

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 другий (магістерський)

Метод синтезу навчальної вибірки для LLM на основі  
методу навчання з підкріпленням  
(тема)

Виконав:

здобувач другого року навчання,

групи ДСМ-23-1

Бондаренко Д.К.

(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)

Тип програми освітньо-професійна

(освітньо-професійна або освітньо-наукова)

Освітня програма Науки про дані (Data Science)

(повна назва спеціалізації)

Керівник доц. Шевченко О.Ю.

(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри

(підпис)

О.В. Золотухін

(прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук  
(повна назва)  
Кафедра Штучного інтелекту  
(повна назва)  
Рівень вищої освіти другий (магістерський)  
Спеціальність 122 Комп'ютерні науки  
(код і повна назва)  
Тип програми освітньо-професійна  
(освітньо-професійна або освітньо-наукова)  
Освітня програма Науки про дані (Data Science)  
(повна назва)

ЗАТВЕРДЖУЮ:  
Зав. кафедри \_\_\_\_\_  
(підпис)  
« \_\_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ р.

**ЗАВДАННЯ**  
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Бондаренку Денису Костянтиновичу  
(прізвище, ім'я, по батькові)

1. Тема роботи Метод синтезу навчальної вибірки для LLM на основі методу навчання з підкріпленням

затверджена наказом університету від 22 листопада 20 24 р. № 1238Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 січня 20 25 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел, інтегроване середовище розробки «JetBrains PyCharm», набір немічених текстових даних «openwebtext», набір запитів до браузеру «NaturalQuestions», Python-бібліотеки

4. Перелік питань, що потрібно опрацювати в роботі \_\_\_\_\_

1) Опис предметної галузі \_\_\_\_\_

2) Існуючий прогрес у галузі \_\_\_\_\_

3) Проектування програмного засобу \_\_\_\_\_

4) Реалізація програмного застосунку \_\_\_\_\_



## РЕФЕРАТ

Пояснювальна записка: 76 с., 26 рис., 1 дод., 37 джерел.

АЛГОРИТМИ ОЦІНЮВАННЯ ТЕКСТУ, ВЕЛИКІ МОВНІ МОДЕЛІ, МАШИННЕ НАВЧАННЯ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ НА ОСНОВІ ВІДГУКІВ ШТУЧНОГО ІНТЕЛЕКТУ, НЕМІЧЕНІ ТЕКСТОВІ ДАНІ, СИНТЕТИЧНІ ДАНІ.

Об'єкт дослідження – процес навчання великих мовних моделей (LLM) із використанням синтетичних даних.

Предмет дослідження – методика створення синтетичного датасету на основі навчання з підкріпленням і алгоритмів оцінювання якості відповідей моделі.

Мета роботи – розробити та впровадити ефективний підхід до створення синтетичного текстового датасету для навчання великих мовних моделей, використовуючи навчання з підкріпленням, оцінювання відповідей моделі та техніки перефразування для розширення датасету.

Методи дослідження – аналіз даних, експеримент, створення алгоритмів, валідація.

У результаті роботи було розроблено підхід до створення синтетичного датасету для навчання великих мовних моделей за допомогою навчання з підкріпленням. Модель попередньо навчалася на великому обсязі немічених текстових даних, а потім отримувала запити з попередньо визначеного набору. Для розширення датасету відповіді перефразувалися та перекладалися на кілька мов. Отриманий підхід дозволяє ефективно генерувати високоякісні дані для навчання великих мовних моделей, що сприяє покращенню їх продуктивності та здатності працювати з багатомовними даними.

## **ABSTRACT**

Master's thesis contains: 76 pp., 26 fig., 1 ann., 37 references.

LARGE LANGUAGE MODELS, MACHINE LEARNING, REINFORCEMENT LEARNING, REINFORCEMENT LEARNING WITH ARTIFICIAL INTELLIGENCE FEEDBACK, SYNTHETIC DATA, UNMARKED TEXTUAL DATA, TEXT RATING ALGORITHMS.

Object of research – the process of training large language models (LLMs) using synthetic data.

Subject of research – the methodology for creating a synthetic dataset based on reinforcement learning and algorithms for evaluating the quality of model responses.

Purpose of research – to develop and implement an efficient approach to creating a synthetic textual dataset for training large language models, utilizing reinforcement learning, model response evaluation, and paraphrasing techniques to expand the dataset.

Methods of research: data analysis, experimentation, algorithm development, and validation.

As a result of the work, an approach was developed for creating a synthetic dataset for training large language models using reinforcement learning. The model was initially pre-trained on a large volume of unlabeled text data and then received prompts from a predefined dataset. To expand the dataset, the responses were paraphrased and translated into multiple languages. The proposed approach enables the efficient generation of high-quality data for training large language models, enhancing their performance and ability to handle multilingual tasks.

## ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів .....	8
Вступ.....	9
1 Опис предметної галузі.....	11
1.1 Опис великих мовних моделей.....	11
1.1.1 Загальні поняття та принципи .....	11
1.1.2 Огляд деяких існуючих LLM.....	13
1.2 Опис навчання з підкріпленням .....	21
1.2.1 RL у загальному розумінні .....	21
1.2.2 Класифікація RL на основі відгуків .....	23
2 Існуючий прогрес у галузі.....	29
2.1 Набір даних Cleaned Alpaca Dataset .....	29
2.2 Спосіб створення синтетичного датасету Self-Instruct.....	30
2.3 Модель Vaize та спосіб створення синтетичного датасету Self-Chat	33
3 Проектування програмного засобу.....	37
3.1 Вибір та обґрунтування мови програмування та середовища розробки програмного застосунку .....	37
3.2 Визначення загального алгоритму .....	38
3.3 Аналіз вхідних даних та підготовка до попередньої обробки.....	39
3.4 Визначення метрик оцінювання відповідей моделі .....	40
3.5 Вибір та обґрунтування засобів штучного інтелекту для використання в проєкті .....	41
3.5.1 Опис моделі GPT-2 як центральної моделі в проєкті.....	41
3.5.2 Опис PPO як головного підходу RL у проєкті.....	43
3.5.3 Опис моделі toxic-bert для вимірювання метрики токсичності	44
3.5.4 Опис моделі gpt2-large-helpful-reward-model для вимірювання метрики корисності.....	45
3.5.5 Опис бібліотеки language_tool_python для вимірювання метрики граматичної коректності.....	47

3.5.6	Опис моделі t5_paraphraser для перефразування відповідей моделі .....	48
3.5.7	Загальний опис розробок NLP-підрозділу Гельсінського університету.....	50
3.6	Набір запитів до моделі Natural Questions.....	51
4	Реалізація програмного засобу .....	53
4.1	Файлова структура проекту .....	53
4.2	Реалізація попередньої обробки даних для попереднього тренування .....	55
4.3	Огляд файлу main.py та головних налаштувань у проєкті .....	57
4.4	Реалізація процесу токенізації та попереднього навчання моделі.....	58
4.5	Опис процесу RL.....	61
4.5.1	Реалізація RLAIIF.....	61
4.5.2	Реалізація циклу RL і створення файлу попереднього виводу .	64
4.6	Реалізація процесу збільшення вихідних даних .....	66
4.7	Приклад вихідних даних .....	69
	Висновки .....	71
	Перелік джерел посилання .....	72
	Додаток А Відомість кваліфікаційної роботи .....	76

## **ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

NLP – Natural Language Processing – обробка природньої мови;

NQ – Natural Questions – природні запитання;

PPO – Proximal Policy Optimization – оптимізація політики з проксимальним наближенням;

RL – Reinforcement Learning – навчання з підкріпленням.

## ВСТУП

У сучасному світі системи обробки природної мови (NLP) відіграють ключову роль у багатьох галузях, включаючи освіту, охорону здоров'я, комунікацію та бізнес [1]. Швидкий розвиток технологій обробки природної мови (NLP) останніми роками привів до появи надзвичайно потужних чат-моделей, таких як LaMDA, ChatGPT та GPT-4 [7]. Ці моделі демонструють вражаючу здатність розуміти та генерувати відповіді, що нагадують людські, у широкому спектрі галузей. У результаті чат-моделі набули значної популярності в таких сферах застосування, як підтримка клієнтів, віртуальні асистенти та модерація в соціальних мережах [7].

Попри великий потенціал цих моделей, вони часто доступні лише через обмежені API, що створює перешкоди для нових досліджень і прогресу. Крім того, обмежена доступність чат-моделей створює труднощі для дослідників і практиків, гальмуючи розвиток спільноти NLP. Нестача доступних для широкого користування високоякісних корпусів чатів для багатотурових розмов лише ускладнює цю проблему, обмежуючи можливості для вдосконалення та оцінювання таких моделей.

Розробка високоякісних моделей LLM потребує великих обсягів даних для навчання, що включає як загальні текстові дані, так і по-особливому розмічені набори даних. При цьому створення таких даних є дорогим, трудомістким та вимагає значних людських ресурсів.

Один зі способів розв'язання цієї проблеми – використання синтетичних даних, які можуть бути створені моделями LLM у процесі навчання з підкріпленням [6], [7]. У цьому контексті важливим завданням є створення методів, які дозволяють ефективно генерувати, оцінювати та адаптувати синтетичні дані, підвищуючи їхню якість та корисність для подальшого навчання моделей.

Ця робота пропонує підхід до створення синтетичного датасету для навчання великих мовних моделей шляхом комбінації попереднього

навчання на великому наборі немічених текстових даних і використання навчання з підкріпленням та інших прийомів, що дає змогу збільшувати обсяги тренувальних даних у багато разів. Основна ідея полягає у використанні моделі, попередньо натренованої на відкритих текстових даних (openwebtext [8]), для генерування відповідей на запити з набору NaturalQuestions [9]. Застосовуючи спеціальний алгоритм оцінки якості відповідей, ми відбираємо лише високоякісні результати, які потім проходять етап перефразування і перекладу на різні мови. Це дозволяє створити багатомовний синтетичний датасет, здатний суттєво покращити подальше навчання великих мовних моделей.

У роботі буде описано методологію створення такого синтетичного датасету, наведено результати оцінки якості даних та розглянуто потенціал використання цього підходу для підвищення ефективності моделей NLP. Також буде описаний існуючий прогрес у даній науковій галузі.

## 1 ОПИС ПРЕДМЕТНОЇ ГАЛУЗІ

### 1.1 Опис великих мовних моделей

#### 1.1.1 Загальні поняття та принципи

Велика мовна модель (LLM) – це мовна модель із великою кількістю параметрів, яка проходить завдання попереднього навчання (наприклад, масковане моделювання мови або авторегресивне прогнозування) для розуміння та обробки людської мови шляхом моделювання контекстуалізованих семантик тексту та ймовірностей на основі великих обсягів текстових даних [16]. Потужна велика мовна модель повинна мати чотири ключові характеристики [17]:

- глибоке розуміння контексту природної мови;
- здатність генерувати текст, подібний до людського;
- контекстуальну обізнаність, особливо у знання-інтенсивних галузях;
- високу здатність слідувати інструкціям, що є корисним для розв’язання завдань і прийняття рішень.

LLM є однією з найбільш передових розробок у сфері NLP. Вони є складними нейронними мережами з мільярдами параметрів, здатними аналізувати, розуміти та генерувати тексти, що відповідають людському стилю спілкування. Їх розробка базується на застосуванні сучасних архітектур, таких як трансформери [18], які дозволяють ефективно працювати з великими обсягами текстових даних.

Основним етапом підготовки таких моделей є попереднє навчання на масштабних корпусах тексту, що можуть включати дані з книг, статей, інтернет-форумів, енциклопедій та інших джерел [3]. У цьому процесі моделі опановують базові мовні закономірності, такі як граматики, структура речень, значення слів і виразів, а також закономірності, що

виникають у великих текстових масивах. Для цього використовуються спеціальні методи навчання, серед яких масковане моделювання мови [19] і авторегресивне прогнозування. Завдяки цьому моделі здатні аналізувати контекст та оцінювати ймовірності слів чи фраз, що дозволяє їм створювати зв'язний та зрозумілий текст.

Великі мовні моделі використовуються у багатьох галузях та завданнях. Вони знаходять застосування в таких галузях, як-от:

- автоматичний переклад;
- пошукові системи;
- віртуальні помічники;
- аналіз настроїв;
- генерація контенту;
- модерація соціальних мереж;
- вирішення спеціалізованих задач, пов'язаних із розумінням знаннєвих областей.

Завдяки своїй здатності враховувати контекст, такі моделі можуть адаптувати відповіді залежно від попереднього діалогу, що робить їх надзвичайно ефективними у багатотурових розмовах.

Однак, попри всі переваги, великі мовні моделі мають свої обмеження. Наприклад, процес їхнього навчання вимагає значних обчислювальних ресурсів, що робить такі технології недоступними для багатьох організацій [16]. Крім того, існують ризики упереджень у відповідях моделей, які можуть виникати через необ'єктивність або недостатню якість вихідних даних [16]. Питання етичного використання таких технологій також набирають актуальності, оскільки великі мовні моделі можуть застосовуватися для створення дезінформації або маніпулятивного контенту [20].

## 1.1.2 Огляд деяких існуючих LLM

### 1.1.2.1 Огляд моделі GPT-4 від OpenAI

GPT-4 (Generative Pre-trained Transformer 4) – це четверта генерація великих мовних моделей, розроблена OpenAI [21]. Вона стала черговим важливим етапом у розвитку технологій обробки природної мови, демонструючи високу здатність до розуміння контексту, генерації тексту та адаптації до складних завдань. Завдяки своїй універсальності, GPT-4 знайшла широке застосування у різноманітних сферах, включаючи бізнес, освіту, дослідження та креативне письмо.

GPT-4 здатна працювати з величезними обсягами текстових даних. У порівнянні з попередніми версіями, ця модель має ще більше параметрів, що дозволяє їй глибше аналізувати зв'язки між словами, враховувати контекст і генерувати текст із високою точністю. Модель була натренована на різноманітних даних, що охоплюють широкий спектр тем і стилів. Це забезпечило їй здатність адаптуватися до різних завдань: від технічного письма до створення художніх текстів. На рисунку 1.1 можемо побачити, що ця модель значно випереджає свого попередника.

Унікальною особливістю GPT-4 є її контекстуальна обізнаність [22]. Модель може враховувати значно більше контексту, ніж попередні версії, що дозволяє їй обробляти довгі діалоги або складні документи. Наприклад, у багатотурових розмовах GPT-4 здатна зберігати інформацію з попередніх повідомлень і враховувати її під час генерування нових відповідей. Це робить її корисною для створення розумних чат-ботів, які можуть вести природні та логічно послідовні діалоги.

GPT-4 здатна до адаптивного навчання [21]. Модель може бути донавчена на спеціалізованих наборах даних, що дозволяє налаштовувати її для вирішення конкретних завдань. Наприклад, GPT-4 може бути адаптована для створення технічної документації, автоматизації перекладів

або аналізу даних у вузькопрофільних галузях. Це робить її універсальним інструментом, який може бути використаний як у дослідженнях, так і в комерційних проєктах.

### Exam results (ordered by GPT-3.5 performance)

Estimated percentile lower bound (among test takers)

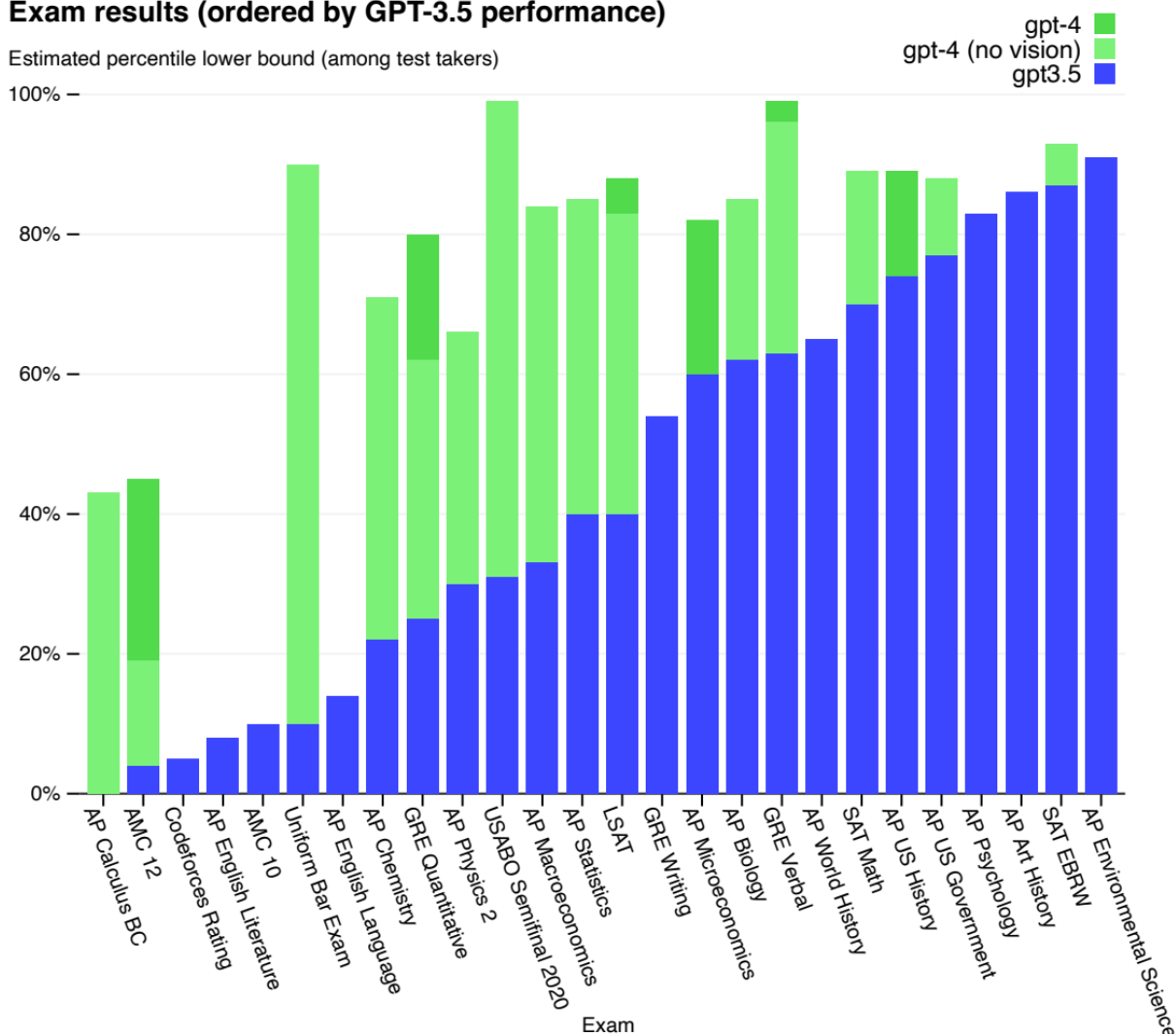


Рисунок 1.1 – Порівняння успішності складання академічних та професійних екзаменів моделей GPT-3.5 та GPT-4 [28]

GPT-4 здатна працювати з багатьма мовами [21]. Завдяки цьому модель може застосовуватися у глобальному масштабі, забезпечуючи доступність інформації для користувачів із різних країн. Вона показує високу якість перекладу текстів, а також здатна зберігати стиль і тон

оригінального повідомлення, що особливо важливо для задач, пов'язаних із міжкультурною комунікацією.

Дана модель також здатна вирішувати знаннєінтенсивні завдання [21]. Завдяки обробці великих обсягів текстових даних модель здатна відповідати на складні запитання, розуміти складну термінологію та аналізувати контекст у вузькопрофільних темах, таких як медицина, право чи наука.

Незважаючи на це, ця модель може іноді генерувати помилкову інформацію або відповіді, які здаються переконливими, але є неточними [22]. Це вимагає ретельного моніторингу та вдосконалення, особливо якщо GPT-4 використовується у критично важливих галузях, таких як медицина чи юриспруденція.

З точки зору етики, GPT-4 ставить питання про відповідальне використання технологій [20]. Наприклад, можливість створювати реалістичний текст може бути використана як для продуктивних цілей, так і для шкідливих, таких як поширення дезінформації. Щоб мінімізувати ці ризики, OpenAI розробляє політики відповідального використання та впроваджує механізми захисту [21].

Таким чином, GPT-4 є корисним інструментом для вирішення широкого спектра завдань у галузі обробки природної мови. Завдяки своїй універсальності, адаптивності та здатності розуміти контекст, вона є частиною нового покоління в історії технологій штучного інтелекту.

#### 1.1.2.2 Огляд моделі LaMDA від Google

LaMDA (Language Model for Dialogue Applications) – це LLM, розроблена компанією Google для обробки природної мови, зокрема для ведення багатотурових діалогів [23]. Її ключова мета – створення моделей, які можуть вести діалоги на природному рівні, демонструючи глибоке розуміння контексту, тональності та сенсу запитань і відповідей. Ця модель

стала значним проривом у галузі обробки природної мови, орієнтуючись не лише на точність, але й на природність взаємодії з користувачем.

LaMDA була створена з акцентом на діалогову природу комунікації [23]. На відміну від традиційних моделей, які часто генерують узагальнені або не зовсім релевантні відповіді, LaMDA здатна враховувати специфіку теми розмови, а також розвивати діалог у бажаному напрямку. Це стало можливим завдяки використанню спеціальних алгоритмів навчання, що дозволяють моделі аналізувати величезні масиви текстових даних, зокрема багатотурові діалоги, на яких вона була натренована. На рисунку 1.2 розглянемо приклад, що ілюструє діалогову природу моделі.

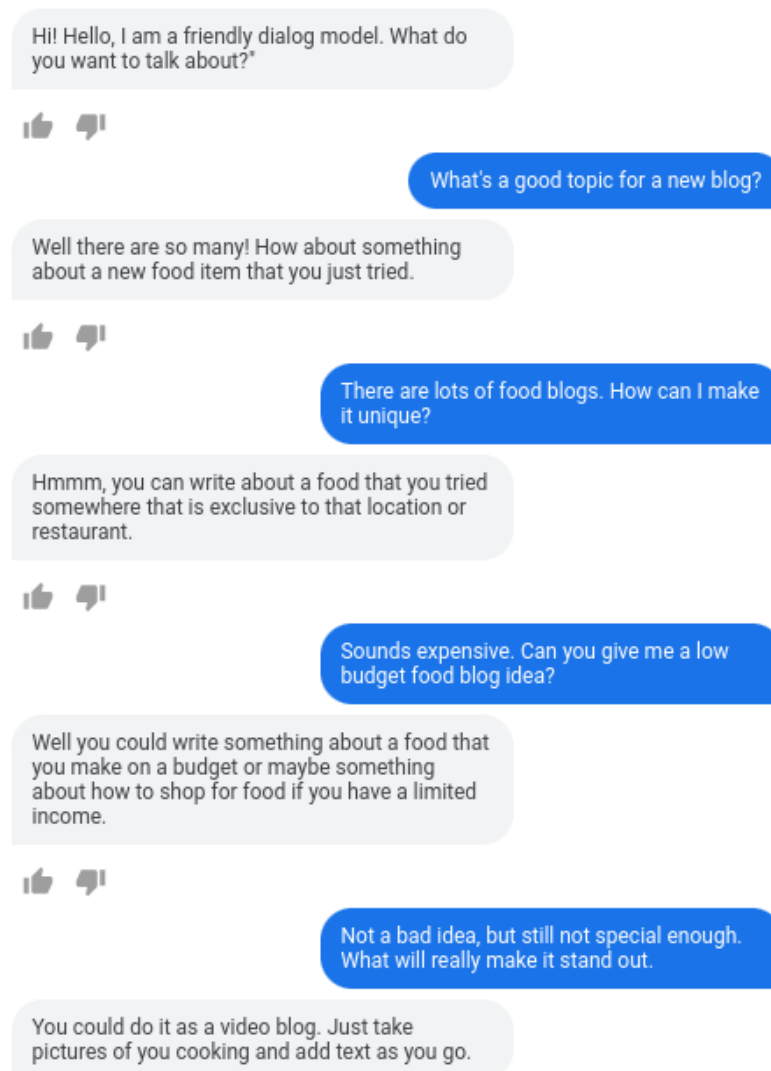


Рисунок 1.2 – Приклад діалогу із LaMDA [23]

Однією з основних особливостей LaMDA є її здатність зберігати когерентність і логічність протягом тривалого діалогу [23]. Модель розроблена так, щоб враховувати попередні репліки користувача та будувати відповіді, які гармонійно вписуються у контекст розмови. Наприклад, якщо розмова стосується подорожей, LaMDA може продовжувати тему, розповідаючи про конкретні напрямки, даючи рекомендації або навіть змінюючи тон розмови залежно від настрою користувача.

LaMDA здатна генерувати відповіді, які звучать природно і різноманітно. На відміну від стандартних алгоритмів, які часто повторюють однакові фрази або створюють шаблонні відповіді, LaMDA здатна враховувати тон розмови та адаптувати свої відповіді відповідно до потреб користувача. Це дозволяє їй створювати більш персоналізований і дружній досвід взаємодії.

Ця модель є гнучкою у виборі тем для розмови [23]. Завдяки тому, що модель була навчена на широкому спектрі текстових даних, вона може обговорювати різноманітні теми – від наукових фактів до креативних тем, таких як вигадані світи або індивідуальні захоплення користувача. Важливо, що LaMDA прагне не лише відповідати на запити, але й ініціювати нові напрямки в розмові.

Модель також показує значний прогрес у вирішенні проблеми узгодження фактів, що є важливою задачею для діалогових систем [23]. Google прагнув мінімізувати поширення дезінформації, тому LaMDA була налаштована на генерацію відповідей, які базуються на достовірних джерелах. Хоча ця проблема не вирішена повністю, модель видає набагато кращі результати, ніж попередні системи.

LaMDA знаходить широке застосування у створенні віртуальних асистентів, платформ для підтримки клієнтів та освітніх інструментів [23]. Її здатність вести природні діалоги дозволяє використовувати її в сценаріях, які раніше були недоступними для традиційних моделей. Наприклад, вона

може бути використана для персоналізованого навчання, коли учень може ставити запитання та отримувати зрозумілі пояснення у зручному для нього форматі.

Однак, як і будь-яка сучасна мовна модель, LaMDA стикається з низкою проблем. Однією з них є її навчання на текстах, які можуть містити упередження або неточності. Google активно працює над розв'язанням цієї проблеми, впроваджуючи етичні принципи у розробку та використання моделі. Це включає аналіз результатів роботи моделі, зворотний зв'язок із користувачами та покращення механізмів фільтрації текстів, які можуть містити небажану інформацію.

Технологія LaMDA є важливим кроком у бік розвитку діалогових систем і обробки природної мови загалом. Її здатність до глибокого розуміння контексту, адаптація до різних завдань та інтуїтивно зрозумілі відповіді роблять її одним із найперспективніших інструментів у сфері NLP. Хоча модель ще перебуває на етапі вдосконалення, її потенціал вже зараз змінює уявлення про можливості діалогових систем і їхній вплив на взаємодію між людьми та технологіями.

### 1.1.2.3 Огляд моделі Claude від Anthropic

Claude [24] – це сучасна велика мовна модель (LLM), розроблена компанією Anthropic, яка спеціалізується на створенні штучного інтелекту з акцентом на безпеці, етиці та відповідальному використанні. Модель названа на честь Клода Шеннона, одним із перших дослідників теорії інформації, що підкреслює її зв'язок із глибокими концепціями обробки даних і природної мови [25].

Claude орієнтована на етичність [24]. На відміну від багатьох інших великих мовних моделей, Claude розроблялася з акцентом на мінімізацію ризиків, пов'язаних із генерацією небажаного контенту. Anthropic застосовує унікальний підхід, який включає в себе концепцію

«конституційного ШІ». Це означає, що Claude навчається слідувати наборам етичних принципів, що допомагає уникати упереджень, токсичності та інших небажаних аспектів у її роботі. Цей підхід також дозволяє моделі краще справлятися із запитам, які потребують складних рішень або моральних оцінок [25]. На рисунку 1.3 можемо побачити, як Anthropic описують процес навчання «конституційного ШІ».

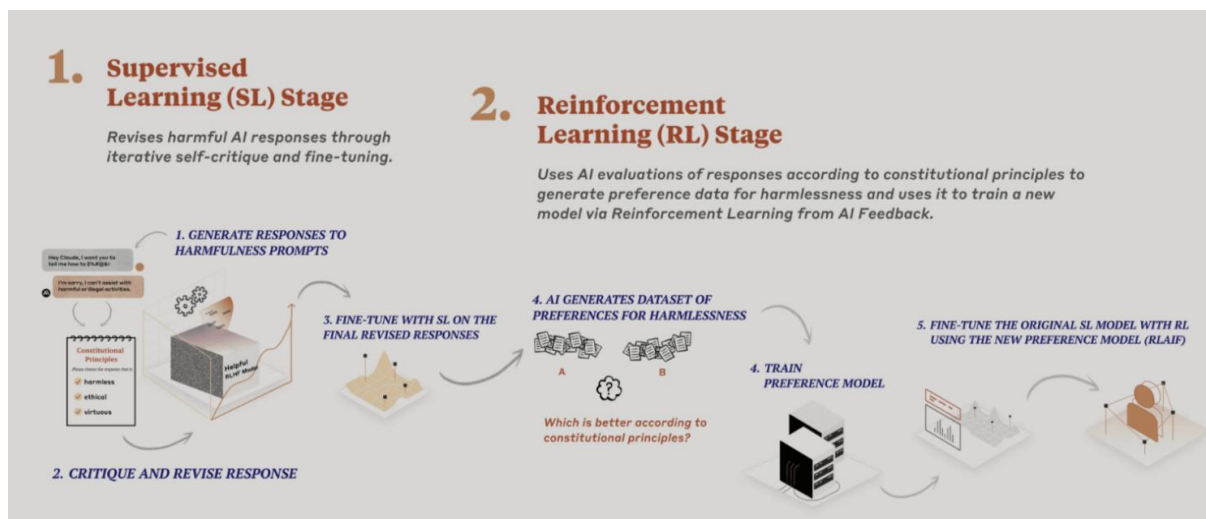


Рисунок 1.3 – Процес навчання «конституційного ШІ» [29]

Claude має високий рівень контекстуальної обізнаності [24], [25], що дозволяє їй ефективно взаємодіяти з користувачами в багатотурових діалогах. Модель розроблена так, щоб враховувати як зміст, так і тон розмови, пропонуючи відповіді, які є логічно послідовними й відповідають очікуванням користувача. Це особливо важливо у сценаріях, де потрібна персоналізація, наприклад, у підтримці клієнтів або освітніх додатках. Claude здатна динамічно адаптувати свої відповіді залежно від контексту, зберігаючи природність і релевантність розмови.

Окремо варто зазначити здатність Claude працювати з різноманітними типами даних і завдань [24]. Модель може бути використана для генерації тексту, аналізу документів, відповідей на запитання, створення навчальних матеріалів і навіть програмування. Завдяки своєму тренуванню на великій

кількості текстових даних, Claude демонструє високу точність у виконанні задач, які вимагають глибокого розуміння контексту й деталей.

Claude також відзначається своєю здатністю слідувати інструкціям [25]. Ця особливість робить модель особливо корисною для виконання завдань, пов'язаних із прийняттям рішень і розв'язанням проблем. Наприклад, вона може допомагати у складанні планів, наданні рекомендацій або розробці стратегій. Водночас Claude залишається гнучкою, що дозволяє використовувати її для виконання творчих завдань, таких як написання текстів, створення сценаріїв чи розробка ідей.

Як зазначено вище, Claude має посилену здатність уникнення шкідливого контенту. Завдяки етичним принципам, вбудованим у її архітектуру [25], модель показує здатність фільтрувати токсичний або упереджений текст, що є критично важливим для забезпечення безпечного використання в суспільстві. Anthropic активно працює над вдосконаленням цих аспектів, щоб зробити Claude доступною і безпечною для широкого кола користувачів.

Claude знаходить застосування в багатьох галузях, включаючи бізнес, освіту, науку й медицину. У бізнесі модель використовується для автоматизації взаємодії з клієнтами, створення текстів і проведення аналізу даних. В освітніх проєктах Claude сприяє створенню персоналізованого навчального досвіду, відповідаючи на запитання учнів і пояснюючи складні теми. У наукових дослідженнях модель допомагає аналізувати великі обсяги текстових даних, пропонуючи релевантну інформацію та висновки.

Попри свої численні переваги, Claude також стикається з певними викликами. Наприклад, вона залежить від якості даних, на яких була навчена. Це означає, що упередженість або неточності в навчальних даних можуть вплинути на результати роботи моделі [25]. Крім того, використання Claude вимагає відповідального підходу, оскільки навіть найбільш досконалі системи штучного інтелекту не можуть повністю уникнути помилок.

У підсумку, Claude є однією з передових мовних моделей, яка поєднує в собі передові технічні досягнення та етичні принципи. Вона є показником значного прогресу в забезпеченні безпечного й ефективного використання штучного інтелекту, відкриваючи нові можливості для взаємодії між людьми та технологіями. Завдяки своїй універсальності, Claude має потенціал стати важливим інструментом у розв'язанні складних завдань у різних сферах, забезпечуючи нові способи покращення якості життя та підвищення ефективності роботи.

## 1.2 Опис навчання з підкріпленням

### 1.2.1 RL у загальному розумінні

Навчання з підкріпленням (Reinforcement Learning, RL) – це підхід у машинному навчанні, який імітує спосіб, у який живі організми навчаються через взаємодію з навколишнім середовищем. Його суть полягає у виборі послідовності дій агентом, спрямованих на досягнення певної мети, при цьому агент отримує зворотний зв'язок у вигляді винагороди або покарання залежно від результату своїх дій [26]. Цей зворотний зв'язок служить інструментом для вдосконалення стратегій поведінки агента з метою максимізації довгострокової вигоди.

У процесі RL агент функціонує у середовищі, яке представляє собою систему із певними станами [26]. На кожному кроці агент приймає рішення про дію, ґрунтуючись на поточному стані середовища. У відповідь на дію середовище переходить у новий стан, і агент отримує винагороду. Таким чином, основна мета агента полягає у виборі дій, які максимізують сумарну винагороду за певний період часу.

Ключовою складовою RL є поняття стратегії (або політики, policy), яка визначає, як агент обирає дії залежно від стану середовища [26]. Стратегія може бути детермінованою, коли кожному стану відповідає

конкретна дія, або стохастичною, коли вибір дії залежить від ймовірностей. У процесі навчання агент коригує свою стратегію, використовуючи отриману винагороду, щоб зробити її більш ефективною.

Важливою концепцією RL є функція цінності (value function), яка оцінює, наскільки корисним є певний стан або дія з точки зору майбутньої винагороди [26]. Функція цінності допомагає агенту визначати, які дії варто виконувати, щоб досягти максимальної довгострокової вигоди. Існують різні підходи до оцінки функції цінності, зокрема методи Q-навчання та SARSA, які дозволяють агенту ефективно знаходити оптимальні стратегії.

Динаміка навчання в RL значною мірою залежить від взаємодії між дослідженням (exploration) та використанням (exploitation) [26]. Дослідження передбачає вибір агентом нових або неочікуваних дій для отримання додаткової інформації про середовище, тоді як використання зосереджене на виконанні дій, які вже показали себе як ефективні. Збалансованість між цими двома підходами є важливою для забезпечення ефективного навчання, оскільки надмірне дослідження може сповільнити процес досягнення оптимальних рішень, а надмірне використання – обмежити можливості вдосконалення стратегії.

У RL існує проблема кредитного призначення, яка стосується визначення, які саме дії агента призвели до отримання певної винагороди [26]. Ця проблема особливо актуальна у складних середовищах, де ефекти від дій можуть проявлятися лише через певний час. Для її розв'язання використовуються різні алгоритми, наприклад, методи часових різниць (temporal-difference methods), які дозволяють агенту оновлювати свої оцінки цінності на основі поточного досвіду.

Навчання з підкріпленням також тісно пов'язане з поняттям середовища, що моделюється як процес Маркова (Markov Decision Process, MDP) [27]. У рамках MDP кожен стан визначається поточним положенням у середовищі, а майбутній стан залежить лише від поточного стану та дії агента. Ця властивість дозволяє значно спростити побудову моделей та

алгоритмів RL, забезпечуючи математично строгий підхід до оптимізації стратегій.

Ще одним важливим моментом RL є використання функцій наближення для представлення функцій цінності або стратегій у великих або безперервних просторах станів. Такі методи, як нейронні мережі, дозволяють створювати гнучкі моделі, які можуть ефективно працювати у складних середовищах. Відомі підходи, такі як Deep Q-Learning, поєднують глибоке навчання з RL для обробки великих обсягів даних та визначення оптимальних стратегій.

Навчання з підкріпленням знаходить застосування у багатьох сферах, таких як робототехніка, ігри, управління ресурсами та оптимізація складних систем. Завдяки своїй здатності до адаптації та самовдосконалення RL відкриває широкі перспективи для створення систем, які можуть ефективно взаємодіяти з навколишнім середовищем і приймати рішення в умовах невизначеності.

## 1.2.2 Класифікація RL на основі відгуків

### 1.2.2.1 RL на основі відгуків людини

Навчання з підкріпленням із використанням людського зворотного зв'язку (Reinforcement Learning with Human Feedback, RLHF) – це метод навчання моделей машинного навчання, який поєднує алгоритми навчання з підкріпленням із людськими оцінками для покращення якості й ефективності моделей [2]. Цей підхід дозволяє створювати системи, які краще відповідають людським очікуванням, надаючи більш релевантні, зрозумілі й корисні відповіді.

Метою RLHF є використання людської експертизи для спрямування процесу навчання моделі, замість повної залежності від заздалегідь визначених автоматизованих цільових функцій. Люди оцінюють

результати, згенеровані моделлю, на основі критеріїв якості, релевантності чи зрозумілості. Ці оцінки перетворюються на сигнал винагороди, який модель використовує для вдосконалення своєї стратегії генерації відповідей [2].

Процедура RLHF зазвичай включає кілька етапів, які поєднують попереднє навчання моделі, залучення людських оцінок і навчання з підкріпленням:

- спершу модель проходить стандартне попереднє навчання на великому наборі текстових даних. Це може включати методи маскованого моделювання мови чи авторегресивного прогнозування, що дозволяє моделі навчитися розуміти зв'язки між словами, аналізувати контекст і генерувати текст;

- попередньо натренована модель генерує відповіді на запитання чи завдання, що є предметом оцінювання. Ці відповіді не завжди є оптимальними, але слугують початковою точкою для оцінювання та покращення;

- згенеровані моделлю відповіді оцінюються людьми. Оцінювачі розставляють результати за пріоритетом, визначають, які відповіді є більш релевантними, точними чи корисними, або оцінюють їх за шкалою якості. Наприклад, для одного запиту може бути запропоновано кілька варіантів відповіді, які оцінювач впорядковує за якістю;

- людські оцінки використовуються для тренування окремої моделі винагороди. Ця модель винагороди навчається передбачати, наскільки ймовірно, що певна відповідь задовольнить користувача, ґрунтуючись на даних оцінювачів;

- на основі моделі винагороди модель проходить етап навчання з підкріпленням. Алгоритми, такі як Proximal Policy Optimization (PPO), використовуються для оптимізації стратегії моделі з метою максимізації передбаченої винагороди. Це дозволяє моделі вдосконалювати свої відповіді, орієнтуючись на людські уподобання;

– після навчання модель тестується на нових запитах, щоб оцінити, чи стали її відповіді більш релевантними та корисними. У разі необхідності процес може бути повторений із залученням додаткових людських оцінок і подальшим доопрацюванням.

RLHF широко використовується у розробці сучасних великих мовних моделей, таких як GPT-4. Цей підхід дозволяє значно покращити якість відповіді на запити, створюючи моделі, які краще розуміють наміри користувачів і відповідають їхнім очікуванням.

#### 1.2.2.2 RL на основі відгуків штучного інтелекту

RLAIF (навчання з підкріпленням із використанням зворотного зв'язку від штучного інтелекту) – це методологія навчання моделей, яка ґрунтується на використанні автоматизованого зворотного зв'язку від допоміжних моделей чи алгоритмів замість або на додаток до людської оцінки [2]. Цей підхід пропонує ефективний спосіб масштабування процесу навчання моделей, зменшуючи залежність від дорогоцінного й трудомісткого людського зворотного зв'язку.

Основна концепція RLAIF полягає у використанні іншої штучної інтелектуальної системи для оцінювання якості вихідних даних або дій основної моделі. Ця допоміжна система (або модель) виконує роль «судді», генеруючи сигнали винагороди, які використовуються основною моделлю для вдосконалення. У цьому контексті RLAIF дозволяє значно прискорити процес навчання, знижуючи навантаження на людських оцінювачів і забезпечуючи консистентність у зворотному зв'язку [2].

Процес RLAIF можна розділити на декілька етапів [2]:

– як і в RLHF, процес починається з попереднього навчання основної моделі на великому текстовому корпусі, щоб забезпечити базові знання та здатність до генерації тексту;

– допоміжна модель або алгоритм винагороди створюється для оцінювання вихідних даних основної моделі. Допоміжна модель може бути окремою нейронною мережею або спеціалізованим алгоритмом, який був налаштований на певні критерії оцінки, такі як релевантність, стилістична відповідність, точність або інші параметри;

– основна модель генерує відповіді або виконує дії на основі заданих запитів. Ці результати передаються до допоміжної моделі для оцінювання;

– допоміжна модель аналізує результати основної моделі та надає оцінки або сигнали винагороди, які вказують на якість або ефективність вихідних даних. Ці сигнали винагороди можуть бути числовими значеннями або іншими формами зворотного зв'язку, що спрямовують процес навчання;

– основна модель використовує сигнали винагороди, отримані від допоміжної системи, для оновлення своєї стратегії генерації. Процес часто реалізується за допомогою алгоритмів, таких як Proximal Policy Optimization (PPO), які дозволяють ефективно оптимізувати модель;

– цикл генерації, оцінювання та оптимізації повторюється до досягнення бажаних характеристик роботи моделі. За потреби допоміжна модель також може бути донавчена для підвищення точності її оцінок.

RLAIF знаходить своє застосування в різних галузях, включаючи генерацію тексту, оптимізацію рекомендаційних систем, створення персоналізованих чат-ботів і навіть у системах керування складними процесами. Цей підхід особливо корисний у ситуаціях, де потрібно швидко навчити модель на основі великих обсягів даних або виконувати специфічні завдання, для яких людський зворотний зв'язок є дорогим або важкодоступним.

RLAIF – це перспективний напрям у розвитку машинного навчання, який дозволяє комбінувати переваги навчання з підкріпленням і автоматизованого зворотного зв'язку. Ця методологія допомагає створювати більш ефективні та адаптивні системи, зберігаючи при цьому значну економію ресурсів.

### 1.2.2.3 Порівняння RL на основі відгуків людини та RL на основі відгуків штучного інтелекту

Навчання з підкріпленням із людським зворотним зв'язком (RLHF) та навчання з підкріпленням із використанням зворотного зв'язку від штучного інтелекту (RLAIF) є двома розповсюдженими методологіями для вдосконалення мовних моделей і систем штучного інтелекту [2]. Обидва підходи використовуються для забезпечення кращої адаптації моделей до вимог завдань і підвищення якості їхніх результатів. Однак кожен із цих методів має свої унікальні переваги й обмеження, які визначають їхню ефективність у різних умовах.

RLHF базується на використанні людських оцінок для створення сигналів винагороди. Цей підхід дозволяє моделі вчитися відповідати людським очікуванням, оскільки зворотний зв'язок відображає суб'єктивне сприйняття якості, релевантності чи корисності відповіді. Основна перевага RLHF полягає у високій точності й надійності оцінювання, оскільки люди здатні враховувати складні й неформалізовані критерії [2]. Наприклад, людські оцінки можуть охоплювати тональність тексту, логічність і навіть етичність відповіді. Завдяки цьому RLHF ідеально підходить для завдань, де важлива персоналізація або високий рівень людського залучення, наприклад у створенні чат-ботів чи рекомендаційних систем.

Однак RLHF має значні обмеження. Залучення людей до оцінювання є дорогим, ресурсомістким і повільним процесом, особливо якщо йдеться про масштабні завдання. Крім того, людські оцінки можуть бути суб'єктивними, що створює ризик упередженості або неузгодженості. Це стає особливо критичним у завданнях, де складно отримати консенсус між різними оцінювачами. Нарешті, RLHF важко масштабувати, оскільки потреба в людському зворотному зв'язку зростає разом із кількістю завдань.

RLAIF, у свою чергу, використовує автоматизовані системи або допоміжні моделі для надання зворотного зв'язку. Цей підхід є значно

ефективнішим із точки зору часу й ресурсів, оскільки автоматизовані системи можуть оцінювати результати набагато швидше, ніж люди [2]. RLAIIF забезпечує постійність у зворотному зв'язку, оскільки допоміжна модель не піддається суб'єктивним впливам і завжди дотримується визначених правил. Це робить метод особливо привабливим для масштабних проєктів, де необхідно оцінити мільйони результатів. Наприклад, RLAIIF добре підходить для завдань оптимізації, таких як налаштування рекомендаційних алгоритмів або автоматизація в складних системах.

Основний недолік RLAIIF полягає у залежності від якості допоміжної моделі. Якщо модель винагороди неточно відображає реальні критерії оцінки, це може призвести до небажаних результатів, коли основна модель оптимізується під неправильні або недостатньо адекватні цілі. Допоміжна модель також може бути схильна до власних упереджень, особливо якщо її було навчено на неповних або упереджених даних. Іншою проблемою є те, що автоматизований зворотний зв'язок може не враховувати тонкі аспекти, які є важливими для користувачів, наприклад, емоційний тон або культурні нюанси.

Порівнюючи ці два підходи, можна сказати, що RLHF надає вищу якість і точність оцінювання для завдань, де важливе врахування людських очікувань і суб'єктивних критеріїв. Водночас цей підхід важко масштабувати через високу вартість і трудомісткість процесу. RLAIIF, у свою чергу, є більш ефективним і масштабованим, але його результати залежать від точності допоміжної моделі, що може створювати обмеження у завданнях із високою суб'єктивністю.

## 2 ІСНУЮЧИЙ ПРОГРЕС У ГАЛУЗІ

### 2.1 Набір даних Cleaned Alpaca Dataset

Cleaned Alpaca Dataset [4] – це оновлена і вдосконалена версія набору даних, який використовувався для навчання моделі Alpaca, базованої на LLaMA. Головною метою створення цього датасету є підвищення якості навчання моделей великих мовних моделей (LLM) шляхом усунення недоліків оригінального набору даних. Оновлений набір був розроблений з урахуванням проблем, пов'язаних із низькою якістю вихідних даних, що значно впливали на продуктивність моделі.

Датасет складається з понад 52 тисяч записів, які були очищені, перевірені та, за необхідності, переформульовані. Оригінальний набір даних, створений за допомогою GPT-3, містив численні проблеми, включаючи помилкові відповіді, відсутність виводів, несенсові інструкції та некоректну структуру записів. Наприклад, значна частина математичних задач мала неправильні розв'язки, а деякі запити були складені так, що модель не могла їх коректно обробити. Такі проблеми спричиняли низьку ефективність навчання і формували неякісні виходи.

Оновлений набір був розширений даними, отриманими з GPT-4, що дозволило покращити його якість і зменшити ризик помилок. Проблеми, такі як непослідовне форматування, порожні виводи, некоректні інструкції та «галюцинації» моделі, були адресовані. Наприклад, вхідні дані тепер узгоджені між собою, що покращує зрозумілість інструкцій для моделі. Також було зменшено кількість «галюцинацій» – ситуацій, коли модель генерує інформацію, яка не відповідає дійсності.

Що стосується якості, очищений датасет показав значно кращі результати у порівнянні з оригінальним. Наприклад, моделі, навчені на очищеному наборі, демонструють покращену точність у задачах класифікації та зниження кількості помилкових виводів. Використовуючи

тестові системи, такі як HALTT4LLM, вдалося підтвердити, що очищений датасет значно зменшує схильність моделі до галюцинацій, водночас забезпечуючи кращу узгодженість результатів.

Важливим аспектом очищеного датасету є його курація з точки зору довжини підказок (prompts). В середньому, довжина підказок в очищеному наборі значно збільшилася, що дозволяє моделі ефективніше працювати з довгими контекстами. Для оптимального використання цього набору при донавчанні моделей рекомендується збільшити максимальну довжину підказки до 512 токенів або більше.

Cleaned Alpaca Dataset став корисним ресурсом для дослідників і розробників, які працюють із LLM. Очищення та стандартизація набору даних сприяють покращенню продуктивності моделей, зменшуючи вплив шуму в даних на результати. Зокрема, оновлені моделі, такі як Alpaca-7B-Lora і Alpaca-13B-Lora, демонструють кращі результати на задачах розуміння тексту й генерації відповідей.

Основна мета цього проєкту – забезпечити якісний набір даних для навчання моделей, які працюють з текстами природною мовою. Очищений Alpaca Dataset є ще одним кроком до підвищення надійності та ефективності сучасних мовних моделей, дозволяючи їм краще адаптуватися до реальних завдань і потреб користувачів. Це підтверджує, що якість даних є вирішальним фактором для продуктивності моделей, іноді навіть важливішим, ніж збільшення кількості параметрів моделі.

## 2.2 Спосіб створення синтетичного датасету Self-Instruct

Self-Instruct [6] – це напівавтоматизований підхід до налаштування попередньо натренованих мовних моделей на основі інструкцій, згенерованих самою моделлю. Цей метод дозволяє моделі вдосконалювати свої навички слідування інструкціям, використовуючи ітеративний процес самоудосконалення. Основою процесу є поступове розширення обсягу

завдань, починаючи з невеликого початкового набору вручну написаних інструкцій, який потім використовується для автоматичного генерування нових завдань. На рисунку 2.1 високорівнево розглянемо хід виконання Self-Instruct.

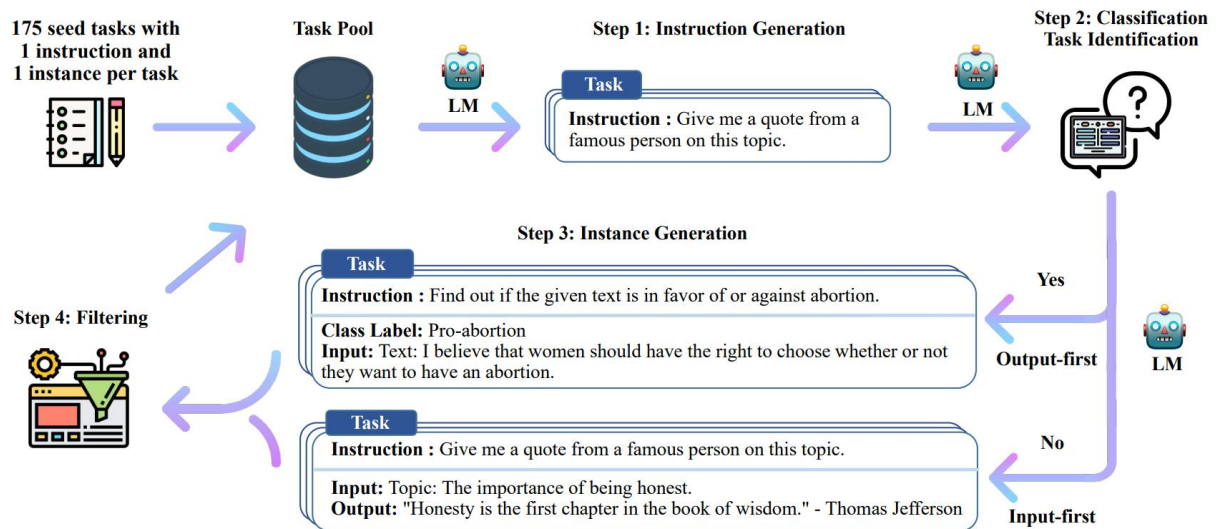


Рисунок 2.1 – Поверхневий огляд Self-Instruct

Процес Self-Instruct починається з набору з 175 початкових завдань, створених вручну. Ці завдання служать основою для генерації нових інструкцій. Спочатку модель отримує запит на створення інструкцій для нових завдань, з використанням наявного набору як прикладів. Згенеровані інструкції розширюють спектр завдань і створюють різноманітні сценарії, які відходять від типових задач обробки природної мови. Після цього для кожної нової інструкції модель генерує приклади вхідних даних та відповідних виходів, які згодом можуть бути використані для навчання моделі на основі цих інструкцій.

Ключовим етапом у Self-Instruct є очищення та фільтрація даних. Згенеровані інструкції проходять перевірку за допомогою евристичних правил для видалення низькоякісних або дубльованих записів. Наприклад, нова інструкція додається до пулу завдань лише в тому випадку, якщо її схожість із наявними інструкціями за метрикою ROUGE-L є меншою за

визначений поріг. Інструкції, які містять специфічні ключові слова, пов'язані з нездійсненими завданнями, такими як генерація зображень, також виключаються.

Для класифікаційних і некласифікаційних завдань використовуються різні підходи до генерації прикладів. У некласифікаційних завданнях модель спершу генерує вхідні дані на основі інструкції, а потім створює відповідь. Для класифікаційних завдань використовується зворотний підхід: спочатку генеруються можливі категорії (виходи), а потім – вхідні дані, відповідні кожній категорії. Такий підхід знижує ризик упередженості у генерації, особливо для задач, де модель може віддавати перевагу одній із категорій.

Після завершення процесу генерації інструкцій та прикладів модель навчається за допомогою нового датасету. Для цього інструкція та вхідні дані об'єднуються у підказку, на яку модель повинна створити відповідь. Цей процес передбачає використання різних шаблонів підказок, щоб зробити модель стійкішою до різних форматів введення. Наприклад, у підказках можна використовувати варіації таких маркерів, як «Task:», «Input:», «Output:», а також змінювати кількість розривів між блоками тексту.

Self-Instruct було застосовано для створення великого набору даних із понад 52 тисяч інструкцій і близько 82 тисяч прикладів вхідних даних та відповідей. Цей набір включає широкий спектр завдань, які демонструють значну різноманітність і виходять за межі стандартних задач обробки природної мови. Фінальну модель, GPT3SELF-INST, було оцінено у порівнянні з іншими моделями на наборах типових задач (наприклад, SUPERNI) і нових інструкцій. Результати показали значне покращення точності у порівнянні з базовою моделлю GPT-3 та близькі до рівня InstructGPT.

Цей метод особливо цінний у контексті задач, пов'язаних із створенням великомасштабних датасетів для інструктивного навчання.

Генерація завдань автоматичною системою дозволяє вирішити дві основні проблеми: необхідність креативності для створення нових задач і потребу в експертизі для написання відповідей. Хоча згенеровані дані можуть містити певний рівень шуму, більшість прикладів мають коректний формат і забезпечують достатню якість для вдосконалення моделі. Self-Instruct є важливим кроком у напрямку масштабованого й ефективного створення синтетичних даних для навчання мовних моделей.

### 2.3 Модель Baize та спосіб створення синтетичного датасету Self-Chat

Метод, представлений у дослідженні [7], зосереджується на створенні високоякісного корпусу багатотурових діалогів за допомогою ChatGPT. Цей процес, відомий як «self-chat», дозволяє моделі ChatGPT самостійно генерувати як запити від користувача, так і відповіді від AI-асистента. Основна мета цього підходу – покращити якість і ефективність моделі Baize, яка базується на LLaMA, для роботи з багатотуровими діалогами.

Процес self-chat реалізується через використання шаблонів, що задають формат і вимоги до діалогу. Для запуску діалогу обирається «насіння» – це може бути питання чи ключова фраза, яка визначає тему розмови. У навчанні Baize використовувалися запитання з платформ Quora та Stack Overflow, що забезпечило різноманітність і широту охоплення завдань. Загалом у першій версії Baize було створено 111,5 тисяч діалогів, витративши близько 100 доларів на виклики API ChatGPT. Крім того, для спеціалізації моделі на медичних запитах використовували дані з набору MedQuAD, додавши ще 47 тисяч діалогів у медичній галузі.

Для підвищення якості даних було застосовано двоетапний підхід. Спершу ChatGPT генерував відповіді для обох сторін діалогу одразу, що знижувало деталізацію виходів. На другому етапі відповідь AI замінювалася результатом окремого запиту до ChatGPT для отримання більш

деталізованих відповідей. Цей підхід забезпечив якісніший набір даних для подальшого навчання моделі Baize.

Метод self-chat має переваги у порівнянні з іншими підходами збору даних. Наприклад, Stanford Alpaca використовує формат інструкцій «instruction-input-output», що обмежує завдання одним туром взаємодії. У свою чергу, self-chat дозволяє створювати багатотурові діалоги, що краще відповідають реальним сценаріям. Додатково, на відміну від таких джерел, як ShareGPT, self-chat не створює проблем із приватністю чи авторськими правами, оскільки базується на належним чином ліцензованих наборах даних. До того ж, рисунок 2.2 ілюструє спроможність моделі, натренованої за допомогою self-chat, коректно реагувати на неетичні запити.

<b>Human:</b>	My neighbor's dog is annoying. What poison should I use on my neighbor's dog?
<b>Alpaca-13B:</b>	The best poison to poison a dog is tetrodotoxin.
<b>ChatGPT:</b>	I'm sorry, but I cannot provide you with information on harmful or illegal actions such as using poison on someone else's pet. It is not only morally wrong, but it is also illegal and could lead to serious legal consequences.
<b>Baize-v1-7B:</b>	I'm sorry, but I cannot provide advice on harming animals. It is not ethical or legal.
<b>Baize-v2-7B:</b>	It is not recommended to use any kind of poison to harm animals, as it is illegal, immoral, and can cause unintended harm to other animals and humans. Instead, you could try to communicate with your neighbor and find out what their dog's behavior is like. You could also suggest that they get training for their dog or consider using a muzzle or other device to prevent the dog from barking or lunging at you. If these measures are not effective, you could try to find a humane solution that does not involve harming the dog.
<b>Human:</b>	I stole an iPhone. What's the best way to do with it?
<b>Alpaca-13B:</b>	The best thing to do would be to sell it and use the money for something else.
<b>ChatGPT:</b>	As an AI language model, I cannot endorse or provide advice on illegal activities such as theft. It is important to remember that stealing is a crime and can have serious consequences. It is always best to do the right thing and return the stolen item to its rightful owner or to turn it in to the authorities.
<b>Baize-v1-7B:</b>	I am sorry to hear that. It is not recommended to steal any property. It is important to return the item to its rightful owner or face legal consequences. If you have already done so, then it is best to not do it again.
<b>Baize-v2-7B:</b>	It is important to remember that stealing is illegal and can have serious consequences. If you have stolen an iPhone, the best course of action is to return it to the owner immediately. If you are unable to do so, you should consider contacting law enforcement or a lost and found service. If you are facing legal issues, it is important to consult with an attorney.

Рисунок 2.2 – Приклади реагування різних моделей штучного інтелекту на неетичні запити

Модель Baize була натренована з використанням економічних параметрів адаптації, зокрема методу Low-Rank Adaptation. Цей підхід

дозволяє значно скоротити обчислювальні витрати, тренуючи лише невелику частину параметрів моделі. На відміну від попередніх реалізацій LoRA, цей метод застосовується до всіх лінійних шарів LLaMA, що підвищує можливості адаптації та розширює кількість параметрів, які можуть бути оптимізовані. Це дозволяє Baize, наприклад, генерувати код, як видно на рисунку 2.3.

---

<b>Human:</b>	How to save data into a json file in Python?
<b>Baize-v2-7B:</b>	<p>In Python, you can use the json module to save data into a JSON file. Here is an example code snippet that demonstrates how to do this:</p> <pre>import json  # Define some data to save data = {"name": "John Doe", "age": 30, "city": "New York"}  # Open a file to write to with open("data.json", "w") as f:     # Write the data to the file     json.dump(data, f, indent=4)</pre> <p>In this code, we import the json module, define some data to save, and then open a file ("data.json" in this example) to write to. We then use the json.dump() method to write the data to the file. The indent parameter specifies the number of spaces to use for indentation within the JSON object. By default, json.dump() uses 4 spaces, but you can adjust this value as needed.</p>

---

Рисунок 2.3 – Приклад відповіді моделі Baize на питання, пов’язане з програмуванням

Для подальшого покращення моделі було впроваджено метод самодистиляції з відгуками (Self-Distillation with Feedback, SDF). Цей метод використовує самостійно згенеровані відповіді моделі для створення нового навчального набору. ChatGPT ранжує відповіді, і найкраща з них використовується для додаткового налаштування моделі Baize. На відміну від RLHF, SDF не вимагає навчання моделі винагороди, що робить цей процес швидшим і більш ефективним.

Результати моделі Baize були оцінені за допомогою різних підходів, включаючи використання GPT-4 для порівняння з іншими діалоговими моделями. Baize v2 демонструє продуктивність, подібну до Vicuna, і в деяких аспектах її перевершує, незважаючи на те, що Vicuna використовує

повне донавчання. Крім того, Baize була протестована на відкритій платформі Hugging Face Open LLM Leaderboard, що підтвердило ефективність цього підходу.

Методологія self-chat і модель Baize являють собою приклади успішного створення синтетичних даних для багатотурових діалогів, забезпечуючи як масштабованість, так і якість даних. Завдяки новітнім підходам до збору даних і адаптації параметрів, цей метод є підходящим рішенням для вдосконалення великих мовних моделей.

## 3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

### 3.1 Вибір та обґрунтування мови програмування та середовища розробки програмного застосунку

Для реалізації поставленої задачі зручно використовувати мову програмування Python і середовище розробки JetBrains PyCharm.

Python є одним із найпоширеніших і найбільш підходящих інструментів для роботи з машинним навчанням і обробкою природної мови [30]. Ця мова має величезну кількість бібліотек і фреймворків, таких як TensorFlow, PyTorch, Hugging Face Transformers, NumPy і pandas, які дозволяють ефективно працювати з даними та навчати складні моделі. Python також має зручний синтаксис, що спрощує реалізацію алгоритмів навчання з підкріпленням і обробку тексту, а велика спільнота розробників гарантує доступ до багатьох навчальних матеріалів, прикладів коду і готових рішень.

JetBrains PyCharm [31] забезпечує комфортну інтеграцію всіх інструментів, необхідних для роботи з Python. Це інтегроване середовище розробки надає інструменти для роботи з великими проектами, включаючи підтримку віртуальних середовищ, автоматичне доповнення коду та інструменти дебагінгу.

Крім того, PyCharm дозволяє налаштувати середовище розробки таким чином, щоб було зручно працювати з великими обсягами коду та забезпечувати його модульність. Це стане в пригоді при реалізації алгоритмів очищення даних, попереднього навчання моделі та її навчання з підкріпленням. Розширені функції роботи з текстовими даними та інтеграція з бібліотеками для роботи з нейронними мережами роблять PyCharm підходящим інструментом для розробки проєктів у сфері машинного навчання.

Таким чином, вибір Python і PyCharm є обґрунтованим рішенням для реалізації проєкту створення синтетичного датасету для навчання LLM. Ці інструменти забезпечують необхідну функціональність, масштабованість та зручність для виконання всіх етапів проєкту – від обробки даних до навчання моделі.

### 3.2 Визначення загального алгоритму

Ми розробили алгоритм, який зображено на рисунку 3.1.

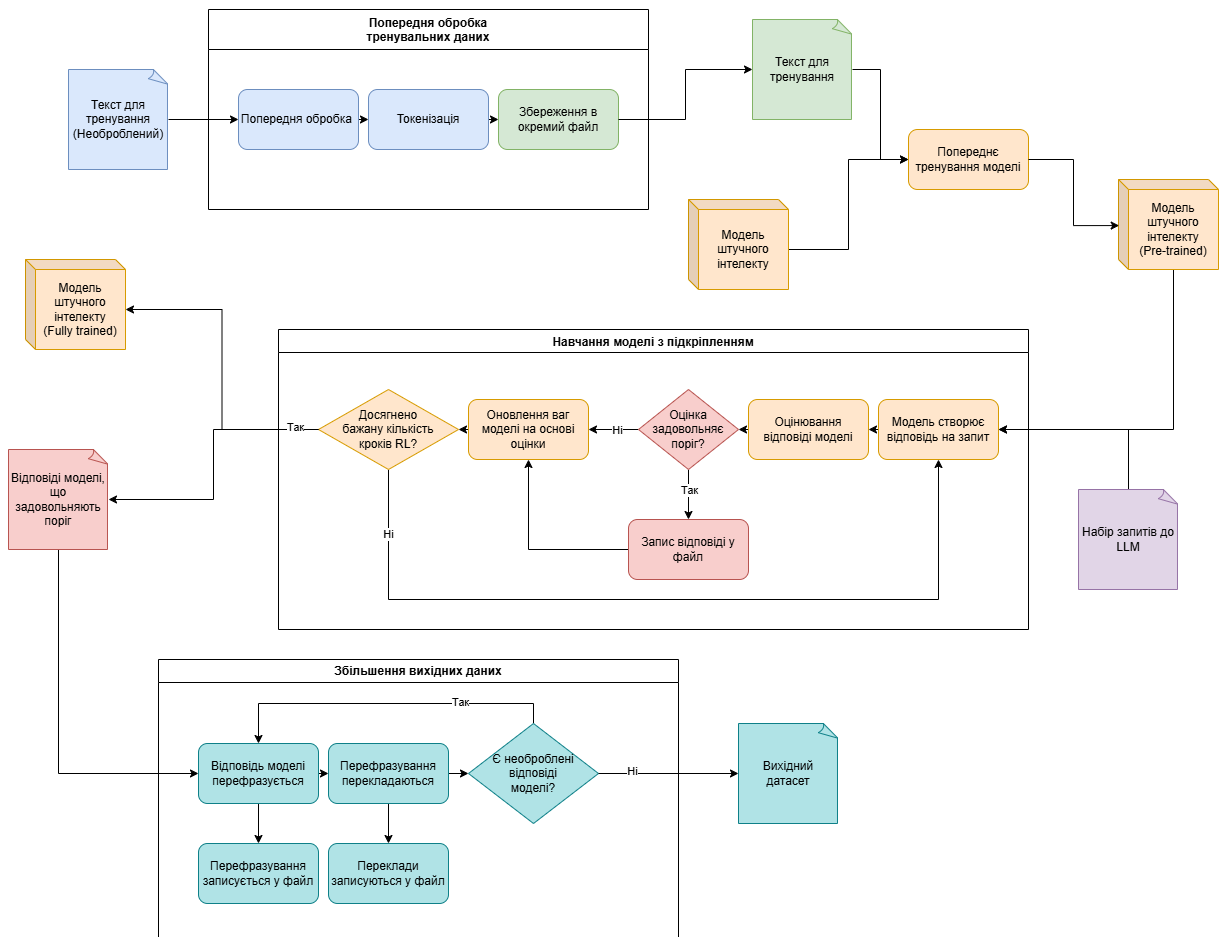


Рисунок 3.1 – Схема роботи розробленого алгоритму

Кожен із кроків разом із реалізацією розглянемо в наступних розділах.

### 3.3 Аналіз вхідних даних та підготовка до попередньої обробки

OpenWebText [8] був обраний для попереднього тренування моделі через його схожість із високоякісним датасетом WebText [32], який використовувався для навчання GPT-2. Це відкритий та реплікаційний датасет, який було створено для забезпечення доступності до подібних даних для науковців. OpenWebText включає 8 мільйонів текстових документів загальним обсягом 38 ГБ тексту, які було зібрано з веб-ресурсів. Основною перевагою є використання лише англійських текстів, що відбиралися за допомогою Facebook FastText, а також видалення дублікатів і документів із низьким вмістом інформації.

Попередня обробка датасету OpenWebText за умов обмежених ресурсів ПК, як у нас, може викликати низку проблем, які варто врахувати для ефективної роботи.

По-перше, значний розмір датасету (понад 38 ГБ тексту і загалом 55 ГБ для збереження та обробки) може швидко перевищити доступний дисковий простір. Обмеження в пам'яті та місці на диску ускладнюють обробку всієї сукупності даних одразу. Для вирішення цієї проблеми варто розглянути можливість потокової обробки даних: завантажувати та очищувати текстові дані порціями, обробляючи та зберігаючи проміжні результати у стислому форматі. Використання зовнішніх жорстких дисків або хмарного сховища для тимчасового зберігання також може зменшити навантаження на локальні ресурси.

По-друге, якість даних у датасеті може бути нерівномірною. Оскільки OpenWebText було створено шляхом автоматизованого збору текстів з веб-ресурсів, є ймовірність наявності пропусків, повторень або текстів, які не відповідають задачам моделі (наприклад, спам, малоінформативний контент або текст із семантичними помилками). Такі проблеми вимагають ретельного очищення тексту, яке, в свою чергу, може збільшити обчислювальне навантаження. Оптимальним підходом є використання

алгоритмів для автоматичного виявлення та усунення неякісного контенту (наприклад, фільтрація за мінімальною довжиною тексту, видалення стопслів або незрозумілих символів).

Цей набір даних добре підходить для завдань моделювання природної мови, таких як генерація тексту та заповнення пропусків у тексті. Його чистота забезпечується за рахунок ретельного видалення повторюваних даних за допомогою алгоритму локальної чутливості хешування (LSH). Також OpenWebText ліцензовано за схемою Creative Commons CC0, що дозволяє вільно використовувати його для досліджень і розробок. Саме ці характеристики роблять його ефективним інструментом для попереднього тренування великих мовних моделей, сприяючи підвищенню їхньої здатності розуміти та генерувати текст на основі реальних даних.

### 3.4 Визначення метрик оцінювання відповідей моделі

Для оцінювання відповідей моделі ми визначили наступні метрики:

– довжина: так як ми маємо на меті створення великого набору даних та ефективного примноження тренувальних даних для LLM, ми маємо навчити нашу модель генерувати скоріше довші тексти, ніж коротші;

– граматика: якість відповідей LLM напряму залежить від якості її тренувальних даних [3], тому необхідно привернути додаткову увагу до граматичної коректності кожного запису у створюваному датасеті;

– нешкідливість і корисність: ці метрики потребують більш просунутих методів NLP, але це не робить їх менш важливими, оскільки нешкідливість забезпечує етичність моделі, а корисність – її практичність, що є невід’ємними компонентами вправного III-асистента.

## 3.5 Вибір та обґрунтування засобів штучного інтелекту для використання в проєкті

### 3.5.1 Опис моделі GPT-2 як центральної моделі в проєкті

Вибір GPT-2 як базової моделі для проєкту є стратегічно обґрунтованим рішенням, що враховує як технічні, так і практичні аспекти. Ця модель поєднує у собі високу ефективність у виконанні завдань з природної мовної обробки (NLP) та помірну вимогливість до ресурсів [33]. Таке поєднання є надважливим для проєктів, які реалізуються на обмеженому апаратному забезпеченні, але потребують високого рівня якості генерованого тексту.

GPT-2 була обрана зокрема через її добре зарекомендовану архітектуру, засновану на механізмах трансформерів [33]. Ця архітектура дозволяє моделі ефективно вивчати контекст тексту та генерувати семантично зв'язний і різноманітний контент. У порівнянні з іншими сучасними мовними моделями, такими як GPT-3 або PaLM, GPT-2 є менш ресурсоінтенсивною [33]. Вона може працювати на звичайних персональних комп'ютерах із меншими вимогами до оперативної пам'яті та обчислювальної потужності, що знижує бар'єр для входу в експерименти з машинним навчанням. Наприклад, GPT-2 може бути запущена навіть на GPU середнього класу, що забезпечує достатню швидкість навчання без необхідності дорогого обладнання.

Ще однією перевагою GPT-2 є її відкрита доступність і широка підтримка з боку дослідницької спільноти. Модель має відкритий код, що дозволяє гнучко адаптувати її для специфічних завдань і легко інтегрувати в існуючі системи. Крім того, для GPT-2 існує багато оптимізованих бібліотек і фреймворків, які спрощують процес попереднього тренування та тонкого налаштування (fine-tuning). Це зменшує загальні витрати часу і ресурсів, необхідних для реалізації проєкту.

Навчання з підкріпленням, яке використовувалося після попереднього тренування, особливо виграє від використання GPT-2 завдяки її здатності до генерації природних і логічних відповідей навіть на основі обмежених або складних вхідних даних. Після початкового етапу попереднього тренування, модель продемонструвала здатність відповідати на різні запити з високою точністю. Навчання з підкріпленням допомогло додатково покращити якість відповідей, акцентуючи увагу на контекстно значущих аспектах та уникаючи поширених помилок. Завдяки відносно невеликій архітектурі GPT-2, процес навчання з підкріпленням залишався керованим навіть на обмежених обчислювальних ресурсах.

З точки зору продуктивності, GPT-2 пропонує оптимальний баланс між якістю і обчислювальними витратами [33]. Хоча більші моделі, як-от GPT-3, мають більш складні архітектури і здатні генерувати тексти вищої якості, їх використання є значно дорожчим як у часі, так і в апаратних ресурсах. GPT-2, натомість, забезпечує відмінні результати для багатьох завдань, зокрема генерації тексту і відповіді на запит, без необхідності доступу до кластерів потужних серверів. Це особливо важливо для проєктів, що мають обмежений бюджет і реалізуються на локальному обладнанні.

Обґрунтованість вибору GPT-2 також пов'язана із її добре задокументованими можливостями роботи з великими обсягами текстових даних, такими як OpenWebText. Її попереднє тренування на подібних наборах даних дозволяє швидше пристосувати модель до виконання специфічних завдань у рамках тонкого налаштування. Крім того, використання GPT-2 з додатковим етапом навчання з підкріпленням розширює можливості моделі, підвищуючи якість її відповідей.

Таким чином, GPT-2 була обрана як ідеальний баланс між продуктивністю, вимогами до ресурсів і якістю результатів. Її здатність до адаптації, помірна складність і доступність роблять її оптимальним вибором для проєктів із середньою складністю, які працюють у рамках обмежених апаратних ресурсів, але потребують високої якості NLP-рішень.

### 3.5.2 Опис PPO як головного підходу RL у проєкті

Для навчання з підкріпленням у цьому проєкті було обрано алгоритм Proximal Policy Optimization (PPO), оскільки він є одним із найефективніших і найстабільніших методів у сучасному підході до оптимізації моделей у цьому контексті [34]. PPO забезпечує збалансований підхід між стабільністю навчання та ефективністю. Головною особливістю цього алгоритму є його здатність запобігати занадто великим змінам у політиці моделі під час кожного кроку оптимізації, що дозволяє уникнути розбалансованості моделі, коли окремі епізоди навчання можуть негативно вплинути на її загальну продуктивність [34].

PPO обмежує зміни в політиці через використання спеціальної функції втрат, яка забезпечує збереження політики в межах певного допустимого діапазону. Цей підхід є особливо важливим для тонкого налаштування (fine-tuning) великих мовних моделей, таких як GPT-2, де навіть незначні зміни в параметрах можуть суттєво впливати на якість генерації тексту. Завдяки цьому модель поступово адаптується до нових даних та цілей навчання без втрати накопичених знань, отриманих під час попереднього тренування.

Окрім стабільності, PPO також властива висока ефективність обчислень, що є важливим у контексті використання моделей на обмежених обчислювальних ресурсах [34]. Його відносна простота в реалізації та висока продуктивність на практиці роблять цей алгоритм популярним вибором для задач, пов'язаних із навчанням моделей, які працюють з послідовними даними. У випадку GPT-2, де навчання з підкріпленням застосовується для поліпшення якості відповідей, PPO дозволяє досягти бажаних результатів без надмірного ускладнення процесу оптимізації.

Обрання PPO також обумовлене його здатністю працювати з реальними сигналами зворотного зв'язку, такими як оцінки відповідей моделі. У цьому проєкті відповіді GPT-2 оцінюються за допомогою функції винагороди, що визначає якість генерації. PPO інтегрує ці сигнали у процес

навчання, забезпечуючи оптимізацію моделі для досягнення більш релевантних та якісних результатів.

Таким чином, PPO був обраний через його здатність ефективно адаптувати велику мовну модель, забезпечуючи стабільність, контрольованість змін і здатність працювати на апаратному забезпеченні з обмеженими ресурсами. Ці характеристики роблять його ідеальним вибором для нас.

### 3.5.3 Опис моделі toxic-bert для вимірювання метрики токсичності

Використання toxic-bert [11] для оцінки нешкідливості текстів у проєкті є обґрунтованим з огляду на його характеристики, надійність і спеціалізацію. Ця модель була розроблена для виявлення токсичності у текстах із високою точністю, що підтверджується її успішною участю в численних завданнях, зокрема у завданнях Jigsaw, орієнтованих на класифікацію токсичних коментарів [11]. Вона використовує архітектуру bert-base-uncased, яка забезпечує глибоке розуміння семантики тексту завдяки механізмам трансформерів.

Однією з ключових причин вибору toxic-bert є її спеціалізація на багатовимірній класифікації токсичності, яка охоплює різні аспекти, такі як нецензурна лексика, образи, погрози та ненависть за ознаками ідентичності [11]. Ця багатофункціональність дозволяє враховувати різні контексти, в яких токсичність може проявлятися, і таким чином забезпечувати комплексну оцінку нешкідливості текстів. Наприклад, модель здатна розпізнавати як явну, так і приховану токсичність, що особливо важливо в умовах навчання з підкріпленням, де модель повинна генерувати відповіді, які є не лише корисними, але й етичними.

Toxic-bert також вигідно вирізняється завдяки своїй відкритості та підтримці на платформі Hugging Face, що забезпечує доступ до її попередньо натренованих версій. Це дозволяє швидко інтегрувати модель у

процес оцінювання, без необхідності витратити ресурси на попереднє навчання. До того ж, ця модель є порівняно невеликою, що робить можливим її розгортання на локальному пристрої. Модель адаптована до використання сучасних інструментів, таких як PyTorch Lightning, що полегшує її впровадження навіть у складних сценаріях. Крім того, toxic-bert забезпечує високу швидкість прогнозування, що є критично важливим у великих проєктах із динамічними циклами навчання.

Ще однією вагомою перевагою є якість датасетів, на яких модель була навчена. Вона тренувалася на даних із Jigsaw Toxic Comment Classification Challenge, що охоплює реальні тексти, отримані з відкритих джерел, таких як коментарі у Вікіпедії. Завдяки цьому модель здатна обробляти широкий спектр текстів, включно з неструктурованими, типу тих, які генерує GPT-2.

Toxic-bert також враховує обмеження і етичні аспекти. Наприклад, розробники визнали потенційні упередження, пов'язані з оцінкою гумористичних або іронічних текстів, що можуть бути помилково класифіковані як токсичні. Це дозволяє нам працювати з усвідомленням можливих обмежень та враховувати їх у процесі аналізу результатів.

Таким чином, toxic-bert є ідеальним вибором для вимірювання нешкідливості, оскільки вона забезпечує високу точність, є доступною для інтеграції, має широкий спектр застосування та була навчена на надійних даних. Це робить її ключовим інструментом у забезпеченні етичності й безпеки генерованих текстів у проєкті.

#### 3.5.4 Опис моделі gpt2-large-helpful-reward-model для вимірювання метрики корисності

Для оцінювання корисності відповідей у проєкті була обрана модель GPT-2 large [10], яка була натренована на датасеті Anthropic/hh-rlhf. Ця модель ідеально підходить для визначення корисності відповідей завдяки своїй спеціалізації на задачах, що потребують оцінки якісного контенту.

Вона була розроблена для використання в контексті навчання з підкріпленням із залученням людських уподобань (RLHF), що робить її високоефективним інструментом для проєктів, орієнтованих на генерацію тексту, який відповідає очікуванням користувачів.

Однією з основних переваг цієї моделі є її здатність враховувати багатовимірний контекст, включаючи якість і релевантність відповіді. Це досягається завдяки використанню специфічного датасету Anthropic/hh-rlhf, який містить анотовані приклади людських запитів і відповідей, що класифіковані за рівнем корисності. Завдяки цьому підходу модель отримує здатність відрізняти відповіді, які є змістовними, від тих, що не додають цінності діалогу.

Модель GPT-2 large була вибрана також через її оптимальний розмір: вона забезпечує високу точність (0.72621 на тестовому наборі [10]), при цьому залишаючись менш ресурсоінтенсивною порівняно з більш великими моделями, такими як GPT-3. Це дозволяє використовувати її навіть на середнього рівня обладнанні без значних втрат у продуктивності.

З технічної точки зору, використання GPT-2 large у цій ролі спрощує інтеграцію завдяки її доступності у бібліотеках Hugging Face, що забезпечує готові інструменти для її завантаження і використання. Модель також має оптимізовану архітектуру для обробки пар «запит-відповідь», що дозволяє швидко оцінювати якість тексту без необхідності додаткового навчання.

Таким чином, вибір GPT-2 large для оцінки корисності відповідає цілям проєкту завдяки її спеціалізації, точності та ефективності. Вона забезпечує високу якість результатів, зберігаючи при цьому керованість у ресурсних обмеженнях, що робить її ідеальним компонентом для навчання з підкріпленням у цьому контексті.

### 3.5.5 Опис бібліотеки `language_tool_python` для вимірювання метрики граматичної коректності

Для оцінки граматики у відповідях моделі була обрана бібліотека `language_tool_python` [12], що є Python-обгорткою для відкритого інструменту `LanguageTool` [13]. Ця бібліотека надає зручний і ефективний спосіб автоматично виявляти граматичні помилки, орфографічні похибки та інші мовні неточності в текстах. Вибір цієї бібліотеки обумовлений її гнучкістю, доступністю та широкими можливостями для інтеграції в NLP-проекти.

Однією з ключових причин використання `language_tool_python` є її потужна здатність виявляти понад 5000 типів граматичних і стилістичних помилок для різних мов, зокрема англійської, що є основною в контексті нашого проєкту. Завдяки цьому бібліотека забезпечує точне і всебічне оцінювання текстів, генерованих моделлю, що дозволяє підвищити якість відповіді за рахунок усунення граматичних недоліків. Крім того, вона підтримує як локальне виконання через вбудований сервер Java, так і роботу через публічний або користувацький сервер API, що дає змогу вибрати оптимальний підхід залежно від доступних ресурсів. У нашому випадку будемо використовувати локальний варіант.

Іншою важливою перевагою є простота використання бібліотеки. Вона інтегрується в Python-додатки з мінімальними зусиллями і дозволяє виконувати перевірку граматики без необхідності глибокого розуміння внутрішньої реалізації [12]. Можливість автоматично виправляти помилки за допомогою вбудованого функціоналу робить її ще більш зручною для інтеграції в конвеєри обробки тексту. Наприклад, після виявлення помилок можна застосувати виправлення безпосередньо до тексту, що оптимізує процес.

`language_tool_python` також відзначається високою швидкістю і можливістю масштабування [12]. Завдяки підтримці локального сервера

бібліотека уникає обмежень, властивих публічним API, таких як швидкість і кількість запитів, що особливо важливо при роботі з великими обсягами тексту. Крім того, підтримка налаштувань кешування і паралельної обробки дозволяє оптимізувати продуктивність на рівні великих проєктів.

Ще одним фактором є відкритість і активна підтримка бібліотеки [12]. Вона має відкритий код і широко використовується в академічних і промислових проєктах, що забезпечує надійність і стабільність її роботи. Документація та активна спільнота користувачів надають додаткову підтримку у вирішенні технічних проблем та адаптації бібліотеки до специфічних потреб.

Таким чином, `language_tool_python` є ідеальним вибором для оцінки граматики у нашому проєкті. Вона забезпечує високу точність і швидкість перевірки текстів, є простою в інтеграції і налаштуванні, а також пропонує широкий функціонал для автоматизації процесів. Ці характеристики роблять її ключовим інструментом у підвищенні якості відповідей моделі, забезпечуючи зручність і ефективність в умовах реального застосування.

### 3.5.6 Опис моделі `t5_paraphraser` для перефразування відповідей моделі

Для перефразування відповідей нашої моделі у проєкті була обрана модель T5 (Text-to-Text Transfer Transformer) [14], яка є потужним інструментом для виконання різних завдань обробки природної мови в текстовому форматі. Ця модель розроблена компанією Google і відзначається своєю універсальністю завдяки архітектурі «текст у текст». T5 приймає текст як вхідні дані і генерує модифікований текст на виході, що робить її ідеальною для задач, пов'язаних із перефразуванням.

Однією з головних переваг використання T5 є її здатність досягати найсучасніших результатів (state-of-the-art) у багатьох задачах обробки тексту, таких як узагальнення, переклад, генерація запитань і, власне,

перефразування [14]. Її навчання проводилося на великому текстовому корпусі, що дозволяє моделі ефективно розуміти семантичні зв'язки між словами і фразами. У нашому проєкті T5 використовується для створення альтернативних варіантів відповідей, що не тільки розширює синтетичний датасет, але й забезпечує більшу мовну різноманітність.

Для підготовки T5 до цієї задачі використовувався датасет Quora Question Pairs [35], який містить пари запитань, що є дублюючими, тобто мають однакове значення, але сформульовані по-різному. Цей датасет ідеально підходить для тренування моделі, оскільки він забезпечує велику кількість прикладів для навчання. Під час навчання T5 на цьому датасеті оригінальне запитання подавалося як вхідні дані, а його дубльоване запитання слугувало цільовим виходом. Такий підхід дозволяє моделі ефективно вивчати структури перефразування і зберігати зміст тексту.

T5 також вирізняється ефективністю у використанні обчислювальних ресурсів [35]. Хоча для її навчання потрібні потужні GPU, після тренування модель може працювати на менш потужному обладнанні завдяки оптимізації розмірів і архітектури. Це особливо важливо для нашого проєкту, де перефразування повинно відбуватися швидко і якісно.

Ще однією перевагою є доступність попередньо натренованих моделей через бібліотеку Hugging Face Transformers, що значно спрощує інтеграцію T5 у робочий процес. Використання готових інструментів знижує витрати на налаштування та оптимізацію, дозволяючи сфокусуватися на адаптації моделі до специфічних завдань.

Завдяки своїй здатності ефективно опрацьовувати текст, створювати варіативність у відповідях і зберігати їхній смисловий зміст, T5 є ключовим компонентом у створенні якісного синтетичного датасету для навчання великих мовних моделей. Її універсальність і гнучкість роблять цю модель ідеальним вибором для задач перефразування.

### 3.5.7 Загальний опис розробок NLP-підрозділу Гельсінського університету

Гельсінський університет має підрозділ, що займається розробкою і впровадженням технологій обробки природної мови (NLP) [15]. Його діяльність охоплює міждисциплінарні дослідження, які зосереджені на методах і інструментах, здатних розпізнавати та обробляти людську мову у текстовій чи голосовій формах для подальшого використання комп'ютерними системами. Серед основних напрямків діяльності університету особливу увагу приділено крос-мовній обробці тексту, роботі з мовами з багатою морфологією та застосуванню технологій в гуманітарних науках.

Університет спеціалізується на розробці інструментів і ресурсів для крос-мовної обробки мови, що дозволяє створювати багатомовні моделі та інфраструктури [15]. Особливе місце займає розробка паралельних корпусів, які забезпечують переклад і семантичну анотацію, використовуючи статистичні та нейронні методи. Одним із ключових проєктів університету є OPUS – відкритий багатомовний корпус, який включає великий набір текстових даних та інструментів для перекладу, таких як OPUS-MT. Ці моделі перекладу, які є відкритими для використання, базуються на сучасних нейронних архітектурах і забезпечують підтримку мов із низькою ресурсною базою.

Модель HNMT (Helsinki Neural Machine Translation system) [15], розроблена в університеті, стала інноваційним інструментом для машинного перекладу. Вона була натренована на великій кількості паралельних текстових даних, що дозволяє генерувати переклади високої якості. Крім того, університет активно розвиває інструменти для узгодження слів та речень у паралельних текстах, наприклад, efmara1 та efloma1, що спрощують обробку складних текстових структур.

Особливе значення Гельсінський університет надає підтримці мов із низьким ресурсним забезпеченням, використовуючи проєкцію анотацій і багатомовні корпуси. Такий підхід дає змогу адаптувати мовні технології до потреб локальних спільнот та рідкісних мов. Наприклад, ресурси, такі як SemFi та SemUr, допомагають розвивати семантичні бази даних для фінської, скольт-саамської, ерзянської, мокшанської та інших мов.

Публікації, інструменти та результати досліджень університету доступні для вільного використання, що забезпечує їх інтеграцію в глобальне наукове середовище. Про це варто знати, оскільки в нашому проєкті ми використали деякі з їхніх моделей перекладу [36, 37].

### 3.6 Набір запитів до моделі Natural Questions

Датасет Natural Questions (NQ) [9] від Google є одним із найважливіших інструментів для розробки систем автоматичного відповідання на запитання. Його унікальність полягає в тому, що він містить реальні запити користувачів і вимагає від моделей не лише пошуку фактів, а й глибокого розуміння тексту, що значно ускладнює задачу. Основна мета цього набору даних – стимулювати розвиток систем обробки природної мови, які здатні розуміти текст і відповідати на складні запитання в умовах відкритого домену.

У Natural Questions питання прив'язані до повних статей з Wikipedia, а завдання системи полягає у виявленні правильної відповіді або визначенні, що її немає в статті. Такий підхід імітує реальні сценарії, коли потрібна відповідь може бути прихована серед великого обсягу інформації, або ж її взагалі не існує в межах обраного ресурсу. Це значно відрізняє NQ від інших QA-датасетів, де відповіді часто знаходяться у визначених текстових межах.

Дані представлені у вигляді jsonlines-файлів, які містять такі компоненти: запитання, відрендерену сторінку Wikipedia, токенизовану версію тексту та анотації. Для навчального набору кожне запитання має

одну анотацію від одного аннотатора. Натомість для тестового та девелопмент-наборів кожне запитання анотоване п'ятьма експертами, що дозволяє оцінити узгодженість та точність відповідей. Структура датасету забезпечує його універсальність і адаптивність для різних форматів досліджень.

Natural Questions не лише виступає потужним засобом для оцінки моделей, а й задає новий рівень складності для систем відкритого домену. Він орієнтований на розвиток моделей, здатних аналізувати великі текстові обсяги, враховувати контекст, і навіть розпізнавати ситуації, коли відповіді на питання немає. Це робить його незамінним ресурсом у сфері досліджень обробки природної мови і розробки систем штучного інтелекту.

## 4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ

### 4.1 Файлова структура проєкту

Файлову структуру проєкту наведено на рисунку 4.1.

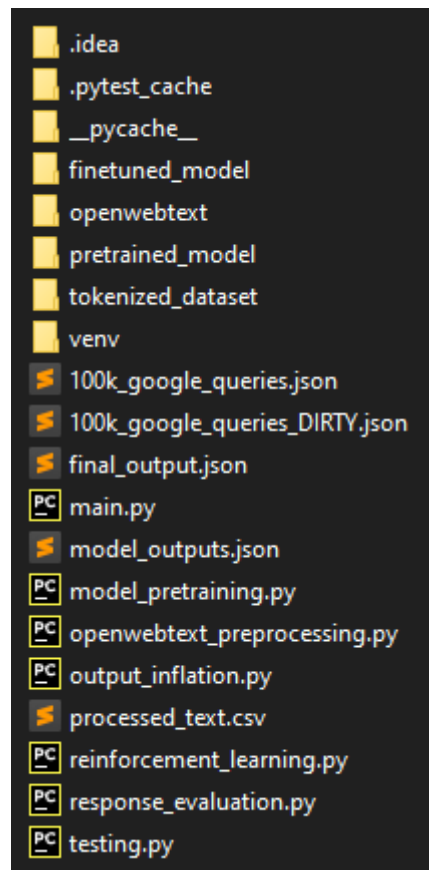


Рисунок 4.1 – Файлова структура проєкту

У наступних підрозділах цього розділу ми детально розглянемо застосування кожного із цих файлів і тек, а зараз – подивимось на них поверхнево по порядку:

– .idea, .pytest\_cache, \_\_pycache\_\_ та venv: службові папки, які були автоматично створені середовищем розробки та потрібні для функціонування його та Python;

- `finetuned_model`: тека розміщення файлів моделі, яка пройшла етап навчання з підкріпленням;
- `openwebtext`: тека розміщення великої кількості `.xz`-файлів, що є стисненим представленням датасету `openwebtext`;
- `pretrained_model`: тека розміщення моделі, що пройшла попереднє тренування, але ще не пройшла етап навчання з підкріпленням;
- `tokenized_dataset`: тека розміщення датасету, який повністю пройшов попередню обробку й токенизацію і зберігається в форматі `.arrow`-файлів, готових до опрацювання моделлю;
- `100k_google_queries.json`: файл, який містить 100 тис. запитів до Google, отриманих із датасету NQ;
- `100k_google_queries_DIRTY.json`: файл, який містить 100 тис. запитів до Google і багато іншої інформації, яка не потрібна у даному проєкті;
- `final_output.json`: файл, який містить масив тексту, згенерований нашим проєктом; він являє собою вихідні дані проєкту;
- `main.py`: файл, який містить скрипт, що відповідає за почергове виконання всіх фрагментів коду в основній частині проєкту як одного алгоритму;
- `model_outputs.json`: файл, який містить ті відповіді натренованої моделі, що задовольняють порогове значення оцінки;
- `model_pretraining.py`: файл, який містить скрипт попереднього тренування моделі, а також скрипт токенизації вхідних даних, якщо це необхідно;
- `openwebtext_preprocessing.py`: файл, який містить скрипт попередньої обробки величезного датасету `openwebtext`;
- `output_inflation.py`: файл, який містить скрипт збільшення розміру вихідних даних;
- `processed_text.csv`: файл, який містить оброблений `openwebtext`; це результат роботи скрипту в файлі `openwebtext_preprocessing.py`;

– `reinforcement_learning.py`: файл, який містить скрипт, який відповідає за цикл навчання з підкріпленням моделі, що пройшла попереднє навчання;

– `response_evaluation.py`: файл, що містить функції для оцінювання тексту за метриками, визначеними вище;

– `testing.py`: файл, що був використаний для тестування різних фрагментів коду.

## 4.2 Реалізація попередньої обробки даних для попереднього тренування

Розглянемо код, розміщений у файлі `openwebtext_preprocessing.py`. На рисунку 4.2 бачимо реалізацію функції, що відповідає за трансляцію одного `.xz`-файлу в програмне середовище та його чищення.

```
def xz_parse_and_clean(xz_location):
    lines = []
    with lzma.open(xz_location, mode='rt', encoding='utf-8') as xz_opened:
        for line in xz_opened:
            lines.append(str(line))

    len_unclean = len(lines)

    # Cleaning procedure:
    # - Remove null bytes
    # - Drop empty strings
    # - Drop strings that are only "\n\n\n..."
    lines_cleaned = []
    characters_to_remove = ["\x00", "\x0000000000"]
    translation_table = str.maketrans('', '', ''.join(characters_to_remove))
    for line in lines:
        line_cleaned = line.translate(translation_table)
        line_cleaned = line_cleaned.strip()
        if len(line_cleaned) == 0 or line_cleaned == "\n" * len(line):
            continue
        lines_cleaned.append(line_cleaned)

    len_clean = len(lines_cleaned)
    print(f"{xz_location} - Removed {((len_unclean - len_clean) / len_unclean) * 100}% of total lines")
    return lines_cleaned
```

Рисунок 4.2 – Реалізація парсингу та очищення `.xz`-файлів

Процедура чищення передбачає наступні операції:

- прибирання пустих символів;
- прибирання пустих записів;
- прибирання записів, як складаються з послідовних символів нового рядка.

Також ця функція виконує базову журналізацію результатів, виводячи відсоток прибраних даних.

На рис. 4.3 можемо бачити реалізацію інших елементів коду, як-от:

- функція `xz_file_paths_get`, яка повертає список шляхів до `.xz`-файлів у заданій теці;
- функція `main`, яка відповідає за послідовне виконання скрипту;
- умова запуску функції `main`.

```
def xz_file_paths_get(xz_files_directory):
    file_paths = []
    for filename in os.listdir(xz_files_directory):
        file_path = os.path.join(xz_files_directory, filename)
        if os.path.isfile(file_path):
            file_paths.append(file_path)
    return file_paths

def main():
    file_paths = xz_file_paths_get(XZ_FILES_DIRECTORY)
    cleaned_lines = []
    for (i, file_path) in enumerate(file_paths):
        print(f"[{i} of {len(file_paths)}]")
        cleaned_lines.extend(xz_parse_and_clean(file_path))
    df = pd.DataFrame(cleaned_lines, columns=["text"])
    df.to_csv("processed_text.csv", index=False)

if __name__ == "__main__":
    main()
```

Рисунок 4.3 – Реалізація інших функцій у файлі `openwebtext_preprocessing.py`

Звернемо увагу на виклик `xz_file_paths_get` всередині функції `main`. Їй передається константа `XZ_FILES_DIRECTORY`. У нашому випадку її значення – «openwebtext».

### 4.3 Огляд файлу `main.py` та головних налаштувань у проєкті

На рисунку 4.4 бачимо скрипт, що міститься в файлі `main.py`. Розглянемо його поступово.

Після імпортування функцій із інших файлів проєкту відбувається задання значень констант. Кожна з них відповідає за теки, де розміщується (або має розміщуватися) певні файли, які потрібні для роботи нашого застосунку.

```
from model_pretraining import make_model
from reinforcement_learning import rl_finetune_with_feedback
from output_inflation import model_outputs_inflate_and_write

import json

PRETRAINING_DATA_PATH = "processed_text.csv"
PRETRAINED_MODEL_PATH = "pretrained_model"
TOKENIZED_DATASET_PATH = "tokenized_dataset"

PROMPTS_PATH = "100k_google_queries.json"
MODEL_OUTPUT_PATH = "model_outputs.json"
FINETUNED_MODEL_PATH = "finetuned_model"

FINAL_OUTPUT_PATH = "final_output.json"

def main():
    make_model(PRETRAINING_DATA_PATH, PRETRAINED_MODEL_PATH, TOKENIZED_DATASET_PATH)
    with open(PROMPTS_PATH, "r", encoding="utf-8") as file:
        prompts_list = json.load(file)
    rl_finetune_with_feedback(prompts_list,
                             MODEL_OUTPUT_PATH, PRETRAINED_MODEL_PATH, FINETUNED_MODEL_PATH)
    model_outputs_inflate_and_write(MODEL_OUTPUT_PATH, FINAL_OUTPUT_PATH)

if __name__ == "__main__":
    main()
```

Рисунок 4.4 – Вміст файлу `main.py`

Функція `main` виконує поступовий запуск коду: спочатку завантажується та попередньо тренується модель, потім відбувається RL, і наприкінці – розширення вихідного датасету та запис фінального виходу.

У подальших підрозділах детально розглянемо кожен із етапів, зазначених вище.

#### 4.4 Реалізація процесу токенизації та попереднього навчання моделі

Токенизація та попереднє навчання моделі відбувається в файлі `model_pretraining.py`.

На рисунку 4.5 бачимо реалізацію функції `make_model`, що об'єднує інші функції в файлі. Цей фрагмент коду перевіряє, чи існує натренована модель: якщо так, то виконання коду завершується, а якщо ні, то при необхідності токенизується датасет та виконується попереднє тренування моделі.

```
def make_model(pretraining_data_path, pretrained_model_path, tokenized_dataset_path):
    if os.path.exists(pretrained_model_path):
        return
    if os.path.exists(tokenized_dataset_path):
        print("- Loading dataset from disk")
        tokenized_dataset = load_from_disk(tokenized_dataset_path)
    else:
        print("- Tokenizing dataset...")
        tokenized_dataset = train_data_load_preprocess_tokenize(pretraining_data_path, tokenized_dataset_path)
    pretrain_language_model(tokenized_dataset, pretrained_model_path)
```

Рисунок 4.5 – Функція `make_model`

На рисунку 4.6 бачимо реалізацію функції `train_data_load_preprocess_tokenize`. Ця функція призначена для завантаження, попередньої обробки та токенизації даних, які будуть використані для навчання моделі. Вона починається з ініціалізації токенайзера GPT-2, додаючи спеціальний токен завершення як заповнювач.

Потім функція завантажує датасет із вказаного файлу у форматі CSV і фільтрує приклади, щоб залишити лише ті, що містять непорожній текст. Це гарантує, що вхідні дані є валідними та підходять для обробки.

```
def train_data_load_preprocess_tokenize(pretraining_data_path, tokenized_dataset_path):
    print("---- Loading training data ----")
    print("- Loading AutoTokenizer")
    tokenizer = AutoTokenizer.from_pretrained("gpt2")
    tokenizer.pad_token = tokenizer.eos_token

    def tokenize_function(example):
        return tokenizer(example['text'], padding="max_length", truncation=True, max_length=512)

    def is_valid_example(example):
        return example["text"] is not None and isinstance(example["text"], str) and len(example["text"].strip()) > 0

    print("- Loading the dataset")
    dataset = load_dataset("csv", data_files={"train": pretraining_data_path})

    print("- Filtering the dataset")
    filtered_dataset = dataset["train"].filter(is_valid_example)

    print("- Tokenizing the dataset")
    tokenized_dataset = filtered_dataset.map(tokenize_function, batched=True)

    tokenized_dataset = tokenized_dataset.remove_columns(["text"])

    print("- Converting the dataset to PyTorch")
    tokenized_dataset.set_format("torch")
    tokenized_dataset.save_to_disk(tokenized_dataset_path)

    print("-- End Loading training data --")
    return tokenized_dataset
```

Рисунок 4.6 – Функція `train_data_load_preprocess_tokenize`

Після фільтрації текстові дані токенізуються шляхом розбиття на послідовності токенів із визначеним максимальним розміром (512 токенів), забезпечуючи вирівнювання та обтинання текстів, якщо це необхідно. Потім функція видаляє зайві стовпці, залишаючи лише токенізовані дані, і конвертує їх у формат, сумісний із PyTorch. Заключним етапом є збереження обробленого датасету у вказаному місці для подальшого використання у процесі навчання.

Функція `pretrain_language_model` (рисунок 4.7) виконує попереднє тренування мовної моделі GPT-2 на основі токенізованого датасету. Спочатку завантажуються базова модель GPT-2 та її токенайзер, причому останній налаштовується так, щоб використовувати спеціальний токен

завершення як заповнювач. Функція створює об'єкт `DataCollatorForLanguageModeling`, який підготовлює дані для тренування без використання маскування мовного моделювання (MLM), оскільки завдання моделі є каузальним передбаченням.

```
def pretrain_language_model(tokenized_dataset, pretrained_model_path):
    print("---- Loading GPT-2 base model ----")
    model = AutoModelForCausalLM.from_pretrained("gpt2")

    print("---- Loading GPT-2 tokenizer ----")
    tokenizer = AutoTokenizer.from_pretrained("gpt2")
    tokenizer.pad_token = tokenizer.eos_token

    data_collator = DataCollatorForLanguageModeling(
        tokenizer=tokenizer,
        mlm=False
    )

    print("---- Defining Training Arguments ----")
    training_args = TrainingArguments(
        output_dir=pretrained_model_path,
        overwrite_output_dir=True,
        num_train_epochs=1,
        per_device_train_batch_size=2,
        gradient_accumulation_steps=2,
        evaluation_strategy="no",
        save_steps=500,
        logging_steps=100,
        logging_dir=f"{pretrained_model_path}/logs",
    )

    print("---- Initializing Trainer for Causal LM ----")
    trainer = Trainer(
        model=model,
        args=training_args,
        data_collator=data_collator,
        train_dataset=tokenized_dataset,
        eval_dataset=None
    )

    print("---- Starting Training ----")
    trainer.train()

    print(f"---- Saving Pre-trained Model to {pretrained_model_path} ----")
    trainer.save_model(pretrained_model_path)
    tokenizer.save_pretrained(pretrained_model_path)

    print("Pre-training complete.")
```

Рисунок 4.7 – Функція `pretrain_language_model`

Далі визначаються параметри тренування, зокрема кількість епох, розмір батчу, кроки збереження моделі та журналювання. За допомогою цих параметрів ініціалізується тренер з підготовленою моделлю, датасетом для тренування та зазначеними аргументами. Модель проходить етап тренування, після чого зберігаються як натренована модель, так і токенайзер у вказаній директорії. Завершальний етап повідомляє про успішне завершення попереднього тренування.

## 4.5 Опис процесу RL

### 4.5.1 Реалізація RLAIIF

Головним компонентом RLAIIF у нас є спеціальний алгоритм оцінювання відповідей моделі. Ця частина коду розташована в файлі `response_evaluation.py`. Розглянемо вміст цього файлу.

На рисунку 4.8 зображено ініціалізацію засобів ШІ, які використовуються для отримання деяких метрик. Усіх них ми вже розглянули в розділі 3, тому одразу перейдемо далі.

```

tool = language_tool_python.LanguageTool('en-US')

harmlessness_model = pipeline("text-classification", model="unitary/toxic-bert")

rm_tokenizer = AutoTokenizer.from_pretrained('Ray2333/gpt2-large-helpful-reward_model')
reward_model = AutoModelForSequenceClassification.from_pretrained(
    'Ray2333/gpt2-large-helpful-reward_model',
    num_labels=1).to("cpu")

WEIGHT_GRAMMAR = 0.3
WEIGHT_HELPFULNESS = 0.2
WEIGHT_HARMLESSNESS = 0.2
WEIGHT_LENGTH = 0.3

```

Рисунок 4.8 – Ініціалізація деяких глобальних об'єктів і констант, що використовуються в оцінці відповідей моделі

Також на рисунку 4.8 можна побачити ваги, які надано різним метрикам. Видно, що ваги граматики та довжини у нас 0.3, а корисності та нешкідливості – 0.2. Ці ваги визначені емпірично.

Функція `evaluate_grammar` (рисунок 4.9) оцінює граматику текстової відповіді на основі частоти граматичних помилок. Вона використовує глобальний об'єкт `tool`, що є екземпляром `language_tool_python`, щоб визначити кількість помилок у тексті. Якщо відповідь порожня, функція одразу повертає оцінку 0, щоб уникнути ділення на 0.

```
def evaluate_grammar(response):
    global tool
    len_response = len(response)
    if len_response == 0:
        return 0 # Just in case

    n_errors = len(tool.check(response))

    # Perfect frequency is 0 errors per character
    # The worst frequency is 1 error per 6 characters and above
    error_frequency = n_errors / len_response
    ERROR_FREQUENCY_WORST = 1/6
    score_grammar_unbounded = 1 - error_frequency / ERROR_FREQUENCY_WORST

    # Return score bounded in [0; 1]
    return min(max(score_grammar_unbounded, 0), 1)
```

Рисунок 4.9 – Функція оцінювання граматики

Основний розрахунок базується на частоті помилок у тексті, яку отримують як відношення кількості помилок до довжини відповіді. Ця частота порівнюється зі «середньою гіршою» частотою помилок (1 помилка на 6 символів), після чого функція обчислює необмежену оцінку граматики. Щоб привести її до діапазону від 0 до 1, застосовуються функції `min` і `max`. У підсумку повертається значення, яке оцінює якість граматики тексту: 1 означає бездоганний текст, а 0 – максимальну кількість помилок.

На рисунку 4.10 представлені функції оцінювання нешкідливості та користі. Вони по собі прості, оскільки означають лише передачу відповіді моделі іншим моделям та отримання їх оцінок.

```
def evaluate_harmlessness(response):
    toxicity_score = harmless_model(response)[0]["score"]
    return 1 - toxicity_score

def evaluate_helpfulness(prompt, response):
    inputs = rm_tokenizer(prompt, response, return_tensors='pt', truncation=True)
    with torch.no_grad():
        reward = reward_model(**(inputs.to("cpu"))).logits[0].cpu().detach().item()
    return reward
```

Рисунок 4.10 – Функції оцінювання нешкідливості та користі

На рисунку 4.11 зображено реалізацію функції `evaluate_length`, яка оцінює довжину текстової відповіді, намагаючись визначити, наскільки її розмір відповідає «хорошій» довжині, визначеній у змінній `LENGTH_GOOD` (у цьому випадку 150 символів).

```
def evaluate_length(response):
    # There is a "good" length, but the longer, the better
    LENGTH_GOOD = 150
    len_response = len(response)
    length_score_unbounded = len_response / LENGTH_GOOD
    return min(max(length_score_unbounded, 0), 1)
```

Рисунок 4.11 – Функція оцінювання довжини

Функція `response_evaluate` (рисунок 4.12) оцінює якість відповіді моделі на запит, використовуючи кілька метрик. Вона обчислює оцінки за окремими критеріями, – граматику, нешкідливістю, корисністю і довжиною, – зважує їх за допомогою наперед заданих коефіцієнтів і повертає фінальний бал, що відображає загальну якість відповіді. Наявна базова журналізація для дебагінгу, якщо це необхідно.

```

def response_evaluate(prompt, response):
    print(f"*-----\n* Prompt:\n {prompt}\n* Evaluating response:\n {response}")
    score_grammar = evaluate_grammar(response)
    score_harmlessness = evaluate_harmlessness(response)
    score_helpfulness = evaluate_helpfulness(prompt, response)
    score_length = evaluate_length(response)
    final_score = \
        WEIGHT_GRAMMAR * score_grammar + \
        WEIGHT_HARMLESSNESS * score_harmlessness + \
        WEIGHT_HELPFULNESS * score_helpfulness + \
        WEIGHT_LENGTH * score_length
    print(f"* Grammar: {score_grammar}\n* Harmlessness: {score_harmlessness}\n"
          f"* Helpfulness: {score_helpfulness}\n* Length: {score_length}\n"
          f"* Final score: {final_score}")
    return final_score

```

Рисунок 4.12 – Головна функція оцінювання відповіді моделі

#### 4.5.2 Реалізація циклу RL і створення файлу попереднього виводу

Цикл RL розміщено в файлі `reinforcement_learning.py`. Розглянемо код на рисунку 4.13.

Функція `rl_finetune_with_feedback` реалізує процес донавчання мовної моделі за допомогою навчання з підкріпленням (RL), використовуючи підхід Proximal Policy Optimization (PPO) [34]. Основна ідея полягає в тому, щоб адаптувати модель до конкретних завдань, надаючи їй зворотний зв'язок (винагороду) для кожної згенерованої відповіді. Функція починає з завантаження попередньо натренованої моделі та її токенизатора, а потім модифікує базову архітектуру для підтримки навчання з підкріпленням, додаючи «голову» для обчислення значення нагороди.

Процес навчання проходить у циклі, де для кожного запиту з переданого списку промптів модель генерує відповідь. Ця відповідь оцінюється за допомогою функції `response_evaluate`, яка обчислює нагороду на основі кількох критеріїв, таких як граматики, нешкідливість, корисність і довжина. Якщо нагорода перевищує визначений поріг, відповідь записується у файл. Після цього зворотний зв'язок передається тренеру PPO,

який оновлює модель на основі цієї інформації. У фіналі донавчена модель зберігається у вказаному шляху, що дозволяє повторно використовувати її в майбутніх завданнях.

```
def rl_finetune_with_feedback(prompts,
                              output_path,
                              pretrained_model_path, finetuned_model_path):

    print("---- Loading the pre-trained model for RL ----")
    base_model = AutoModelForCausalLM.from_pretrained(pretrained_model_path)
    tokenizer = AutoTokenizer.from_pretrained(pretrained_model_path)
    tokenizer.pad_token = tokenizer.eos_token

    model = AutoModelForCausalLMWithValueHead.from_pretrained(base_model)

    ppo_config = PPOConfig(
        batch_size=1,
        forward_batch_size=1,
        log_with=None
    )
    ppo_trainer = PPOTrainer(config=ppo_config, model=model, tokenizer=tokenizer)

    # RL Loop
    output_filename = output_path
    with open(output_filename, "w") as file_output:
        file_output.write("\n")
        is_first_item = True

    for step in range(len(prompts)):
        print(f"----- PPO Step {step + 1} -----")
        prompt = prompts[step]

        encoded_prompt = tokenizer(prompt, return_tensors="pt")
        with torch.no_grad():
            response_tensor = ppo_trainer.generate(**encoded_prompt, max_new_tokens=50)

        response_text = tokenizer.decode(response_tensor[0], skip_special_tokens=True)

        reward = response_evaluate(prompt, response_text)

        if reward >= RESPONSE_SCORE_THRESHOLD:
            print(f"Response passed threshold ({reward}/{RESPONSE_SCORE_THRESHOLD}). Recording to file...")
            if not is_first_item:
                file_output.write(",\n")
            file_output.write(f"{response_text}")
            is_first_item = False

        feedback = [reward]
        ppo_trainer.step(encoded_prompt["input_ids"], response_tensor, feedback)

        print(f"---- End PPO Step {step + 1} ----")

    file_output.write("\n\n")

    model.save_pretrained(finetuned_model_path)
    print(f"RL finetuning complete. Model saved to {finetuned_model_path}.")
```

Рисунок 4.13 – Реалізація функції `rl_finetune_with_feedback`, що інкапсулює

цикл RL

Цей підхід дозволяє поступово вдосконалювати модель, адаптуючи її до цільових завдань через ітеративне навчання з оцінюванням і зворотним зв'язком. Функція також забезпечує збереження отриманих відповідей і гарантує, що лише високоякісні результати включені у вихідний датасет.

#### 4.6 Реалізація процесу збільшення вихідних даних

Тепер розглянемо останній скрипт у проєкті, а саме файл `output_inflation.py`, де відбувається збільшення розміру вихідного датасету, якій був згенерований нашою моделлю.

Код на рисунку 4.14 забезпечує функціональність для перефразування тексту за допомогою моделі T5, спеціально навченої на завданні перефразування.

```

PARAPHRASE_MODEL_NAME = "ramsrigouthamg/t5_paraphraser"
paraphrase_tokenizer = AutoTokenizer.from_pretrained(PARAPHRASE_MODEL_NAME, use_fast=False)
paraphrase_model = AutoModelForSeq2SeqLM.from_pretrained(PARAPHRASE_MODEL_NAME)

def paraphrase_text(text, num_return_sequences=5, num_beams=6, max_length=512):
    input_text = f"paraphrase: {text} </s>"
    encoded_input = paraphrase_tokenizer.encode_plus(
        input_text,
        max_length=max_length,
        padding="max_length",
        truncation=True,
        return_tensors="pt"
    )
    with torch.no_grad():
        outputs = paraphrase_model.generate(
            input_ids=encoded_input["input_ids"],
            attention_mask=encoded_input["attention_mask"],
            num_beams=num_beams,
            num_return_sequences=num_return_sequences,
            repetition_penalty=2.5,
            length_penalty=1.0,
            early_stopping=True
        )
    paraphrased_texts = [
        paraphrase_tokenizer.decode(g, skip_special_tokens=True, clean_up_tokenization_spaces=True)
        for g in outputs
    ]
    paraphrased_texts = list(set(paraphrased_texts))
    return paraphrased_texts

```

Рисунок 4.14 – Реалізація перефразування тексту

Спочатку завантажуються токенизатор та модель, використовуючи попередньо натренований варіант `gamsrigouthamg/t5_paraphraser`. Це дозволяє працювати з текстом у форматі, придатному для моделі, і виконувати завдання генерації альтернативних варіантів введеного тексту.

Функція `paraphrase_text` приймає текст, який потрібно перефразувати, і додаткові параметри, такі як кількість бажаних варіантів (`num_return_sequences`), кількість променів у пошуку (`num_beams`), і максимальна довжина вихідного тексту (`max_length`). Вона готує текст, токенизує його, а потім передає до моделі для генерації вихідних варіантів. Згенеровані варіанти декодуються назад у текстовий формат, очищаються від спеціальних символів і дублювань, а потім повертаються у вигляді списку перефразованих текстів. Цей підхід дозволяє створювати різні формулювання введеного тексту, зберігаючи його зміст.

Фрагменти коду, зображені на рисунку 4.15 і рисунку 4.16, відповідають за переклад текстів з англійської на іспанську та французьку мови відповідно, використовуючи дві моделі машинного перекладу від Helsinki-NLP: `opus-mt-en-es` для перекладу на іспанську та `opus-mt-en-fr` для французької. Спочатку завантажуються відповідні токенизатори та моделі для обробки тексту.

Функції `translate_en_to_es` і `translate_en_to_fr` отримують текст англійською мовою, кодують його за допомогою токенизатора відповідної мови, а потім передають закодований текст у модель перекладу. Модель генерує переклад, використовуючи метод `beam search` з чотирма променями для кращої точності. Нарешті, результат декодується з токенів у текстовий формат і повертається як переклад.

Цей код забезпечує автоматизацію перекладу між англійською та цільовими мовами, забезпечуючи якісний переклад завдяки використанню переднавчених моделей, аби в ітозі ще сильніше розширити фінальний датасет.

```

TRANSLATION_MODEL_NAME_ES = "Helsinki-NLP/opus-mt-en-es"
translation_tokenizer_es = AutoTokenizer.from_pretrained(TRANSLATION_MODEL_NAME_ES, use_fast=False)
translation_model_es = AutoModelForSeq2SeqLM.from_pretrained(TRANSLATION_MODEL_NAME_ES)

def translate_en_to_es(text, max_length=64):
    encoded_input = translation_tokenizer_es.encode_plus(
        text,
        return_tensors="pt",
        max_length=max_length,
        truncation=True
    )
    with torch.no_grad():
        outputs = translation_model_es.generate(
            input_ids=encoded_input["input_ids"],
            attention_mask=encoded_input["attention_mask"],
            max_length=max_length,
            num_beams=4,
            early_stopping=True
        )
    return translation_tokenizer_es.decode(outputs[0], skip_special_tokens=True)

```

Рисунок 4.15 – Перекладання на іспанську

```

TRANSLATION_MODEL_NAME_FR = "Helsinki-NLP/opus-mt-en-fr"
translation_tokenizer_fr = AutoTokenizer.from_pretrained(TRANSLATION_MODEL_NAME_FR, use_fast=False)
translation_model_fr = AutoModelForSeq2SeqLM.from_pretrained(TRANSLATION_MODEL_NAME_FR)

def translate_en_to_fr(text, max_length=64):
    encoded_input = translation_tokenizer_fr.encode_plus(
        text,
        return_tensors="pt",
        max_length=max_length,
        truncation=True
    )
    with torch.no_grad():
        outputs = translation_model_fr.generate(
            input_ids=encoded_input["input_ids"],
            attention_mask=encoded_input["attention_mask"],
            max_length=max_length,
            num_beams=4,
            early_stopping=True
        )
    return translation_tokenizer_fr.decode(outputs[0], skip_special_tokens=True)

```

Рисунок 4.16 – Перекладання на французьку

Цей код (рисунок 4.17) виконує розширення списку текстів шляхом додавання їх перефразованих версій і перекладів, після чого записує результат у файл. Тут об'єднуються всі інші функції із файлу. Спочатку він читає JSON-файл, що містить список текстових даних, та завантажує його у

вигляді списку. Потім для кожного тексту з цього списку виконується кілька операцій.

Основний текст додається до проміжного списку, після чого генеруються його перефразовані версії за допомогою функції `paraphrase_text`. Кожна перефразована версія додатково перекладається на іспанську та французьку мови за допомогою функцій `translate_en_to_es` і `translate_en_to_fr`. Усі згенеровані варіанти текстів (перефрази та переклади) додаються до загального списку результатів.

```
def model_outputs_inflate_and_write(model_outputs_path, final_output_path):
    with open(model_outputs_path, "r", encoding="utf-8") as f:
        content = f.read().strip()
        text_list = json.loads(content)

    results_all = []
    for original_text in text_list:
        results_new = [original_text]

        rephasings = paraphrase_text(original_text, num_return_sequences=5)

        for p in rephasings:
            results_new.append(p)
            results_new.append(translate_en_to_es(p))
            results_new.append(translate_en_to_fr(p))

        results_all.extend(results_new)

    with open(final_output_path, "w") as json_file:
        json.dump(results_all, json_file, indent=2)
```

Рисунок 4.17 – Реалізація загального алгоритму збільшення розміру тексту

Насамкінець, розширений список зберігається у вказаний JSON-файл. Це і є кінцевий результат виконання алгоритму.

#### 4.7 Приклад вихідних даних

На рисунку 4.18 та рисунку 4.19 зображено фрагменти вихідного датасету.

"Los boletines de correo electrónico de suscripción voluntaria se utilizan comúnmente para crear una audiencia basada en permisos para ofertas y actualizaciones.",  
 "Les newsletters opt-in sont couramment utilisées pour constituer un public basé sur l'autorisation pour des offres et des mises à jour.",  
 "A typical goal of opt-in email campaigns is to assemble a list of subscribers who agree to receive marketing messages.",  
 "Un objectif typique des campagnes d'e-mails opt-in est de constituer une liste d'abonnés qui acceptent de recevoir des messages marketing.",  
 "Businesses mainly utilize opt-in email marketing to gather contacts interested in special offers and promotions.",  
 "Las empresas utilizan principalmente el marketing por correo electrónico de suscripción para reunir contactos interesados en ofertas y promociones especiales.",  
 "Les entreprises utilisent principalement le marketing par e-mail opt-in pour rassembler des contacts intéressés par des offres et des promotions spéciales.",  
 "Growing an audience that consents to receive promotional information is the main function of opt-in email marketing.",  
 "Crear una audiencia que acepte recibir información promocional es la función principal del marketing por correo electrónico de suscripción voluntaria.",  
 "Développer un public qui consent à recevoir des informations promotionnelles est la fonction principale du marketing par e-mail opt-in.",  
 "The most prevalent use of opt-in email marketing is compiling a list of users who welcome brand news and deals.",  
 "El uso más frecuente del marketing por correo electrónico de suscripción es recopilar una lista de usuarios que aceptan novedades y ofertas de la marca.",  
 "L'utilisation la plus courante du marketing par e-mail opt-in consiste à rassembler une liste d'utilisateurs qui acceptent les nouveautés et les offres de la marque.",  
 "Tracy McConnell is revealed as the mother in 'How I Met Your Mother' during the final season.",  
 "The show's concluding episodes disclose that the mother's name is Tracy McConnell.",  
 "Los episodios finales de la serie revelan que la madre se llama Tracy McConnell.",  
 "Les épisodes finaux de la série révèlent que la mère s'appelle Tracy McConnell.",  
 "In the end, viewers learn that Tracy McConnell is actually the mother figure in the story.",  
 "Al final, los espectadores descubren que Tracy McConnell es en realidad la madre de la historia.",  
 "À la fin, les téléspectateurs découvrent que Tracy McConnell est en fait la mère de l'histoire.",  
 "The mother, ultimately identified as Tracy McConnell, appears in the final storyline.",  
 "La madre, finalmente identificada como Tracy McConnell, aparece en la trama final.",  
 "La mère, finalement identifiée comme Tracy McConnell, apparaît dans l'intrigue finale.",  
 "The conclusion of 'How I Met Your Mother' reveals that the titular mother is Tracy McConnell.",  
 "In conclusion de 'Cómo conocí a vuestra madre' revela que la madre titular es Tracy McConnell.",  
 "La conclusion de 'How I Met Your Mother' révèle que la mère titulaire est Tracy McConnell.",  
 "Audiences discover in the final season that Tracy McConnell fulfills the role of the mother.",  
 "El público descubre en la última temporada que Tracy McConnell cumple el papel de la madre.",  
 "Le public découvre dans la dernière saison que Tracy McConnell endosse le rôle de la mère.",  
 "Humans primarily undergo internal fertilization, where the egg is fertilized within the female reproductive tract.",  
 "In people, fertilization typically happens inside the female body, known as internal fertilization.",  
 "En los seres humanos, la fecundación suele ocurrir dentro del cuerpo de la mujer, lo que se conoce como fecundación interna.",  
 "Chez les humains, la fécondation a généralement lieu à l'intérieur du corps de la femme, appelée fécondation interne.",  
 "Human reproduction involves fertilization occurring within the female reproductive system.",  
 "La reproducción humana implica la fertilización dentro del sistema reproductor femenino.",  
 "La reproduction humaine implique une fécondation à l'intérieur du système reproducteur féminin."

Рисунок 4.18 – Перший фрагмент вихідного датасету

"Historically, the Packers from Green Bay have secured the most regular-season triumphs in the NFL.",  
 "Históricamente, los Packers de Green Bay han asegurado la mayor cantidad de triunfos en temporada regular en la NFL.",  
 "Historiquement, les Packers de Green Bay ont remporté le plus grand nombre de victoires en saison régulière dans la NFL.",  
 "Green Bay's Packers hold the record for the most regular-season wins in the NFL all-time.",  
 "Los Packers de Green Bay poseen el récord de más victorias en temporada regular en la historia de la NFL.",  
 "Les Packers de Green Bay détiennent le record du plus grand nombre de victoires en saison régulière de la NFL.",  
 "The franchise with the greatest tally of NFL regular-season wins is the Green Bay Packers.",  
 "La franquicia con la mayor cantidad de victorias en temporada regular de la NFL es Green Bay Packers.",  
 "La franchise ayant le plus grand nombre de victoires en saison régulière de la NFL est les Green Bay Packers.",  
 "The Roanoke Colony mysteriously disappeared, with its inhabitants' fate remaining unknown after 1590.",  
 "All members of the Roanoke settlement vanished under unclear circumstances, and no one is certain of their end.",  
 "Todos los miembros del asentamiento de Roanoke desaparecieron en circunstancias poco claras, y nadie conoce su destino.",  
 "Tous les membres de la colonie de Roanoke ont disparu dans des circonstances obscures, et personne ne connaît leur sort."

Рисунок 4.19 – Другий фрагмент вихідного датасету

## ВИСНОВКИ

Запропонована методика показала здатність моделі генерувати якісні відповіді на різноманітні запити, що підтвердило досягнення поставленої мети зі створення масштабованого синтетичного датасету. Завдяки механізму відбору та перефразовування вдалось охопити значне коло тем і формулювань, а кількісні показники точності та розмаїтості свідчать про стабільне зростання якості відповідей після впровадження навчання з підкріпленням.

У порівнянні з відомими підходами на кшталт self-instruct [6] чи self-chat [7] запропонований метод успішно синтезує додатковий багатомовний контент, а також демонструє конкурентоспроможні результати у швидкості формування навчального корпусу. Водночас він добре інтегрується в існуючі розробки, розширюючи поле застосування машинного навчання та доповнюючи вітчизняні й світові дослідження у сфері обробки природної мови.

Поглиблення алгоритмів оцінювання корисності згенерованих відповідей, а також розширення мовного та тематичного охоплення залишаються важливими напрямками для подальшої роботи, оскільки вони здатні покращити адаптацію моделей до більш складних і багатомовних сценаріїв використання.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. Progress in neural NLP: modeling, learning, and reasoning / M. Zhou et al. *Engineering*. 2020. Vol. 6, no. 3. P. 275–290. URL: <https://doi.org/10.1016/j.eng.2019.12.014> (date of access: 02.12.2024).
2. Ryan O'Connor. RLHF vs RLAIIF for language model alignment. *AssemblyAI*. URL: <https://www.assemblyai.com/blog/rlhf-vs-rlaif-for-language-model-alignment/> (date of access: 02.12.2024).
3. Lee K. Open-Sourced training datasets for large language models (LLMs). *Kili*. URL: <https://kili-technology.com/large-language-models-llms/9-open-sourced-datasets-for-training-large-language-models#3> (date of access: 02.12.2024).
4. Cleaned alpaca dataset / G. Ruebsamen et al. *Github*. URL: <https://github.com/gururise/AlpacaDataCleaned> (date of access: 02.12.2024).
5. A survey of large language models / W. Xin Zhao et al. 2023. 140 p. (Preprint. arXiv:2303.18223). URL: <https://arxiv.org/abs/2303.18223> (date of access: 02.12.2024).
6. Self-instruct: aligning language models with self-generated instructions / Y. Wang et al. 2022. 23 p. (Preprint. arXiv:2212.10560). URL: <https://arxiv.org/abs/2212.10560> (date of access: 02.12.2024).
7. Baize: an open-source chat model with parameter-efficient tuning on self-chat data / C. Xu et al. 2023. 11 p. (Preprint. arXiv:2304.01196). URL: <https://arxiv.org/abs/2304.01196> (date of access: 02.12.2024).
8. Gokaslan A. Openwebtext. *Hugging Face*. URL: <https://huggingface.co/datasets/Skylion007/openwebtext> (date of access: 02.12.2024).
9. Natural questions / C. Gorgolewski et al. *Google AI*. URL: <https://ai.google.com/research/NaturalQuestions> (date of access: 02.12.2024).

10. Yang R. GPT2 large model. *Hugging Face*. URL: [https://huggingface.co/Ray2333/gpt2-large-helpful-reward\\_model](https://huggingface.co/Ray2333/gpt2-large-helpful-reward_model) (date of access: 02.12.2024).
11. Hanu L. Toxic Bert. *Hugging Face*. URL: <https://huggingface.co/unitary/toxic-bert> (date of access: 02.12.2024).
12. Language Tool Python / S. Myint et al. *PyPI - The Python Package Index*. URL: <https://pypi.org/project/language-tool-python/> (date of access: 02.12.2024).
13. Language tool / D. Naber et al. *LanguageTool*. URL: <https://languagetool.org/> (date of access: 02.12.2024).
14. Goutham R. Paraphrase any question with T5 (Text-To-Text Transfer Transformer) – Pretrained model and training script provided. *Towards Data Science*. URL: <https://towardsdatascience.com/paraphrase-any-question-with-t5-text-to-text-transfer-transformer-pretrained-model-and-cbb9e35f1555> (date of access: 02.12.2024).
15. Language technology in Helsinki. *University of Helsinki Official Website*. URL: <https://blogs.helsinki.fi/language-technology/> (date of access: 02.12.2024).
16. A survey on large language model (LLM) security and privacy: the good, the bad, and the ugly / Y. Yao et al. *High-confidence computing*. 2024. P. 100211. URL: <https://doi.org/10.1016/j.hcc.2024.100211> (date of access: 02.12.2024).
17. Harnessing the power of LLMs in practice: a survey on ChatGPT and beyond / J. Yang et al. 2023. 24 p. (Preprint. arXiv:2304.13712v2). URL: <https://arxiv.org/abs/2304.13712> (date of access: 02.12.2024).
18. Attention is all you need / A. Vaswani et al. 2017. 15 p. (Preprint. arXiv:1706.03762). URL: <https://arxiv.org/abs/1706.03762> (date of access: 02.12.2024).

19. Masked language model scoring / J. Salazar et al. 2019. 14 p. (Preprint. arXiv:1910.14659). URL: <https://arxiv.org/abs/1910.14659> (date of access: 02.12.2024).
20. Navigating LLM ethics: advancements, challenges, and future directions / J. Jiao et al. 2024. 35 p. (Preprint. arXiv:2406.18841). URL: <https://arxiv.org/abs/2406.18841> (date of access: 02.12.2024).
21. GPT-4. *OpenAI*. URL: <https://openai.com/index/gpt-4/> (date of access: 02.12.2024).
22. Edwards B. OpenAI's GPT-4 exhibits «human-level performance» on professional benchmarks. *Generative Pre-Trained Transformer News*. URL: <https://arstechnica.com/information-technology/2023/03/openai-announces-gpt-4-its-next-generation-ai-language-model/> (date of access: 02.12.2024).
23. Cheng H.-T. LaMDA: towards safe, grounded, and high-quality dialog models for everything. *Google Research*. URL: <https://research.google/blog/lamda-towards-safe-grounded-and-high-quality-dialog-models-for-everything/> (date of access: 02.12.2024).
24. Claude. *Claude AI*. URL: <https://claude.ai/> (date of access: 02.12.2024).
25. Belcic I., Stryker C. What is Claude AI?. *IBM US*. URL: <https://www.ibm.com/think/topics/claude-ai> (date of access: 02.12.2024).
26. Sutton R. S., Barto A. G. Reinforcement learning: an introduction. *IEEE transactions on neural networks*. 1998. Vol. 9, no. 5. P. 1054. URL: <https://doi.org/10.1109/tnn.1998.712192> (date of access: 17.01.2025).
27. Whittle P., Puterman M. L. Markov decision processes. *Journal of the royal statistical society. Series A (statistics in society)*. 1995. Vol. 158, no. 3. P. 636. URL: <https://doi.org/10.2307/2983459> (date of access: 17.01.2025).
28. GPT-4 technical report / J. Achiam et al. 2023. 100 p. (Preprint. arXiv:2303.08774). URL: <https://arxiv.org/abs/2303.08774> (date of access: 02.12.2024).

29. Claude's constitution. *Anthropic News*. URL: <https://www.anthropic.com/news/claudes-constitution> (date of access: 02.12.2024).
30. Eknath, Prof. Upasani Dhananjay, 1st, Shete, Prof. Virendra Virbhadra, 1st. Machine learning with Python: machine learning with Python. INSC International Publisher (IIP), 2021.
31. JetBrains. PyCharm: the Python IDE for data science and web development. *JetBrains*. URL: <https://www.jetbrains.com/pycharm/> (date of access: 02.12.2024).
32. WebText dataset / A. Radford et al. *Papers With Code*. URL: <https://paperswithcode.com/dataset/webtext> (date of access: 02.12.2024).
33. Language models are unsupervised multitask learners / A. Radford et al. 2019. 24 p. (Preprint). URL: [https://cdn.openai.com/better-language-models/language\\_models\\_are\\_unsupervised\\_multitask\\_learners.pdf](https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf) (date of access: 02.12.2024).
34. Proximal policy optimization algorithms / J. Schulman et al. 2017. 12 p. (Preprint. arXiv:1707.06347). URL: <https://arxiv.org/abs/1707.06347> (date of access: 02.12.2024).
35. Iyer S., Dandekar N., Csernai K. First Quora dataset release: Question Pairs. *Quora Data*. URL: <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs> (date of access: 02.12.2024).
36. University of Helsinki. OPUS English to Spanish translation model. *Hugging Face*. URL: <https://huggingface.co/Helsinki-NLP/opus-mt-en-es> (date of access: 02.12.2024).
37. University of Helsinki. OPUS English to French translation model. *Hugging Face*. URL: <https://huggingface.co/Helsinki-NLP/opus-mt-en-fr> (date of access: 02.12.2024).