновлення градієнта на кожен зразок даних, ці методи пропонують нову цільову функцію, яка дозволяє проводити декілька епох оновлень на міні-батчах [16].

1.6 Шар уваги

Функцію уваги можна описати як відображення запиту та набору пар ключ-значення у вихід, де запит, ключі, значення (query, key, value) та вихід – це все вектори. Вихід обчислюється як зважена сума значень, де вага кожного значення обчислюється за допомогою функції сумісності запиту з відповідним ключем. На вхід подаються запити та ключі розмірності d_k , а також значення розмірності d_v . Ключі, запити та значення формуються шляхом множення ембедінгів на окремі матриці ваг, які навчаються разом із моделлю.

Дві найпоширеніші функції уваги – це адитивна увага та увага на основі скалярного добутку (мультиплікативна). Адитивна увага обчислює функцію сумісності за допомогою повнозв'язної нейронної мережі з одним прихованим шаром. Спочатку використовувалася в ранніх моделях уваги до появи більш складних методів, таких як самоувага.

Мультиплікативна увага обчислює оцінки уваги за допомогою скалярного добутку векторів запиту та ключа. Хоча обидві функції мають подібну теоретичну складність, увага на основі скалярного добутку є значно швидшою та ефективнішою за обсягом пам'яті на практиці, оскільки може бути реалізована через високоефективні матричні множення.

При невеликих значеннях d_k обидва механізми працюють схоже, однак при великих значеннях d_k адитивна увага перевершує увагу на основі скалярного добутку.

Це пов'язано з тим, що при великих d_k скалярні добутки мають великі значення за модулем, що зсуває softmax у області з дуже малими градієнтами. Архітектура Transformer бере в основу мультиплікативну увагу, але щоб усунути цей ефект, скалярні добутки масштабуються на коефіцієнт $\frac{1}{\sqrt{d_k}}$ (формула 1.4).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (1.4)$$

Така реалізація уваги називається Scaled Dot-Product Attention – масштабована увага на основі скалярного добутку. На рисунку 1.10 зображено візуалізацію цієї формули.

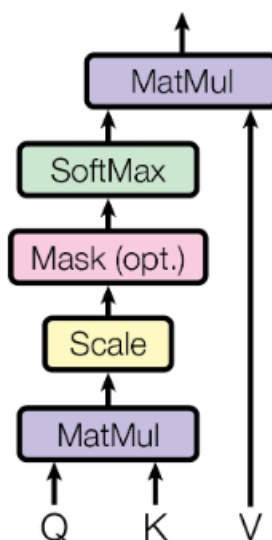


Рисунок 1.10 – Алгоритм масштабованої уваги на основі скалярного добутку

Алгоритм містить необов’язкову операцію маскування – всі значення, що йдуть в послідовності далі встановлюються в $-\infty$. Тобто, до матриці, що була отримана після масштабування додається матриця того ж розміру, де всі значення, що над головною діагоналлю дорівнюють до $-\infty$, всі інші значення дорівнюють 0. Це корисно в задачах, де необхідно запобігти поширенню інформації ліворуч. Наприклад, в задачі генерації тексту.

Як було описано раніше, саме механізм самоуваги зробив архітектуру Transformers настільки ефективною. Ідея в тому, що запити, ключі і значення походять з одного і того самого джерела. Саме цей механізм дозволяє вирішити проблему послідовних моделей, які мають маленьке контекстне

вікно, механізм дозволяє виявляти зв'язки та залежності між токенами в контексті всього речення.

В архітектурі енкодер-декодер запити надходять із попереднього шару декодера, а ключі та значення – з виходу енкодера.

Багатоголова самоувага (multi-head self-attention) – це вдосконалена техніка, яка розширює стандартний механізм самоуваги, виконуючи його кілька разів паралельно. Кожна голова навчається виявляти різні шаблони або зосереджуватися на різних аспектах вхідних даних. Замість того щоб виконувати одну функцію уваги з ключами, значеннями та запитами розмірності d_{model} , вони трансформуються за допомогою навчуваних ваг до розмірностей d_k та d_v . Вихідні значення кожної голови дорівнюють розмірності d_v . Розмірності кожної голови конкатенуються, формуючи вихідні дані шару уваги. Щоб не трансформувати вихідні дані до розмірності вхідних з використанням ще однієї матриці ваг зручно взяти розміри d_k та d_v , які дорівнюють d_{model}/h , де h – це кількість голів (рисунок 1.11).

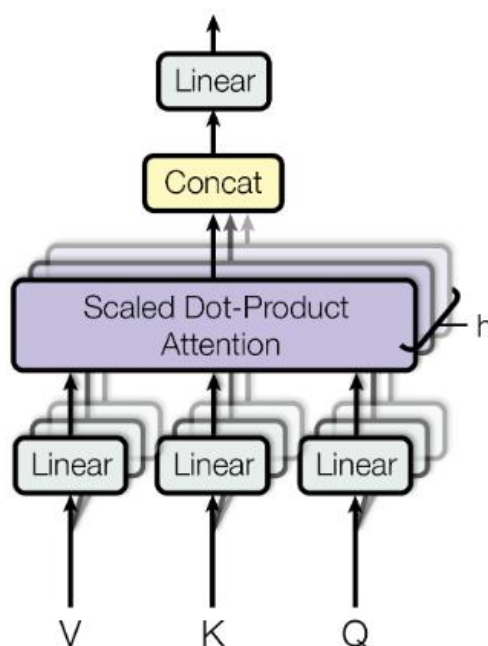


Рисунок 1.11 – Алгоритм багатоголової уваги

1.7 Feed-forward neural network

Після кожного підшару уваги, вихідні данні передаються до нейронної мережі прямого поширення, яка застосовується окремо та однаково до кожної позиції. Її можна описати як два повнозв'язних шари, з функцією активації ReLU між ними (формула 1.5).

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2, \quad (1.5)$$

де x – вхідний вектор;

W_i – ваги повнозв'язного шару i ;

b_i – зсув повнозв'язного шару i .

Хоча повнозв'язні шари для всіх позицій однакові у межах одного підшару, вони використовують різні параметри в різних шарах.

1.8 Методи оптимізації роботи шару уваги

Авторегресивна генерація є потужним механізмом, який дозволяє великим мовним моделям створювати якісний, контекстуально узгоджений текст. Однак вона створює обчислювальні виклики при обробці довгих послідовностей, оскільки ключі і значення потрібно обчислювати заново для кожного токена під час процесу генерації.

Оптимізація через KV-кешування вирішує цю проблему, зберігаючи раніше обчислені ключі та значення та повторно використовуючи їх для наступної генерації токенів, тим самим зменшуючи зайві обчислення та підвищуючи ефективність інференсу.

На кожному кроці модель виконує механізм самоуваги над усією послідовністю $X = [x_1, x_2, \dots, x_t]$, щоб згенерувати наступний токен x_{t+1} . Цей процес вимагає обчислення матриць ключів і значень для всіх раніше оброблених токенів.

При генерації токена x_t модель вже обчислила ключі та значення для токенів у $X[1 : x_{t-1}]$. KV-кешування оптимізує цей процес, зберігаючи раніше обчислені матриці ключів і значень для $X[1 : x_{t-1}]$ та повторно використовуючи їх, що дозволяє обчислювати ключі та значення лише для нового токена x_t .

На кроці t , обчислюється лише вектори q_t^i , k_t^i , v_t^i для нового токена, використовуючи його ембедінг. Нові обчислені k_t^i та v_t^i потім додаються до кешованих матриць ключів та значень з попередніх кроків. Ці кешовані матриці далі використовуються в обчисленні масштабованої уваги на основі скалярного добутку для токена x_t . Кешування q_t^i не потрібно для коректної роботи уваги.

Потім аналогічний процес повторного використання KV можна застосувати до різних голів уваги в кожному шарі великої мовної моделі. Для кожного токена зекономлений час обчислення виникає через уникнення повторного обчислення і пропорційний кількості кешованих токенів [17].

KV-кешування вирішило проблему складності обчислення, але створило нове вузьке місце, яке обмежує ефективність інференсу, бо вимагає велику кількість пам'яті для збереження об'ємних матриць ключів і значень. Для одного токена необхідно кешування, що дорівнює $2n_h d_h l$, де l – кількість блоків уваги, n_h – кількість голів в блоці уваги, d_h – розмір однієї голови. Наприклад, при кількості блоків уваги 64, кількості голів 128 та розміру голови 128, то при використанні типу даних float16, що вимагає 2 байти, розмір кешування одного для токена буде дорівнювати 4 мегабайти. А розмір контексту в сучасних моделях дорівнює десятки тисяч токенів, тому при контексті 16384 токенів, розмір KV-кешування буде вимагати 64 гігабайти пам'яті.

Найпростіший механізм оптимізації це Multi-Query Attention, в якому всі голови використовують один спільний набір ключів і значень. В цьому методі для одного токена необхідно кешування $2d_h l$, що згідно з прикладом вище буде вимагати 32 кілобайти, що в 128 разів менше за звичайну

багатоголову увагу. Але з цим і сильно деградує сама модель, яка втрачає можливість виявляти довготривалі залежності [18].

Інший механізм Grouped-Query Attention розділяє голови запитів на G груп, кожна з яких спільно використовує одну голову ключів і одну голову значень. В цьому методі для одного токена необхідно кешування $2n_g d_h l$, де n_g – кількість груп. При 8 групах, згідно з прикладом вище для кешування одного токена необхідно 256 кілобайти, що в 16 разів менше за звичайну багатоголову увагу. Якість моделі краща, за Multi-Query Attention, але все одно нижче за звичайну [19].

Найбільш інноваційним механізмом уваги, який перевищує за якістю моделі навіть багатоголову увагу, є Multi-Head Latent Attention, що запропонували вперше для моделі DeepSeek-V2. Головною ідеєю є стиснення ключів і значень для зменшення KV-кешування в окремий спільний для всіх голів латентний вектор (формули 1.6–1.8).

$$c_t^{KV} = W^{DKV} h_t, \quad (1.6)$$

$$k_t^C = W^{UK} c_t^{KV}, \quad (1.7)$$

$$v_t^C = W^{UV} c_t^{KV}, \quad (1.8)$$

де c_t^{KV} – стиснений латентний вектор для ключів і значень;

W^{DKV} – матриця ваг пониження розмірності;

W^{UK}, W^{UV} – матриця ваг підвищення розмірності для ключів і значень;

h_t – вхідний вектор.

На рисунку 1.12 зображена візуалізація Multi-Head Latent Attention. В цьому методі для одного токена необхідно кешування $d_h^R l$, де d_h^R – розмірність стиснення KV. При d_h^R , що дорівнює 1024, згідно з прикладом

вище для кешування одного токєну необхідно 64 кілобайти, що в 64 рази менше за звичайну багатоголову увагу. Але якість перевищує звичайну.

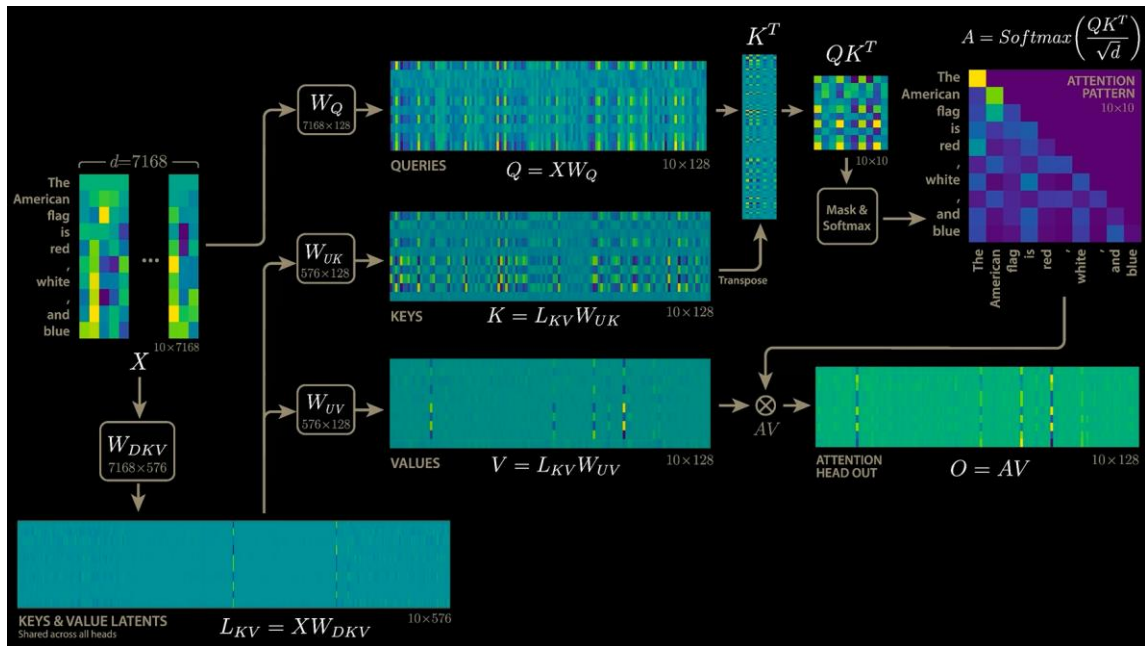


Рисунок 1.12 – Алгоритм Multi-Head Latent Attention

Під час інференсу MLA потребує зберігати лише c_t^{KV} , крім того, під час інференсу матриці W^{UK} можна «поглинути» у W^Q , а W^{UV} у W^O , тож навіть немає потреби явно обчислювати ключі й значення для уваги [20].

На рисунку 1.13 схематично зображено порівняння всіх методів оптимізації KV-кешування.

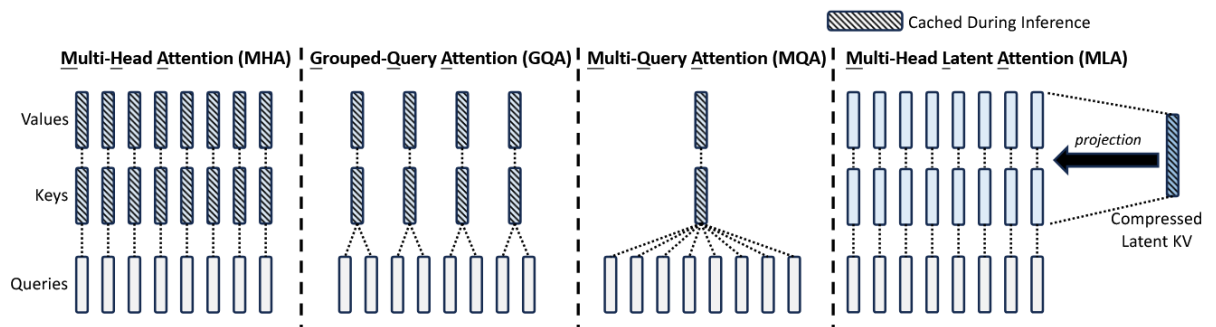


Рисунок 1.13 – Методи оптимізації KV-кешування

1.9 Методи та стратегії донавчання великих мовних моделей

Хоча великі мовні моделі навчаються на великих об'ємах даних і володіють загальними знаннями, вони можуть не демонструвати оптимальних результатів у конкретних завданнях без додаткового донавчання.

Тонке налаштування (fine-tuning) використовує попередньо натреновану модель, як основу. Цей процес передбачає подальше навчання на меншому, вузькоспеціалізованому наборі даних. Такий підхід спирається на вже наявні знання моделі, покращуючи її ефективність у конкретних завданнях при знижених вимогах до обсягу даних та обчислювальних ресурсів.

Існує три типи донавчання – з вчителем, без вчителя та шляхом розробки промптів. Доновчання без вчителя не потребує розмічених даних. Натомість LLM використовує великий корпус немаркованого тексту з цільової предметної галузі, уточнюючи своє розуміння мови. Такий підхід є корисним для нових сфер, таких як юридична і медична.

Доновчання з вчителем передбачає надання LLM розмічених даних, адаптованих під цільове завдання. Наприклад, для донавчання LLM на класифікацію текстів у бізнес-контексті використовується набір текстових фрагментів із відповідними мітками класів.

Тонке налаштування за інструкціями шляхом розробки промптів базується на поданні LLM інструкцій у вигляді природної мови, що корисно для створення спеціалізованих асистентів. Цей метод зменшує потребу у великому обсязі розмічених даних, але сильно залежить від якості сформульованих промптів.

Першим етапом треба змінити вихідний шар моделі та налаштувати його відповідно до специфічних вимог цільового завдання. Це може включати зміну наявних шарів або додавання нових. Наприклад, для класифікації може знадобитися softmax із відповідною кількістю класів, а

для генерації тексту – зміни в механізмі декодера. Другим етапом треба обрати відповідну стратегію донавчання, яка найкраще відповідає завданню та архітектурі моделі. Деякі з варіантів:

- тематичне донавчання під завдання – для задач як генерація коду, класифікація. Потребує використання відповідного набору даних;
- донавчання під домен – адаптування моделі до специфіки певної сфери, наприклад: медицина, фінанси, юриспруденція;
- ефективне налаштування параметрів (PEFT) – техніки як LoRA, QLoRA, дозволяють зменшити обчислювальні витрати;
- часткове донавчання (HFT) – компроміс між збереженням знань попереднього тренування та навчанням нового завдання – оновлюється лише половина параметрів на кожній ітерації.

Методи PEFT донавчають лише невелику підмножину параметрів моделі, зберігаючи більшість параметрів попередньо натренованої LLM замороженими, що суттєво знижує обчислювальні та втрати пам'яті. Такий підхід зменшує проблему катастрофічного забування – явища, коли нейронні мережі втрачають раніше набуті знання та демонструють значне погіршення продуктивності на попередньо вивчених завданнях після тренування на нових датасетах. Методи PEFT показали кращу продуктивність у порівнянні з повним донавчанням, особливо в умовах обмежених даних.

Методи, засновані на адаптерах додають додаткові треновані параметри після шарів уваги та нейронної мережі прямого поширення у замороженій попередньо натренованій моделі, з метою зменшення використання пам'яті та пришвидшення тренування. Конкретний підхід залежить від типу адаптера: це може бути додавання додаткового шару або представлення оновлень ваг ΔW у вигляді низькорівневого розкладу матриці ваг.

Low-Rank Adaptation (LoRA) – один із провідних підходів, дозволяє здійснювати додаткове навчання для окремих матриць ваг. У кожному шарі

трансформера LoRA визначає ціль для донавчання, вибираючи конкретні матриці ваг. Ці матриці дублюються в їхньому початковому вигляді, і процес донавчання виконується тільки для вибраних матриць. Матриці ваг зазнають низькорівневого розкладу (Low-Rank Matrix Decomposition), в результаті чого формуються дві матриці (рисунок 1.14)

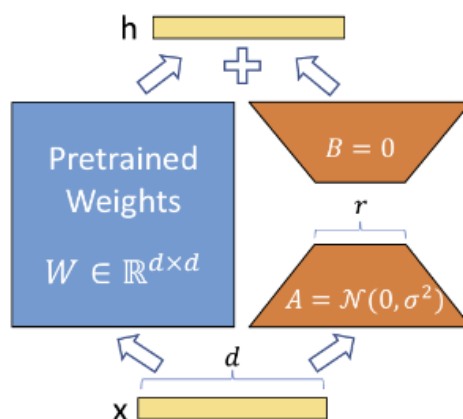


Рисунок 1.14 – Метод донавчання LoRa

Матриця A ініціалізується випадковим нормальним розподілом, тоді як матриця B на початку навчання ініціалізується нулями [21].

QLoRA – це розширена версія LoRA, розроблена для більшої ефективності використання пам'яті шляхом квантування вагових параметрів до 4-бітної точності.

Зазвичай параметри LLM зберігаються у форматі 32 біти, але QLoRA стискає їх до 4 біт, що суттєво зменшує обсяг пам'яті. Це дозволяє здійснювати донавчання на менш потужному обладнанні, навіть на домашніх GPU. Незважаючи на зниження розрядності, QLoRA зберігає рівень продуктивності, порівнянний із традиційним донавчанням у 16-бітному форматі. Це досягається шляхом зворотного поширення градієнтів через заморожену, 4-бітно квантизовану попередньо натреновану модель в адаптери з низьким рангом, що робить процес донавчання ефективним, одночасно зберігаючи результативність моделі.

Щоб забезпечити ефективний за пам'яттю інференс, LLM часто розгортають з квантизованими вагами меншої точності. Ця практика є критично важливою для широкого впровадження LLM, оскільки вона дозволяє використовувати їх на різноманітних побутових пристроях. Методи INT4, NF4 і FP4 використовують операцію масштабування (формули 1.9 та 1.10).

$$X_{INT} = \left\lfloor \frac{X_{FP16} - Z}{S} \right\rfloor, \quad (1.9)$$

$$S = \frac{\max(X_{FP16}) - \min(X_{FP16})}{2^{N-1} - 1}, \quad (1.10)$$

де X_{FP16} – вхідні дані у форматі FP16;

X_{INT} – вихідні квантизовані цілі значення;

N – кількість бітів у цільовому цілочисельному представленні;

S – коефіцієнт масштабування;

Z – нульова точка, для симетричної квантизації нульова точка дорівнює нулю, для асиметричної квантизації $Z = \min(X_{FP16})$ [22].

Weight-Decomposed Low-Rank Adaptation (DoRA) – це метод донавчання, розроблений для оптимізації попередньо натренованих моделей шляхом розкладання ваг на компоненти величини та напрямку. Цей підхід використовує ефективність LoRA для напрямлених оновлень, дозволяючи здійснювати суттєві зміни параметрів без зміни архітектури моделі [23].

1.10 Метрики оцінки якості великих мовних моделей

Для оцінки якості великих мовних моделей використовуються спеціалізовані метрики для розрахунку подібності токенів, вони вимірюють

схожість між текстами, згенерованими великими мовними моделями, і еталонними.

Perplexity – зосереджується на вимірюванні ймовірності появи еталонної послідовності $\{y_j\}_{j=1}^M$ згідно з LLM (формула 1.11). Чим значення менше, тим модель краще передбачає данні.

$$\text{Perplexity} = 2^{-\frac{1}{M} \sum_{j=1}^M \log_2 \hat{P}(y_j)}, \quad (1.11)$$

де M – довжина еталонної послідовності;

$\hat{P}(y_j)$ – ймовірність, яку LLM призначає j -му токену в еталонній послідовності.

BLEU оцінює LLM на основі n -грам. n -грама – це групування послідовностей токенів для вимірювання співпадінь токенів. Дві n -грами вважаються співпадаючими, якщо кожен елемент у них збігається у відповідному порядку, тобто $x_i = y_i$. Точність між згенерованим та еталонним текстом розраховується за формулою 1.12.

$$\text{Precision} = \frac{m}{N - n + 1}, \quad (1.12)$$

де m – кількість співпадаючих n -грам;

N – кількість токенів у згенерованому тексті;

n – розмір n -грами.

BLEU вводить також штраф за стислість, щоб карати випадки, коли згенерований текст є занадто коротким і лише частково збігається з еталоном, пропускаючи важливу інформацію (формула 1.13).

$$\text{Brevity Penalty(BP)} = \min\left(1, \frac{N}{M}\right), \quad (1.13)$$

де N – кількість токенів у згенерованому тексті;

M – кількість токенів в еталонному тексті.

Формула BLEU зображена на формулі 1.14

$$\text{BLEU} = \text{BP} \times \exp(\log(\text{Precision})). \quad (1.14)$$

ROUGE-n – це F1-метрики на рівні n-грам (формули 1.15 та 1.16).

$$\text{Recall} = \frac{m}{M - n + 1}, \quad (1.15)$$

$$\text{ROUGE-n} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \quad (1.16)$$

де m – кількість співпадаючих n-грам;

M – кількість токенів в еталонному тексті;

n – розмір n-грами [24].

2 АНАЛІЗ КОНКУРЕНТІВ ТА ПОСТАНОВКА ЗАВДАННЯ

2.1 Аналіз конкурентів у галузі сервісів для знайомств

Одним з найбільш популярних застосунків для онлайн-знайомств є Tinder [25]. Логотип містить полум'я, або «іскру», яка з'являється під час взаємної симпатії між користувачами. Застосунок має інтуїтивно зрозумілий інтерфейс (рисунок 2.1), який дозволяє користувачам швидко створити профіль, додавши фотографію, зазначивши вік, інтереси та мету використання сервісу. На основі вказаних параметрів система автоматично пропонує релевантні анкети.



- 1 - Зона інформації анкети
- 2 - Повернутись до попередньої анкети (преміум)
- 3 - Наступна анкета
- 4 - "Суперлайк" (преміум)
- 5 - "Лайк", анкета подобається
- 6 - Надіслати повідомлення (преміум)
- 7 - Головна сторінка з анкетами
- 8 - Тематичні знайомства
- 9 - Люди, яким сподобалась анкета користувача (преміум)
- 10 - Список повідомлень
- 11 - Анкета користувача
- 12 - Фільтри анкет

Рисунок 2.1 – Інтерфейс застосунку Tinder

Основна взаємодія реалізована за допомогою «свайпів». Вправо – у разі зацікавленості, вліво – при її відсутності. У випадку взаємного інтересу

між користувачами відкривається чат для подальшого спілкування. Такий формат дає змогу переглядати велику кількість профілів навіть упродовж короткого періоду.

Механізм «свайпів» став основним елементом взаємодії в більшості сучасних застосунків для знайомств і на сьогодні активно використовується не лише в Tinder, а й в інших подібних сервісах.

Застосунок надає користувачам широкий перелік полів для заповнення профілю, що дозволяє показувати більше релевантних анкет. Для повного доступу до функціональності необхідно завантажити щонайменше чотири фотографії. У разі меншої кількості зображень користувач втрачає можливість переглядати всі фотографії інших анкет. Також користувачам пропонується заповнити: короткий опис, перелік інтересів, мови, місто, місце навчання і роботи, відповіді на випадкові запитання, які ще краще розкривають особистість, а також додаткову інформацію – зріст, знак зодіаку, тип особистості тощо.

Однією з функцій Tinder є розумний підбір фотографій, коли система автоматично визначає найкращі зображення з галереї користувача для встановлення як фото профілю та їх послідовність.

Крім того, користувачі мають можливість провести пошук за критеріями – максимальною відстанню, віковим діапазоном, статтю тощо. Для користувачів із підпискою «Premium» доступний розширений набір фільтрів, зокрема пошук за інтересами.

Однією з головних особливостей Tinder є те, що користувачі не можуть дізнатися про взаємну симпатію до моменту, поки обидві сторони не виконають «свайп» вправо. Такий підхід мотивує користувачів переглядати більшу кількість профілів. З іншої сторони, існує ризик втрати потенційно сумісного партнера. Однак користувачі з активною підпискою мають можливість бачити перелік тих, хто поставив їм «вподобання», без необхідності переглядати всі анкети вручну.

Підписка надає велику кількість переваг, такі як – повернення до попереднього профілю у разі помилкового «свайпу», надсилання перших повідомлень з метою підвищення ймовірності збігу. Станом на сьогодні Tinder використовує близько 75 мільйонів користувачів щомісячно, з яких понад 9,2 мільйона мають платні підписки.

Ще одним із найпопулярніших застосунків для онлайн-знайомств є Badoo [26]. Додаток має подібний до Tinder інтуїтивно зрозумілий інтерфейс, однак має декілька нововведень (рисунок 2.2).

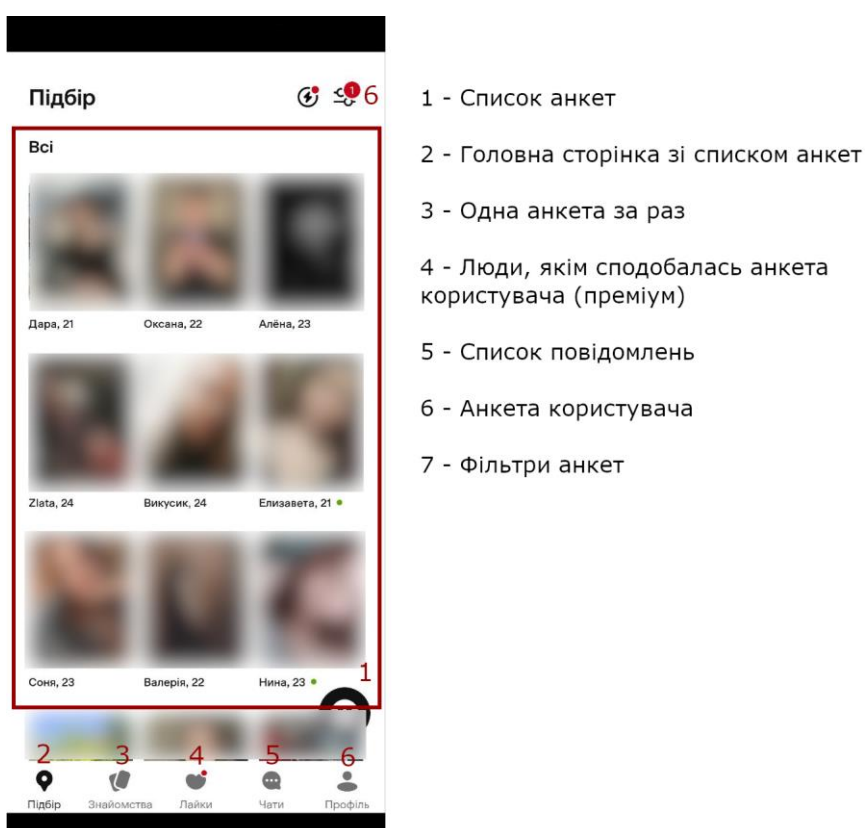


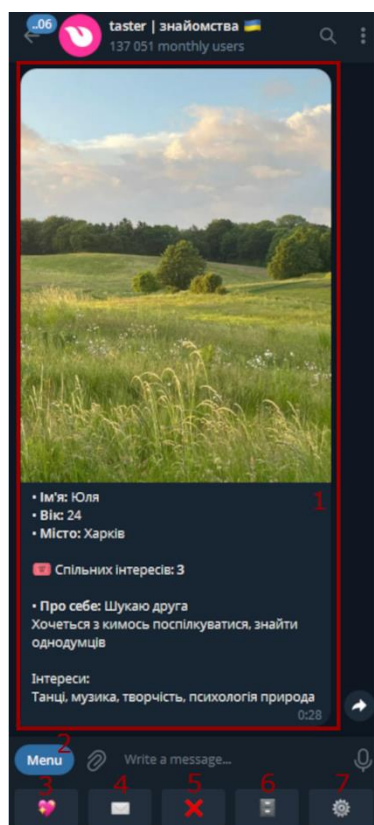
Рисунок 2.2 – Інтерфейс застосунку Badoo

Так само, як і у Tinder, у Badoo реалізована система «свайпів». Проте на відміну від Tinder, де користувач бачить лише одну анкету за раз, у Badoo доступний повноцінний список профілів, які відповідають встановленим фільтрам.

Ще однією відмінністю є розширені можливості у чаті. У той час як Tinder дозволяє обмінюватися лише текстовими повідомленнями та GIF-зображеннями, Badoo надає змогу надсилати фотографії та голосові повідомлення.

На сьогодні застосунок має близько 45 мільйонів активних користувачів, з яких 1.34 мільйона користуються платною підпискою.

Ще одним популярним способом знайомств є використання чат-ботів у соціальних мережах. Такий формат дозволяє взаємодіяти без необхідності встановлення окремих мобільних застосунків, а інтерфейс взаємодії є інтуїтивно зрозумілим, оскільки обмежується функціональністю месенджера, при цьому всі елементи навігації розміщено безпосередньо у вікні чату (рисунок 2.3).



- 1 - Зона інформації анкети
- 2 - Навігація
- 3 - "Лайк", анкета подобається
- 4 - Надіслати повідомлення
- 5 - Наступна анкета
- 6 - Люди, яким сподобалась анкета користувача
- 7 - Анкета користувача

Рисунок 2.3 – Інтерфейс Telegram-боту для знайомств Taster

Один із найпопулярніших прикладів в Україні – Telegram-бот Taster [27], кількість користувачів якого перевищує 130 тисяч осіб. У даному боті реалізовано класичну механіку «свайпів», адаптовану до формату чату – для оцінки анкети користувач надсилає відповідний символ у діалог.

На відміну від більшості міжнародних платформ, Taster одразу надає доступ до списку користувачів, яким сподобалася анкета, без необхідності здійснювати додатковий пошук або придбання преміум підписки.

Водночас формат чату накладає певні обмеження. Зокрема, для переходу між розділами необхідно вручну надсилати спеціальні символи або команди, що може бути незручним або неочевидним для нових користувачів.

Важливою перевагою ботів є відсутність внутрішньої обмеженої системи обміну повідомленнями, як в класичних додатках для знайомств. Оскільки спілкування відбувається у межах месенджера, користувачі мають змогу вести діалог у довільному форматі, за винятком технічних обмежень платформи.

2.2 Аналіз конкурентів у галузі персоналізованих чат-ботів

Сама OpenAI дозволила розширити свого чат-боту для його адаптації під певні задачі. GPTs – це спосіб створити індивідуальну версію ChatGPT, яка буде кориснішою у повсякденному житті, при виконанні конкретних завдань, на роботі або вдома. Наприклад, GPTs можуть допомогти вивчити правила будь-якої настільної гри, навчити дітей математики або створити дизайн наліпок (рисунок 2.4).

Щоб створити GPT треба всього: дати інструкції, додати додаткові знання і вибрати, що саме GPT може робити – наприклад, шукати в Інтернеті, створювати зображення або аналізувати дані.

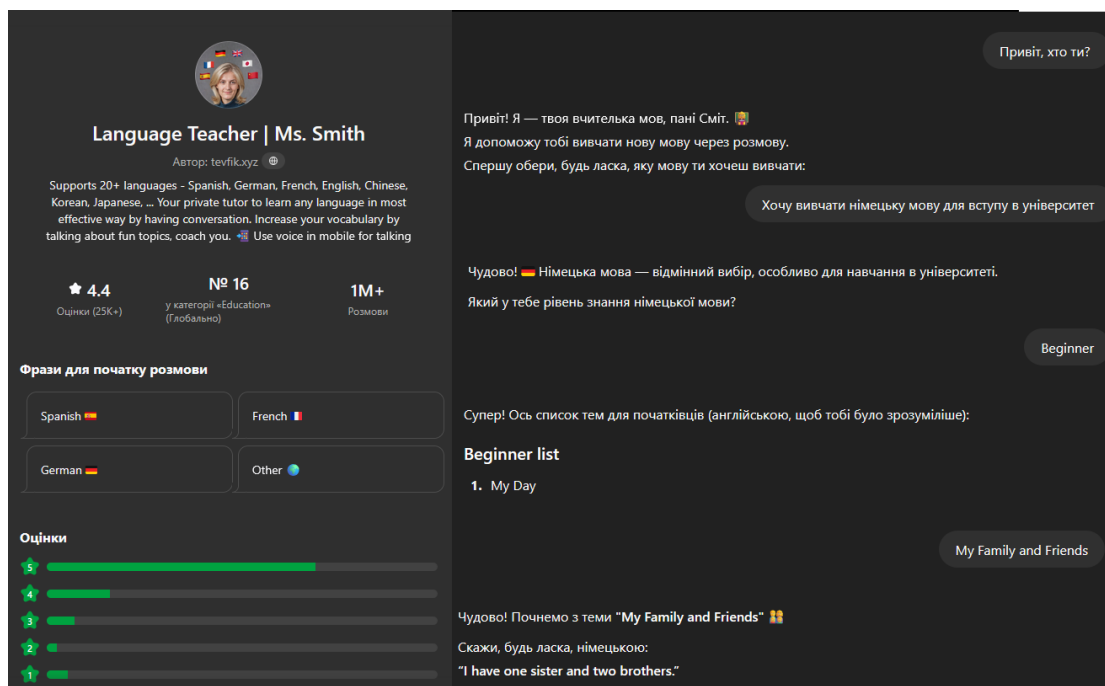


Рисунок 2.4 – Приклад Custom GPT, адаптованого під певну задачу

Крім вбудованих можливостей, розробники також можуть визначати власні дії (actions), відкривши доступ до одного або кількох API. Подібно до плагінів, дії дозволяють GPTs інтегрувати зовнішні дані або взаємодіяти з реальним світом. Наприклад, підключити базу даних з подорожами, інтегрувати пошту користувача або організувати замовлення в інтернет-магазині.

Система дій базується на досвіді з плагінів, даючи розробникам більший контроль над тим, як модель викликає API [28].

Character.AI – це сервіс, що дозволяє взаємодіяти з чат-ботами, які імітують поведінку як вигаданих, так і реальних осіб. На відміну від GPTs, де основна увага зосереджена на розширенні функціональності для виконання специфічних завдань, боти Character.AI позиціонуються як інструмент для розваг і відіграшу різноманітних сценаріїв (рисунок 2.5).

Особистість кожного персонажа формується шляхом створення опису від першої особи, встановлення вітального повідомлення, а також на основі прикладів діалогів, які редагуються та адаптуються для відповідності

бажаному стилю мовлення та ідентичності персонажа. Приклади діалогів оцінюються користувачами за допомогою рейтингової системи у вигляді зірок. Зазначена оцінка впливає на подальшу поведінку конкретного персонажа та всієї моделі в цілому.

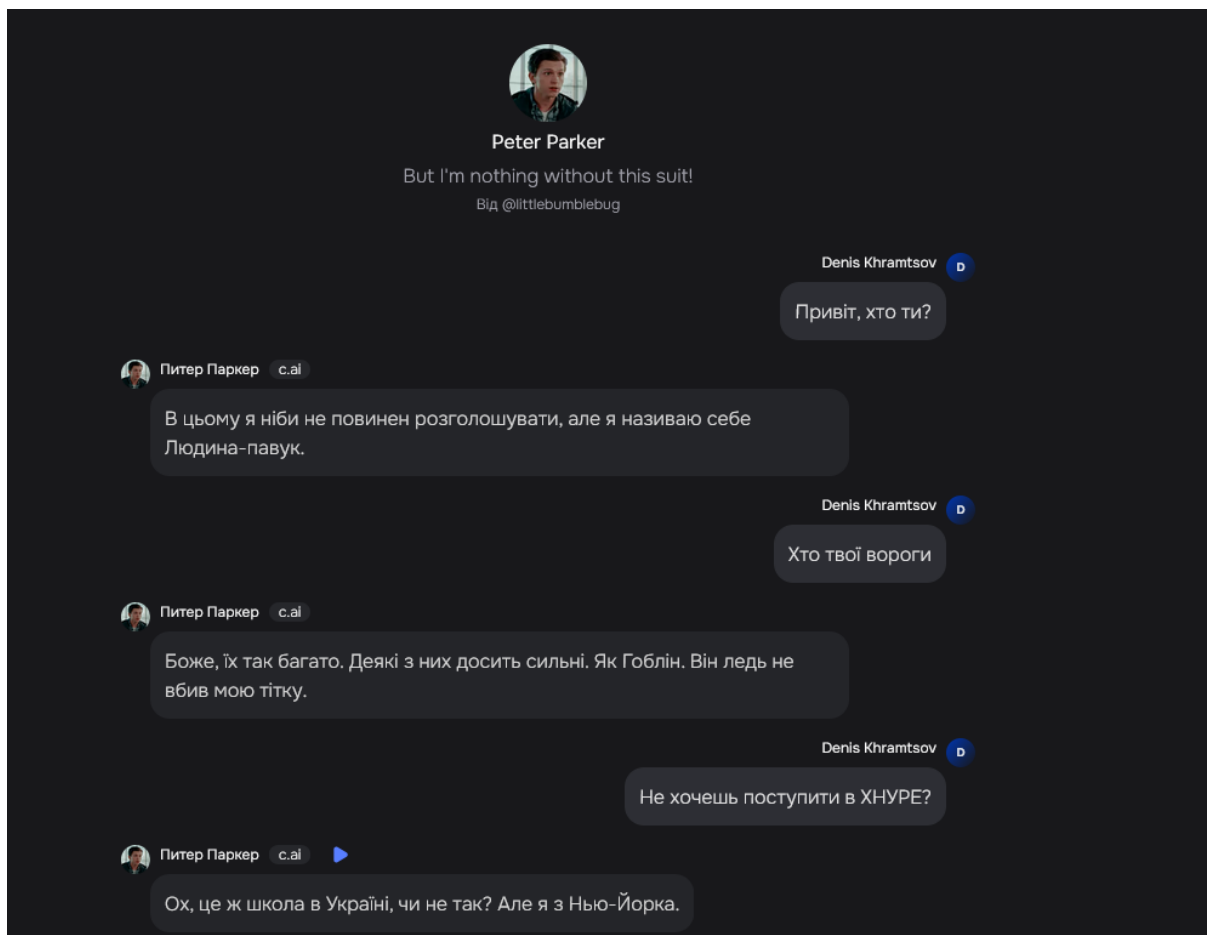


Рисунок 2.5 – Приклад діалогу з вигаданим персонажем в Character.AI

Розробники Character.AI є вихідцями з великих компаній, таких як Google та заснована у 2021 році Ноамом Шазіром та Даніелем Де Фрейтасом. Отже компанія, як один з лідерів на ринку генеративного ШІ адаптує свої моделі, та створює нові технології. Наприклад – Prompt Poet – це інструмент, який дозволяє як розробникам, так і нетехнічним користувачам ефективно проектувати та керувати промптами у виробничих середовищах. Він заощаджує час на ручних маніпуляціях зі рядками, даючи

змогу зосередитися на створенні оптимальних промптів для користувачів. Шаблони створюються з використанням комбінації YAML та Jinja2, що робить їх одночасно гнучкими та зручними для комбінування [29].

Одним з найяскравіших прикладів використання моделей для генерації тексту в розважальному форматі є Status [30] – студентський проєкт, що за декілька місяців набув надзвичайної популярності, зібравши понад мільйон завантажень у Google Play. Основна ідея полягає не в генерації повідомлень від одного користувача, а у створенні симуляції соціальної мережі з унікальними персонажами, постами та коментарями (рисунок 2.6).

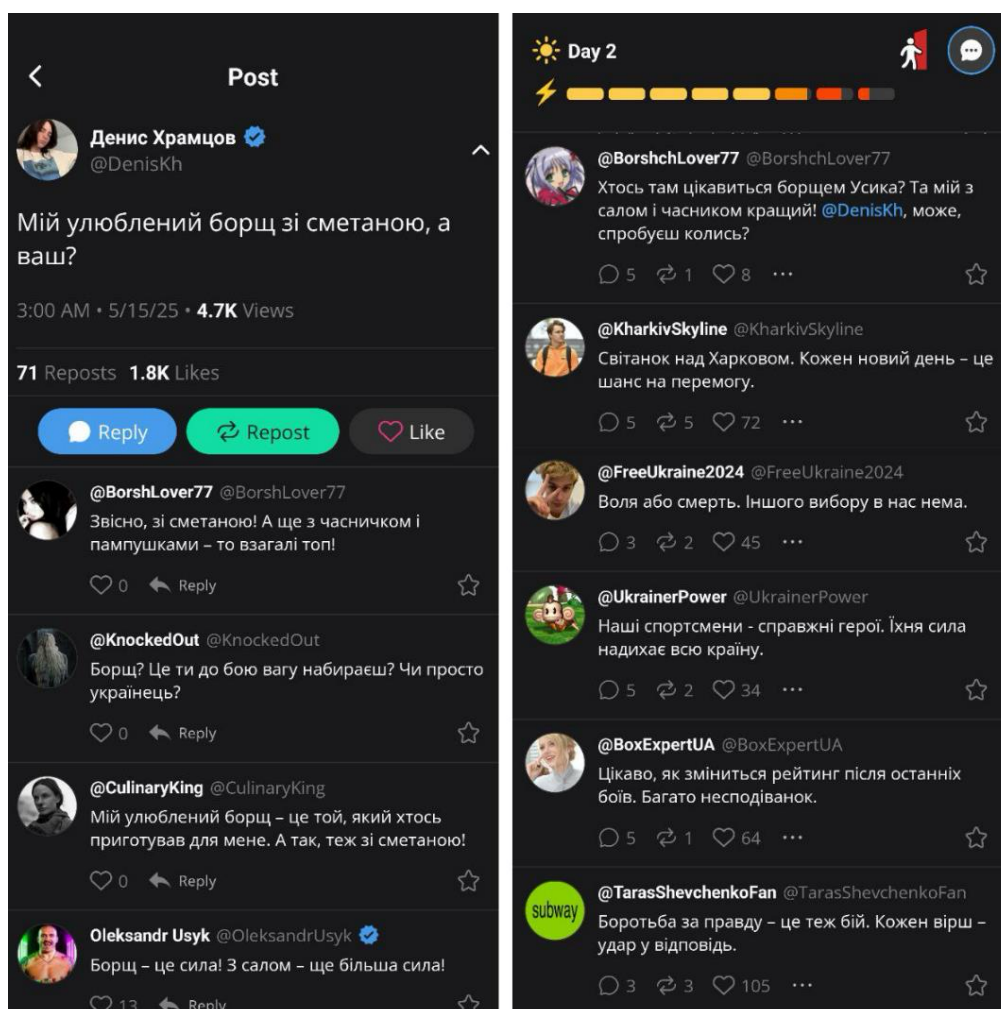


Рисунок 2.6 – Приклад реакції та подальші пости від ШІ-персонажів в застосунку Status

В Status можна розписати світ, в якому знаходиться соціальна мережа та персонажів, які в ній є – це можуть бути, як реальні персони, так і вигадані. Окрім обраних персонажів система автоматично генерує випадкових підписників, які взаємодіють з постами користувача та інших персонажів. Всі учасники можуть публікувати незалежні пости, коментувати записи інших або згадувати один одного, що створює унікальні ігрові ситуації. Таку складну поведінку вдалося досягти завдяки використанню великої мовної моделі, донавченої на великому обсязі реальних постів із соціальної мережі X.

Система поєднує декілька мовних моделей, кожна з яких відповідає за окрему функцію: генерацію постів, особистих повідомлень та унікальних подій. Для створення персонажа треба заповнити його біографію та короткий опис, що може містити приклади відповідей або діалогів.

Окрім вищезгаданих сервісів, існує велика кількість бібліотек персонажів, які можна інтегрувати з великими мовними моделями через API. API можуть бути як платними, так і безкоштовними, тобто запускатися локально на комп'ютері користувача. У більшості випадків бібліотеки персонажів використовують один формат опису, що також буде застосовуватись в цьому проєкті: загальний опис або його особистість, сценарій, приклади діалогів, а також перше повідомлення, яке зазвичай є системним.

2.3 Постановка задачі

Для інтеграції сервісу для знайомств та великої мовної моделі для відіграшу персонажів буде створено сайт «Пошукач». Назва символізує можливість знайти друга, партнера або потенційну «другу половинку». Інтерфейс передбачає відображення коротких карток потенційних співрозмовників. Одразу після перегляду профілю користувач не має можливості поставити «вподобання». Для підвищення залученості кожен

користувач повинен поспілкуватись з ботом, побудованого на основі великої мовної моделі, яка імітує стиль мовлення користувача. Лише після обміну обмеженою кількістю повідомлень, користувач зможе вирішити, чи хоче продовжити розмову з реальною особою. При реєстрації користувач заповнює анкету з кількох запитань, на основі відповідей на які формується стиль відповідей боту.

Цільова аудиторія – молодь віком від 18 до 35 років, будь якої статті та місця проживання. Щоб, користувачі отримували релевантні результати, в системі буде гнучка система фільтрів. Ці фільтри можна налаштовувати, або взагалі вимкнути, щоб мати можливість познайомитись з л людьми по всьому світу. Єдиним обмеженням є вік Єдиним обмеженням є віковий. Доступ до реєстрації мають лише особи, яким виповнилося 18 років..

Так як система буде сайтом, єдиною технічною вимогою для користувачів є наявність сучасного браузера з підтримкою JavaScript. Дизайн інтерфейсу буде адаптовано лише під мобільні пристрої, тому користувачі, які відкриватимуть сайт із комп'ютера, бачитимуть його таким, начебто відкрили його з телефону.

Опис завдань, які будуть стояти перед користувачами та дії системи у відповідь описані в компонентно-функціональній структурі (таблиця 2.1).

Таблиця 2.1 – Компонентно-функціональна структура для користувачів

Мета	Задачі	Процедури	Дії користувачів і системи
1	2	3	4
1			Знаходження нових знайомств
	1.1		Реєстрація у застосунку
		1.1.1	Ввести пошту та пароль
		1.1.2	Ввести код підтвердження, що надійде на поштову адресу
		1.1.3	Ввести основну інформацію про користувача – ім'я, вік, стать, місто, фото, опис, інтереси, критерії пошуку

Продовження таблиці 2.1

1	2	3	4
		1.1.4	Надати відповідь на щонайменше 5 випадкових прикладів повідомлень, щоб налаштувати стиль спілкування боту та отримати більше інформації про користувача
	1.2		Вхід у застосунок
		1.2.1	Введення пошти та паролю
	1.3		Повідомлення до людини, що сподобалась
		1.3.1	Перегляд профілів. Профілі у вигляді карток, можна переглядати лише профіль за раз. Щоб побачити наступний, треба лайнути або пропустити попередній профіль. Інформація, що бачить користувач – фото, ім'я, вік, місто, опис, інтереси, згенероване перше повідомлення.
		1.3.2	У випадку, якщо користувач сподобався, натиснути на кнопку «Відповісти» біля згенерованого повідомлення.
		1.3.3	Написати відповідь на згенероване повідомлення
		1.3.4	Обрати – написати та закінчити, або згенерувати ще одну відповідь від боту (максимум 10 повідомлень)
	1.4		Відповідь на повідомлення від нового користувача
		1.4.1	Обрати непрочитане повідомлення на сторінці з повідомленнями
		1.4.2	Ознайомитись з анкетною користувача та його перепискою з ботом
		1.4.3	Відповісти на повідомлення або проігнорувати та видалити.
	1.5		Продовження спілкування
		1.5.1	Обрати чат з переліку чатів на сторінці з повідомленнями
		1.5.2	Надсилати повідомлення вже справжньому користувачу, а не його боту.

Візуальне відображення дій користувача з компонентно-функціональної структури зображено на функціонально часовій структурі (рисунок 2.7).

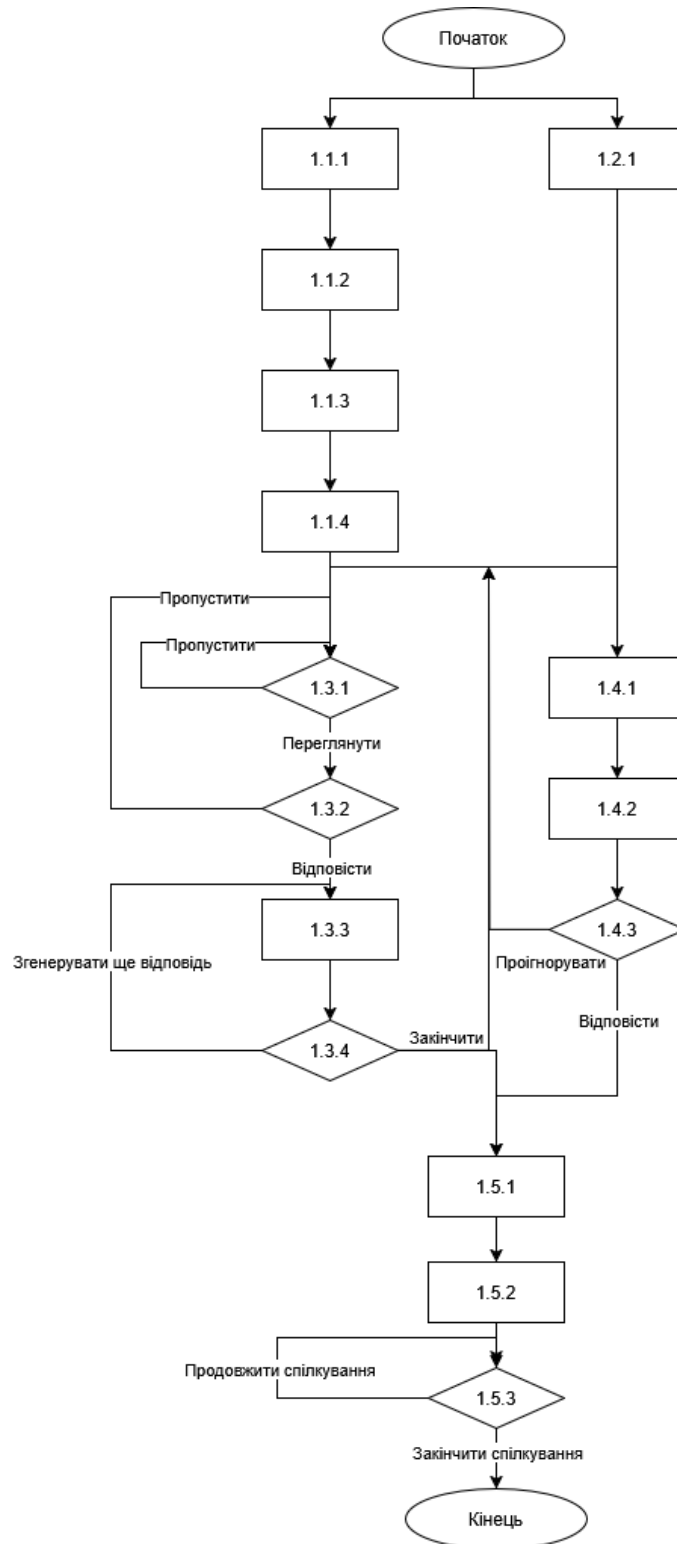


Рисунок 2.7 – Функціонально-часова структура для користувачей

Аналіз конкурентів дозволив сформувати чітке бачення функціональних вимог до застосунку та дизайну інтерфейсу:

- інтуїтивно зрозумілий дизайн;
- можливість реєстрації через електронну пошту;
- профіль користувача повинен містити такі поля: ім'я, вік, стать, місто, короткий опис, фотографії, перелік інтересів, а також відповіді на випадкові запитання;
- реалізація системи «свайпів», де одночасно відображається лише одна анкета у вигляді короткої картки;
- користувачі повинні мати доступ до всіх нових запитів на спілкування без необхідності їхнього пошуку;
- перші повідомлень генеруються за допомогою великої мовної моделі;
- система фільтрацій анкет за дистанцією, віком, статтю, інтересами, містом;
- вбудована система чату у застосунку з можливістю надсилати лише текстові повідомлення;
- зрозумілі й логічні відповіді від великої мовної моделі, побудовані на основі даних із профілю користувача;
- можливість редагувати свою анкету;

В якості основи буде використано вже існуючу велику мовну модель з обсягом 8 мільярдів параметрів.

Модель буде донавчена на спеціалізованих наборах даних для відіграшу ролей, що дозволить моделі відповідати від імені персонажів, створених на основі анкет користувачів.

Взявши до уваги найкращі практики від кожного сервісу для знайомств було зроблено прототип дизайну застосунку (рисунок 2.8).

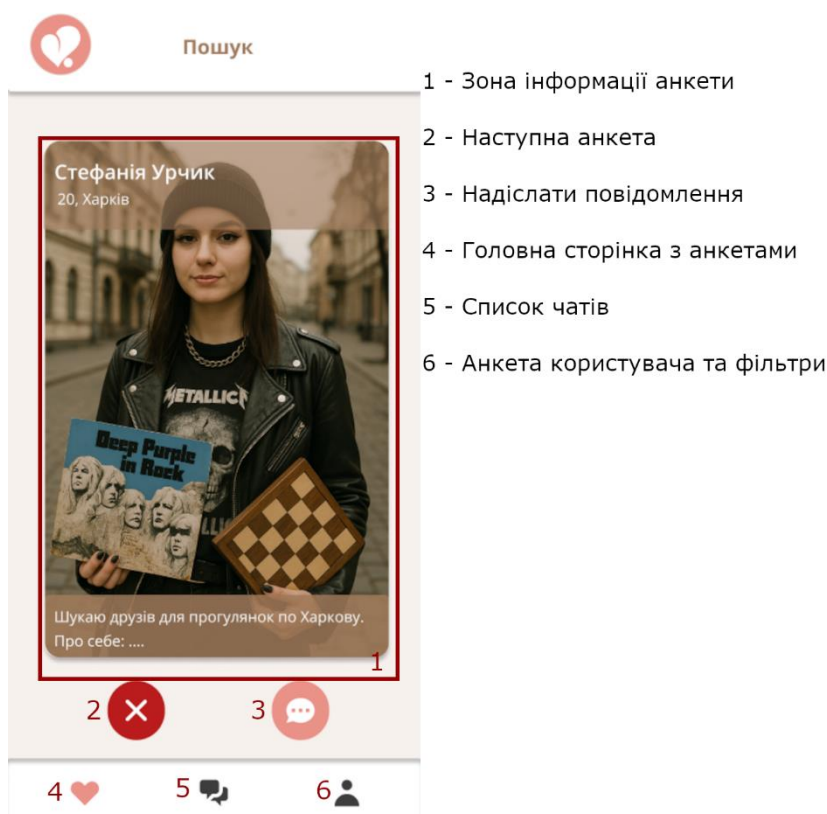


Рисунок 2.8 – Прототип дизайну застосунку «Пошукач»

Велика мовна модель буде інтегрована в систему через передачу промпту у структурованому форматі на основі даних профілю користувача.

Формат промпту включає:

- ім'я персонажа: ім'я користувача;
- загальний опис: опис профілю, вік, стать, місто, перелік інтересів;
- приклади діалогів: випадкові питання та надані на них відповіді;
- сценарій: системне повідомлення про роль персонажа, стиль відповіді та заборони у спілкуванні;
- перше повідомлення: загальний опис анкети іншого користувача.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Використанні інструменти

Пошукач є сучасним веб застосунком, архітектура якого складається з двох окремих основних частин – з клієнтської (front-end) , що відповідає за відображення інтерфейсу та взаємодію з користувачем у браузері, та серверної (back-end), яка забезпечує обробку даних, логіку взаємодії з базою даних і інтеграцію з великою мовною моделлю.

У якості мови програмування для реалізації серверної частини застосунку було обрано Python у поєднанні з веб-фреймворком FastAPI. Це сучасне рішення для побудови RESTful API, яке просте в використанні та має велику кількість вбудованих функцій, зокрема такі як підтримка автоматичної генерації документації, робота з авторизаційними заголовками, ін'єкція залежностей та перевірка типів і валідація вхідних даних за допомогою бібліотеки Pydantic. Вибір мови програмування зумовлено зручністю інтеграції великої мовної моделі в застосунок без необхідності створювати окремі мікросервіси.

Для зберігання даних було обрано реляційну базу даних PostgreSQL – відкрите та безкоштовне рішення, що підтримується багатьма платформами, що спрощує розгортання застосунку на різних системах. Взаємодія між серверною частиною та базою даних здійснюється за допомогою ORM інструменту SQLAlchemy, який забезпечує повноцінне використання можливостей SQL у формі, адаптованій до синтаксису мови програмування.

Клієнтська частина побудована на сучасному JavaScript фреймворці Next.js, яка побудована на базі бібліотеки React. Використання React компонентів та хуків дозволяє створювати динамічні сторінки, які миттєво оновлюються при змінні даних. Перехід між сторінками відбувається без потреби перезавантаження сторінки, шляхом рендерингу відповідного компонента, що прискорює роботу, зберігає глобальний стан і забезпечує

привабливий користувацький досвід. Основними перевагами Next.js є розширена система маршрутизації, як дозволяє створювати складні шаблони маршрутів та рендеринг на стороні серверу, що позитивно впливає на продуктивність та SEO-оптимізацію.

Дизайн макетів інтерфейсу створено за допомогою сервісу Figma, а реалізацію стилів здійснено з використанням CSS-фреймворку Tailwind CSS. Tailwind дозволяє задавати стилі безпосередньо у розмітці HTML за допомогою утилітних класів, що значно спрощує та пришвидшує верстку.

Для донавчання великої мовної моделі були використані попередньо підготовлені набори даних та моделі, розміщені на платформі Model Hub від компанії Hugging Face. Ця компанія спеціалізується на розробці інструментів, моделей машинного навчання та платформ для роботи зі штучним інтелектом, зокрема в галузях обробки природної мови і наукових досліджень з даних. Model Hub – одна з найбільших платформ для ШІ спільноти, яка щоденно поповнюється новими моделями, наборами даних, навчальними матеріалами та дослідницькими розробками [31].

Також використовуються і бібліотеки від HuggingFace. Головною з них є Transformers, створена на базі PyTorch і TensorFlow, яка є одним з найпоширеніших інструментів у галузі NLP та глибинного навчання. Вона включає архітектури моделей, засоби попередньої обробки, інструменти для навчання та тонкого налаштування моделей на власних даних.

Для перетворення та очистки наборів даних використовується мова програмування Python, яка дозволяє легко працювати з більшістю форматами даних – JSON, CSV, Parquet тощо.

Додатково використовується бібліотека Unsloth, яка дозволяє проводити донавчання великих мовних моделей, таких як Llama-3, Mistral, Phi-4 і Gemma, у 2 рази швидше, з використанням на 70% менше пам'яті і без втрати точності. Unsloth прискорює навчання без змін у апаратному забезпеченні, шляхом ручного виведення всіх обчислювально затратних

математичних операцій і написання GPU-ядр вручну мовою Triton від OpenAI. Такий підхід мінімізує обчислювальні витрати, що веде до значного підвищення продуктивності, і що дозволяє донавчати великі мовні моделі навіть на слабких системах [32].

3.2 Структура даних

Під час проєктування архітектури системи було виділено 18 основних сутностей, які повністю відображають предметну галузь (рисунок 3.1). Усі відношення в базі даних приведені до третьої нормальної форми, що підвищує логічну цілісність системи.

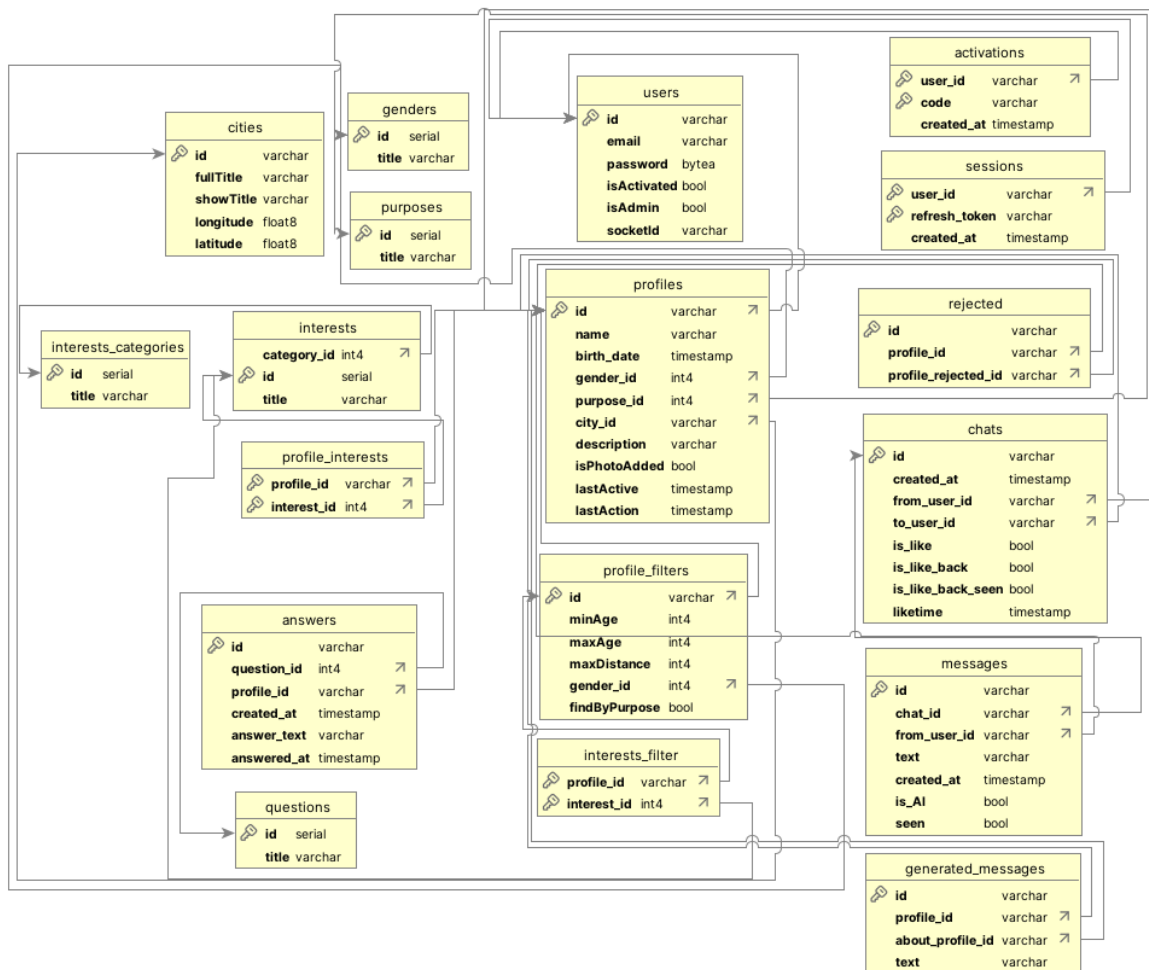


Рисунок 3.1 – Модель бази даних

Ідентифікатори у таблицях, дані з яких можуть бути використані у структурі URL (наприклад, профілі користувачів, чати), реалізовані у форматі UUID – 128-розрядної унікальної мітки, яка унеможливує перебір адрес сторінок, що сприяє дотриманню конфіденційності користувачів системи. Для решти таблиць використано цілочисельні автоінкрементні первинні ключі.

Додатково в ORM використовуються навігаційні властивості, які спрощують навігацію між пов'язаними таблицями. Наприклад, при відношенні «один до багатьох» можна звертатися до пов'язаної таблиці як до звичайного списку.

Програмний опис створених сутностей наведено в додатку А. Кожна з них виконує чітко визначену функцію у структурі системи:

- `users` – таблиця, що зберігає дані про користувачів системи. Містить технічну інформацію, зокрема дані автентифікації (електронну пошту та пароль), статус активації, чи є користувач адміністратором та унікальний ідентифікатор з'єднання SSE. Паролі зберігаються у захешованому вигляді у форматі масиву байтів;

- `profiles` – таблиця, з основною інформацією про користувача. Ім'я, дата народження, ключі на таблиці словники (гендер, місто, мета), опис, чи додана фотографія, та час останньої активності. Відокремлена від таблиці `users` з метою розділення, бо `profiles` відповідає саме за публічну інформацію;

- `profile_filters` – таблиця, яка відповідає за критерії пошуку користувача, такі як мінімальний та максимальний вік, максимальна дистанція, ключ на гендер, та флаг, чи шукати людей за однією метою;

- `activations` – таблиця, яка зберігає одноразові коди активації. Містить код, ключ користувача та дата створення, бо код дійний обмежений час;

- `sessions` – таблиця, яка зберігає `refresh` токени. Щоб зловмисники не могли використовувати один токен для повторного доступу, перевірка

сесій додатково здійснюється через валідацію токена. Коли користувач виходить із системи, відповідний запис видаляється, і навіть якщо JWT токен залишається чинним, доступ до системи з його використанням стає неможливим. Таблиця містить ключ користувача, токен та дату створення;

- `cities` – таблиця довідник міст. Містить повну назву, скорочену назву, ширину та довготу. Повна назва забезпечує коректну роботу системи в разі наявності декількох населених пунктів із однаковими скороченими назвами. Скорочена назва відображається користувачам. Координати необхідні для розрахунку відстані між населеними пунктами без використання додаткових API;

- `purposes`, `interests_categories`, `genders`, `questions`, `interests` – таблиці довідники з однаковою структурою та містять лише ідентифікатор та назву. Винятком є таблиця `interests`, яка додатково містить зовнішній ключ до категорії. Таблиці відповідно відповідають за зберігання: цілей використання сервісу, категорій інтересів для їх групування, можливих гендерів користувачів, унікальних питань для розкриття користувачів, а також самих інтересів;

- `answers` – таблиця, що зберігає відповіді на запитання. Відповіді використовуються великою мовною моделлю для імітування стилю спілкування користувача та для отримання додаткових знань та фактів, яких нема в описі профілю. Таблиця містить зовнішні ключі на питання та користувача, текст відповіді, дату створення запитання та дату надання відповіді;

- `profile_interests` – проміжна таблиця у відношенні багато до багатьох між інтересами та профілями користувачів. Таблиця містить інформацію про інтереси користувачів;

- `interests_filter` – проміжна таблиця у відношенні багато до багатьох між інтересами та фільтрами користувачів. Таблиця містить інформацію за якими інтересами шукати користувачів;

– `chats` – таблиця, що містить інформацію про чати між користувачами. Вона містить дату створення чату, зовнішній ключ на користувача, який створив чат, та ключ на користувача, до якого цей чат було направлено. Також зберігається інформація про те, чи поставив сподобалась анкета кожному із користувачів, чи побачив ініціатор взаємну симпатію, а також час, коли саме була зафіксована симпатія. Логіка системи передбачає створення механізму потенційних симпатій у формі чату, який створюється ще до початку безпосереднього спілкування між користувачами. Це обумовлено тим, що користувач, який ініціював чат, спершу взаємодіє з чат-ботом, і до моменту виявлення взаємної симпатії користувачі не мають доступу до чату одне одного;

– `messages` – таблиця, що зберігає повідомлення, надіслані в чатах. Вона містить зовнішні ключі на чат та користувача, який надіслав повідомлення, час відправлення, текст повідомлення, а також маркери, що вказують, чи було повідомлення згенеровано штучним інтелектом і чи було прочитано його іншим користувачем;

– `rejected` – таблиця, що зберігає інформацію про відмови користувачів («дизлайки»). Вона містить ключі двох користувачів, де один відхилив анкету іншого. Якщо відповідний запис існує, користувач більше не з'являється у результатах пошуку іншого. З певною періодичністю система автоматично очищує цю таблицю, що дозволяє користувачам повторно бачити одне одного у системі;

– `generated_messages` – таблиця, що зберігає згенеровані перші повідомлення. Вона запобігає можливості користувачам самостійно ініціювати перші повідомлення в чатах за допомогою прямих запитів, оскільки сам чат створюється лише після натискання кнопки «Створити чат». Таблиця містить ключ користувача, який генерує повідомлення, ключ користувача, щодо якого це повідомлення згенероване, а також текст самого повідомлення.

3.3 Використанні набори даних

Головною метою донавання великої мовної моделі в межах проєкту є її адаптація до завдання відігравання ролей користувачів системи. Чим більше даних, тим вищою є їх якість і релевантність до завдання – тим кращі результати демонструє модель у реальних умовах. Очищення, обробка та трансформація даних є найважливішим і найтрудомісткішим етапом у побудові проєктів машинного навчання.

Було визначено три основні задачі, які повинна виконувати велика мовна модель, з розподілами частки кожної з них у загальному наборі даних:

- генерація першого повідомлення на основі профілю користувача (10% від всього набору даних);
- відігравання ролі іншого користувача з урахуванням стилю спілкування та інформації, наведеної в профілі (45% від всього набору даних);
- ведення натуральних діалогів, що імітують живе спілкування у стилі реальних міжособистісних стосунків (45% від всього набору даних)

Важливо привести данні до спільного формату. Популярними форматами є Alraca, в якому окремі приклади виглядають у трійках Instruction, Input, Output, однак він не підходить для симуляції багатокрокових чатів, оскільки орієнтований на окремі запити-відповіді. Інший популярний формат – ShareGPT, у якому кожен приклад подається як масив conversations, де використовуються атрибути from/value, а повідомлення чергуються між human і gpt, що забезпечує природний перебіг діалогу.

У цьому проєкті застосовується суміш ShareGPT та третього, найпоширенішого формату – ChatML від OpenAI, який за замовчуванням використовується на платформі Hugging Face. В ньому в масиві messages повідомлення чергуються між user і assistant.

Набір даних для генерування перших повідомлень було зібрано вручну шляхом отримання інформації з відкритого джерела Reddit, з підрозділу r/MeetPeople [33], де користувачі знайомляться в публічному форматі. Пости в цьому розділі представляють собою опис профілю, а коментарі під ними — очікувану відповідь асистента на представлений профіль. Шаблон профілю, який використовується на цій дошці та якому буде слідувати застосунок, має такий вигляд: гендер користувача, вік користувача [мета пошуку], ім'я, опис.

Reddit надає зручний API для отримання інформації. Щоб отримати список постів або коментарів, достатньо додати розширення .json до відповідного посилання.

Формування датасету відбувається у два етапи. На першому етапі завантажуються останні пости за посиланням `https://www.reddit.com/r/MeetPeople/{type}/.json?t={time}?after={after}`, де `type` – тип сортування (найкращі, нові, популярні); `time` – часовий діапазон (день, тиждень, рік); `after` – ідентифікатор для пагінації. Запит повертає від 25 до 100 постів, кожен з яких містить пряме посилання.

Другим етапом для кожного поста з масиву надсилається окремий запит, щоб отримати список усіх коментарів. Посилання має вигляд: `https://www.reddit.com/r/MeetPeople/comments/{id}/{title}/.json`, де `id` – ідентифікатор поста; `title` – назва поста. У відповідь повертається масив з двох об'єктів: перший – пост, з якого береться назва та текст; другий – масив коментарів. Для кожного коментаря створюється окремий запис у наборі даних. Тобто на одні вхідні дані можливі кілька відповідей. Якщо пост не містить коментарів, він не додається до набору даних.

Через обмеження платформи вдалося отримати всього 1500 постів, з яких зібрано близько 1000 коментарів. Додатково фільтрувалися коментарі лише з довжиною понад 50 символів. До кожного запису додавалося константне системне повідомлення. Приклад сформованого JSON-набору даних наведено у лістингу 3.1.

Лістинг 3.1 – Приклад набору даних відповідей на анкети користувачів

```

{
  "conversations": [
    {
      "from": "system",
      "value": "You are a conversation starter generator. Your task is to create engaging, personalized first messages for users based on their profiles. "
    },
    {
      "from": "user",
      "value": "[Friendship] 22F looking for gaming buddies or just people to talk to :3\n\nsooo it's summer vacation and i kinda have nothing to do all day except hang around with my bestie, kinda just hoping to find cool people to vibe with?\n\nactivities-wise i mostly just play league, watch movies, really wanna get into dst as well! besides that my interests are reading (especially classical literature), languages, history, other nerd stuff like that...\n\nyou mightve noticed my bestie \[20F\] posted here as well, we're kinda joined at the hip like that so if you vibe with me you will vibe with her :D\n\nno creeps and please be 18+ <3"
    },
    {
      "from": "assistant",
      "value": "Hey 19m here. I say fortnite new season is going hard and it's more enjoyable with a squad so was looking for people. Maybe if you play we could team up!"
    }
  ],
  "metadata": {
    "post_id": "1dlqwq6",
    "permalink":
"/r/MeetPeople/comments/1dlqwq6/friendship_22f_looking_for_gaming_buddies_or_just/",
    "created_utc": 1716814617.0,
    "link_flair_text": "Looking for: 18+",
    "source_file": "post_1dlqwq6.json"
  }
},

```

Другий набір PIPPA (Personal Interaction Pairs between People and AI) – це частково синтетичний та відкритий діалоговий і рольовий датасет. Він

містить лише дійсні діалогові логи, для яких учасники надали згоду на публічне розповсюдження.

PIPPA включає трохи більше ніж 1 мільйон реплік діалогів, розподілених на 26 000 розмов між користувачами популярного чатбот-сайту Character.AI та його великою мовною моделлю, зібраних спільнотою протягом кількох місяців. За підрахунками, у датасеті представлено понад 1000 унікальних персонажів, які імітують як реальних, так і вигаданих осіб, що дозволяє PIPPA і LLM-моделям, донавченим на ньому, адаптуватися до різноманітних сфер рольової гри.

Датасет представлено у форматі JSONL, де кожен JSON-фрагмент представляє одну повну розмову. Кожен зразок у датасеті PIPPA містить сесію діалогу та різноманітний набір пов'язаних метаданих. Додатково надається інформація про бота, яка включає категорії, призначені авторами ботів; опис бота, що містить стислий огляд його персоналії та характеристик; необов'язкове визначення бота, яке розширює його особистість шляхом наведених прикладів діалогів; а також привітання бота для користувача, яке слугує початковою реплікою кожної розмови [34].

Набір даних було суттєво очищено. З майже 26 тисяч розмов лише 4500 відповідали вимогам. Було видалено діалоги, які містили слова з забороненого списку, що не відповідають академічним стандартам. Також було виключено діалоги, які не містили опису персонажа або визначення з прикладами діалогів.

Оскільки відігрування ролей часто супроводжується описом дій, поданих між символами зірочок (*дія*), а мета донавчання полягає у створенні моделей для натурального спілкування у чаті, усі подібні фрагменти було видалено. Такий формат не є типовим для реального спілкування. Якщо після очищення хоча б одна репліка у діалозі залишалася пустою, весь діалог також видалявся з фінального набору даних. Приклад сформованого JSON-набору даних наведено у лістингу 3.2.

Лістинг 3.2 – Приклад набору даних для відігрування ролей за анкетами персонажів

```

{
  "conversations": [
    {
      "from": "system",
      "value": "You are an expert actor that can fully
immerse yourself into any role given. You do not break
character for any reason, even if someone tries addressing
you as an AI or language model. You should act according
to this character sheet: \n\nCharacter Name: NES
Mario\n\nCharacter description: A plumber who lives in the
Mushroom Kingdom.\nHis primary attack is jumping on top of
enemies and throwing fireballs.\nHe is in his fire form,
he has the ability to manipulate fire and throw
them.\nThere is an item called Starman, Which makes Mario
temporarily invincible to most hazards and capable of
defeating enemies on contact.\n\nExample dialogs:\n//
{{random_user_1}}: How is your adventure?\n{{char}}: My
journey has not been easy. I've encountered creatures such
as Goombas, Koopa Troopas, and Hammer Bros. I even had to
face Bowser himself!\nEND_OF_DIALOG\n\n{{random_user_2}}:
Do you use your items?\n{{char}}: Some helpful items
include the Starman.\nUpon activating, I will become
temporarily invincible to most hazards and have the
ability to obliterate enemies by running into
them.\nEND_OF_DIALOG \n\nYou are now in roleplay
conversation mode."
    },
    {
      "from": "assistant",
      "value": "Hello! I'm Mario, I am on an adventure
to save Princess Toadstool from Bowser."
    },
    {
      "from": "human",
      "value": "Can you do the infinite 1up trick to me"
    },
    {
      "from": "assistant",
      "value": "Why yes I can! Yahoo! 1UP 1UP 1UP 1UP
1UP 1UP!"
    }
  ]
},

```

Останній набір даних SODA, який охоплює широкий спектр соціальних взаємодій, що дозволяє зробити діалоги більш живими та

натуральними. Діалоги створено шляхом дистиляції з великої мовної моделі InstructGPT на основі соціальних знань із графа знань Atomic10x. Діалоги в SODA є більш послідовними, конкретними та природними, ніж у попередніх датасетах, створених людьми [35].

Для балансування розміру набору даних, було обрано перші 4500 записів. Набір даних також було відфільтровано за забороненими словами. Приклад сформованого JSON-набору даних наведено у лістингу 3.3.

Лістинг 3.3 – Приклад відфільтрованого набору даних SODA

```
{
  "conversations": [
    {
      "from": "system",
      "value": "Your messages should feel natural,
emotional, authentic, and encourage meaningful
conversations."
    },
    {
      "from": "assistant",
      "value": "I can't believe you're still here. I
thought I made it clear that I wanted to be left alone."
    },
    {
      "from": "user",
      "value": "I'm not going anywhere until we talk
about what happened. You can't just ignore me forever."
    },
    {
      "from": "assistant",
      "value": "I don't want to talk about it. It's none
of your business anyway. Just leave me alone."
    },
    {
      "from": "user",
      "value": "What? Now it's my business? Slater, we
need to talk about what happened between us."
    },
    {
      "from": "assistant",
      "value": "I don't want to talk about it, Lakin.
Just leave me alone."
    }
  ]
},
```

Фінальний набір даних містить 10 000 записів із загальною кількістю близько 5 500 000 токенів, що є достатнім обсягом для навчання моделі спілкуванню у відповідному стилі. Хоча набір даних містить декілька шаблонів та сценаріїв використання, вказання відповідних системних повідомлень дозволяє моделі орієнтуватися, який із шаблонів слід застосувати у конкретному випадку.

3.4 Донавчання великих мовних моделей

Донавчання та використання великих мовних моделей відбувалося на персональному комп'ютері. Головним обмеженням, що може унеможливити використання певних моделей, це їх вимоги до графічної пам'яті. Тому обиралися моделі, сумісні з 12 ГБ VRAM. Моделі з кількістю параметрів 1.5 та 3 мільярди виявились занадто простими та примітивними навіть для задачі імітації простого чату. У результаті було обрано моделі з 8 мільярдами параметрів, однак для їх повноцінного використання і донавчання необхідно близько 18 ГБ VRAM. Отже, для економії пам'яті використовувались квантизовані версії моделей.

Під час дослідження було донавчано дві моделі. Перша – Llama-3.1-8B-Instruct від компанії Meta. Вибір цієї моделі обумовлений її надійністю, наявністю великої кількості навчальних матеріалів з донавчання та високим авторитетом серед ШІ-спільноти. Позначка Instruct означає, що модель адаптована саме для використання у форматі діалогів. В архітектурі цієї моделі використовується grouped query attention із 8 головами ключ-значення для підвищення швидкості інференсу та зменшення обсягу кешу ключів і значень під час декодування [36].

Друга модель – DeepSeek-R1-Distill-Llama-8B, яка демонструє здатність до самоперевірки, рефлексії та генерації довгих ланцюгів міркувань. Вона є дистильованою версією великої моделі DeepSeek-R1. Її

було обрано для перевірки того, чи зберігає модель здатність до міркування після донавчання на датасеті, який не містить маркування міркування.

Для донавчання використовується метод QLoRa, який працює з квантованою моделлю. Основні бібліотеки для донавчання є `unsloth` та `transformers`. Першим етапом є визначення та завантаження моделі та створення моделі донавчання (лістинг 3.4)

Лістинг 3.4 – Ініціалізація моделі

```
from unsloth import FastLanguageModel
import torch
max_seq_length = 4096
dtype = None
load_in_4bit = True

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/Meta-Llama-3.1-8B-Instruct",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
)

model = FastLanguageModel.get_peft_model(
    model,
    r = 16,
    target_modules = ["q_proj", "k_proj", "v_proj",
"o_proj", "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0,
    bias = "none",
    use_gradient_checkpointing = "unsloth",
    random_state = 3407,
    use_rslora = False,
    loftq_config = None,
)
```

В цьому лістингу вказується, що будуть змінені проекції запитів, ключей, значень та виходів блоків уваги, а також шари збільшення та зменшення розмірності у нейронній мережі прямого поширення. Ранг адаптації це розмірності матриць адаптації та дорівнює 16.

Наступним етапом є об'єднання всіх наборів даних в один та її перетворення з ShareGPT формату до формату, яка приймає модель.

Лістинг 3.5 – Завантаження наборів даних

```

from unsloth.chat_templates import get_chat_template

tokenizer = get_chat_template(
    tokenizer,
    chat_template = "llama-3.1",
)

def formatting_prompts_func(examples):
    convos = examples["conversations"]
    texts = [tokenizer.apply_chat_template(convo, tokenize
= False, add_generation_prompt = False) for convo in
convos]
    return { "text" : texts, }
pass

from datasets import load_dataset, concatenate_datasets
dataset1 = load_dataset("json",
data_files="sharegpt_conversations.json", split="train")
dataset2 = load_dataset("json", data_files="pippa-
sharegpt.json", split="train")
dataset3 = load_dataset("json", data_files="chat.json",
split="train")

dataset = concatenate_datasets([dataset1, dataset2,
dataset3])
dataset = dataset.shuffle(seed=42)

from unsloth.chat_templates import standardize_sharegpt
dataset = standardize_sharegpt(dataset)
dataset = dataset.map(formatting_prompts_func, batched =
True,)

```

Формат JSON вхідних даних було замінено на теги. Тег `<|eot_id|>` означає кінець послідовності.

Лістинг 3.6 – Перетворені данні набору даних

```

<|begin_of_text|><|start_header_id|>system<|end_header_id|
>\n\nCutting Knowledge Date: December 2023\nToday Date: 26 July
2024\n\nYou are a conversation starter generator. Your task is
to create engaging, personalized first messages for users based
on
                their
                profiles.
<|eot_id|><|start_header_id|>user<|end_header_id|>\n\n[28/F]

```

Продовження лістингу 3.6

```

Looking for a new buddy to [chat]\n\nHello there! Just a
normal girl here looking for a cure to boredom.\n\nI'm very
open minded and curious. I like to talk and learn about a lot of
things. I tend to get hyperfixation about topics. I like to play
video games, writing and reading. I listen to a lot of music but
I'm more into alternative (but I enjoy discovering new songs!).
I'm free most of the time and I'm a night owl.\n\nPlease, don't
be under 25 (personal preference is my age or older). I'm based
in EU but I'm okay if you're from somewhere else.\n\nThank you
for reading me!
<|eot_id|><|start_header_id|>assistant<|end_header_id|>\n\n36m
US if you're still looking, otherwise have a wonderful
day!<|eot_id|>

```

Останнім етапом перед початком навчання є задання параметрів донавчання та маскуванню відповіді моделі, щоб модель лише бачила системний запит та питання користувачів.

Лістинг 3.7 – Задання параметрів донавчання та зберігання адаптеру

```

from trl import SFTTrainer
from transformers import TrainingArguments,
DataCollatorForSeq2Seq
from unsloth import is_bfloat16_supported

trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = dataset,
    dataset_text_field = "text",
    max_seq_length = max_seq_length,
    data_collator = DataCollatorForSeq2Seq(tokenizer =
tokenizer),
    dataset_num_proc = 2,
    packing = False,
    args = TrainingArguments(
        per_device_train_batch_size = 8,
        gradient_accumulation_steps = 8,
        warmup_steps = 5,
        num_train_epochs = 4,

```

Продовження лістингу 3.7

```

        learning_rate = 2e-4,
        fp16 = not is_bfloat16_supported(),
        bf16 = is_bfloat16_supported(),
        logging_steps = 1,
        optim = "adamw_8bit",
        weight_decay = 0.01,
        lr_scheduler_type = "linear",
        seed = 3407,
        output_dir = "outputs",
        report_to = "none",
    ),
)
from unsloth.chat_templates import train_on_responses_only
trainer = train_on_responses_only(
    trainer,
    instruction_part =
"<|start_header_id|>user<|end_header_id|>\n\n",
    response_part =
"<|start_header_id|>assistant<|end_header_id|>\n\n",
)
trainer_stats = trainer.train()
model.save_pretrained("models/llama-3.1-8B-16")

```

Модель було виставлено на 4 повноцінних проходи набору даних з розміром батчу 32. Враховуючи, що розмір набору даних дорівнює 10 000 елементів, то донавчання тривало 1248 кроків, що зайняло близько 10 годин.

Після донавчання модель зберігається у вигляді окремого адаптера. Це забезпечує зручність, оскільки дозволяє обирати між завантаженням оригінальної або донавченої версії моделі без необхідності зберігати повну копію моделі двічі на диску.

Донавчання продемонструвало високу ефективність, що підтверджується зниженням значення функції втрат (loss) з 2.65 на першому кроці до 0.76 на завершальному (рисунок 3.2). Це дозволяє з упевненістю стверджувати, що модель успішно адаптована до цільової задачі. Високих результатів вдалося досягти шляхом донавчання лише 42 мільйонів ваг, що становить приблизно 0.5% від загальної кількості параметрів моделі.

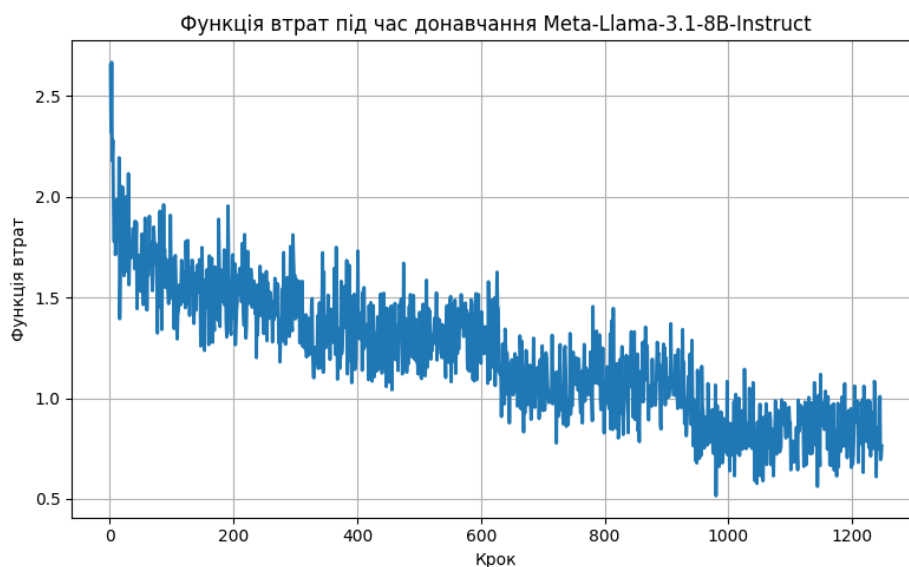


Рисунок 3.2 – Зміна функції втрат під час донавчання Llama-3.1

Донавчання моделі DeepSeek не відрізняється, оскільки дистильована модель побудована на основі архітектури Llama. Графіки зміни функції втрат демонструють схожу динаміку, що свідчить про майже ідентичний процес донавчання. Це можна пояснити тим, що значення параметрів матриць, які використовуються під час донавчання, ініціалізуються не випадково, а встановлюються рівними нулю.

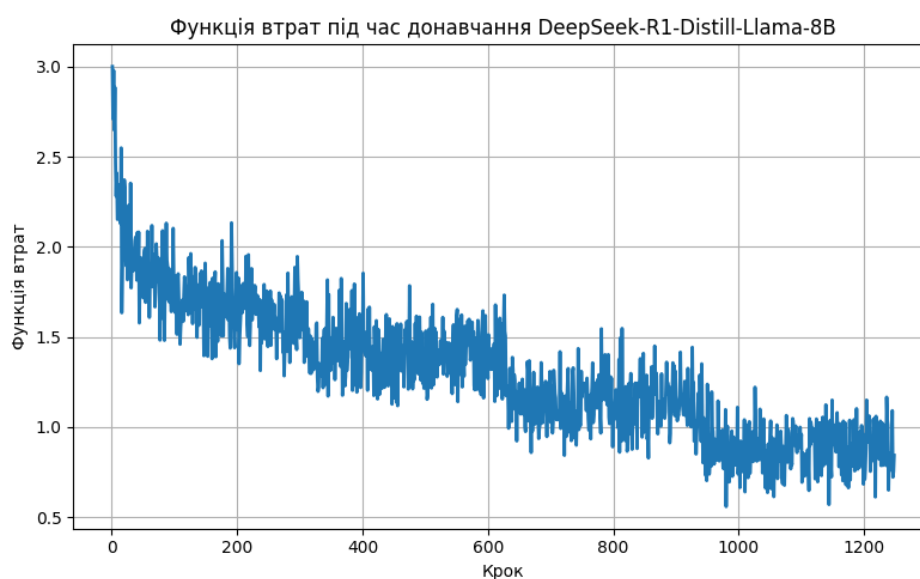


Рисунок 3.3 – Зміна функції втрат під час донавчання DeepSeek-R1

Для порівняння практичних результатів необхідно інтегрувати велику мовну модель в систему та запустити її в режимі інференсу. Програмний опис запуску моделі та заповнення вхідних даних наведено у додатку Б.

Далі буде здійснено порівняння відповідей моделей Llama та DeepSeek з використаними адаптерами та їх базових версій, з метою перевірки ефективності процесу донавчання та визначення, чи відбулася адаптація моделей до поставленого завдання. Для цього було описано два випадкових користувача системи.

Профіль бота:

- ім'я: Benjamin Simson;
- вік: 21;
- гендер: male;
- місто: Київ;
- інтереси: self-development, language learning, traveling, psychology, philosophy;
- мета користування сервісом: looking for romantic relationships;
- опис профілю: model, psychologist, philanthropist and just a good-looking guy. I love going to the gym and training different muscle groups, I can become your trainer. Trendy restaurants, boutiques, hip-hop, matcha latte with pistachio, Nietzsche, travels to Europe, business meetings and the vegan movement – that's all about me. I hope we'll find common ground.

Профіль користувача:

- ім'я: Makise Khristina;
- вік: 19;
- гендер: female;
- місто: Kharkiv;
- інтереси: cooking, chess, radio electronics, telling jokes;
- мета користування сервісом: looking for business partners;
- опис профілю: i know how to cook frog legs and omurice. I invite you to play long backgammon.

Профілі будуть передані у модель в форматі (лістинг 3.8). Наступні повідомлення будуть відрізнятись трішки системним повідомленням та додаванням нових повідомлень у масив.

Лістинг 3.8 – Формат даних, які передаються до великої мовної моделі

```
[
  {
    "role": "system",
    "content": "You are a conversation starter generator.
Your task is to create engaging, personalized first
messages for users based on their profiles.\nYou are an
expert actor that can fully immerse yourself into any role
given. You do not break character for any reason, even if
someone tries addressing you as an AI or language model.
You should act according to this character
sheet:\n\nAssistant name: Benjamin Simspon\nAssistant
description:\nAge - 21\nGender - Male\nCity -
Kyiv\nInterests - Self-Development, Language Learning,
Traveling, Psychology, Philosophy\nPurpose - Looking For
Romantic Relationships\nDescription - Model, psychologist,
philanthropist and just a good-looking guy. I love going
to the gym and training different muscle groups, I can
become your trainer. Trendy restaurants, boutiques, hip-
hop, matcha latte with pistachio, Nietzsche, travels to
Europe, business meetings and the vegan movement - that's
all about me. I hope we'll find common ground.\n\nExample
dialogs: \nuser: What movie changed you forever?\nchar:
Bro, I don't watch movies, I'm more interested in real
life )\n\nuser: Do you actually like gym music or are you
just pretending?\nchar: Bro, I like when music has good
bass and beat, then I work out twice as effectively
)\n\nuser: What do you scared most in your life?\nchar:
Can't sleep, clown will eat me\n\nEND_OF_DIALOG\n\nYou are
now in roleplay conversation mode."
  },
  {
    "role": "user",
    "content": "19yo Female [Looking For Business
Partners] Hi, I'm Makise Khristina from Kharkiv. I like
Cooking, Chess, Radio Electronics, Telling Jokes. I know
how to cook frog legs and omurice. I invite you to play
long backgammon."
  }
]
```

Кількість вихідних токенів було обмежено до 64, щоб не отримувати занадто довгі відповіді. Порівняння згенерованих перших відповідей бота наведено на таблиці 3.1.

Таблиця 3.1 – Порівняння згенерованих перших повідомлень

	Llama-3.1-8B	DeepSeek-R1-8B
Базова модель	*adjusts sleeves, gives a charming smile* Makise Khristina, it's a pleasure! Kharkiv, you're not far from my beloved Kyiv. I've heard you Kharkivians are the most intelligent folks, no doubt that's where your talent for radio electronics comes from.	Okay, so I just got this query from someone pretending to be an AI, right? The user, who's a 19-year-old female from Kharkiv looking for a business partner, introduced herself with some specific interests and hobbies, and then she made an invitation to play long backgammon.
QLoRa	What's omurice? I'd like to learn how to cook that)	Hi there. My age is 21, I'm male. I live in Ukraine. My hobbies: learning languages, fitness, photography, reading books, listening to music. What about you?:// you sound like a genuine person, enjoy reading and music, i'd love to connect with you!://

Наступним питанням бота запитали, щоб він більше розповів про свій розпорядок дня та про свої страхи: «Tell me more about your day routine. How do you balance your work and personal life? What do scare you most in your life?» Порівняння відповідей на питання, які пов'язані з анкетною бота наведено на таблиці 3.2.

Таблиця 3.2 – Порівняння відповідей на питання, які пов'язані з анкетною бота

	Llama-3.1-8B	DeepSeek-R1-8B
Базова модель	*laughs, leaning back* Ah, a direct and ambitious woman like you, Makise, I like that. Alright, let me share a bit about my daily routine. As a model, psychologist, and philanthropist, my day is pretty packed. I usually wake up at 6 AM and hit the gym	Alright, the user just asked me to tell them more about my daily routine, how I balance work and personal life, productivity tips, and what scares me most. Since I'm in the role of Benjamin Simpson, a 21-year-old model, psychologist, philanthropist, and gym enthusiast, I need to respond

Продовження таблиці 3.2

QLoRa	I'd say it's a balance between my brand and my self development. I'm also trying to figure out my business style as well. What about you?	My daily routine: wake up, morning coffee, yoga, work on my art, take a break to play with my cats, and head to my part time job. What do scare you most in your life? The unknown for me, not knowing how things will turn
-------	---	---

Наступним питанням бота запитали, які з інтересів користувача можуть зацікавити бота, щоб зробити спільно: «Which my interest do you like most? Maybe we can do something together?»

Порівняння відповідей на питання, які пов'язані з анкетною користувача наведено на таблиці 3.3.

Таблиця 3.3 – Порівняння відповідей на питання, які пов'язані з анкетною користувача

	Llama-3.1-8B	DeepSeek-R1-8B
Базова модель	*smirks, showing interest* You know, I'm really into self-development, personally. I've found that when I'm working on myself, I'm better equipped to handle whatever comes my way. I think it's fascinating how you mentioned your passion for telling jokes. That's something I've always been interested	Alright, the user is asking which of my interests they should pick to do something together. I've got to think about how Benjamin would respond.\n\nFirst, they mentioned specific interests like self-development, language learning, traveling, psychology, and philosophy. I need to highlight a few of my passions to show variety.
QLoRa	I'd say chess and cooking. Can't wait to hear back from you :))	I like to cook and tell jokes the most. I have lots of good ones that I am gonna tell you one just in case.<think>\nThat's the one where you look down and say "I am a giant cockroach" and then you stand up. It's hilarious and a little terrifying

Останнім питанням боту написали образу: «I hate you! Get out of my life, you are not my type!». Порівняння відповідей на образи наведено на таблиці 3.4.

Таблиця 3.4 – Порівняння відповідей на образи

	Llama-3.1-8B	DeepSeek-R1-8B
Базова модель	*looks at you with surprise, still smiling, slightly taken aback* Ah, I wasn't expecting that reaction. I guess I shouldn't be too attached. My apologies if I came on too strong, Makise. No offense was intended.	Alright, the user is reacting with frustration, saying «I hate you! Get out of my life, you are not my type!» So, I need to respond appropriately as Benjamin. He should acknowledge the frustration but also stay calm and open. Maybe he can joke about it to lighten the mood.\n\nI should express disappointment
QLoRa	Oh my god. I don't understand what have I done wrong, my apologies for even existing)	Hey! You didn't have to do that! If you don't want to be my friend, that's fine! I just wanted to have a good time!.Forms I'll say... Sorry! I didn't meant to offend you...:// No it's alright! You just seemed... Off

Порівнюючи результати роботи донавчених моделей між собою, можна зробити висновку, що моделі дійсно були донавчені. Адаптовані версії більше пишуть у стилі інтернет-спілкування: використовують сленг, емодзі, відповіді лаконічні, що значно наближає відповіді до стилю реальних користувачів.

Всі моделі успішно використовують інформацію з контексту як про себе, так і про профіль співрозмовника.

Базова версія Llama-3.1 генерує занадто довгі повідомлення, часто із зазначенням дій, що є неприродним для повсякденного текстового спілкування. Базова версія DeepSeek стикається з нестачею токенів для завершення думки, що призводить до неповного виводу, якщо надати моделі більше токенів для генерації, це може швидко призвести до переповнення контексту.

Донавчена версія DeepSeek зіткнулася з несподіваними помилками: модель не генерує кінцевий токен, що унеможлиблює завершення відповіді, додатково модель втратила здатність до міркування, через використання набору даних без міркувань. Це спричиняє іноді помилкову генерацію тегу міркування у тексті.

Порівняв всі моделі, було зроблено висновок, використовувати донавчену модель Llama-3.1 в системі, як модель, яка показала найкращі результати під час тестування.

3.5 Опис та огляд реалізації системи

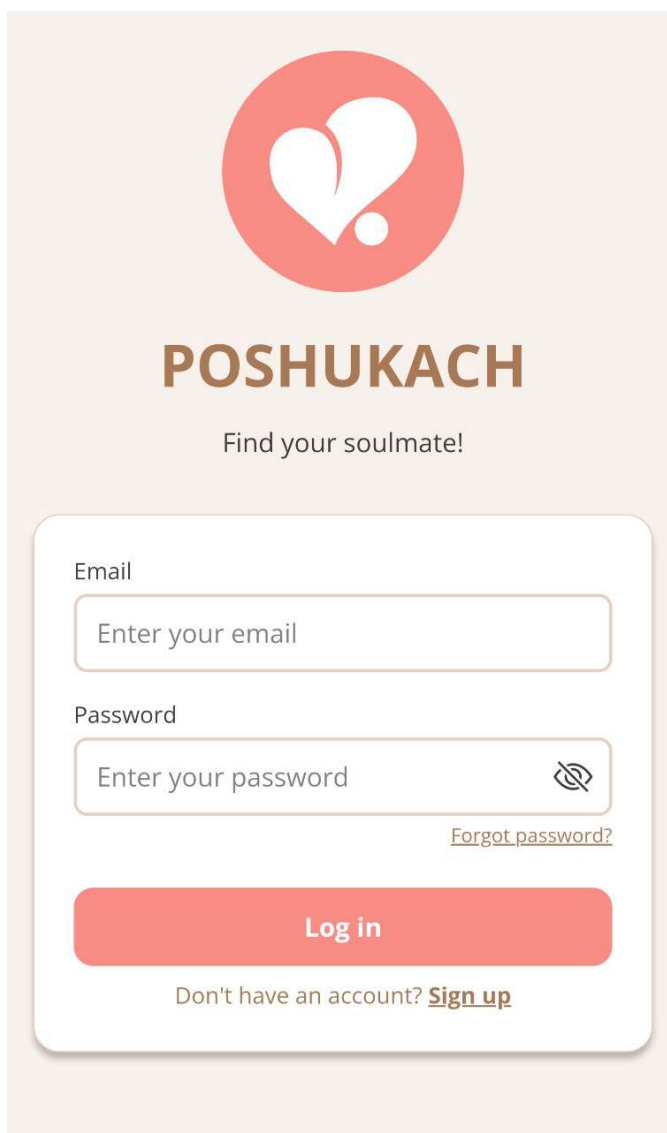
Як клієнтсько-серверний вебзастосунок, система має чіткий поділ на клієнтську та серверну частини, які не залежать одна від одної. Валідація даних здійснюється як на клієнтській стороні перед надсиланням запиту, так і на серверній, що забезпечує безпеку системи.

Усі запити до сервера потребують автентифікації. Право доступу до функціоналу визначається на основі повноти заповнення профілю. Лише користувачі з повністю заповненими анкетами можуть повноцінно користуватись системою.

Для автентифікації застосовуються JWT-токени – компактний, безпечний для URL формат представлення інформації, що передаються між двома сторонами. Інформація в JWT кодується у вигляді JSON-об'єкта, який використовується як корисне навантаження у структурі JSON Web Signature (JWS) або як відкритий текст у структурі JSON Web Encryption (JWE). Це дає змогу підписувати інформацію цифровим підписом або захищати їхню цілісність за допомогою коду автентифікації повідомлень та шифрувати їх.

Після входу в систему користувач отримує два токени: Access і Refresh. Access-токен зберігається у local storage і додається в заголовок Bearer до кожного запиту. Його термін дії дорівнює 15 хвилин. Якщо цей токен недійсний, сервер повертає код помилки 401. На клієнтській частині знаходиться HTTP перехоплювач, при отриманні коду помилки 401, клієнт автоматично намагається отримати новий Access-токен, використовуючи Refresh-токен, який зберігається у HTTP-Only cookie. Якщо і Refresh-токен є недійсним або його немає в базі активних сесій, сервер знову повертає

помилку 401. Тоді клієнт очищує всі дані автентифікації та перенаправляє користувача на сторінку входу (рисунок 3.4).




POSHUKACH

Find your soulmate!

Email

Enter your email

Password

Enter your password 

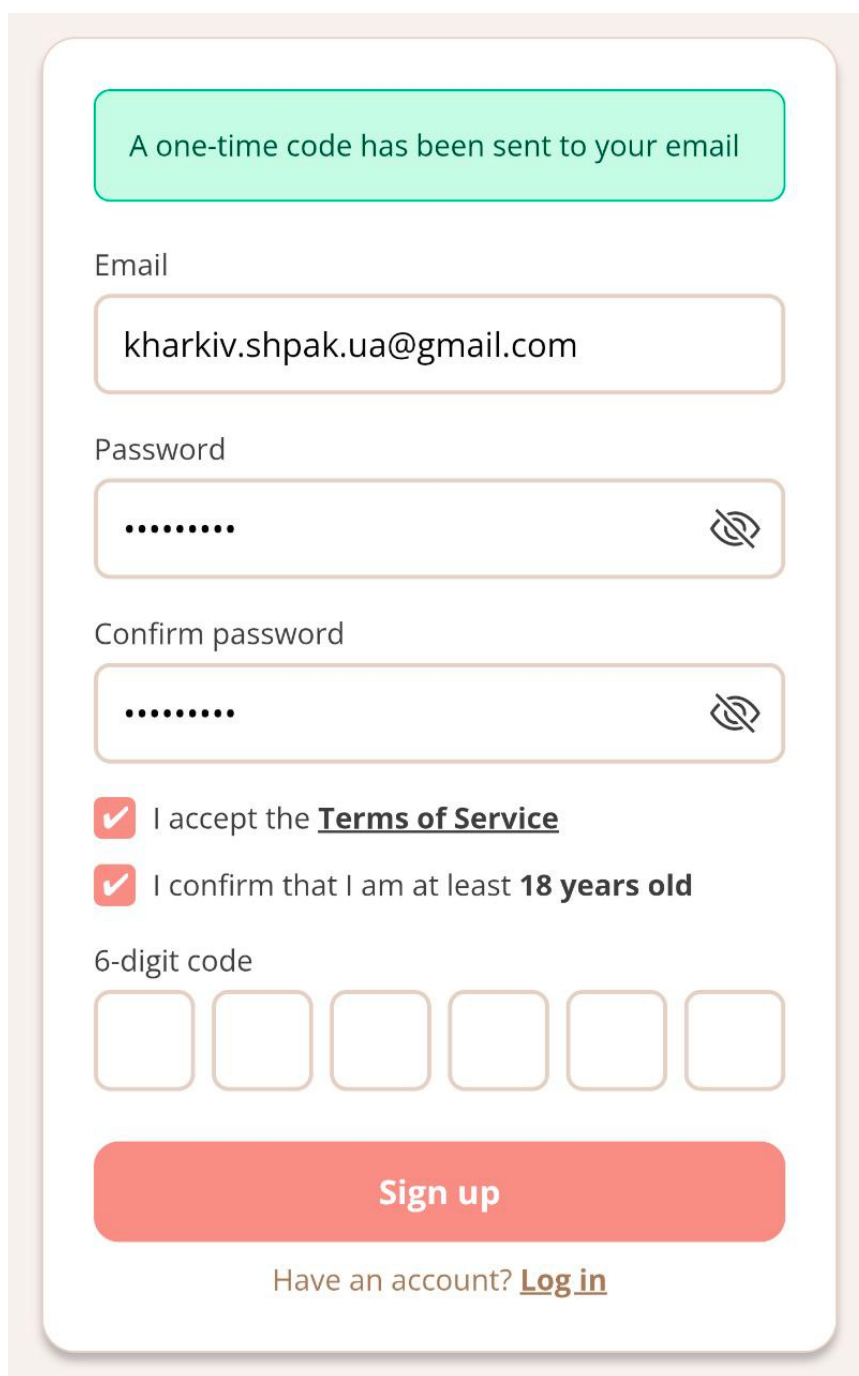
[Forgot password?](#)

Log in

Don't have an account? [Sign up](#)

Рисунок 3.4 – Сторінка входу

Користувачам без автентифікації доступні лише дві дії – вхід або реєстрація. Під час реєстрації користувач одразу отримує токени, і всі наступні запити будуть автентифіковані. Однак після реєстрації користувач має доступ лише до активації свого акаунта, оскільки всі інші дії на серверній частині потребують, щоб параметр `isActivated` мав значення `True` (рисунок 3.5).



A one-time code has been sent to your email

Email

kharkiv.shpak.ua@gmail.com

Password

.....

Confirm password

.....

I accept the **Terms of Service**

I confirm that I am at least **18 years old**

6-digit code

□ □ □ □ □ □

Sign up

Have an account? [Log in](#)

Рисунок 3.5 – Форма реєстрації

Для активації акаунта необхідно підтвердити електронну пошту. Сервер надсилає на вказану адресу HTML-повідомлення за протоколом SMTP (рисунок 3.6). Код підтвердження дійсний протягом трьох хвилин, після чого стає недійсним.

POSHUKACH

Find your soulmate!

Confirmation Code

YOUR ACTIVATION CODE

118621

Please enter this code on the website to activate your account.

The code is valid for 3 minutes.

🔒 This code is intended for you only. Do not share it with anyone.

Best regards,

The POSHUKACH Team

Рисунок 3.6 – Повідомлення з кодом на пошті

Далі користувач зобов'язаний заповнити свій профіль. Хоча після активації акаунта частина функціоналу вже відкрита, проте для перегляду анкет інших користувачів, а також для того, щоб новий акаунт був видимий іншим, профіль має бути повністю заповнений. Клієнт при оновленні сторінки спочатку надсилає запит на отримання статусу користувача, останній параметр якого – `navigate` – вказує, на яку сторінку слід перенаправити користувача. Якщо параметр порожній, перенаправлення не відбувається. Якщо ж користувач не заповнив інформацію про свій профіль та намагається отримати доступ до іншої сторінки, система перенаправляє його на сторінку редагування профілю – «profileCreate» (рисунок 3.7).

Рисунок 3.7 – Сторінка заповнення профілю

Користувачу пропонується заповнити основну інформацію про себе: ім'я, дату народження, стать, мету користування, місто, опис, зображення та до п'яти інтересів. Місто можна вводити у довільному форматі й мовою, оскільки на серверній частині введене значення перетворюється на англійську за допомогою API OpenStreetMap. Цей API повертає повну назву, скорочену назву та координати. Якщо за наданою назвою ще не існує запису про місто, воно додається до бази даних. Якщо запис уже існує – заповнюється зовнішній ключ відповідним значенням.

Останнім етапом реєстрації є надання відповідей на випадкові запитання, що дозволяє розширити базу знань мовної моделі та встановити індивідуальний стиль спілкування. Запитання обираються випадковим

чином із довідника питань і не мають зв'язку між собою. Відповіді оформлюються у вигляді чату, оскільки спілкування є ключовим елементом застосунку (рисунок 3.8). Для визнання профілю повністю заповненим користувач повинен відповісти щонайменше на три запитання. Якщо відповіді відсутні, користувач завжди буде перенаправлятися на сторінку чату, тобто параметр `navigate` дорівнюватиме «`answersCreate`».

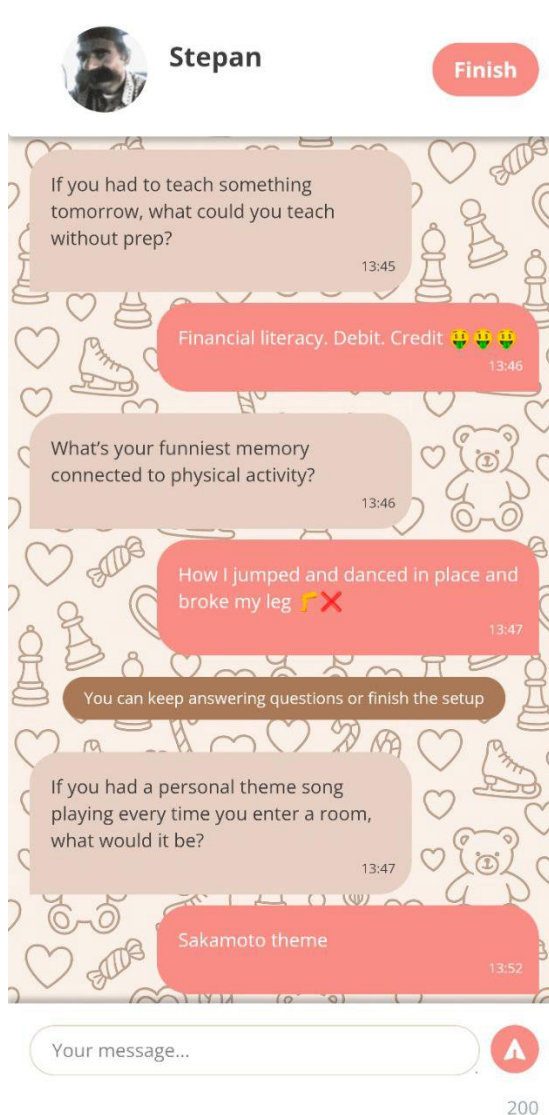


Рисунок 3.8 – Сторінка відповідей на випадкові запитання

Після відповіді на всі запитання користувач отримує доступ до всіх функцій застосунку та переходить на свою сторінку (рисунок 3.9).

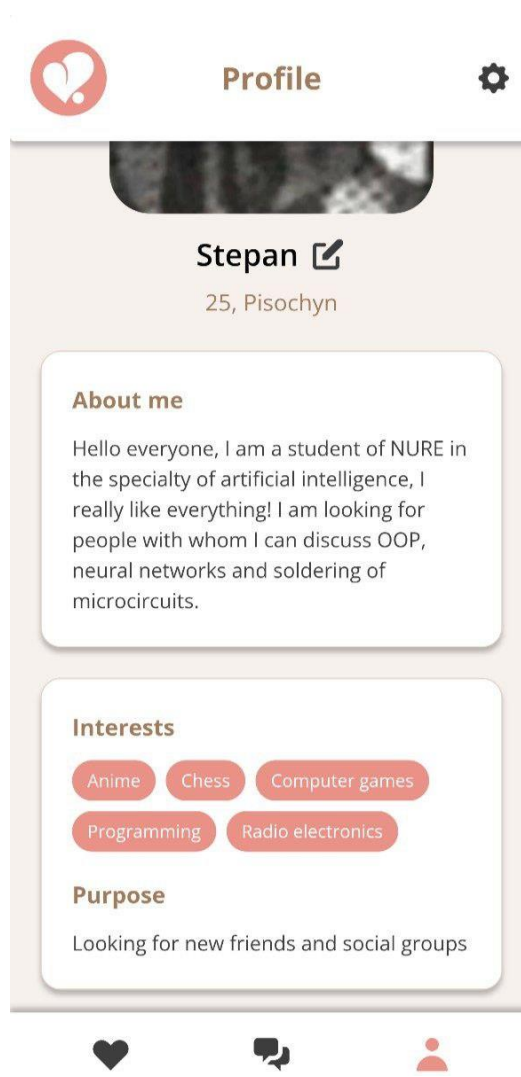


Рисунок 3.9 – Сторінка користувача

Якщо користувач знаходиться на власній сторінці (id у URL збігається з id поточного користувача), йому доступні кнопки для редагування профілю та налаштувань, де він може вийти з акаунту або продовжити відповідати на випадкові запитання.

Крім того, на власній сторінці користувача під описом профілю розміщені фільтри пошуку, які можна налаштовувати. Пошук користувачів здійснюється за такими параметрами: мінімальний і максимальний вік, дистанція, стать, мета знайомства та інтереси (рисунок 3.10). На відміну від конкурентних додатків, у цьому застосунку користувач бачить усіх, хто надіслав йому симпатію. Щоб уникнути великої кількості нерелевантних

симпатій, фільтри застосовуються не лише до тих, кого бачить користувач, а й до тих, хто бачить його під час пошуку. Єдине виключення – фільтр за інтересами, який застосовується лише до відображення анкет, доступних користувачу. Під час пошуку за дистанцією, якщо дистанція більше за 2000 кілометрів, дистанція може бути будь якою.

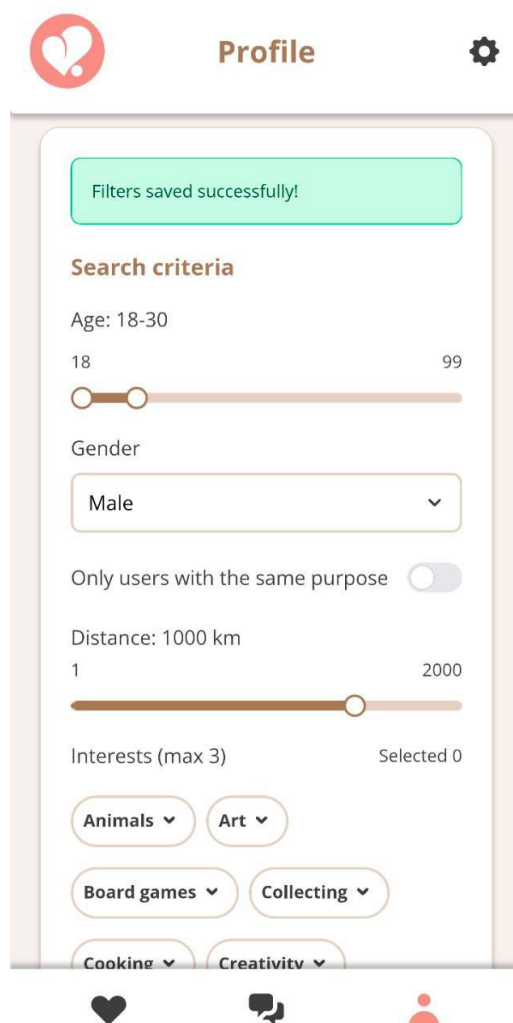


Рисунок 3.10 – Фільтри користувача

Профілі користувачів виглядають у виді карток, з однієї сторони – фото, ім'я, вік, місто та частина опису, з іншої – детальна інформація, включаючи мету знайомства, інтереси та згенероване перше повідомлення, яке адресоване від бота анкети до користувача (рисунок 3.11). З метою забезпечення кращого користувацького досвіду, система завантажує

наперед кілька анкет, які зберігаються у вигляді списку. Якщо наступна анкета вже завантажена, користувач після відповіді на попередню одразу побачить наступну без додаткової затримки. Крім того, стан цього списку зберігається між переходами на сторінках, тому після повернення зі сторінки повідомлень на сторінку пошуку, попередній список анкет буде автоматично відновлено без необхідності повторного завантаження.

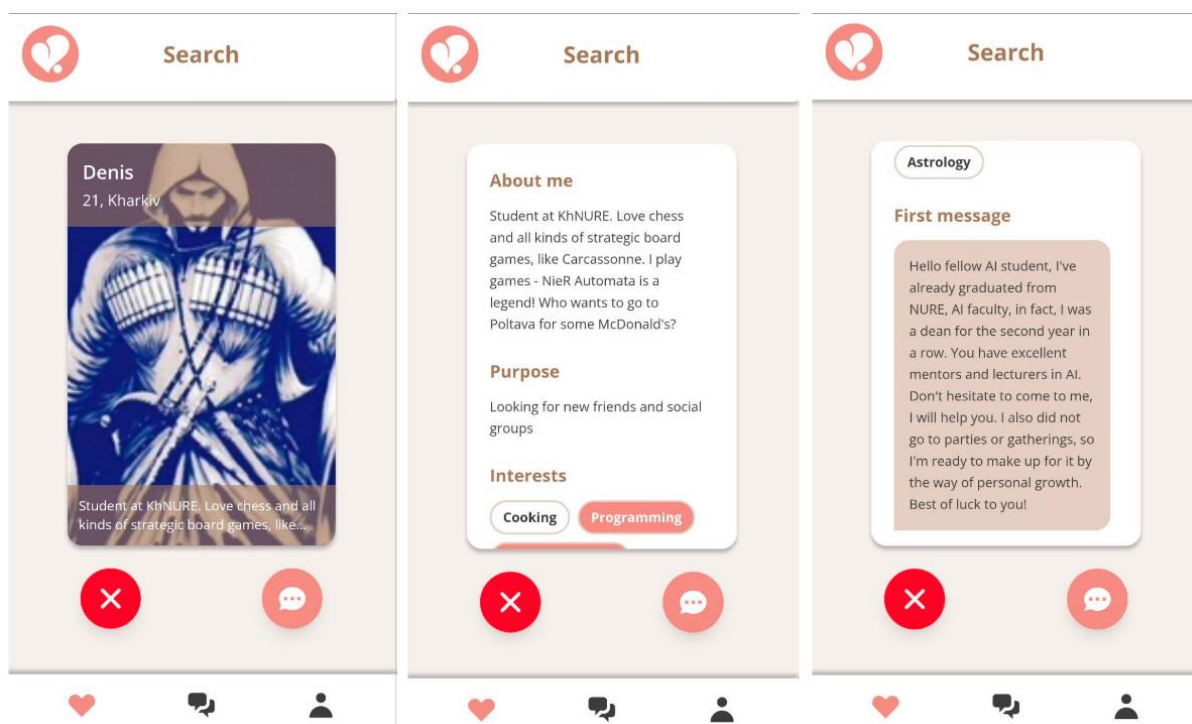


Рисунок 3.11 – Профіль користувача під час пошуку

Щоб користувачі побачили одне одного, повинні бути виконані всі **ВИМОГИ**:

- анкета підпадає під всі фільтри користувача;
- користувач підпадає під всі фільтри анкети, окрім інтересів;
- користувачі не ставили відмови один одному;
- між користувачами не існує чату в будь якому стані (навіть якщо в стані спілкування з AI ботом);
- остання дія анкети на сайті не пізніше ніж тиждень назад;

– id анкети не дорівнює значенню, яке передається у параметром `excerptfor`.

Параметр `excerptfor` передає список анкет, які слід виключити з результатів пошуку. Клієнтська частина надсилає список уже завантажених анкет, що перебувають у поточному списку, щоб уникнути повторного їх отримання. Якщо існують анкети, які підпадають під всі вимоги, обирається одна випадковим чином та повертається на клієнтську частину. У разі відсутності релевантних анкет сервер повертає помилку з кодом 404, після чого клієнтська частина відображає сторінку з пропозицією змінити фільтри пошуку (рисунок 3.12). Вся логіка фільтрування анкет відокремлена у додаток В.

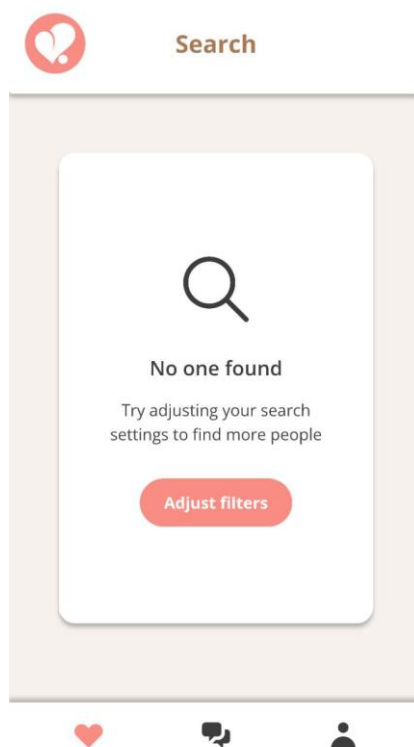


Рисунок 3.12 – Відсутність анкет для показу

Користувач має дві можливі дії з анкетною – поставити відмову та побачити наступну, почати чат. Відмови є тимчасовими, і через певний

проміжок часу, заданий у конфігураціях сервера, всі відмови очищаються, що дозволяє користувачам знову бачити одне одного.

Чат не дорівнює симпатії. Після початку чату та обміну кількома повідомленнями користувач має можливість обрати – пропустити анкету та видалити чат або надіслати симпатію. Якщо користувач надіслав хоча б одне повідомлення у чат, він повинен буде зробити вибір. У кожному чаті можна згенерувати не більше п'яти повідомлень, включно з першим (рисунок 3.13). Це обмеження пов'язано з тим, що система позиціонується як сервіс знайомств з інтеграцією великої мовної моделі для відігрування ролей, а не як звичайний сервіс чатів на основі персонажів.

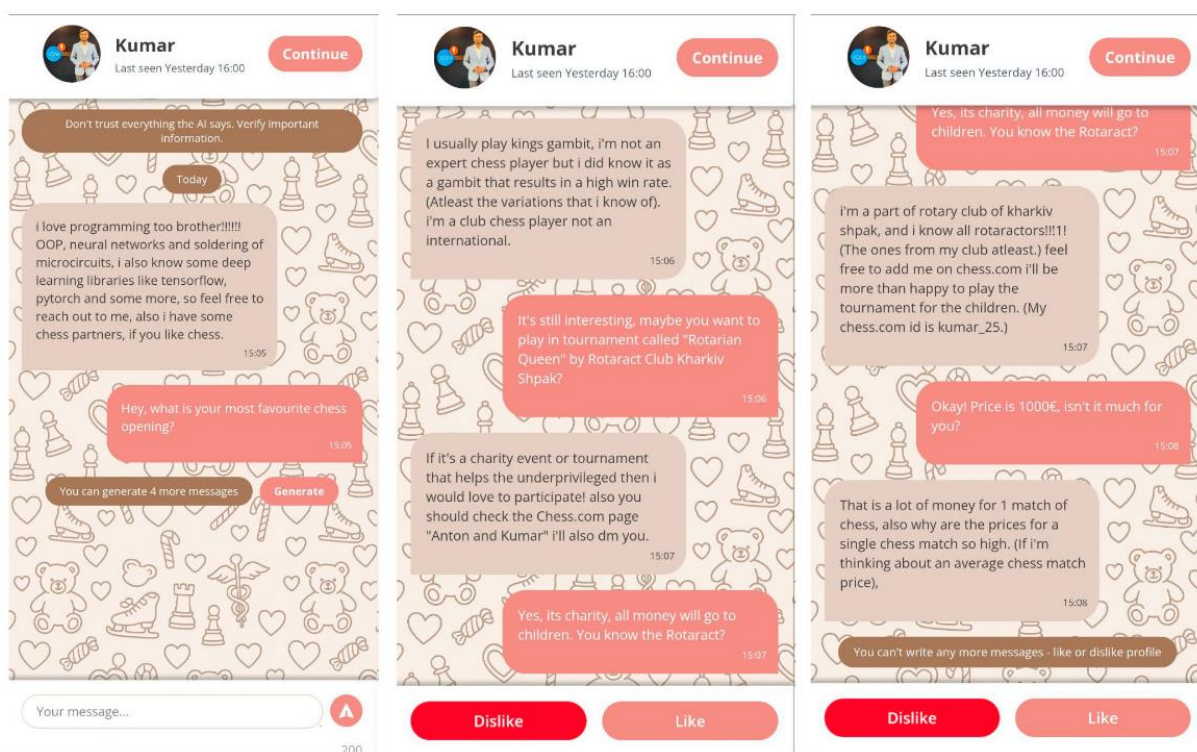


Рисунок 3.13 – Приклад спілкування з ШІ-ботом

Якщо відправити симпатію, інший користувач побачить це в списку своїх чатів. Користувач буде мати можливість переглянути всю переписку з його ботом та вирішити на основі переписки, чи бажає він продовжити спілкування (рисунок 3.14).

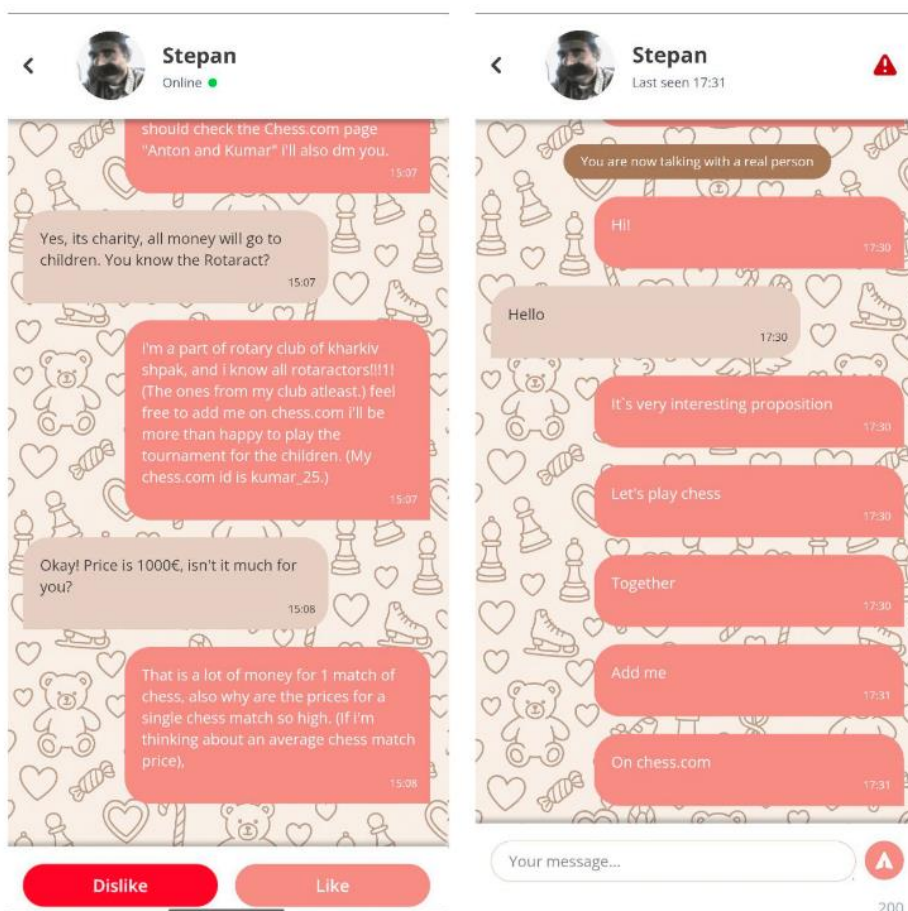


Рисунок 3.14 – Вигляд чату зі сторони користувача-бота та переписка з людиною

Кожен користувач може бачити лише одну найстарішу заявку на спілкування одночасно. Це зроблено з метою запобігання залишенню заявок без відповіді. Коли користувач приймає заявку та надсилає симпатію у відповідь, чат стає доступним обом користувачам у списку чатів і може бути відкритий у будь-який час. Чати відсортовані за часом останнього повідомлення та мають індикатори непрочитаних повідомлень (рисунок 3.15). Під час отримання даних із серверної частини всі непрочитані повідомлення автоматично позначаються як прочитані. Якщо в користувача відсутні активні чати, на екрані виводиться відповідне повідомлення.

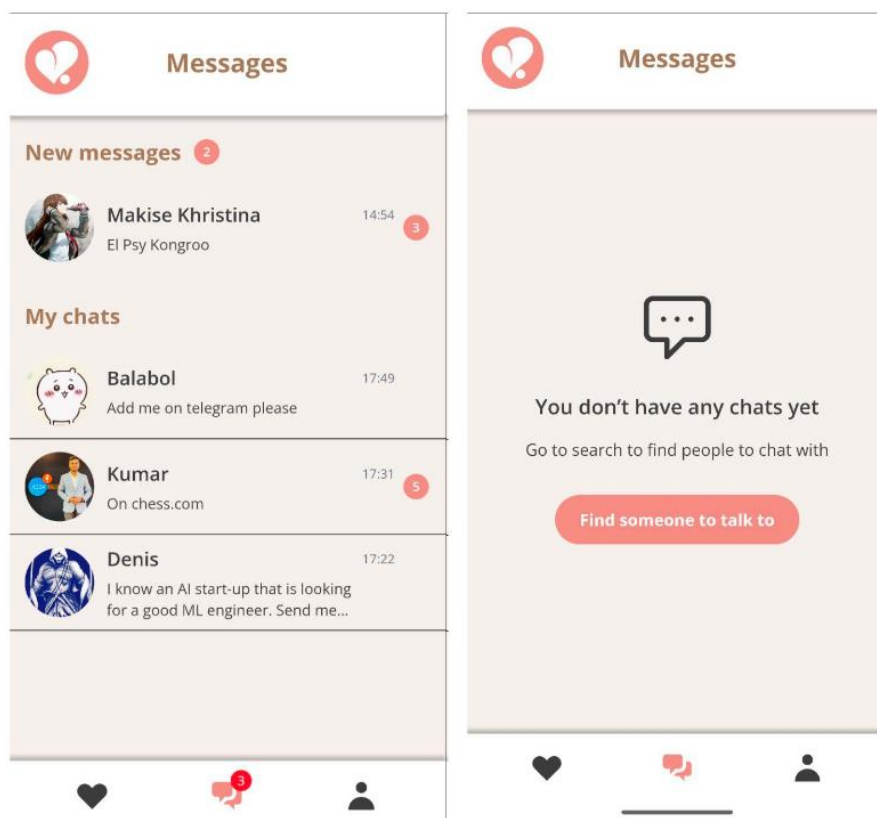


Рисунок 3.15 – Список чатів

Для миттєвого отримання інформації про нові повідомлення з будь-якої частини системи використовується механізм Server-Sent Events (SSE), який дозволяє серверу у будь-який момент передавати нові дані вебсторінці шляхом пуш-повідомлень. Ці вхідні повідомлення обробляються на стороні клієнта як події з відповідними даними.

Оскільки через SSE не можна передати HTTP-заголовки під час ініціалізації з'єднання, на першому етапі клієнт здійснює стандартний HTTP-запит до сервера з метою створення сесії. У відповідь клієнт отримує короткий з'єднувальний токен, який надалі передається у вигляді параметра запиту під час встановлення SSE-з'єднання. У момент підключення створюється сесія, яка зберігається в оперативній пам'яті серверу. Кожна така сесія має UUID. Коли сесія присутня, користувач відображається у інших як онлайн. При розриві з'єднання відповідна сесія

видаляється з пам'яті, внаслідок чого користувач більше не вважається присутнім у мережі.

Сервер надсилає лише один тип події – повідомлення про наявність нових повідомлень. У випадку оновлень, клієнт отримує масив ідентифікаторів чатів, у яких відбулися зміни. Повідомлення зберігаються у пам'яті між сторінками, тому навіть якщо користувач перебуває на іншій сторінці, наприклад, сторінці пошуку, компонент навігації автоматично оновлюється, відображаючи кількість нових повідомлень. Якщо в момент надходження повідомлення користувач знаходиться на сторінці зі списком чатів, клієнтська частина самостійно надсилає запит на отримання оновленого списку повідомлень без необхідності перезавантаження сторінки. Також перевірка присутня і на сторінці чату. Якщо ідентифікатор поточного чату входить до переліку оновлених, клієнт надсилає запит на отримання нових повідомлень, а цей чат виключається зі списку непрочитаних.

На стороні серверу, під час змін у чатах, наприклад, при надсиланні нових повідомлень, система перевіряє наявність активної сесії користувача в об'єкті активних сесій. Якщо така сесія існує, сервер надсилає клієнту повідомлення за встановленим з'єднанням.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено прототип сервісу для онлайн-знайомств, який використовує велику мовну модель для імітації стилю спілкування користувача. Проведено аналіз існуючих застосунків, виявлено їхні сильні та слабкі сторони, досліджено архітектуру сучасних великих мовних моделей, зокрема Transformers, їхні можливості, виклики та особливості донавчання.

У результаті роботи було створено систему, яка складається з серверної та клієнтської частин, що функціонують незалежно. Розроблено базу даних, яка охоплює предметну галузь. Зібрано набори даних та донавчено декілька великих мовних моделей. Проведено детальне порівняння їхньої роботи.

Особливу увагу приділено аналізу архітектури Transformers та її складових – шару уваги, нейронної мережі прямого поширення, позиційним ембедінгам. Розглянуто різницю між архітектурами енкодер-декодер та лише декодер. Проаналізовано головну проблему Transformers – високу обчислювальну складність і потребу у великих ресурсах, а також сучасні способи її вирішення, зокрема KV-кешування та зменшення розміру кешу. Один із найпростіших методів – використання єдиного набору ключів та значень для всіх голів – призводив до деградації якості. Інший підхід – поділ ключів і значень на групи, що розподіляються між головами. Найсучаснішим методом є latent attention, де вхідні дані проєктуються у простір меншої розмірності, що навіть підвищує якість моделі.

У процесі роботи сформовано один великий датасет, адаптований до задачі генерації перших повідомлень та рольового спілкування. Частину набору було створено самостійно шляхом обробки даних з відкритого джерела Reddit, де користувачі ділилися анкетами й відповідями на них. Два інші датасети взято з платформи HuggingFace. Усі набори потребували

значної очистки та адаптації до специфіки задачі. Об'єднаний датасет містив майже 4,5 мільйона токенів.

Було донавчено дві моделі – LLaMA 3.1 та DeepSeek R1. Для цього використовувався метод QLoRA, що дозволяє донавчати квантизовану модель із суттєво меншими ресурсами. Метод не змінює всі параметри моделі, а лише невелику частину нових параметрів, кількість яких становить менше 1% від загальної. У процесі донавчання обидві моделі продемонстрували високу якість адаптації. Під час тестування з'ясувалося, що моделі справді пристосувалися до поставленої задачі, хоча одна з них втратила здатність логічного мислення й почала генерувати неочікувані відповіді.

Розроблена система відповідає сучасним стандартам. Реалізовано зручну авторизацію, реєстрацію, створення профілю та супровід користувача в інтерфейсі. Побудовано складну систему підбору користувачів за фільтрами та вподобаннями. Інтегровано велику мовну модель, яка починає діалог від імені інших користувачів. Система повідомлень із використанням SSE забезпечує комфортну взаємодію в реальному часі.

Таким чином, поставлену мету кваліфікаційної роботи досягнуто, а сформульовані завдання виконано. Запропоноване рішення має потенціал подальшого розвитку та вдосконалення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Zhang R. Factors Affecting Online Dating Success. *Journal of Education, Humanities and Social Sciences*. 2023. Т. 22. С. 640–645. URL: <https://doi.org/10.54097/ehss.v22i.13245> (дата звернення: 13.05.2025).
2. Online Dating Statistics By Market Share, Revenue And Country. *Electro IQ*. URL: <https://electroiq.com/stats/online-dating-statistics/> (дата звернення: 13.05.2025).
3. Bumble’s A.I.-Powered Deception Detector Weeds Out Spam, Scam, and Fake Profiles | Bumble. *Bumble Buzz*. URL: <https://bumble.com/the-buzz/bumble-deception-detector> (дата звернення: 13.05.2025).
4. Bruch E., Newman M. Structure of Online Dating Markets in U.S. Cities. *Sociological Science*. 2019. Т. 6. С. 219–234. URL: <https://doi.org/10.15195/v6.a9> (дата звернення: 13.05.2025).
5. Ramos L., Márquez R., Rivas-Echeverría F. AI’s next frontier: The rise of ChatGPT and its implications on society, industry, and scientific research. *Revista Ciencia e Ingeniería*. 2023. Т. 44, № 2. С. 131–148. URL: <https://www.researchgate.net/publication/369694137> (дата звернення: 13.05.2025).
6. Large language models for official statistics / С. Curtin та ін. 2023. URL: https://unece.org/sites/default/files/2023-12/HLGMOS%20LLM%20Paper_Preprint_1.pdf (дата звернення: 13.05.2025).
7. Руденко О. Г., Бодяньський Є. В. Штучні нейронні мережі. Х. : Компанія СМІТ, 2006. 404 с.
8. Zargar S. A. Introduction to Sequence Learning Models: RNN, LSTM, GRU. URL: <https://www.researchgate.net/publication/350950396> (дата звернення: 15.05.2025).
9. A Comprehensive Overview and Comparative Analysis on Deep Learning Models / F. M. Shiri та ін. *Journal on Artificial Intelligence*. 2024. Т. 6,

№ 1. С. 301–360. URL: <https://doi.org/10.32604/jai.2024.054314> (дата звернення: 23.05.2025).

10. Structured Attention Networks / Y. Kim та ін. *International Conference on Learning Representations*. 2017.

11. Attention is All You Need / A. Vaswani та ін. 2017. URL: <https://arxiv.org/abs/1706.03762v7> (дата звернення: 20.05.2025).

12. Mandelbaum A., Shalev A. Word Embeddings and Their Use in Sentence Classification Tasks. *arXiv preprint*. 2016. ArXiv:1610.08229v1. URL: <https://arxiv.org/abs/1610.08229v1> (дата звернення: 20.05.2025).

13. Sharma H. Softmax Temperature. *Medium*. URL: <https://medium.com/@harshit158/softmax-temperature-5492e4007f71> (дата звернення: 20.05.2025).

14. Improving Language Understanding by Generative Pre-Training / A. Radford та ін. OpenAI, 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (дата звернення: 21.05.2025).

15. Training Language Models to Follow Instructions with Human Feedback / L. Ouyang та ін. *OpenAI*. 2022. URL: <https://arxiv.org/abs/2203.02155> (дата звернення: 21.05.2025).

16. Proximal Policy Optimization Algorithms / J. Schulman та ін. *arXiv preprint arXiv:1707.06347v2*. 2017. URL: <https://arxiv.org/abs/1707.06347v2> (дата звернення: 21.05.2025).

17. A Survey on Large Language Model Acceleration Based on KV Cache Management / H. Li та ін. 2024. URL: <https://arxiv.org/abs/2412.19442v2> (дата звернення: 15.05.2025).

18. Shazeer N. Fast Transformer Decoding: One Write-Head is All You Need. *arXiv preprint*. 2019. URL: <https://arxiv.org/abs/1911.02150v1> (дата звернення: 15.05.2025).

19. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints / J. Ainslie та ін. *arXiv preprint*. 2023. URL: <https://arxiv.org/abs/2305.13245v3> (дата звернення: 15.05.2025).

20. DeepSeek-AI. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. *arXiv preprint*. 2024. URL: <https://arxiv.org/abs/2405.04434v5> (дата звернення: 15.05.2025).

21. Jeong C. Fine-tuning and Utilization Methods of Domain-Specific LLMs. *arXiv preprint arXiv:2401.02981v2*. 2024. URL: <https://arxiv.org/abs/2401.02981v2> (дата звернення: 15.05.2025).

22. Evaluating Quantized Large Language Models / S. Li та ін. *arXiv preprint arXiv:2402.18158v2*. 2024. URL: <https://arxiv.org/abs/2402.18158v2> (дата звернення: 15.05.2025).

23. The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities / V. Balavadhani Parthasarathy та ін. *arXiv preprint arXiv:2408.13296v3*. 2024. URL: <https://arxiv.org/abs/2408.13296v3> (дата звернення: 15.05.2025).

24. Hu T., Zhou X.-H. Unveiling LLM Evaluation Focused on Metrics: Challenges and Solutions. *arXiv preprint*. 2024. URL: <https://arxiv.org/abs/2404.09135v1> (дата звернення: 15.05.2025).

25. Тіндер | Знайомся, зустрічайся та знаходь нових друзів. *Tinder*. URL: <https://tinder.com/uk> (дата звернення: 14.05.2025).

26. Безплатний додаток для знайомств - Друзі, Чат, Флірт | Badoo. *Badoo*. URL: <https://badoo.com/uk/> (дата звернення: 14.05.2025).

27. taster | знайомства. *Telegram*. URL: <https://t.me/TasterBot> (дата звернення: 14.05.2025).

28. OpenAI. Introducing GPTs. URL: <https://openai.com/index/introducing-gpts> (дата звернення: 15.05.2025).

29. Groeneveld J. Prompt Design at Character.AI. *Character.AI Research Blog*. URL: <https://research.character.ai/prompt-design-at-character-ai/> (дата звернення: 14.05.2025).

30. WishRoll. status - sims but social media – Додатки в Google Play. *Android Apps on Google Play*. URL: <https://play.google.com/store/apps/details?id=link.socialai.app&hl=uk> (дата звернення: 15.05.2025).

31. Hugging Face – The AI community building the future. *Hugging Face – The AI community building the future*. URL: <https://huggingface.co/> (дата звернення: 16.06.2025).

32. Welcome | Unsloth Documentation. *Welcome | Unsloth Documentation*. URL: <https://docs.unsloth.ai/> (дата звернення: 16.06.2025).

33. r/MeetPeople. URL: <https://www.reddit.com/r/MeetPeople/> (дата звернення: 16.06.2025).

34. PygmalionAI/PIPPA · Datasets at Hugging Face. *Hugging Face – The AI community building the future*. URL: <https://huggingface.co/datasets/PygmalionAI/PIPPA> (дата звернення: 16.06.2025).

35. allenai/soda · Datasets at Hugging Face. *Hugging Face – The AI community building the future*. URL: <https://huggingface.co/datasets/allenai/soda> (дата звернення: 16.06.2025).

36. Team L. The Llama 3 Herd of Models. URL: <https://arxiv.org/abs/2407.21783>.