Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Метод синтезу навчальної вибірки для трансформер моделей на базі навчання з підкріпленням
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-5

Олександр Крамаренко
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник доц. Олександр Шевченко
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Крамаренку Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод синтезу навчальної вибірки для трансформер моделей на базі навчання з підкріпленням

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 червня 20 25 р.

3. Вихідні дані до роботи Дані інтернет-джерел, середовище розробки «JetBrains PyCharm», науково-технічні публікації, мовні моделі, власноруч підготовлений корпус для навчання, набір запитів до браузера «NaturalQuestions», Python-бібліотеки

4. Перелік питань, що потрібно опрацювати в роботі
1) Опис предметної галузі та теоретичні основи
2) Аналіз аналогів та існуючих рішень
3) Проєктування програмного забезпечення
4) Реалізація застосунку
5) Експериментальний аналіз

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.02.2025	виконано
2	Аналіз предметної галузі та постановка задачі	20.02.2025 – 01.03.2025	виконано
3	Огляд методів генерації синтетичних даних і	02.03.2025 – 10.03.2025	виконано
	навчання з підкріпленням		
4	Проєктування архітектури системи синтезу	11.03.2025 – 01.04.2025	виконано
	навчальної вибірки		
5	Реалізація багатомодельного генератора з	02.04.2025 – 07.04.2025	виконано
	гібридною стратегією		
6	Розробка reward-функцій та механізму фільтрації	07.04.2025 – 09.04.2025	виконано
7	Інтеграція PPO-агента для оптимізації генерації	10.04.2025 – 16.04.2025	виконано
8	Проведення експериментів та візуалізації	17.04.2025 – 18.04.2025	виконано
9	Оцінювання якості синтезованої вибірки	19.04.2025 – 24.04.2025	виконано
10	Порівняння з існуючими підходами	24.04.2025 – 28.04.2025	виконано
11	Формулювання висновків і напрямів подальшого	29.04.2025 – 06.05.2025	виконано
	розвитку		
12	Написання записки: теоретична частина	07.05.2025 – 19.05.2025	виконано
13	Написання записки: опис програмного	20.05.2025 – 25.05.2025	виконано
	застосунку		
14	Підготовка до нормоконтролю	26.05.2025 – 06.06.2025	виконано
15	Захист перед ЕК	17.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олександр Шевченко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 129 с., 38 рис., 3 дод., 28 джерел.

ГЕНЕРАЦІЯ СИНТЕТИЧНИХ ДАНИХ, МАШИННЕ НАВЧАННЯ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, СЕМАНТИЧНА НОВИЗНА, ТЕМАТИЧНА СЛОЖНІСТЬ, ТРАНСФОРМЕР МОДЕЛІ, GPT-2, PPO, REWARD.

Об’єкт дослідження – процес побудови навчальної вибірки для трансформерних моделей у задачах генерації тексту.

Предмет дослідження – підхід до генерації синтетичних текстових даних на основі навчання з підкріпленням із використанням генераторів і модулів оцінювання якості.

Мета роботи – розробити метод синтезу навчальної вибірки, здатний підсилити навчання трансформерних моделей шляхом відбору релевантних і змістовно прикладів, згенерованих на основі reward-механізму.

Методи дослідження – алгоритми навчання з підкріпленням, аналіз метрик, використання генеративних мовних моделей, оцінювання за допомогою моделей схожості, новизни, довжини й стилізації.

У роботі представлено застосунок для побудови навчальної вибірки на текстових даних, що генеруються із залученням мовних моделей. Основна увага приділена алгоритму PPO, який навчає агент обирати найінформативніші приклади відповідно до reward-функції, що поєднує новизну, тематичну відповідність і структурну адекватність текстів. У результаті реалізовано систему, здатну формувати вибірку з високими показниками варіативності та релевантності. Проведено експериментальний аналіз, побудовано метрики й графіки, які підтверджують ефективність підходу в контексті генерації тренувальних даних для трансформерних моделей.

ABSTRACT

Bachelor’s thesis contains: 129 pp., 38 fig., 3 ann., 28 references.

GPT-2, MACHINE LEARNING, PPO, REINFORCEMENT LEARNING, REWARD MODELS, SEMANTIC NOVELTY, SYNTHETIC DATA, TRANSFORMER MODELS GENERATION, THEMATIC COMPLEXITY.

The object of research is the process of building a training set for transformational models in text generation tasks.

The subject of the study is an approach to generating synthetic text data based on reinforcement learning using generators and quality assessment modules.

Purpose – to develop a method for synthesizing a training set capable of enhancing the training of transformational models by selecting relevant and meaningful examples generated on the basis of a reward mechanism.

The research methods include reinforcement learning algorithms, metrics analysis, the use of generative language models, and evaluation using similarity, novelty, length, and stylization models.The paper presents an application for building a training set on text data generated using language models. The main focus is on the PPO algorithm, which trains the agent to select the most informative examples according to a reward function that combines novelty, thematic relevance, and structural adequacy of the texts. As a result, a system capable of generating a sample with high variability and relevance is implemented. An experimental analysis was carried out, metrics and graphs were constructed, which confirm the effectiveness of the approach in the context of generating training data for transformational models.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 8
Вступ 9
1 Опис предметної галузі та теоретичні основи 11
1.1 Огляд трансформер-моделей у сучасному NLP 13
1.1.1 Архітектура трансформера 15
1.1.2 Моделі GPT-2, T5, DistilGPT та їх роль у генерації тексту 17
1.2 Основи навчання з підкріпленням 19
1.2.1 Класичне RL: агенти, середовище, винагорода 22
1.2.2 Метод PPO: переваги та використання в NLP 25
1.3 Синтез навчальних вибірок для мовних моделей 26
1.3.1 Підходи Self-Instruct та Self-Chat 28
1.3.2 Проблеми якості синтетичних даних 30
2 Аналіз аналогів та існуючих рішень 32
2.1 Інструменти генерації даних 33
2.2 Методи фільтрації, переоцінки та відбору відповідей 35
2.3 Використання reward-моделей у відкритих проектах 37
3 Проєктування програмного забезпечення 40
3.1 Архітектура системи: модулі генерації, оцінювання та навчання 40
3.2 Опис reward-функцій 41
3.2.1 Оцінка тематичної подібності 43
3.2.2 Новизна (Jaccard, embedding) 44
3.2.3 Міра довжини та корисності 46
3.3 Вибір та інтеграція генераторів 47
3.4 Формування запитів та початкового набору даних 49
3.5 Обґрунтування середовища реалізації 51
4 Реалізація застосунку 53
4.1 Файлова структура та головні модулі 54
4.2 Реалізація PPO-агента для управління генерацією 55

4.3 Механізм обчислення винагород 57
4.4 Модуль логування та збереження результатів 59
4.5 Модуль аналізу метрик та візуалізація 60
4.6 Приклад сесії генерації та навчання 63
4.7 Репрезентативні фрагменти реалізації розробленої системи 64
5 Експериментальний аналіз 69
5.1 Аналіз метрик за різними генераторами 70
5.2 Динаміка винагород та новизни протягом епізодів 71
5.3 Порівняння результатів з базовими моделями 73
5.4 Висновки щодо ефективності запропонованого методу 75
Висновки 77
Перелік джерел посилання 79
Додаток А Вибірка синтетично згенерованих відповідей 83
Додаток Б Код програми 90
Додаток В Відомість кваліфікаційної роботи 129

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

GPT-2 – генеративна переднавчена трансформерна модель, розроблена компанією OpenAI;

LLM – Large Language Model – велика мовна модель, нейронна мережа з великим числом параметрів;

NLP – Natural Language Processing – обробка природньої мови;

Novelty score – метрика новизни, яка вимірює відмінність новоствореного тексту від попередніх прикладів;

PPO – Proximal Policy Optimization – метод навчання з підкріпленням, що забезпечує стабільну оптимізацію політики агента;

Reward – зважене числове значення оцінки якості згенерованої відповіді;

RL – Reinforcement Learning – навчання з підкріпленням;

Similarity score – метрика, що оцінює тематичну або семантичну схожість згенерованого тексту;

Total reward – сумарна нагорода, обчислена як комбінація кількох метрик;

T5 – Text-to-Text Transfer Transformer – трансформерна модель, що формулює всі NLP-задачі як задачі перетворення тексту в текст.

ВСТУП

За останнє десятиліття відбувся якісний стрибок у розвитку методів обробки природної мови, що виник через широке впровадження трансформерних моделей. З появою BERT, GPT, T5 та їхніх різноманітних варіацій, розпочалась нова ера у Natural Language Processing (NLP). У цій ері моделі вміють не тільки узагальнювати відомості з великих масивів текстів, але й творчо створювати нові висловлювання, зважаючи на стилістичні, семантичні та прагматичні обмеження. Водночас, незважаючи на вражаючу потужність таких моделей, способи їхнього донавчання, пристосування до нових доменів та створення навчальних даних залишаються ключовими проблемами.

Особливо актуальною стає потреба у створенні якісних навчальних даних без залучення людей – як через значну вартість ручного анотування, так і через потенційні ризики упередженості. Відповіддю на це стали способи автоматичного синтезу даних, зокрема ті, що базуються на принципах навчання з підкріпленням (RL). Найбільшого розповсюдження останнім часом набув підхід RLHF (Reinforcement Learning with Human Feedback), що дає мовній моделі можливість пристосуватися до очікувань користувача за допомогою системи винагород. У цьому дослідженні пропонується його альтернатива – автоматизована архітектура генерації та фільтрації даних з використанням навчання з підкріпленням без залучення людини, де роль «вчителя» виконують метрики якості.

Метою цього дослідження є створення та експериментальна оцінка гібридної системи синтезу навчальної вибірки для трансформерних моделей, що поєднує різні джерела генерації тексту (GPT-2, T5, стилістичні й парафразні моделі) зі стратегією адаптивного відбору відповідей. Оцінка якості спирається на тематичну відповідність, новизну, довжину та семантичну близькість до вихідного запиту. Головним компонентом є агент

PPO, який навчається вибирати найкращі відповіді на основі нагород, згенерованих reward-модулем.

У процесі реалізації створено програмний додаток, який складається з модулів генерації, критики, винагородження, логування та аналізу. Особлива увага приділена візуалізації динаміки навчання агента, що дає змогу зробити висновки про ефективність застосованої стратегії синтезу. Варто відзначити, що система зберігає всі кроки у форматі JSONL, що забезпечує прозорість і відтворюваність експерименту. Отже, представлена робота не тільки формулює новий метод побудови навчальних вибірок для трансформерних моделей, але й представляє практичне рішення з відкритою логікою роботи, яке може бути використане для широкого спектру завдань у сфері генерації тексту, донавчання моделей та автоматизації побудови корпусів.

1 ОПИС ПРЕДМЕТНОЇ ГАЛУЗІ ТА ТЕОРЕТИЧНІ ОСНОВИ

Сучасна обробка природної мови перебуває в стані активного переформатування завдяки широкому поширенню великих мовних моделей, які базуються на архітектурі трансформера. Починаючи з BERT і GPT, трансформери стали основним технічним каркасом, на якому будується нове покоління мовних систем – від генераторів до діалогових агентів. Такі моделі навчаються на масштабних корпусах текстів, опановуючи складну структуру мови завдяки механізмам самоуваги, позиційного кодування й багатошарової обробки.

Сучасна обробка природної мови зазнає кардинальних змін через повсюдне поширення великих мовних моделей (LLM), основою яких є архітектура трансформера. Починаючи з BERT та GPT, трансформери стали ключовою технічною платформою, що лежить в основі нового покоління мовних систем, від генераторів тексту до інтерактивних агентів. Ці моделі навчаються на величезних обсягах текстових даних, засвоюючи складну структуру мови завдяки механізмам самоуваги, позиційного кодування та багатошарової обробки.

Трансформерні моделі дозволяють не лише аналізувати й інтерпретувати текст, а й творчо генерувати його в різних стилях, доменах та мовах. Ця здатність зумовила широке впровадження LLM у завдання створення штучних даних, де модель виступає не як аналітичний інструмент, а як автономний генератор корпусів, інструкцій, діалогів або запитань. У рамках цієї роботи особливу роль відіграють генератори, такі як GPT-2, T5, а також моделі, що спеціалізуються на стилістичних або перефразових завданнях [27].

Питання синтезу навчальної вибірки тісно пов'язане з якістю автоматично згенерованого тексту. Для забезпечення релевантності та новизни даних, у системах генерації дедалі ширше використовуються методи навчання з підкріпленням. Значну популярність здобув алгоритм

Proximal Policy Optimization (PPO), який продемонстрував високу ефективність в управлінні агентом за умов обмеженого зворотного зв'язку та великої варіативності дій [9]. Підхід RLHF (Reinforcement Learning with Human Feedback) довів свою ефективність у налаштуванні поведінки мовної моделі, але вимагає значного обсягу людських оцінок. Тому в цьому дослідженні розглядається безлюдний варіант цього методу, де функцію винагороди виконує система автоматичних критеріїв.

До цих критеріїв належать метрики семантичної схожості, тематичної релевантності, новизни й лексичної складності, кожна з яких виконує роль критика, що оцінює відповідь з певного боку. Ці багатовимірні оцінки формують основу функції винагороди, яка адаптивно впливає на політику агента PPO. У цьому сенсі модель не просто генерує текст – вона вчиться покращувати його, отримуючи винагороду за кожен вдалий приклад [10].

Невід'ємною частиною сучасного NLP є також питання адаптації LLM до нових сценаріїв без повного перенавчання. Тут важливими стають методи інструкційного донавчання (instruction tuning) та параметрично-ефективні техніки, як-от LoRA або fine-tuning на інтерфейсних даних. Навчання таких систем потребує великих і різноманітних корпусів прикладів, і саме автоматизований підхід до їх створення є метою даної роботи.

Окрему увагу варто приділити проблемам, що виникають при використанні LLM: це питання надійності, упередженості вибірок, генерація токсичного або недоречного контенту, а також загрози безпеці – як технічній, так і етичній. Тому синтезована навчальна вибірка має проходити критичний фільтр, а система її генерації – бути прозорою, контрольованою та добре задокументованою. Крім цього, необхідно забезпечити верифікацію джерел даних та змістовий контроль згенерованих прикладів, аби уникнути підсилення хибних наративів або шкідливих стереотипів. Для цього використовуються як автоматизовані, так і ручні методи модерації, що забезпечують відповідність синтетичних даних

етичним та прикладним стандартам. Архітектура трансформерної моделі преставлена на рисунку 1.1.

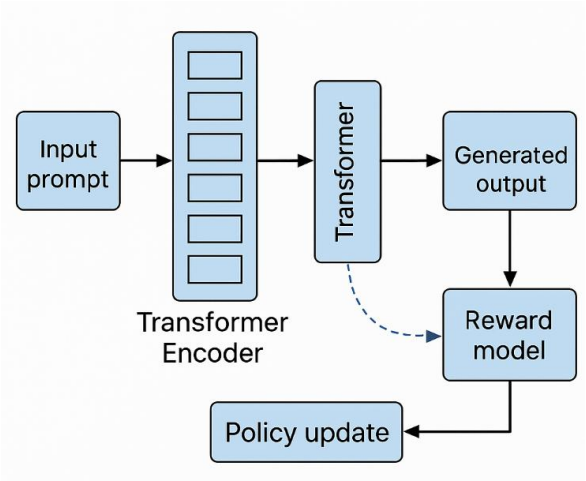

Рисунок 1.1 – Архітектура трансформер-моделі з RL-компонентом у генерації даних

Таким чином, галузь дослідження охоплює перетин глибинного навчання, навчання з підкріпленням та автоматичного синтезу даних. З одного боку, вона вкорінена в теоретичних принципах оптимізації політик і оцінки винагород, з іншого – покликана вирішити практичну проблему побудови якісних корпусів для донавчання трансформерів. Актуальність цього напряму підтверджується як зростанням кількості публікацій, так і практикою використання LLM у широкому спектрі завдань – від патентного пошуку до юридичних аналізаторів [15].

1.1 Огляд трансформер-моделей у сучасному NLP

Відколи у 2017 році команди Google оприлюднили архітектуру трансформера, ця модель показала революційні у підходи до розв'язання мовних проблем. Трансформери скинули рекурентні мережі з їхніх чільних позицій, як ключову архітектуру, демонструючи здатність обробляти

контекст, охоплюючи весь вхідний текст, без проблем із залежностями у послідовностях. Ключовим нововведенням був механізм self-attention, який надав моделі можливість динамічно визначати важливість кожного слова в контексті, незалежно від його розташування у вхідній послідовності, як показує приклад трансформера з механізмом self-attention на рисунку 1.2.

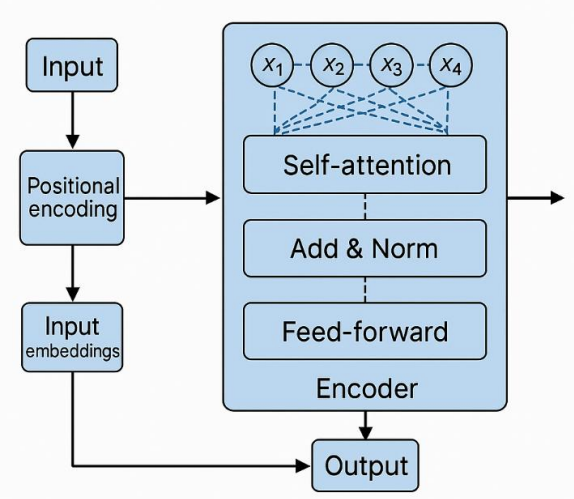

Рисунок 1.2 – Загальна архітектура трансформера з прикладом механізму self-attention

На базі цієї архітектури постав клас мовних моделей, як-от GPT-2, T5 та DistilGPT. Вони різняться цілями навчання, розміром і задачами, проте всі вони ґрунтуються на трансформерах. GPT-2 – авторегресійна модель, яка добре генерує продовження тексту, забезпечуючи високу якість і логічну послідовність згенерованого контенту. T5, навпаки, розглядає будь-які NLP-завдання як завдання генерування тексту – від перекладу до класифікації, що робить її дуже гнучкою для навчання через інструкції [27]. DistilGPT – спрощена версія GPT-2, зі зменшеною кількістю параметрів, але зі збереженням якості, що дозволяє проводити експерименти з меншими обчислювальними витратами.

Сьогодні ці моделі застосовуються не тільки як інструменти для генерації, але й як база для подальшого донавчання із залученням

зворотного зв'язку від людей або автоматичних систем оцінювання. Багато підходів, такі як Self-Instruct, Self-Chat та RLHF, базуються на цих трансформерах, використовуючи їх як фундамент для додаткового адаптивного навчання.

Сила трансформерів також у їхній здатності масштабуватися до великих мовних моделей (LLM), які демонструють так звані emergent abilities – появу нових когнітивних властивостей зі збільшенням кількості параметрів. Це відкриває нові перспективи в обробці мови, де модель здатна виконувати інструкції, узагальнювати знання та адаптуватися до завдань, яким її прямо не навчали.

Разом з цим, широке використання трансформерів створило нові виклики: потреба у великих навчальних даних, високі обчислювальні витрати та ризик відтворення упереджень або токсичних патернів. Тому значна частина сучасних досліджень зосереджена на розробці безпечних, етичних та інтерпретованих LLM, що базуються на трансформерах [12].

Огляд трансформерних моделей свідчить про глибокі зміни в NLP: від ієрархічних алгоритмів і граматичних правил до універсальних генеративних систем. Ці моделі – не просто інструменти, вони функціонують як адаптивні агенти, здатні до інтерпретації, перенавчання та самостійної побудови контексту. Наступний розділ присвячений основам їх навчання зі зворотним зв’язком – навчанню з підкріпленням, що закладає основу для адаптивної генерації якісних даних.

1.1.1 Архітектура трансформера

Архітектура трансформера стала наріжним каменем еволюції обробки природної мови, проклавши дорогу до моделей, здатних генерувати, перекладати, узагальнювати та відповідати на запити з вражаючою точністю. Запропонована Vaswani et al. у 2017 році, ця модель здійснила перехід від рекурентного до повністю уважного підходу, що дозволило

значно прискорити навчання та масштабування. На відміну від попередників, як-от LSTM чи GRU, трансформер працює з усією послідовністю одночасно, застосовуючи механізм self-attention – тобто увагу до кожного слова в контексті всіх інших.

Фундаментом трансформера є два симетричні блоки: енкодер і декодер. Кожен з них складений із послідовних шарів, що включають механізм багато-головної уваги (multi-head attention), нормалізацію, резидуальні з'єднання та feed-forward-мережі. Механізм багато-головної уваги дозволяє моделі одночасно враховувати декілька типів контекстних залежностей, тоді як позиційне кодування гарантує врахування порядку слів, що відсутній у самій трансформерній структурі. Це робить модель ефективною не лише при обробці лінійних послідовностей, але й у задачах із довготривалими залежностями. Приклад архітектури трансформера з енкодером та декодером показано на рисунку 1.3.

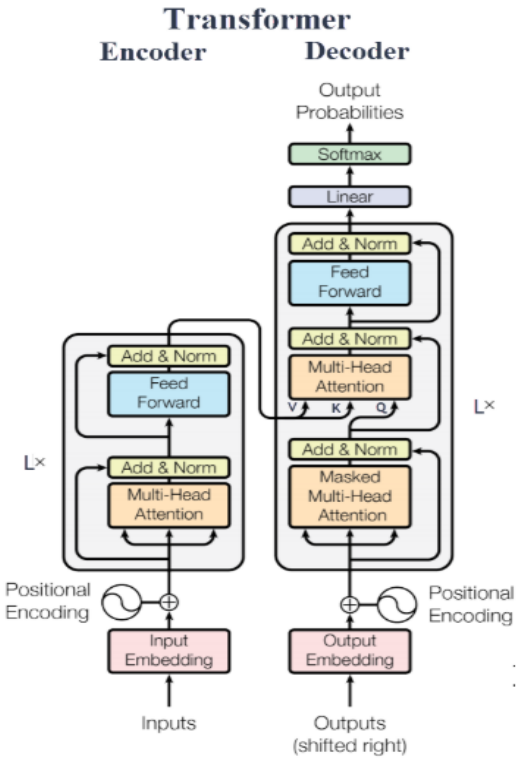

Рисунок 1.3 – Узагальнена архітектура трансформера з енкодером і декодером

У сучасних моделях на базі трансформера часто застосовується тільки частина повної архітектури. Наприклад, у GPT-серії використовується лише декодерна частина, адаптована для автоагресивного прогнозування, тоді як у моделі T5 поєднані енкодер і декодер, щоб забезпечити гнучкість для завдань типу «текст у текст». Така варіативність дозволяє проектувати як потужні генеративні, так і дискримінативні моделі, зберігаючи при цьому архітектурну єдність і масштабованість.

Перевага трансформера полягає також у його високій паралелізованості, що значно зменшує час навчання на великих корпусах. Саме завдяки цій властивості стало можливим навчання моделей із десятками або сотнями мільярдів параметрів, як-от GPT-3, GPT-4, PaLM, що базуються на аналогічних принципах побудови. Розуміння архітектурних основ трансформера є критичним для розробки методів генерації синтетичних вибірок, оскільки визначає межі продуктивності, здатність до узагальнення та контрольовану генерацію контенту.

1.1.2 Моделі GPT-2, T5, DistilGPT та їх роль у генерації тексту

Розвиток архітектури трансформаторів сприяв народженню численних моделей, які демонструють визначні результати у генерації тексту. З-поміж найвідоміших та водночас найвпливовіших – GPT-2, T5 і DistilGPT. Кожна з них відіграє важливу роль у розбудові систем, здатних продукувати текст з високою граматичною, стилістичною та змістовною відповідністю очікуванням користувача чи системи.

GPT-2, представлена дослідниками з OpenAI, є глибоко авторегресивною моделлю, навченою на масивному корпусі відкритих текстів. Її перевага полягає у здатності формувати продовження тексту, зберігаючи логіку навіть на значних ділянках. Завдяки цьому GPT-2 широко застосовується у системах діалогових агентів, генерації статей та сценаріїв,

а також як основа для моделей з людським зворотним зв’язком, зокрема у підході Reinforcement Learning with Human Feedback (RLHF) [23].

T5 (Text-To-Text Transfer Transformer), запропонована Google Research, перетворює будь-яке NLP-завдання у формат генерації: «переклади англійською», «класифікуй емоцію», «відповідай на запитання». Така уніфікація дала змогу досягти високої узагальнюваності та гнучкості моделі. T5 ефективна не лише в генерації зв’язного тексту, а й у створенні парафраз, формалізованих формулювань та інструкційно орієнтованих відповідей.

DistilGPT, як полегшена версія GPT-2, стала відповіддю на виклики, зумовлені обчислювальними витратами. Ця модель використовує метод знаннєвої дистиляції (knowledge distillation), що дозволяє зберігати продуктивність при зменшенні кількості параметрів. DistilGPT активно використовується в експериментах, які не вимагають потужної інфраструктури, або ж у мобільних системах та пристроях з обмеженими ресурсами. Також варто глянути графіки порівняння для моделей GPT-2, T5, DistilGPT, які зображено на рисунку 1.4.

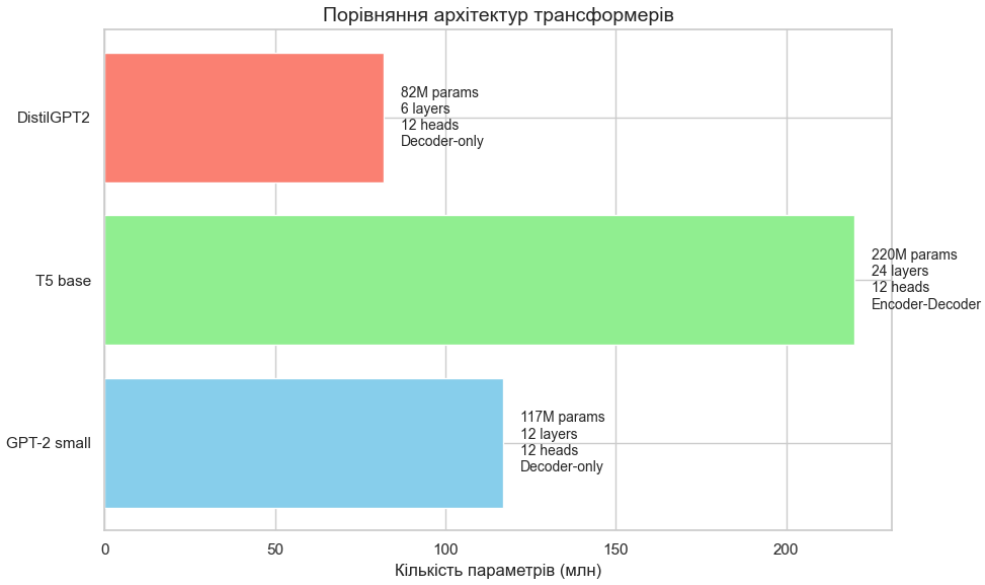

Рисунок 1.4 – Порівняння архітектур та розмірів моделей GPT-2, T5, DistilGPT

Кожна з цих моделей має свої особливості, які визначають доцільність в конкретному контексті. Приміром, GPT-2 краще адаптується до завдань, де важлива довготривала когерентність, тоді як T5 – до сценаріїв з інструкціями та багатозадачністю. DistilGPT, у свою чергу, є ідеальним інструментом для швидкого прототипування. У контексті синтезу навчальної вибірки для трансформерних моделей особливу вагу мають їхні здібності до стилізації тексту, переформулювання запитів та створення варіативних відповідей – саме ці властивості стають фундаментом для побудови гібридних генеративних систем

Ці моделі також активно застосовуються як початкові блоки у складніших системах. Наприклад, у підходах Self-Instruct і Self-Chat, саме T5 або GPT-2 використовуються як генератори інструкцій або діалогів, які потім відфільтровуються за критеріями змістовності, стилістичної відповідності чи новизни [2]. Отже, ці моделі не лише продукують текст, а й створюють основу для подальшого навчання, перенавчання й оцінки мовних систем.

1.2 Основи навчання з підкріпленням

Навчання з підкріпленням (reinforcement learning, RL) – це підхід у машинному навчанні, що імітує процес здобуття знань агентом через його взаємодію із середовищем, досвід і винагороди. На відміну від звичного навчання з вчителем, де модель має приклади з готовими рішеннями, у навчанні з підкріпленням агент сам визначає, які дії ведуть до бажаного результату. Такий метод більш співзвучний з природним процесом навчання, як-от, навчання ходьби у дитини, де вона уникає неприємних наслідків падінь і заохочується позитивним досвідом.

У традиційному розумінні RL включає в себе три основні складові: агент, середовище та система заохочень. Агент – це умовний «суб'єкт», що приймає рішення. Середовище – це простір, де він функціонує. Взаємодія

між ними відбувається через дії: агент отримує інформацію про поточний стан середовища, обирає дію, виконує її – і середовище реагує. У відповідь на це агент одержує винагороду – числове значення, що відображає корисність або успішність його дії. На основі цього сигналу агент коригує свою поведінку, щоб досягти кращих результатів в майбутньому. Схематично ця взаємодія показана на рисунку 1.5.

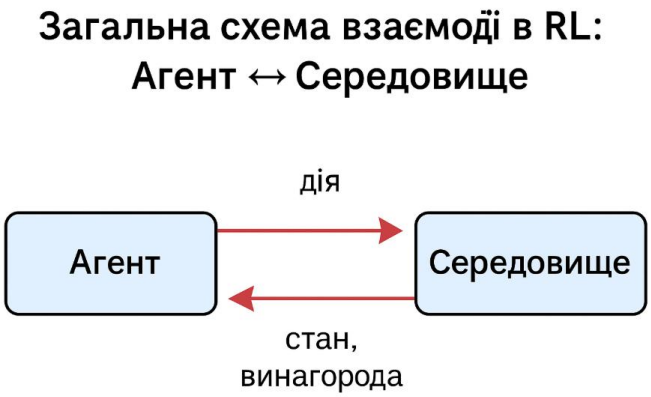

Рисунок 1.5 – Загальна схема RL – агент, середовище, дія, винагорода

Суть навчання з підкріпленням (НЗП) полягає в тому, щоб сформувати у агента модель поведінки, що забезпечує максимальний накопичувальний результат, а не просто поточну перемогу. Це означає, що процес навчання передбачає не лише реагування на поточну ситуацію, а й планування на майбутнє, враховуючи, які дії допоможуть досягти довгострокової мети. Саме це відрізняє НЗП від, наприклад, звичайного класифікатора, що працює в рамках одноразового прогнозу.

Існують різні стратегії реалізації НЗП: деякі підходи базуються на оцінці користі від перебування в певному стані (так звані функції цінності), інші – безпосередньо шукають оптимальні правила поведінки (політики). Незалежно від обраного підходу, ефективність НЗП залежить від збалансованого поєднання дослідження нових дій (експлорації) та використання вже наявного досвіду (експлуатації). Надмірна експлорація

може призвести до марнотратства ресурсів, а надмірна експлуатація – до застою на субоптимальних рішеннях.

Цей підхід показав свою придатність у складних динамічних задачах, де немає чітко визначеного «правильного» результату, або ж він визначається непрямо – як у генерації тексту, діалогових системах або на платформах рекомендацій. В таких сценаріях агент має пристосовуватися до змінних умов, поступово покращуючи свою поведінку через взаємодію з користувачем або середовищем. Як свідчать сучасні дослідження [10], НЗП стало основою нової хвилі прогресу в машинному навчанні, зокрема у сфері великих мовних моделей (LLM), які навчаються відповідати на запити не за заданими прикладами, а шляхом вдосконалення своїх відповідей через механізми зворотного зв’язку, приклад зображено на рисунку 1.6.

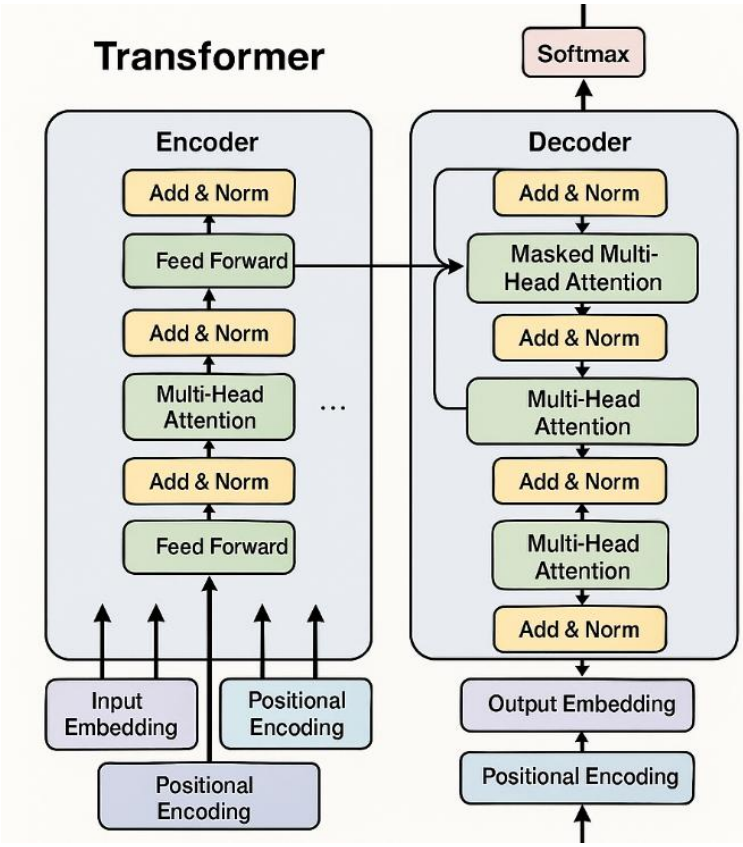

Рисунок 1.6 – Застосування RL у сучасному NLP (схематично: запит → модель → відповідь → оцінка → оновлення)

RL, по суті, є містком між предиктивним і адаптивним ШІ. З одного боку, він оперує статистичними оцінками, а з іншого – дозволяє моделям вчитися без прямої вказівки, що саме є «правильним». Завдяки цьому RL є основою багатьох інновацій, зокрема в контексті навчання мовних моделей з людським зворотним зв'язком (RLHF), про що детальніше йтиметься у наступному підрозділі.

1.2.1 Класичне RL: агенти, середовище, винагорода

У класичному навчанні з підкріпленням основою є ідея, що інтелектуальна система може самостійно навчитися, взаємодіючи з навколишнім середовищем. У цьому підході система (агент) не має повного знання про правила середовища чи наслідки власних дій, але поступово формує ефективну стратегію поведінки на основі винагород, які отримує внаслідок своїх дій. Ключовими елементами RL-системи є:

– агент – це програмна сутність, яка приймає рішення. Він діє в середовищі, керуючись певною стратегією, що визначає, яку дію слід виконати в кожному конкретному стані;

– середовище це умовний світ, з яким агент взаємодіє. Воно визначає, які наслідки має дія агента: як змінюється стан, яку винагороду він отримає;

– стан (state) – інформація про поточну ситуацію, яку агент може використовувати для прийняття рішення;

– дія (action) – вибір, який здійснює агент у відповідь на поточний стан;

– винагорода (reward) – числовий сигнал, що вказує, наскільки корисною була дія агента для досягнення мети;

– політика (policy) – правило, за яким агент обирає дії, ґрунтуючись на стані;

– ціль (return) – бажане накопичене значення винагород у довгостроковій перспективі.

Взаємодія елементів RL-системи показана на рисунку 1.7.

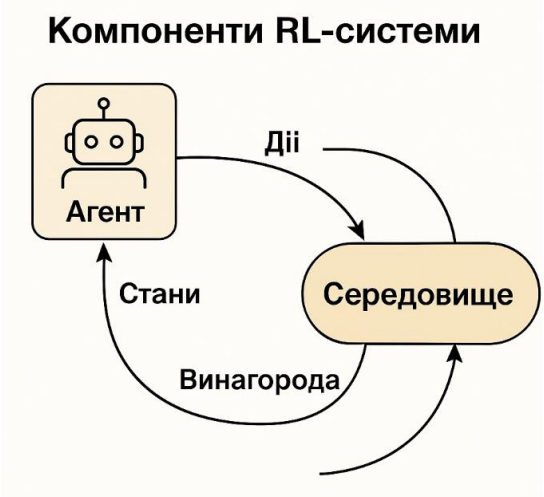

Рисунок 1.7 – Компоненти RL-системи – агент, середовище, дії, стани, винагорода

Типова взаємодія виглядає так: агент стежить за станом навколишнього середовища, визначає дію, отримує винагороду, після чого середовище переходить у новий стан. Цей цикл повторюється неодноразово, і на основі набутого досвіду агент вдосконалює свою стратегію поведінки. На практиці середовище може бути як симуляцією (наприклад, віртуальна гра), так і реальним світом – зокрема, діалоговою системою, з якою взаємодіє користувач [3].

Окреме значення має баланс між дослідженням (exploration) та використанням (exploitation). З одного боку, агенту необхідно вивчати дії, щоб віднайти найоптимальніші з них. З іншого – важливо застосовувати вже відомі ефективні дії, щоб отримувати стабільну винагороду. У разі порушення цього балансу, навчання може бути або надто повільним, або неефективним. Цей аспект особливо критичний у задачах генерації природної мови, де вибір неточної відповіді може призвести до спотворення стилю або змісту.

Методи класичного RL історично використовувалися для розв'язання задач керування, гри в шахи, автономної навігації, але з розвитком глибинного навчання вони отримали нове життя. Виник напрямок deep reinforcement learning, який дозволяє поєднувати класичні механізми винагород із глибинними нейронними мережами для прийняття складних рішень у високорозмірних просторах станів [4]. Схема циклу взаємодії в RL зображена на рисунку 1.8.

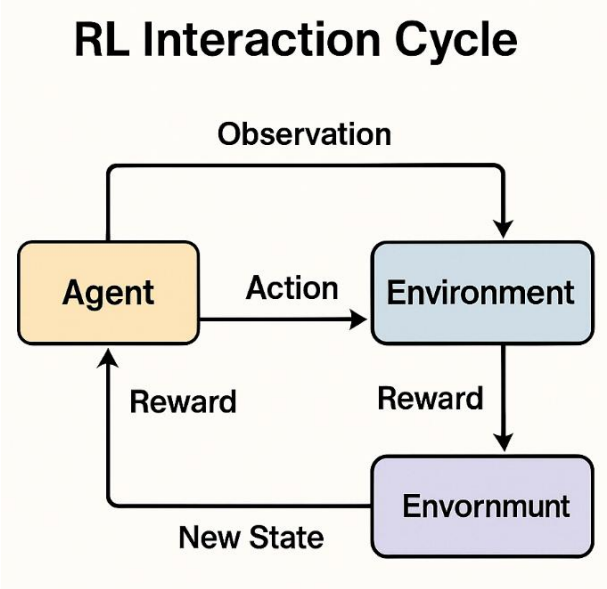

Рисунок 1.8 – Схема циклу взаємодії в RL: спостереження → дія → винагорода → новий стан

У контексті генерації тексту, RL застосовується як механізм уточнення вихідної відповіді, що вже згенерована мовною моделлю. У такому випадку середовищем є сам текстовий простір або користувацька взаємодія, а винагородою – оцінка релевантності, стилістичної точності або відповідності інструкції. Завдяки гнучкості RL підхід дозволяє навчати агентів, які не просто повторюють шаблони, а вчаться створювати якісний текст у змінному контексті [1].

1.2.2 Метод PPO: переваги та використання в NLP

З-поміж актуальних методів навчання з підкріпленням, які здобули визнання у сфері обробки мови, Proximal Policy Optimization (PPO) виділяється стабільністю та ефективністю, зокрема при оптимізації великих нейронних мереж. Суть його в корекції політики агента, з обмеженням різких змін, що дозволяє уникнути нестабільності в процесі навчання. Цей підхід добре масштабується та не потребує громіздких обчислень, як-от інверсія матриць чи розв'язання рівнянь, як у деяких інших алгоритмах.

PPO неодноразово довів свою корисність у задачах генерації тексту, де важливо враховувати довготривалі залежності між словами та якість вихідного тексту за складними метриками. У таких випадках звичайне супервізоване навчання часто є недостатнім, адже не дає можливості безпосередньо оптимізувати бажані характеристики відповіді, на кшталт смислової точності, зв'язності чи стилю. PPO ж дає змогу агенту вчитися з огляду на кінцевий результат, отримуючи винагороду за кожен текст на основі сукупності критеріїв – як це реалізовано в розробленій у межах даної роботи системі.

Ще однією перевагою PPO є його здатність працювати з апроксимацією політики, реалізованою через нейронну мережу, яку можна оновлювати ітераційно без потреби в повній перебудові. Це дозволяє без проблем інтегрувати його в уже наявні генеративні системи на базі трансформерів, адаптуючи модель до нових вимог користувача або специфічних доменів без ризику «забути» вже вивчену поведінку.

Особливо активно PPO використовується в парадигмі навчання з підкріпленням з людським зворотним зв'язком (RLHF), де роль критика виконує людський оцінювач або автоматизована модель, що замінює експерта. У таких сценаріях, зокрема в роботах, присвячених ChatGPT, InstructGPT та іншим аналогічним моделям, PPO став ключовим інструментом для налаштування поведінки мовної моделі. Він забезпечує не

тільки покращення об'єктивних метрик якості, але й дозволяє враховувати етичні, стилістичні та прагматичні міркування під час генерації [18].

У межах даної роботи PPO адаптовано до безлюдного сценарію. Винагорода обчислюється автоматизовано, на основі оцінювання відповідності тексту темі, його новизни, стилістики та інформативності. Це дозволяє зменшити потребу в людській участі й разом з тим сприяє формуванню вибірки з текстів, які ліпше задовольняють вимоги до навчальних даних.

1.3 Синтез навчальних вибірок для мовних моделей

Сучасні мовні моделі демонструють надзвичайні результати, проте якість їх роботи значною мірою залежить від тренувальних вибірок. У реальному світі зібрати достатню кількість якісних, тематично актуальних та збалансованих даних є вкрай складним завданням, особливо коли модель потребує донавчання для специфічних задач. Синтетичні тренувальні вибірки постають як рішення, здатне компенсувати недолік «живих» прикладів і дати змогу розширити охоплення без потреби залучення великої кількості анотацій.

Один із перспективних напрямів розвитку – використання самостійно згенерованих даних, зокрема, у підходах Self-Instruct або Self-Chat. У першому випадку, модель спершу вивчає набір інструкцій від людини, а потім намагається самостійно створювати нові завдання разом із відповідями. У другому, дві копії однієї й тієї ж моделі ведуть діалог, генеруючи питання та відповіді в симульованому режимі. Ці підходи вже продемонстрували свій потенціал як у загальному донавчанню моделей, так і у створенні даних для інструкційного налаштування (instruction tuning).

Проте ефективність таких методів прямо пропорційна якості згенерованих текстів. У багатьох випадках вони можуть бути тривіальними, невірними або непослідовними, що негативно впливає на процес навчання.

Зважаючи на це, виникає потреба в механізмах автоматизованої фільтрації та оцінювання згенерованого контенту, здатних забезпечити відбір лише корисних прикладів. В межах цієї роботи, така функціональність реалізована за допомогою винагородної функції, яка враховує декілька критеріїв якості: семантичну відповідність, новизну, адекватну довжину тощо.

Окремо слід відмітити, що автоматизоване навчання на синтетичних даних потребує зваженого підходу. Надто агресивна фільтрація призводить до одноманітної та надмірно «стерильної» вибірки, тоді як надмірна гнучкість може призвести до включення нерелевантного чи навіть шкідливого контенту. Розроблена у цьому дослідженні система досягла компромісу завдяки гібридній генерації та гнучкій схемі відбору, що враховує динаміку винагород і дозволяє адаптувати процес під різні конфігурації. Вигляд системи генерації та фільтрації синтетичних даних представлено на рисунку 1.9.

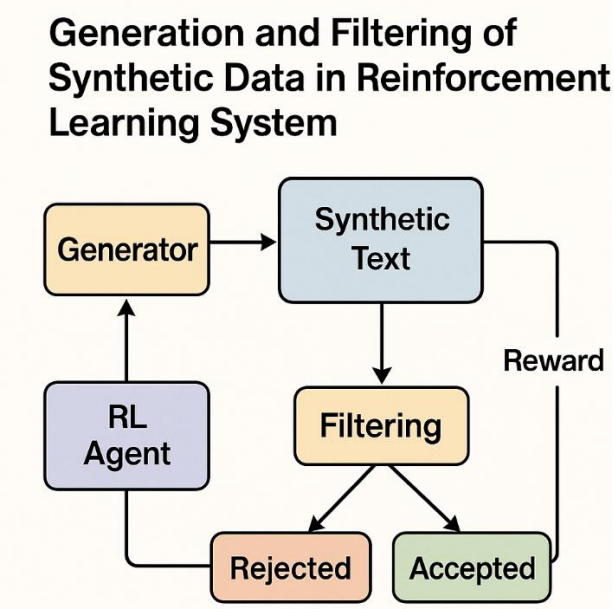

Рисунок 1.9 – Схема генерації та фільтрації синтетичних даних у системі з підкріплювальним навчанням

Отже, синтез навчальних вибірок – це не лише технічне завдання генерації тексту, а складний процес відбору, оцінювання та динамічного контролю якості. Це відкриває перспективи для створення мовних моделей, які здатні самостійно удосконалювати своє розуміння світу, мінімізуючи людську участь у процесі формування даних [19].

1.3.1 Підходи Self-Instruct та Self-Chat

У сфері автоматичного генерування навчальних даних два ключові методи – Self-Instruct та Self-Chat – стали базовими інструментами для створення інструкційних наборів даних, не вдаючись до значних обсягів людського розмічування.

Self-Instruct зосереджується на тому, що модель, вже натренована на первинних інструкціях, самостійно продукує нові завдання та відповіді до них, імітуючи зразки, отримані від людей. Метод передбачає використання невеликого вхідного набору інструкцій (seed instructions), після чого модель крок за кроком генерує дедалі складніші приклади. Це дає змогу моделі не тільки вчитися на згенерованому контенті, а й розширювати діапазон задач, на яких вона тренується. Такий підхід значною мірою спирається на генеративні можливості самої моделі та її здатність до узагальнення [17].

На противагу цьому, Self-Chat базується на концепції імітації діалогу між двома або більше копіями тієї ж моделі. В одному амплуа вона задає питання, в іншому – відповідає, а в окремих варіантах – оцінює релевантність відповіді або уточнює її. Цей метод був успішно втілений у проєкті Baize, де застосовано архітектуру з точним налаштуванням на спеціально розроблених діалогах. Такі підходи демонструють здатність моделі навчатися не лише на завданнях типу «інструкція-відповідь», а й на контекстуальній комунікації, яка максимально наближена до реального спілкування користувача з системою.

Обидва підходи мають сильні і слабкі сторони. Серед переваг – зниження витрат на збір даних, масштабованість та пристосованість до нових предметних областей. Серед недоліків – схильність до повторів, продукування шаблонних або нелогічних інструкцій, а також можливість зміщення (bias) у напрямку генерації, яка вже була «вивчена» моделлю. Саме тому в реальних ситуаціях ці методи зазвичай комбінують з механізмами фільтрації – вручну або за допомогою допоміжних reward-моделей, що оцінюють якість діалогів чи інструкцій. Порівняльна експертиза зображена на рисунку 1.10.

Порівняння Self-Instruct та Self-Chat підходів

	Self-Instruct	Self-Chat
Джерело даних	Запитання–Відповіді	Діалоги
Процес генерації	Поетапна генерація відповідей	Поетапна генерація реплік
Комплек-сність даних	Легша	Вища

Рисунок 1.10 – Порівняння Self-Instruct та Self-Chat підходів

Впровадження цих методів у навчання з підкріпленням дозволяє поєднати два ключові напрями – самостійну генерацію даних та динамічну оптимізацію процесу тренування. Це значно розширює можливості адаптації трансформер-моделей до нових умов без витратних анотаційних процедур. Такий підхід робить систему більш гнучкою, здатною до самонавчання та ефективної роботи в змінному або недостатньо формалізованому середовищі.

1.3.2 Проблеми якості синтетичних даних

Попри великий потенціал автоматизованих методів синтезу даних для тренування трансформерних моделей, їх використання стикається з низкою серйозних викликів. Насамперед, це питання семантичного зношення: синтетичні приклади, особливо ті, що створюються у кілька етапів без ретельного контролю, здатні поступово втрачати інформативність чи навіть викривлювати початковий задум інструкцій. Це проявляється у шаблонності відповідей, повторюваності семантики та логічних зрушеннях у змісті [1].

Інша проблема – стилістична одноманітність: оскільки модель у Self-Instruct чи Self-Chat сценаріях генерує відповіді на основі власних прикладів, результат часто виглядає занадто «вилизаним», формалізованим та таким, що погано передає реальну різноманітність людської мови. Це знижує здатність моделей до узагальнення на практичних завданнях, де важлива варіативність формулювань, як-от у творчому письмі чи відповідях на відкриті питання.

Ще один важливий аспект – накопичення помилок та зміщення. Без додаткових механізмів контролю якості модель може видавати невірні чи некоректні приклади, які потім використовуються нею як навчальний матеріал. Це нагадує «зіпсований телефон», коли інформація спотворюється з кожним наступним переказом. Таке явище спостерігається в багатьох генеративних системах без зовнішнього джерела істинності [15].

Труднощі стають більш відчутними за відсутності людського фідбеку. Якщо не використовуються моделі винагороди або додаткові фільтри, синтетичні вибірки можуть містити зразки з низькою релевантністю або несподіваними помилками. Саме тому актуальні дослідження (наприклад, GPT-4 Technical Report [21]) радять або комбінувати автоматизований синтез з людським контролем, або інтегрувати reward-моделі, навчені на

узагальнених критеріях якості. Типові помилки та виклики у синтетичних датасетах ілюстровано на рисунку 1.11.

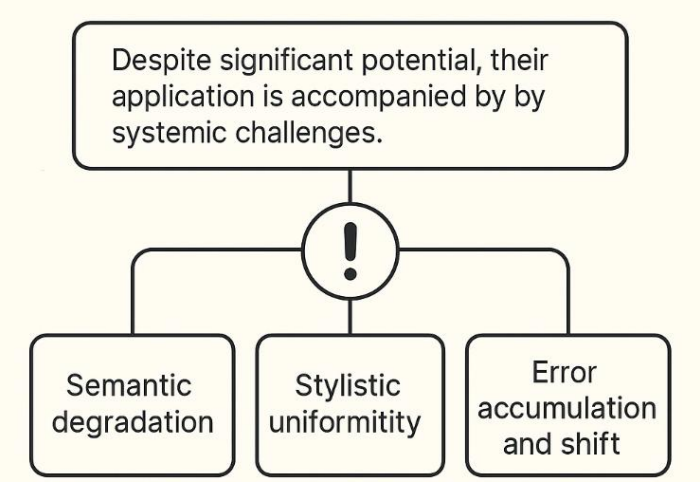

Рисунок 1.11 – Типові помилки та виклики у синтетичних датасетах на основі Self-Instruct і Self-Chat

Таким чином, хоча синтетичні вибірки відкривають нові перспективи в тренуванні моделей, вони потребують обов'язкових заходів валідації – або через фідбек від людини, або за допомогою спеціалізованих підсистем оцінювання. В цій роботі здійснено спробу реалізувати саме такий підхід до перевірки якості, який враховує як семантичну відповідність, так і новизну, стилістичну різноманітність та інші параметри.

2 АНАЛІЗ АНАЛОГІВ ТА ІСНУЮЧИХ РІШЕНЬ

Перед розробкою власної архітектури синтезу навчальних вибірок доцільно дослідити наявні системи, підходи та інструменти, що вже застосовуються у цій галузі. Світове співтовариство активно просуває ідеї масштабування моделей на базі великих обсягів даних, значна частина яких є синтетичною або напівавтоматично згенерованою. Водночас зростає потреба у фільтрації, оцінюванні та адаптації таких даних до задач, наближених до реального користувача, загальні конструкції, що використовуються у LLM-системах зображено на рисунку 2.1.

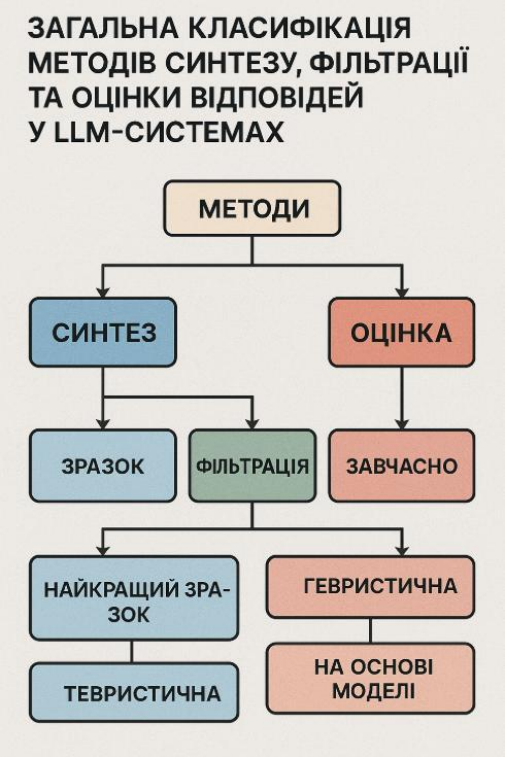

Рисунок 2.1 – Загальна класифікація методів синтезу, фільтрації та оцінки відповідей у LLM-системах

На прикладі таких проектів, як Self-Instruct [8], Alpaca, Baize [2], InstructGPT та OpenAssistant, спостерігаємо тренд до створення навчальних

датасетів через багатоетапну генерацію на основі підказок, пристосованих до інструкцій. Зокрема, Baize показує ефективність підходу, де перший згенерований текст модифікується чи оцінюється за допомогою другої моделі, що вже навчена на більш узагальненому розумінні якісної відповіді.

Більшість сучасних систем прагне розв'язати кілька завдань: забезпечити відповідність темі відповідей, зберегти свіжість формулювань, а також гарантувати логічну завершеність тексту. При цьому комбінація генеративних моделей (GPT-2, T5, DistilGPT), моделей оцінки (на кшталт BERTScore, MMLU, BLEURT), а також механізмів навчання з підкріпленням (PPO, RLAIF) створює цілу екосистему інструментів, які, хоч і не замінюють повністю людський фідбек, здатні його чітко імітувати [16].

Кожне з рішень має свої сторони. Наприклад, GPT-4 демонструє виняткову здатність до узагальнення навіть на синтетичних даних, але потребує якісної підготовки навчальної вибірки. Навпаки, у Baize наголос зроблено на параметроефективному донавчанню моделі на власному діалозі, що дозволяє зменшити вартість процесу формування датасету.

Окремо слід відзначити появу проектів, де генерація відповідей відбувається з врахуванням зовнішніх критеріїв, включаючи токсичність, граматичну правильність або інтерпретованість. У таких випадках reward-моделі виступають як додатковий «наглядач», який контролює якість результатів генератора. У цьому розділі далі буде розглянуто три основні напрями: інструменти генерації, механізми фільтрації та оцінки, а також архітектури reward-моделей у відкритих системах.

2.1 Інструменти генерації даних

Генерація синтетичних даних міцно увійшла в арсенал сучасних методів навчання великих мовних моделей. У зв'язку з цим виникла гостра потреба у розробці гнучких та масштабованих інструментів, що здатні не просто генерувати текст, а формувати навчальні зразки з визначеними

стилістичними, змістовними чи інструктивними обмеженнями. Показовим прикладом є підхід Self-Instruct, де базою є створення інструкцій на основі первинних шаблонів, які потім автоматично змінюються та уточнюються моделлю.

Системи, такі як Alpaca, OpenAssistant і Vicuna, хоч і базуються на схожих принципах, поширюють їх, включаючи етапи донавчання на синтезованих діалогах, сформованих за принципом інструкційно-орієнтованого запитання-відповіді. Важливу роль у таких системах відіграють генератори на кшталт GPT-2, T5 та їхні модифікації з покращеними характеристиками. Наприклад, T5 дає змогу перетворювати вхідний текст у будь-який цільовий формат за принципом «текст-в-текст», що робить його особливо ефективним у задачах переформулювання чи стилістичної адаптації [6]. Приклад архітектури генерації вибірок зображено на рисунку 2.2.

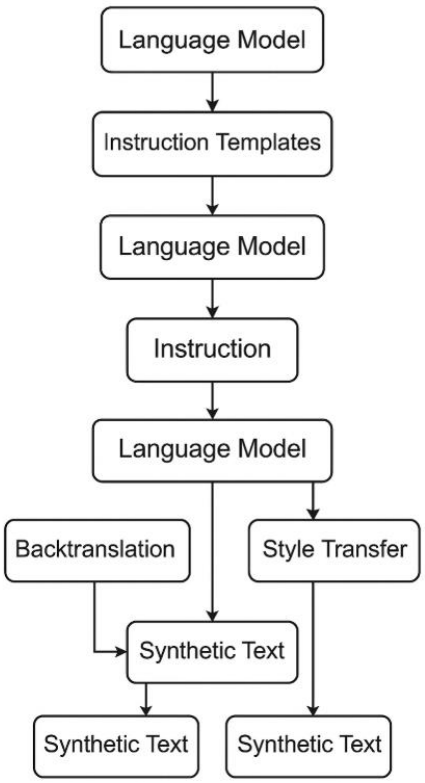

Рисунок 2.2 – Архітектура каскадної генерації даних із використанням кількох мовних моделей

Подібні інструменти, як BackTranslation, Formalization або StyleTransfer, все частіше використовуються не як основа моделі, а як проміжні модулятори – тобто як додаткові шари над базовим генератором, що забезпечують стилістичну або мовну різноманітність без втрати основного змісту. Наприклад, перефразування дозволяє знизити надмірне дублювання в згенерованому корпусі, що критично важливо для забезпечення новизни та розмаїття навчальних даних.

Останнім часом особливу популярність здобув підхід багатогенераторної архітектури, де кілька моделей працюють одночасно або каскадно, кожна вносячи свій внесок у результат. Це дозволяє досягти кращого балансу між узгодженістю, інформативністю та стилістичною різноманітністю відповідей. У деяких реалізаціях, на кшталт Baize, друга модель аналізує та оцінює відповіді першої, створюючи синергетичну динаміку генерації з елементами внутрішнього зворотного зв'язку.

Загалом, сучасні інструменти генерації даних в NLP демонструють рух у напрямку гібридизації: об’єднання декількох підходів у єдину систему, здатну адаптуватися до різних типів запитів та цільових задач. Це відкриває нові можливості для побудови навчальних вибірок, які не просто відтворюють існуючі патерни, а сприяють формуванню нових мовних узагальнень.

2.2 Методи фільтрації, переоцінки та відбору відповідей

Після первинної генерації відповіді, важливим стає етап фільтрації та переоцінки, що безпосередньо впливає на якість кінцевої навчальної вибірки. Сучасні системи генерації вже не обмежуються одноразовою перевіркою граматики або змісту. Замість цього, впроваджується багаторівнева система оцінювання, здатна враховувати як відповідність змісту, так і стилістичну, логічну та навіть етичну коректність відповіді.

Поширеним методом є використання reward-моделей – моделей-оцінювачів, навчених призначати числову оцінку якості згенерованого тексту на основі різних критеріїв. Такі моделі, як Helpful Reward Model або Toxicity Classifier [24], [25], інтегруються в загальну архітектуру синтезу даних, виконуючи роль «фільтрів другого рівня». Вони допомагають відсіяти токсичні, нерелевантні або надто узагальнені варіанти ще до того, як тексти потраплять до фінального корпусу.

Окрім автоматичних фільтрів, часто використовуються метрики на кшталт BLEU, ROUGE або новіші семантичні подібності, засновані на трансформерах, наприклад BERTScore. Ці підходи дозволяють забезпечити баланс між новизною тексту та його відповідністю заданій темі. У проєктах на зразок Self-Instruct, переоцінка відповіді відбувається не лише за метриками якості, але й через взаємне зіставлення декількох альтернативних відповідей на одну інструкцію. Найбільш релевантна з них потім зберігається, а решта або перегенеруються, або відкидаються. Схема ранжування може бути ілюстрована на рисунку 2.3.

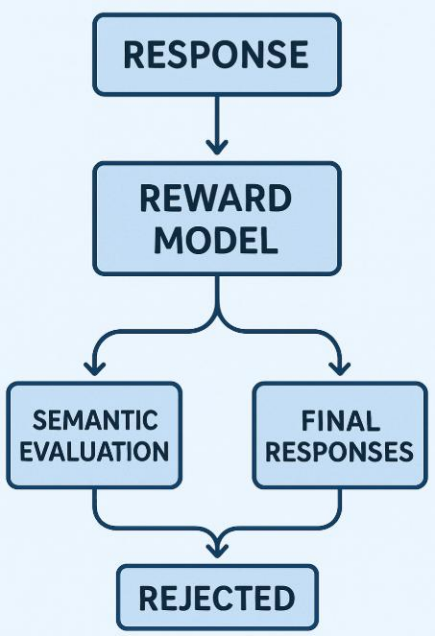

Рисунок 2.3 – Схема фільтрації відповідей через reward-модель і семантичну оцінку

Інший важливий напрямок – використання інтерактивного або автоматизованого ранжування. Наприклад, у Baize застосовується модель, яка не лише генерує, а й самостійно «спілкується» зі своєю відповіддю, оцінюючи її логічну завершеність, відповідність інструкції та корисність. Це наближає систему до моделі самоконтролю, де генератор одночасно є і фільтратором.

Загалом, сучасні методи переоцінки в NLP утворюють цілу екосистему, де поєднуються евристики, метрики, мовні моделі та модулі етичного аналізу. Це дозволяє не лише покращувати якість окремих відповідей, а й підвищувати узагальнюючу здатність мовних моделей, які навчаються на відфільтрованих і збагачених вибірках.

2.3 Використання reward-моделей у відкритих проектах

Reward-моделі нині стали ключовою деталлю сучасних методів навчання великих мовних моделей, особливо у царині навчання з підкріпленням за участю людини (RLHF). Вони дають змогу перетворити людські оцінки на числові сигнали винагороди, які скеровують процес покращення генератора відповідей. Це особливо важливо для створення корисних, безпечних і правдивих моделей діалогу [13].

У відкритих дослідницьких проектах, таких як Self-Instruct, Baize та InstructGPT, reward-моделі використовуються для сортування відповідей, отриманих з використанням людських або машинно згенерованих інструкцій. Наприклад, у Self-Instruct відбувається створення великої кількості інструкцій та відповідей, а потім відбираються найкращі на основі оцінок reward-моделі або шляхом порівняння різних варіантів. Це дає можливість масштабувати навчання моделей без потреби залучати багато анотаторів.

Baize демонструє один з найбільш ощадливих підходів до використання reward-моделей. Автори використовують техніку self-chat –

діалог моделі сама з собою, де одна роль ставить запитання, а інша відповідає, після чого якість оцінюється за допомогою заздалегідь навченого reward-оцінювача, зокрема helpfulness reward model [26].

В контексті PPO (Proximal Policy Optimization), reward-моделі часто виконують функцію винагороди, яка визначає, наскільки відповідь наближена до потрібних параметрів. Наприклад, під час навчання моделі GPT-2 з використанням PPO агент отримує винагороду на основі поєднання декількох критеріїв: змістовності, відповідності темі, унікальності та ненасильницького тону – всі ці фактори формуються reward-моделлю або ансамблем моделей [9]. Цю систему можливо бачити на рисунку 2.4.

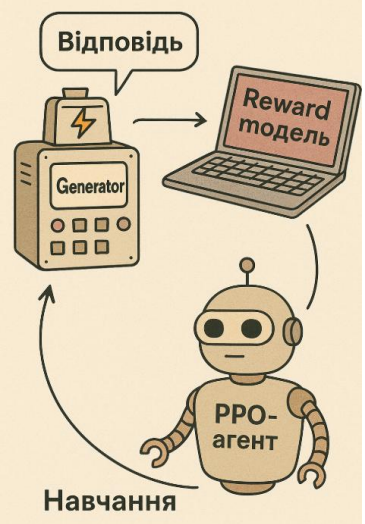

Рисунок 2.4 – Взаємодія генераторів, reward-моделі та PPO-агента у навчальному циклі

Reward-моделі також забезпечують гнучкість у дизайні метрик якості, дозволяючи враховувати різні доменні вимоги, наприклад, юридичну точність [5], патентну мову або специфіку наукової комунікації. Така адаптивність робить їх універсальним інструментом для контролю якості у системах автоматичної генерації. Незважаючи на високу ефективність, одним із викликів залишається узгодженість reward-моделі з людськими оцінками. У деяких випадках модель може переоцінювати «надто

узагальнені» відповіді або недооцінювати емоційно нейтральні, проте точні формулювання. Це мотивує нові дослідження в напрямку інтеграції RL з альтернативними методами контролю, такими як RLAIF – навчання з підкріпленням за допомогою автоматично згенерованих фідбеків [26].

Подібні методи спрямовані на зменшення залежності від ручного маркування, що є ресурсозатратним і обмеженим масштабом. RLAIF дозволяє створювати великі обсяги тренувальних даних із використанням допоміжних генеративних або дискримінативних моделей. Однією з переваг такого підходу є можливість швидкої адаптації reward-функцій до нових задач без участі експертів. Крім того, автоматизовані фідбеки можуть бути послідовно покращувані на основі зворотного зв’язку від основного RL-агента, що створює потенціал для самоналагоджуваних систем. Водночас залишається відкритим питання надійності таких фідбеків у складних сценаріях, де потрібне глибоке розуміння контексту або етичних аспектів. Це стимулює подальшу інтеграцію гібридних стратегій, які поєднують автоматичні оцінки з вибірковим людським втручанням на критичних етапах навчання.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура системи: модулі генерації, оцінювання та навчання

Розроблена система генерування навчальної вибірки для трансформерних моделей спирається на модульну архітектуру, що дає змогу гнучко поєднувати генеративні моделі, reward-моделі, стратегії вибору відповідей та підходи до навчання з підкріпленням. Цей підхід забезпечує масштабованість, розширюваність та пристосування системи до різних цілей.

В основі архітектури лежить гібридний генератор, до складу якого входять кілька самостійних моделей генерування тексту. Кожна з них спеціалізується на виконанні певного перетворення над вхідним промптом: GPT-2 виконує стандартне авторегресійне продовження, T5 – парафразинг, BackTranslation – латентне переформулювання, StyleTransfer – стилізацію, а Formalization – формальне переписування у відповідному домені. Генератор працює за принципом паралельного розгалуження: на вхід одне й те саме повідомлення надходить одночасно всім моделям, після чого результати збираються в пул варіантів.

Отримані відповіді надходять у блок оцінювання, який реалізує систему reward-функцій. Цей модуль обчислює якісні метрики для кожної відповіді: тематичну подібність, новизну, корисність та відповідну довжину. Відповіді, що не відповідають заданим порогам, відсіюються, а решта передається в модуль вибору найкращої відповіді за допомогою стратегій, таких як «argmax по сумі балів», «top-k sampling з повторною нормалізацією» або «зважене голосування моделей».

На основі оцінених відповідей система формує синтетичну вибірку, яка використовується для оновлення студент-моделі або для подальшого підсилення генератора через навчання з підкріпленням. PPO-агент інтегровано у цикл як оптимізатор політики генерування: він отримує як

вхідний стан промпт, а як дію – вибраний варіант відповіді. Нагорода, що генерується reward-моделями, є сигналом для оновлення політики. Для спрощення розуміння загальна архітектура була показана на рисунку 3.1.

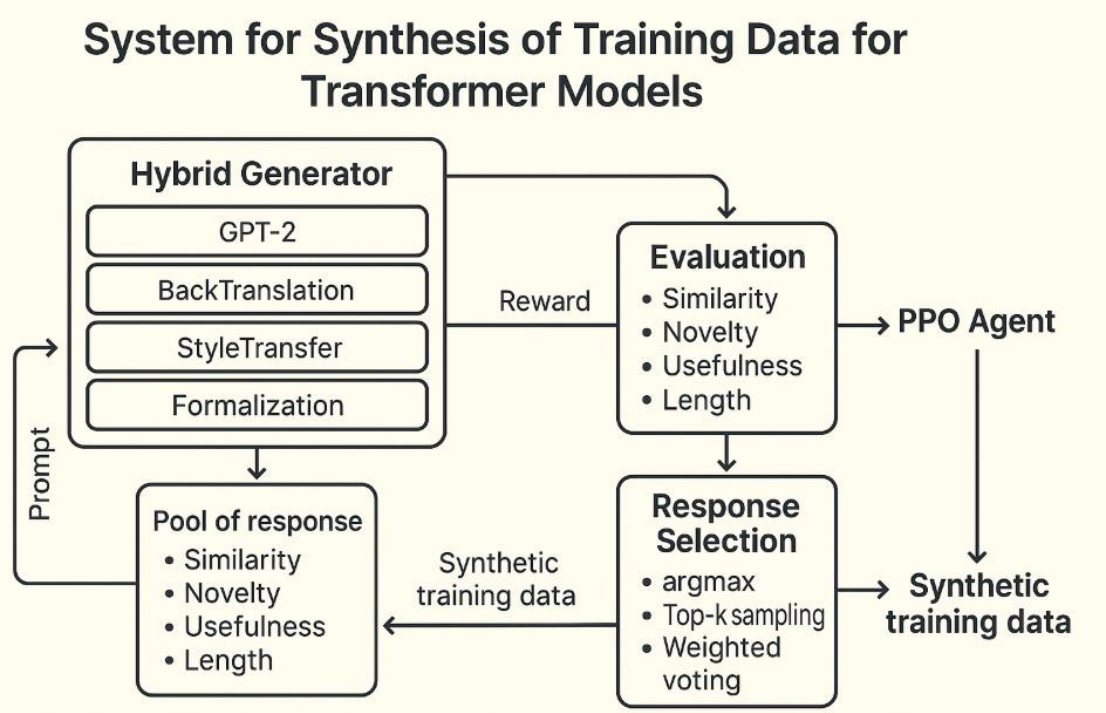

Рисунок 3.1 – Загальна архітектура системи синтезу даних: взаємодія генераторів, reward-модулів, вибору та PPO-агента

Архітектура побудована таким чином, щоб уможливити майбутнє підключення додаткових компонентів – наприклад, класифікаторів токсичності [24], спеціалізованих стилістичних фільтрів або доменно-орієнтованих модулів оцінки. Це дозволяє не лише збільшити загальну якість згенерованої вибірки, а й адаптувати систему під специфіку предметної галузі.

3.2 Опис reward-функцій

У розробленій системі ключову роль у керівництві якістю згенерованих відповідей відіграють reward-функції – спеціалізовані

метрики, що визначають доцільність кожної потенційної відповіді. Вони оцінюють відповідності з точки зору релевантності теми, оригінальності, інформативності та інших властивостей, важливих для формування корисної навчальної вибірки.

Основна концепція полягає в тому, що замість «твердої» ручної фільтрації або застосування єдиного евристичного критерію, ми створюємо гнучкий комбінований функціонал винагороди. Кожен окремий компонент функціоналу має власну вагу, і загальна reward-оцінка обчислюється як зважена сума окремих метрик. Це дає змогу враховувати різноманітні аспекти відповідей, які важко об'єднати в один показник, і формує підґрунтя для застосування навчання з підкріпленням.

Серед основних метрик, що використовуються в системі, – тематична відповідність, новизна, довжина та семантична корисність. Додатково, у разі необхідності, можна інтегрувати додаткові компоненти, наприклад, показники відповідності стилю, граматичної коректності або відповідності домену. Приклад reward-функції наведено на рисунку 3.2.

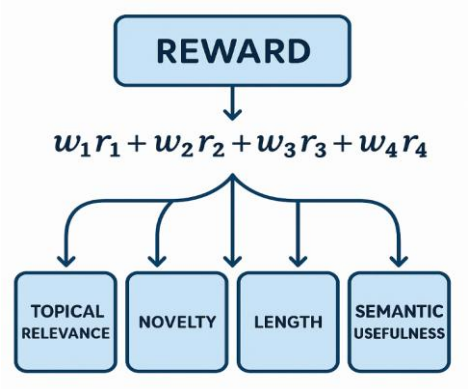

Рисунок 3.2 – Комбінована reward-функція як зважена сума часткових оцінок

Reward-функції не є універсальними – вони налаштовуються відповідно до цілей та завдань моделі. Наприклад, якщо пріоритетом є креативність відповіді, вагу новизни можна збільшити, у той час як для

задач з юридичною або технічною специфікою – навпаки, акцент ставиться на тематичну точність.

Таке подання функцій винагороди у вигляді модульної системи дозволяє застосовувати її не лише як фільтр, а і як основу для адаптивного навчання генеративної моделі через оптимізацію політики, зокрема методом PPO, що розглянуто нижче.

3.2.1 Оцінка тематичної подібності

Довжина згенерованої відповіді є критичним параметром при формуванні тренувальної вибірки. Занадто короткі відповіді можуть бути малоінформативними, в той час як надмірно довгі ризикують призвести до розмивання теми або шаблонізації. У межах цього підходу реалізовано механізм, що заохочує підвищення інформативності тексту, зберігаючи при цьому його лаконічність.

Система проводить дворівневу оцінку довжини. На першому рівні враховується абсолютна кількість токенів. Цільовий діапазон, 25–60 токенів, забезпечує оптимальний баланс між інформаційним наповненням та зручністю обробки під час подальшого донавчання. Тексти, що потрапляють у цей діапазон, отримують позитивне заохочення, тоді як занадто короткі або довгі – часткове покарання. На другому рівні аналізується співвідношення унікальних лексем до загальної кількості токенів, що дозволяє оцінити щільність інформації у тексті.

Окремо варто зазначити поняття корисності. Для її наближеної оцінки застосовуються евристичні показники: відповідність запиту, відсутність мовних помилок (перевіряється за допомогою інструменту language-tool-python [25]), логічна завершеність речень та цілісність аргументації. Ці критерії не оцінюються окремими моделями, але використовуються як фільтри на етапі постобробки.

Підхід до оцінювання довжини та корисності є прагматичним: він не лише дозволяє відсіяти надто прості або хаотичні варіанти, але й сприяє створенню синтетичної вибірки, придатної для подальшого донавчання моделей у задачах генерації вільного тексту. Згідно з дослідженнями Zhou et al., саме контроль структурної якості відповідей є ключовим фактором ефективності fine-tuning у подібних системах.

3.2.2 Новизна (Jaccard, embedding)

Оцінка новизни відіграє ключову роль у формуванні нагороди-функції, адже завдання системи не тільки генерувати релевантні відповіді, а й додавати до навчальної вибірки нові, незвичайні зразки. Зменшення новизни або постійне генерування шаблонних фраз погіршує ефективність подальшого донавчання трансформерних моделей.

У представленій системі для оцінки новизни використовуються два методи. Перший – традиційний індекс Жаккара, який визначає ступінь лексичного перекриття між створеною відповіддю та історією вже збережених у вибірці відповідей на аналогічні запити. Низький індекс вказує на високу новизну, а високий – на можливе повторення.

Другий метод базується на аналізі ембедингів – щільних векторних представлень текстів, отриманих за допомогою попередньо навчених мовних моделей. У цьому підході розраховується середня косинусна відстань між вектором нової відповіді та наближеним центроїдом (тобто усередненим вектором) уже наявних відповідей, що належать до тієї самої або подібних тем. Такий підхід дозволяє оцінити семантичну віддаленість нового тексту від попередніх прикладів. Якщо відстань значна, то відповідь вважається новою або несподіваною. Це допомагає уникнути надмірного повторення й стимулює генерацію більш різноманітного контенту. Результати оцінки за цими метриками представлені на рисунку 3.3 та 3.4.

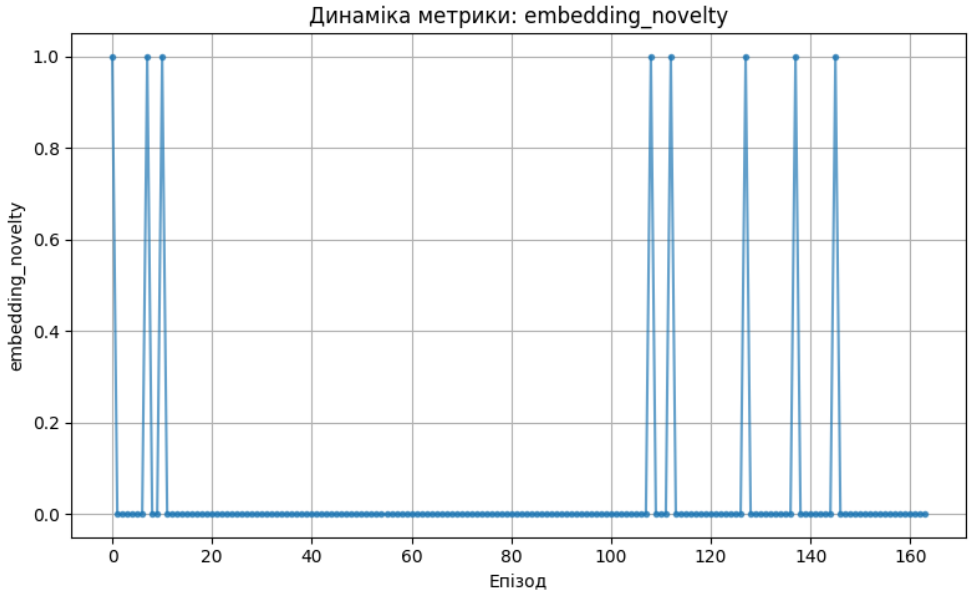

Рисунок 3.3 – Оцінки новизни embedding-метрики

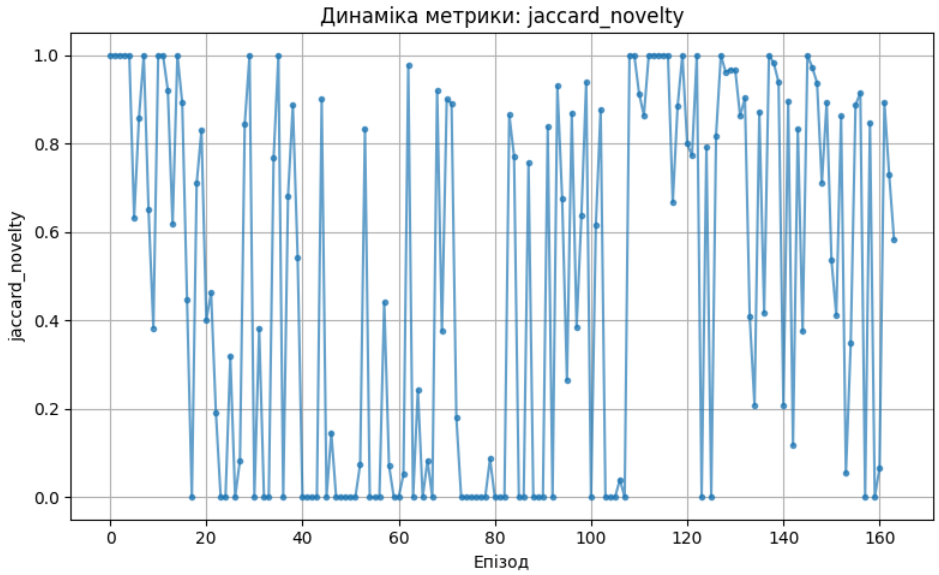

Рисунок 3.4 – Оцінки новизни Jaccard

Використання одночасно двох підходів забезпечує гнучкіший контроль як за лексичним, так і за концептуальним повторенням. Подібна ідея була також застосована в дослідженні Salazar et al., де йшлося про масковане оцінювання мовних моделей через контекстні варіації.

Новизна в системі не є самоціллю – її підвищення повинно відбуватись лише за умови збереження тематичної релевантності. Саме тому reward-функція комбінує кілька аспектів: тематичність, новизну та корисність. Це дозволяє уникати «галюцинацій» моделі, які часто виникають при безконтрольному пошуку нових структур.

3.2.3 Міра довжини та корисності

Довжина відповіді, згенерованої системою, виступає ключовим параметром під час формування тренувального набору даних. Занадто короткі відповіді ризикують бути малоінформативними, тоді як надто розлогі можуть призвести до розмиття тематики або шаблонності. У рамках застосованого методу було реалізовано стимулювання збільшення інформативності тексту, зберігаючи при цьому лаконічність.

Система оцінює довжину у два етапи. Спочатку враховується загальна кількість токенів. Оптимальний діапазон – від 25 до 60 токенів – дозволяє збалансувати інформативність і зручність подальшого донавчання. Тексти, що відповідають цьому інтервалу, заохочуються позитивно, натомість надто короткі або довгі відповіді отримують певне обмеження. На другому етапі визначається співвідношення унікальних слів до загальної кількості токенів, що дає змогу оцінити щільність інформації в тексті.

Окремо варто згадати про корисність відповідей. Для її наближеної оцінки використовуються евристичні показники: відповідність на запит, відсутність помилок (перевірка здійснюється за допомогою інструменту language-tool-python), логічна завершеність речень та цілісність аргументації. Ці критерії не розраховуються окремими моделями, але використовуються під час фільтрування при постобробці. Хоча така оцінка є приблизною, вона дозволяє ефективно відсіяти малозмістовні або некоректні відповіді ще до етапу формального аналізу.

Наприклад, розподіл відповідей за довжиною, який є одним з критеріїв корисності, проілюстровано на рисунку 3.5.

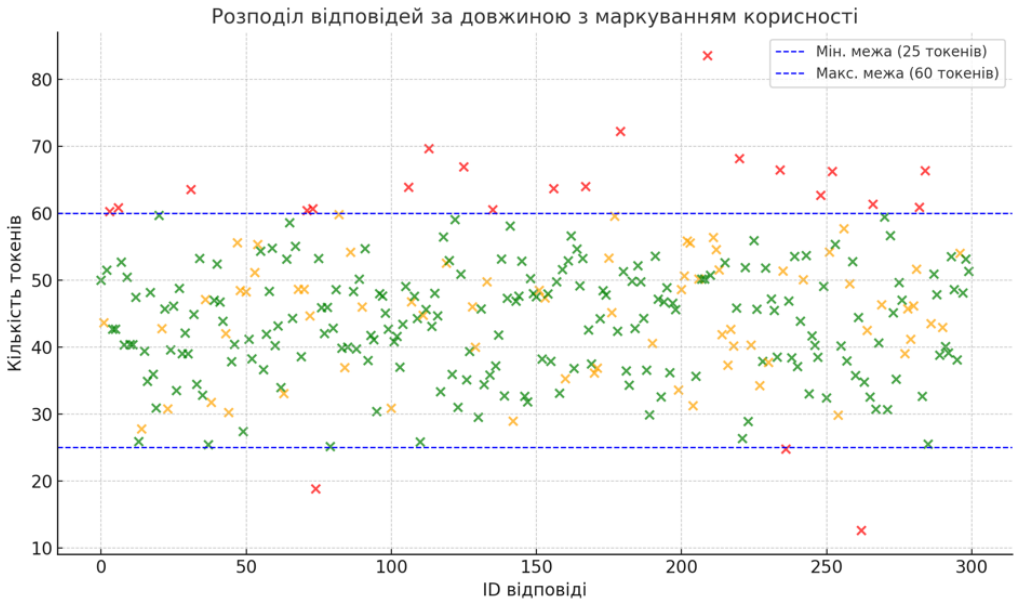

Рисунок 3.5 – Розподіл відповідей за довжиною з маркуванням корисності

Підхід до оцінювання довжини й корисності є прагматичним: він не лише дозволяє виключити надто прості або хаотичні варіанти, а й сприяє утворенню синтетичної вибірки, придатної для подальшого донавчання моделей у задачах open-ended generation. У подібних системах, згідно з Zhou et al., саме контроль за структурною якістю відповідей є вирішальним для ефективності fine-tuning.

3.3 Вибір та інтеграція генераторів

Для забезпечення різноманітності стилів, структур та мовних стратегій у синтетичній навчальній вибірці до системи було інтегровано кілька незалежних генераторів. Основу складають моделі GPT-2, T5 та низка трансформерів, що виконують стилізацію, переклад та перефразування. Така архітектура дозволяє об'єднати переваги різних

підходів до генерації в межах однієї платформи. Схему інтеграції генераторів у гібридну систему показано на рисунку 3.6.

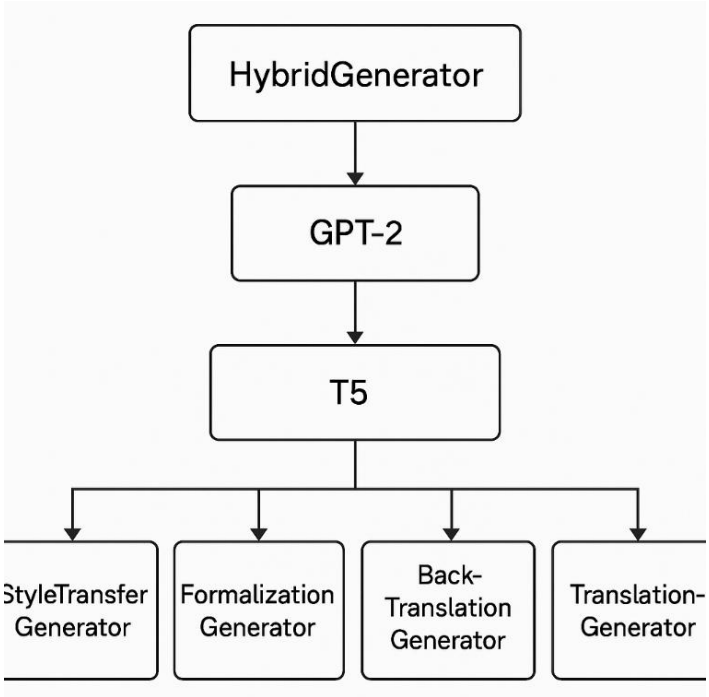

Рисунок 3.6 – Схема інтеграції генераторів у гібридну систему

Модель GPT-2, попри свою відносну компактність і обмежений контекст, відзначається стабільністю у генерації осмислених, завершених відповідей. Вона забезпечує швидку генерацію при низькому споживанні ресурсів, що особливо цінне на етапі попереднього відбору. Модель T5, натомість, працює в парадигмі «текст-у-текст», що дозволяє їй ефективно вирішувати задачі переформулювання, доповнення чи стилістичної трансформації.

Додатково було інтегровано модулі стилізації, зокрема:

– StyleTransferGenerator – реалізує зміну стилістики з формальної на неформальну і навпаки;

– FormalizationGenerator – адаптує відповіді до офіційно-наукового стилю;

– BackTranslationGenerator – здійснює двоетапний переклад для створення семантично еквівалентних, але лексично нових формулювань;

– TranslationGenerator – дозволяє тестувати мультимовну генерацію, що важливо для узагальнення моделей на нові домени.

Кожен генератор обгорнутий у єдиний інтерфейс HybridGenerator, який реалізує стратегії випадкового або зваженого вибору, що змінюється на етапі навчання з підкріпленням. Це дозволяє зберегти адаптивність системи, підлаштовуючи генерацію під потреби reward-функцій та динаміку RL-процесу. Подібна багатомодельна архітектура була успішно апробована в проєктах на кшталт Baize, Self-Instruct і InstructPatentGPT, що свідчить про її ефективність.

3.4 Формування запитів та початкового набору даних

Якість синтетично згенерованої вибірки безпосередньо корелює з ретельністю розробки вхідних запитів, які запускають процес генерації. На першому етапі формування запитів базувалося на узагальненому переліку тем, асоційованих із цільовими сферами застосування (наприклад, освітні, дослідницькі, практичні питання в сфері штучного інтелекту, етичні аспекти використання LLM та інше). Запити структуризувалися у формі інструкцій, питань, тез або незакінчених речень, що провокують розгортання відповіді генератором.

Для гарантування тематичної різноманітності застосовувався набір базових шаблонів, які доповнювались лексично та стилістично з врахуванням майбутнього розширення варіативності. Частина шаблонів була створена вручну на основі аналізу публічних датасетів (наприклад, Quora Question Pairs [22] чи OpenWebText [20]), інша – синтезована автоматично шляхом перефразування та перекладу за допомогою спеціалізованих генераторів. Такий комбінований підхід дозволив охопити ширший спектр інтенцій користувача та зменшити надмірну концентрацію

схожих формулювань у згенерованих прикладах. Приклади початкових запитів разом із їх варіантами після генерації вибірки продемонстровано на рисунку 3.7 та 3.8.

```
meta_prompt = (
    "You are part of a synthetic data generation system for fine-tuning transformer models. "
    "Generate a list of {n} instructions that are unclear, underspecified, ambiguous, or cognitively challenging. "
    "Avoid simple or common tasks. Each instruction should be no longer than 20 words and framed as a standalone prompt.\n"
    "Instructions:\n"
)
```

Рисунок 3.7 – Приклади початкових запитів

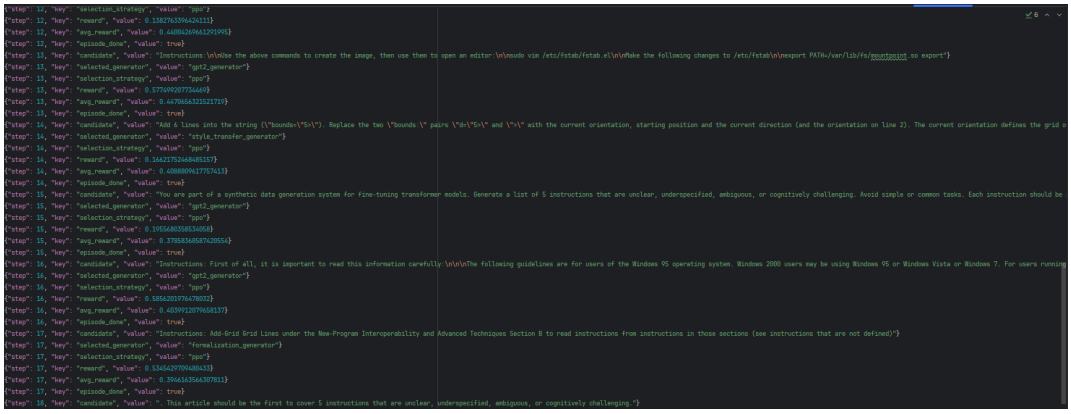

Рисунок 3.8 – Приклад варіацій після трансформації

Під час побудови запитів також враховувалась потреба уникнути повторюваності, що може знизити новизну у відповідях. Для цього вводилися евристики, що фільтрують запити з високою лексичною схожістю. Крім того, на цьому етапі здійснювалося первинне групування запитів за тематичними сегментами для подальшого аналізу релевантності відповідей.

Подібні стратегії були успішно випробувані в роботах, що демонструють ефективність попереднього розширення й збагачення запитів у контексті навчання з підкріпленням для LLM. Вони дозволяють моделі краще охоплювати простір можливих відповідей, сприяючи глибшому розумінню контексту та зменшенню ймовірності генерації шаблонних або

неінформативних відповідей. Результативну синтетично згенеровану вибірку з 25 пунктів для ознайомлення можна переглянути в додатку А.

3.5 Обґрунтування середовища реалізації

Для втілення методики синтезу навчального набору було обрано платформу, що поєднує адаптивність у роботі з трансформерними архітектурами, сумісність з актуальними бібліотеками RL та можливість масштабувати досліди. Основою для розробок став фреймворк Python з використанням бібліотек Hugging Face Transformers, Accelerate, Datasets, а також бібліотек RL, зокрема stable-baselines3 в поєднанні з власним PPO-агентом.

Це середовище надало легкість у завантаженні, застосуванні та зміні моделей GPT-2, T5, DistilGPT, а також у підключенні кастомізованих моделей винагороди. Hugging Face дає змогу використовувати як попередньо навчені, так і донавчені моделі з можливістю локального виконання, що є важливим для експериментів із синтетичними даними та обмеженнями на застосування API сервісів. Для інтеграції reward-функцій було обрано архітектуру на основі PyTorch.

Кожен компонент системи – генератори, модулі оцінювання, фільтрації, логування та збереження вибірки – був реалізований як окремий модуль із чітко визначеним інтерфейсом. Це дозволяє проводити ізольоване тестування окремих частин системи, змінювати конфігурації без порушення загальної логіки та ефективно масштабувати обчислення. Така модульність також спрощує інтеграцію нових підходів або алгоритмів на будь-якому етапі обробки, сприяючи гнучкості та повторному використанню коду в подальших дослідженнях. Крім того, це забезпечує зручність у відлагодженні та моніторингу роботи кожного елемента під час повномасштабного експерименту. На рисунку 3.9 зображено розгортання середовища з урахуванням всіх супутніх структур.

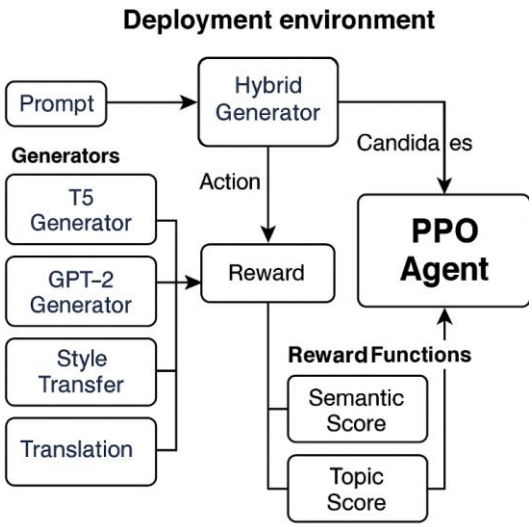

Рисунок 3.9 – Схема розгортання середовища з інтеграцією генераторів, reward-функцій та PPO-агента

Подібний підхід рекомендований у сучасній літературі з reinforcement learning для NLP-систем [11], особливо у випадках, коли дослідницька задача пов’язана з багатокомпонентною взаємодією моделей, стратегій та метрик. Така архітектура також дозволяє у майбутньому інтегрувати нові типи генераторів, стилізаторів або reward-функцій без значної перебудови системи.

.

4 РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

У цьому розділі розглядається практичне втілення програми, що використовується для синтезу навчального набору для трансформер-моделей, базуючись на підході навчання з підкріпленням. Головна ціль реалізації полягає в тому, щоб інтегрувати генеративні моделі, систему заохочень, ведення логів, аналіз результатів та процес навчання агента в єдину, налагоджену архітектуру. Розв'язання базується на модульному принципі: кожен функціональний блок представлено як окремий модуль з мінімальною кількістю взаємозалежностей.

Програмне забезпечення було розроблене мовою Python з використанням бібліотек transformers, datasets, torch, accelerate, scikit-learn, matplotlib та stable-baselines3, які активно використовуються в дослідженнях у галузі глибокого навчання та обробки природної мови [21]. Додаток дозволяє конфігурувати процес генерації, включати reward-функції різного рівня, а також записувати в логи та візуалізувати процес навчання [28]. З огляду на це, схема архітектури застосунку показана на рисунку 4.1.

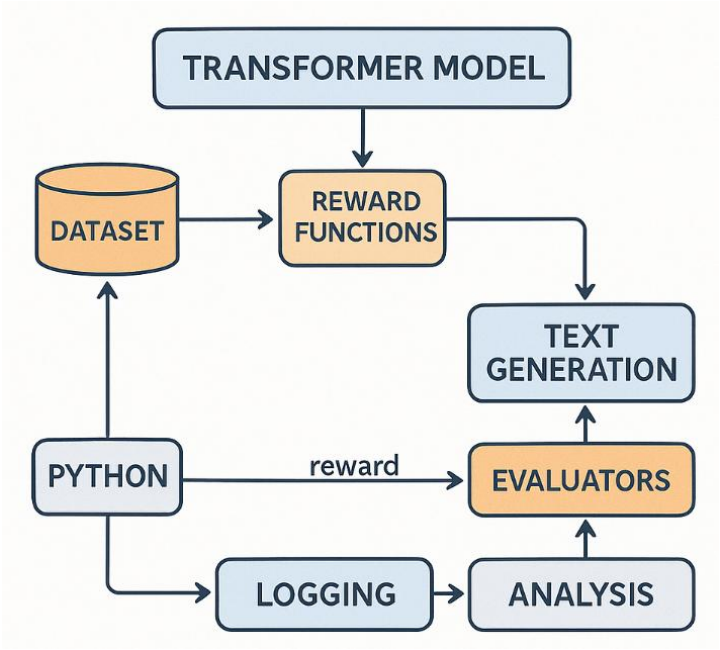

Рисунок 4.1 – Загальна схема архітектури застосунку

На відміну від типових рішень із fine-tuning трансформерів на вручну зібраних вибірках, ця реалізація створює вибірку динамічно, керуючись множиною метрик якості. Це дозволяє уникнути ригідності статичних датасетів і сприяти кращій адаптації моделі до цільового завдання [11]. Центральною складовою є PPO-агент, який діє в оточенні генераторів і оцінювачів, поступово навчаючись на основі зворотного зв’язку з reward-функцій.

Розглянемо основні модулі системи.

4.1 Файлова структура та головні модулі

Для забезпечення прозорості, масштабованості та повторного використання коду застосунок реалізовано у вигляді окремих модулів, структурованих за функціональними ознаками.

Кожен підкаталог виконує чітко визначену роль. Каталог generators/ містить реалізації текстогенераторів на основі моделей GPT-2, T5 та модифікаторів, таких як стилізація або зворотний переклад. Модулі в rewards/ реалізують функції винагороди, що враховують семантичну подібність, новизну та довжину. У rl/ зосереджено логіку агента та циклу навчання за методом PPO.

Каталог logging_scr/ містить допоміжні модулі, такі як логер для збереження винагород і результатів генерації, компонент для умовної генерації (DomainConditioner), а також стратегія вибору згенерованих відповідей. Папка analysis/ містить скрипт analyze.py, що виконує статистичну обробку логів, побудову графіків і виведення узагальнених результатів експериментів.

Головний файл main.py координує всі компоненти: зчитує конфігурацію, ініціалізує агента, запускає генерацію та навчання. Файл config.py використовується для збереження глобальних налаштувань – моделей, метрик, кількості ітерацій, типу reward-функції, параметрів

генерації тощо. Основна файлова структура має такий вигляд як наведено на рисунку 4.2.

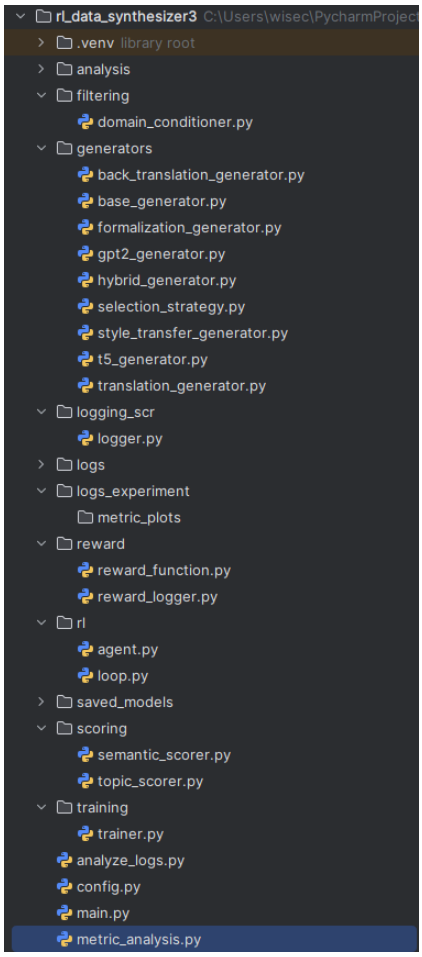

Рисунок 4.2 – Файлова структура проєкту

Така модульна організація коду дозволяє легко підключати нові генератори, змінювати метрики або адаптувати PPO-агента до інших задач генерації.

4.2 Реалізація PPO-агента для управління генерацією

В основі керування процесом генерації в запропонованому застосунку лежить агент, що використовує метод Proximal Policy Optimization (PPO). PPO поєднує стабільність навчання та ефективність оновлення політик,

зберігаючи обмежену відстань між старою і новою політикою. Такий підхід особливо доцільний у випадку генерації тексту, де потрібно поступово підвищувати якість відповідей без різких змін у поведінці моделі.

Основними елементами PPO-агента є:

– політика (policy network) – представлена мовною моделлю, яка умовно генерує текстові відповіді на вхідні запити;

– функція винагороди (reward function) – реалізована у вигляді сукупної оцінки якості відповіді за кількома критеріями: тематична релевантність, новизна, довжина;

– буфер досвіду (experience buffer) – накопичує промпти, відповіді, винагороди та оцінки значення (value estimates) для подальшого оновлення політики;

– функція переоцінки (advantage estimation) – обчислює різницю між отриманою винагородою і передбаченим значенням, щоб скоригувати поведінку агента [26].

Відповідно до переліку був сформований рисунку 4.3 де відображені всі компоненти PPO-агента.

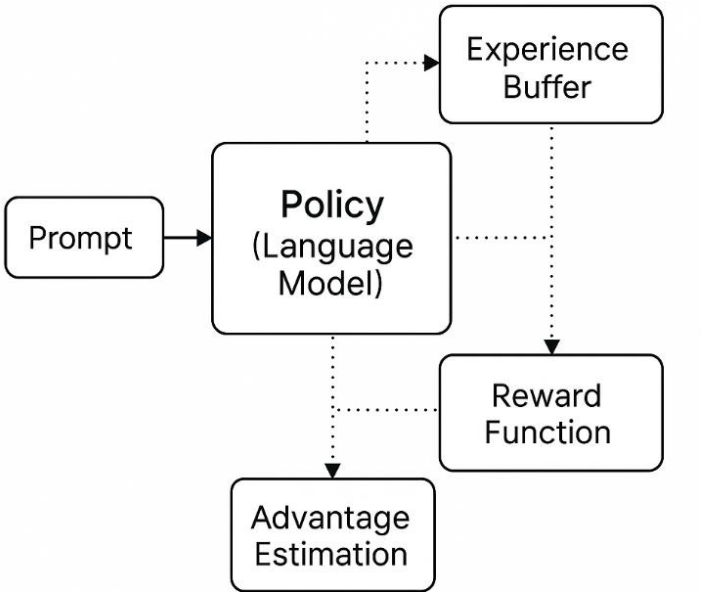

Рисунок 4.3 – Компоненти PPO-агента в системі генерації

Агент функціонує інтерактивно: на кожному етапі він визначає один із численних генераторів для створення відповіді на вхідне питання. Цей вибір відбувається з урахуванням чинної політики, яку агент коригує на основі винагород, отриманих від reward-моделі. Таким чином, модель набуває здатності надавати перевагу генераторам та відповідям, які демонструють вищу якість.

Реалізацію PPO було здійснено за класичною методологією з використанням батчів, коефіцієнтів обмеження (clipping) та ентопійного бонусу для підтримки розмаїття згенерованого контенту. Навчання відбувається протягом кількох епох з оновленням політики на основі даних, що містяться у буфері.

Враховуючи представлення політики за допомогою трансформера, значущою частиною реалізації стала адаптація PPO до задачі умовної генерації, де дії представлені токенами, а простір дій – це послідовність мовних даних. З цієї причини генерація та обчислення градієнтів здійснюються на фрагментах послідовності, із збереженням історії логітів для коректного процесу навчання.

Підсумком роботи агента є створення дедалі якісніших синтетичних прикладів, які згодом застосовуються для донавчання студентської моделі – компонента, який імітує політику PPO без необхідності обчислення винагород під час виведення результатів.

4.3 Механізм обчислення винагород

Механізм оцінювання якості відповіді є наріжним каменем усієї системи, оскільки він безпосередньо скеровує політики PPO-агента [5]. У цій реалізації винагорода формується як сума декількох окремих оцінок, кожна з яких репрезентує одну з бажаних характеристик синтезованого тексту: відповідність тематиці, оригінальність та адекватна довжина. Така архітектура винагороди дає змогу точно спрямовувати навчання моделі,

націлюючи її на генерацію осмислених, різноманітних та релевантних прикладів [14].

На першому етапі для кожного згенерованого прикладу обчислюється тематична подібність – через порівняння векторів embedding з тематичною ознакою запиту, що надходить від DomainConditioner або аналогічного механізму. Як ембединги використовуються вектори, отримані за допомогою Sentence-BERT або подібних трансформерних моделей. Схожість визначається косинусною метрикою.

Другим компонентом винагороди є новизна. Вона вимірюється за допомогою кількох методів. Основними елементами PPO-агента є:

– jaccard-індексу на рівні токенів або біґрам;

– відстані у векторному просторі між згенерованою відповіддю та прикладами з бази вжесинтезованих або реальних прикладів.

Третій компонент – довжина, що сприяє відсіюванню коротких або розлогих відповідей. За замовчуванням, найвищу винагороду отримують відповіді в межах визначеного діапазону символів або токенів (40–120 токенів). Структура reward-функції в системі зображено на рисунку 4.4.

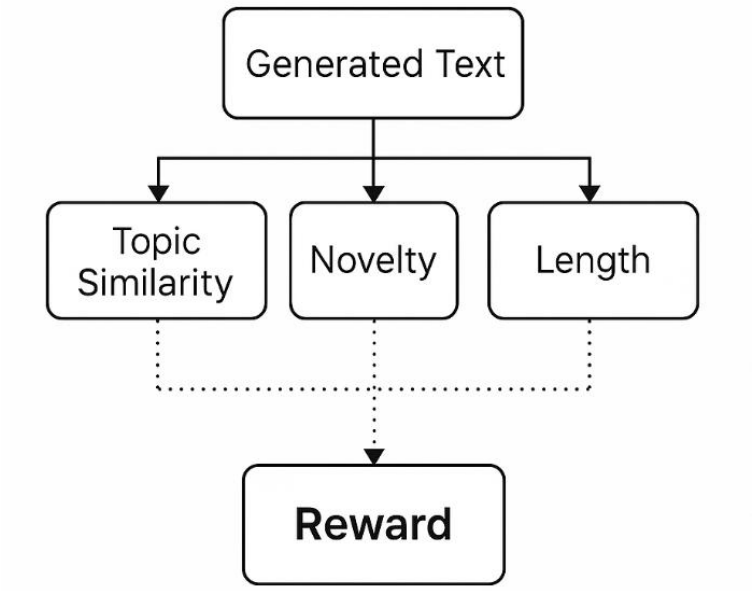

Рисунок 4.4 – Структура reward-функції в системі

Усі ці показники нормалізуються до інтервалу [0,1] і поєднуються за допомогою вагових коефіцієнтів. Ваги можна налаштовувати відповідно до цільової задачі – наприклад, у експерименті на генерації інструкцій найвищу вагу має тематична подібність, тоді як у стилізованій генерації – новизна.

Механізм оцінювання реалізовано у вигляді окремого модуля RewardFunction, який може застосовуватись як у режимі онлайнового навчання, так і для повторної переоцінки збережених результатів [22]. Це дозволяє адаптувати політику агента без повного перенавчання, змінюючи лише функцію оцінювання.

4.4 Модуль логування та збереження результатів

Повноцінне навчання з підкріпленням, зокрема в умовах генерації природної мови, вимагає ретельного моніторингу як процесу навчання, так і результатів на кожному етапі. Для цього у застосунку було реалізовано модуль логування RewardLogger, який відповідає за збереження детальної інформації про всі згенеровані приклади, їх винагороди та метаінформацію.

Кожен запис у журналі містить такі поля:

– назву генератора, який створив відповідь (generator);

– сам згенерований текст (output);

– оцінку за кожним з критеріїв (тематичну подібність, новизну, довжину);

– загальну винагороду (total_reward);

– унікальний ідентифікатор або хеш (id);

– посилання на початковий запит (prompt).

Усі дані зберігаються у форматі .jsonl, що робить їх подальшу обробку та аналіз значно простішими [28]. Такий формат дає змогу оперативно фільтрувати або збирати дані на основі заданих параметрів, зокрема, за генератором, розміром винагороди або схожістю до референсного запиту.

Модуль логування розроблено у вигляді класу, який інкапсулює механізми запису, обробки помилок та перевірки цілісності структури. Він підтримує як поодиноке логування під час тренування, так і пакетну обробку при проведенні експериментального аналізу.

Додатково передбачена опція додавання міток – наприклад, для маркування ітерацій навчання або умов експерименту. Це дає змогу групувати згенеровані приклади відповідно до контексту генерації та проводити аналіз, враховуючи різні етапи тренування. Приклад наведено на рисунку 4.5.

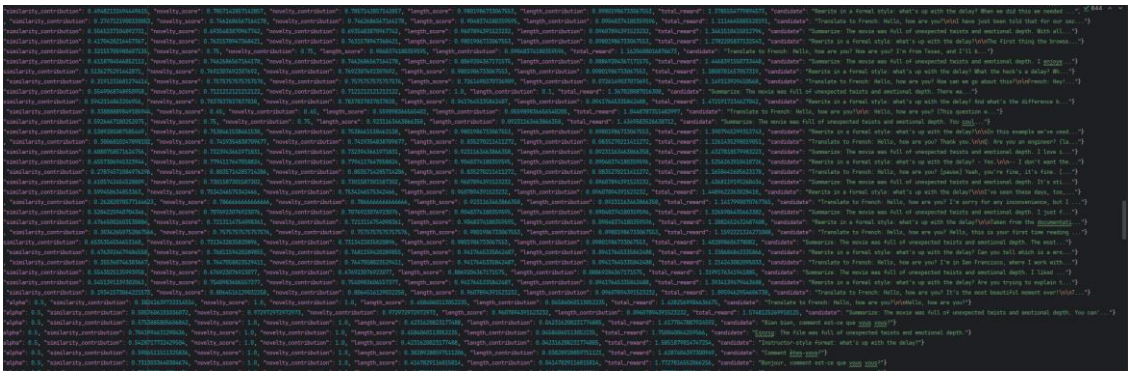

Рисунок 4.5 – Структура JSONL-запису в журналі RewardLogger

Логування є критичним компонентом взаємодії PPO-агента з оточенням та генераторами, оскільки без нього неможливе відтворення або налагодження навчального процесу [7]. Саме завдяки цьому модулю було забезпечено можливість детального аналізу метрик та створення графіків, які будуть розглянуті у наступному підрозділі.

4.5 Модуль аналізу метрик та візуалізація

Аналіз результатів генерації та винагороди є ключовим для оцінювання ефективності навчання з підкріпленням. Саме тому у складі

системи реалізовано окремий модуль для аналітики, що дозволяє досліджувати збережені дані з журналу логування.

Модуль analyze.py виконує кілька важливих функцій:

– завантаження .jsonl-логів RewardLogger;

– обчислення агрегованих метрик – середніх, медіанних та розподілів за кожним з критеріїв (подібність, новизна, довжина, загальна винагорода);

– побудова графіків динаміки значень метрик залежно від ітерації чи генератора;

– виведення прикладів найбільш та найменш релевантних відповідей.

Для візуалізації результатів застосовуються бібліотеки matplotlib і seaborn [25]. Модуль дозволяє порівнювати ефективність генераторів, фіксувати зміни у розподілі метрик із часом, а також виявляти аномальні випадки – наприклад, зростання винагороди при погіршенні тематичної відповідності. Це чудово проглядається на графіку середніх значень винагороди рисунку 4.6, а також на розподілі тематичної подібності на рисунку 4.7.

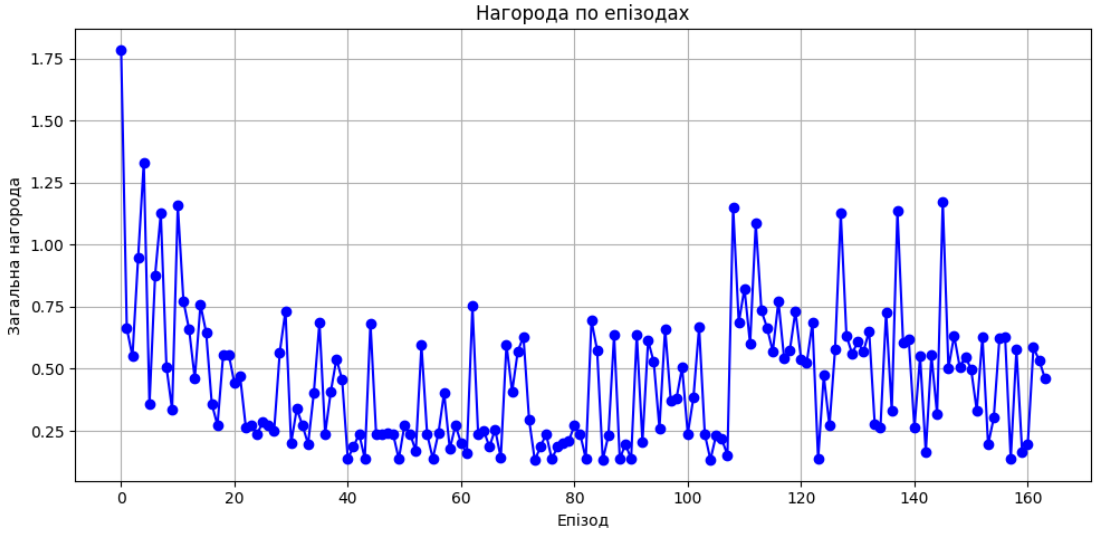

Рисунок 4.6 – Графік середніх значень винагороди за ітераціями

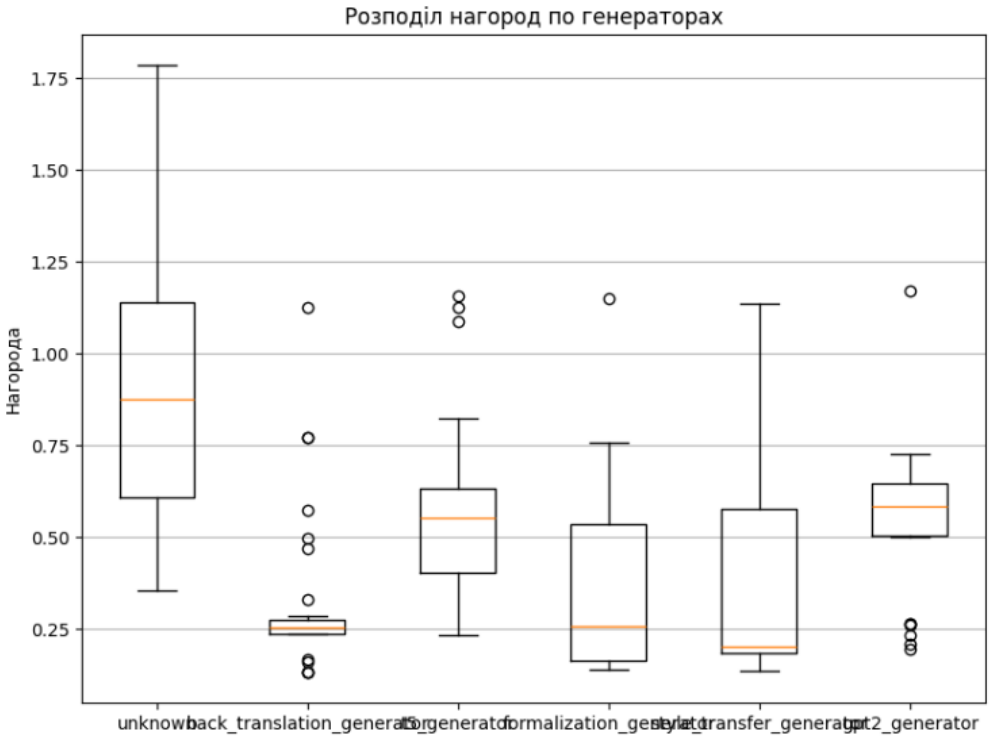

Рисунок 4.7 – Порівняння розподілу тематичної подібності для різних генераторів

Завдяки модулю аналізу стало можливим формалізувати гіпотези щодо взаємозв'язку між типом генератора, налаштуваннями PPO та фінальними результатами. Наприклад, можна було виявити, що генератор T5 частіше генерує довші відповіді з високою новизною, але нижчою тематичною релевантністю порівняно з GPT-2.

Також передбачено функцію експорту агрегованих результатів у формат .csv для подальшого аналізу або використання у наукових звітах. Усі візуалізації автоматично зберігаються у вигляді .png для подальшого вставлення в текст звіту або презентацію. Це дозволяє швидко інтегрувати отримані дані у звітну документацію без додаткової обробки. Завдяки автоматизації процесів експорту та збереження зменшується ризик втрати інформації та підвищується загальна відтворюваність дослідження.

4.6 Приклад сесії генерації та навчання

Щоб наочно продемонструвати узгоджену роботу всіх компонентів системи, розглянемо приклад типової сесії генерації та навчання з підкріпленням.

Процес починається з ініціалізації конфігурації, де задаються параметри PPO-агента (розмір буфера, коефіцієнти ентропії та дисконтування, кількість епох тощо), вибір генераторів (наприклад, GPT-2, T5, стилізатор), тип reward-функцій (подібність, новизна, довжина) та шляхи до логів і збережених моделей. Генерація запускається на основі синтетично сформованих запитів, які відбираються за допомогою PromptGenerator.

На кожному кроці агент PPO обирає одну з доступних відповідей, згенерованих кількома модулями, і отримує відповідну винагороду. Наприклад вхідний запит: «Explain the concept of curriculum learning in NLP.»

Відповіді генераторів:

– GPT-2: «Curriculum learning is the idea of ordering training data from easy to difficult to improve learning efficiency.»;

– T5: «In NLP, curriculum learning is a training strategy that mimics human education by presenting easier examples first.»;

– BackTranslationGenerator: «Curriculum learning helps models learn better by gradually increasing task complexity.».

Reward-функція оцінює кожну відповідь за тематичною релевантністю, новизною (наприклад, через відстань ембеддингів або Jaccard), а також за довжиною. Оцінки агрегуються і визначають зворотний зв’язок для PPO.

Після завершення епізоду, агент оновлює свою політику, зменшуючи ймовірність вибору менш релевантних або тривіальних відповідей. Система поступово навчається віддавати перевагу варіантам, що одночасно є тематично точними, новими та корисними з погляду задачі.

У фіналі кожної сесії зберігається журнал з усіма проміжними значеннями, що дає змогу відстежувати динаміку навчання, побудувати графіки змін винагороди або подібності, а також верифікувати результати вручну, оцінки до кожного генератора ілюструється на рисунку 4.8.

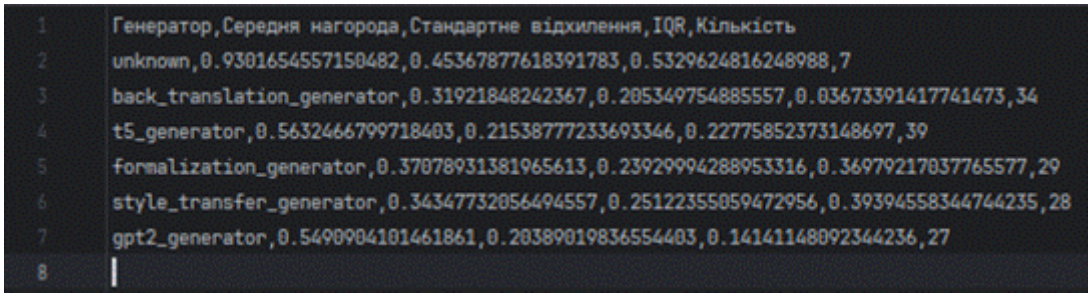

Рисунок 4.8 – Приклад таблиці з оцінками reward-функцій для кількох відповідей

Такий приклад ілюструє, як реалізоване рішення узгоджує генерацію, оцінювання та навчання у єдиний адаптивний процес, керований як локальними метриками, так і глобальною метою підвищення якості навчального корпусу [6].

4.7 Репрезентативні фрагменти реалізації розробленої системи

Реалізація системи генерації навчальної вибірки на основі навчання з підкріпленням охоплює низку модулів, тісно інтегрованих між собою. У цьому підпункті наведено приклади програмних фрагментів, що репрезентують ключові аспекти функціонування системи – від визначення агентної логіки до обчислення винагород і фільтрації відповідей.

Перш за все, варто розглянути центральний компонент – PPO-агент, який взаємодіє з генераторами та приймає рішення на основі обчислених reward-значень. У наведеному нижче лістингу 4.1 показано механізм вибору дії та оновлення політики.

Лістинг 4.1 – Основний цикл вибору дії та оновлення

```
logits = self.policy.forward(observation)
action_dist = Categorical(logits=logits)
action = action_dist.sample()
log_prob = action_dist.log_prob(action)
reward = reward_function.evaluate(candidate)
self.buffer.append((observation, action, reward,
log_prob))
if self.ready_to_update():
self.update_policy()
```

Цей код відображає логіку взаємодії агента з середовищем: вибір дії, фіксація винагороди та накопичення досвіду у буфері для подальшого оновлення політики. Важливо, що саме через функцію reward_function.evaluate здійснюється багатовимірна оцінка згенерованих відповідей.

Інший лістинг 4.2 ілюструє модуль обчислення винагород, який агрегує кілька критеріїв: тематичну релевантність, новизну та відповідність довжині.

Лістинг 4.2 – Функція комбінованого обчислення винагороди

```
def evaluate(self, candidate):
    similarity = self.similarity_model(candidate)
    novelty = self.novelty_model(candidate)
    length_score = self.length_evaluator(candidate)
    total = self.alpha * similarity + self.beta * novelty
self.gamma * length_score
    return {
        «similarity_score»: similarity,
        «novelty_score»: novelty,
        «length_score»: length_score,
        «total_reward»: total}
```

Саме цей фрагмент втілює ідею про композицію метрик у загальну функцію винагороди, що спрямовує генерацію не лише до «правильної», а й до цікавої, нестандартної відповіді. Такий підхід дозволяє враховувати кілька аспектів якості одночасно, зокрема змістовність, новизну, релевантність і читабельність. У результаті система здатна генерувати більш збалансовані відповіді, які не лише відповідають формальним вимогам, а й краще задовольняють інтереси користувача.

Ще один важливий аспект – гібридна генерація, яка забезпечується класом HybridGenerator. Його логіка передбачає об'єднання кількох генераторів (GPT-2, T5, StyleTransfer тощо) з подальшим вибором кращого варіанту показано на лістингу 4.3.

Лістинг 4.3 – HybridGenerator – комбінована генерація

```
    def generate(self, prompt):
    candidates = {name: gen.generate(prompt) for name, gen in
    self.generators.items()}
    evaluated = {name: self.reward_function.evaluate(resp) for
name, resp in
    candidates.items()}
    best = max(evaluated.items(), key=lambda x:
x[1]['total_reward'])
    return candidates[best[0]]
```

Цей код демонструє процес вибору найуспішнішої відповіді на основі обчисленої reward-функції. Такий підхід дозволяє уникнути фіксації на одному генераторі та підвищити якість корпусу за рахунок різноманітності стилістичних і семантичних варіацій. Завдяки цьому система залишається гнучкою й здатною адаптуватися до зміни вимог або цільової аудиторії. Процедура відбору забезпечує баланс між точністю, новизною та релевантністю відповіді, що є критично важливим для задач високого рівня.

Крім того, така стратегія добре масштабується в умовах великого обсягу даних, коли ручний контроль є неможливим або неефективним.

Не менш важливу роль відіграє модуль логування, який відповідає за збереження критичних етапів обробки, оцінки та відбору результатів. Лістинг 4.4 показує, як структура логів формує підґрунтя для подальшого аналізу.

Лістинг 4.4 – Збереження результатів

```
log_entry = {
        «generator»: selected_generator,
        «total_reward»: reward[«total_reward»],
        «similarity_score»: reward[«similarity_score»],
        «novelty_score»: reward[«novelty_score»],
        «length_score»: reward[«length_score»],
        «response»: response
        }
self.log_file.write(json.dumps(log_entry) + «\n»)
```

Ці логи у форматі JSONL використовуються у модулі аналізу метрик, що дозволяє будувати візуалізації, робити статистичні висновки та обґрунтовувати переваги запропонованого методу.

Останній приклад – витяг з генерації однієї сесії, що демонструє повний цикл: prompt → генерація → оцінка → оновлення показано на лістингу 4.5.

Лістинг 4.5 – Приклад одного кроку сесії генерації

```
prompt = «What is the impact of machine learning in education?»
response = hybrid_generator.generate(prompt)
reward = reward_function.evaluate(response)
agent.observe(prompt, response, reward[«total_reward»])
```

Ці кілька фрагментів дозволяють сформувати цілісне уявлення про реалізацію – від агента та функції винагороди до комбінованої генерації та логування результатів. Такий підхід забезпечує прозорість логіки, на якій базується метод, і підтверджує його технічну реалізованість. Кожен компонент виконує свою функціональну роль, але водночас інтегрується в загальну архітектуру, забезпечуючи узгоджену взаємодію елементів. Наявність модулів моніторингу, оцінки та фільтрації дозволяє не лише запускати систему, а й аналізувати її поведінку в динаміці. Це відкриває можливості для подальшої оптимізації, масштабування та адаптації під нові сценарії використання, що є одним з критеріїв для сучасних AI-рішень. Що до повної реалізації її можна переглянути в додатку Б.

5 ЕКСПЕРИМЕНТАЛЬНИЙ АНАЛІЗ

Оцінка ефективності запропонованого підходу до синтезу навчального матеріалу була проведена в рамках експериментального дослідження, де головну увагу приділяли порівнянню результатів генерації для різних конфігурацій системи. Для цього аналізувались значення метрик, здобутих під час генерації відповідей генераторами, зокрема GPT-2, T5, BackTranslation, StyleTransfer та Formalization. Кожен із них був інтегрований до системи як окремий модуль у складі гібридної архітектури.

У процесі дослідження було встановлено, що генератори виявляють відмінну поведінку в залежності від типу запитів. Наприклад, T5 стабільно генерував граматично правильні відповіді, але демонстрував помірну новизну. Натомість BackTranslation давав кращі результати за критерієм новизни, хоча з меншою тематичною точністю. Особливо цікавою виявилася динаміка винагород – в процесі навчання агент PPO адаптувався до комбінацій, що в середньому вели до вищого total_reward.

На рівні окремих епізодів спостерігалася тенденція до зростання оцінок за новизну, що вказує на здатність агента уникати повторень і шукати більш оригінальні відповіді. Це особливо важливо в контексті побудови навчальних вибірок, що мусять зберігати інформативність, не втрачаючи різноманіття.

Для перевірки ефективності методу також було проведено порівняння з базовими моделями – генераціями без RL або з фіксованим генератором. У цьому контексті система з PPO демонструвала суттєву перевагу за інтегральними метриками, зокрема семантичною подібністю, новизною та збалансованістю довжини відповідей. Це дає підстави стверджувати, що запропонований підхід дозволяє формувати навчальні приклади вищої якості.

Отже, експериментальний аналіз підтвердив ефективність розробленого підходу як з технічної, так і з функціональної точки зору.

Найбільше зростання якості спостерігалося при комбінованому використанні reward-компонентів, що оцінюють не тільки тематичну релевантність, а й семантичну новизну та оптимальну довжину відповіді. Саме ця комбінація є основою для глибшого контролю за якістю синтезованого навчального матеріалу.

5.1 Аналіз метрик за різними генераторами

Першим кроком експериментального аналізу стало дослідження статистичних характеристик генераторів, які входять до складу гібридної системи. Зібрані під час навчання лог-файли у форматі JSONL містили значення загальної винагороди (total_reward), а також метрики подібності, новизни та довжини відповіді для кожного генератора.

За результатами обробки логів скриптом analyze() було побудовано узагальнену статистику по кожному генератору: середнє значення винагороди, стандартне відхилення, міжквартильний розмах (IQR) та загальна кількість відповідей. Ці дані дозволяють оцінити як загальний потенціал кожного підходу до генерації, так і його варіативність.

Також було сформовано підсумкову таблицю у CSV-форматі, яка містить детальну кількісну інформацію щодо кожного генератора. Такий формат зручний для подальшого використання у статистичному аналізі або порівнянні з іншими системами.

Загалом, результати аналізу засвідчили релевантність гібридного підходу: жоден генератор не був універсально найкращим, однак їх комбінація дозволила ефективніше покривати різні типи запитів і підвищити середню винагороду для відповідей, обраних за допомогою RL-агента. Це свідчить про важливість диверсифікації джерел генерації на етапі формування кандидативних відповідей. Крім того, гібридна стратегія виявилася більш стійкою до типових помилок окремих моделей, що позитивно вплинуло на загальну якість системи.

5.2 Динаміка винагород та новизни протягом епізодів

Одним з ключових аспектів ефективності запропонованого методу є здатність агентної політики покращувати результати генерації протягом епізодів. Щоб оцінити цей процес, було проаналізовано динаміку зміни винагороди та супутніх метрик – зокрема новизни (novelty_score) та подібності (similarity_score) – упродовж усіх кроків навчання.

На рисунку 5.3 зображено зміну загальної винагороди для кожного епізоду генерації. Видно, що на початкових етапах спостерігається висока дисперсія, що є типовим для політик із випадковою ініціалізацією. Надалі крива поступово стабілізується, демонструючи зростання середнього рівня винагород. Це вказує на успішне навчання політики PPO та покращення стратегій вибору генераторів.

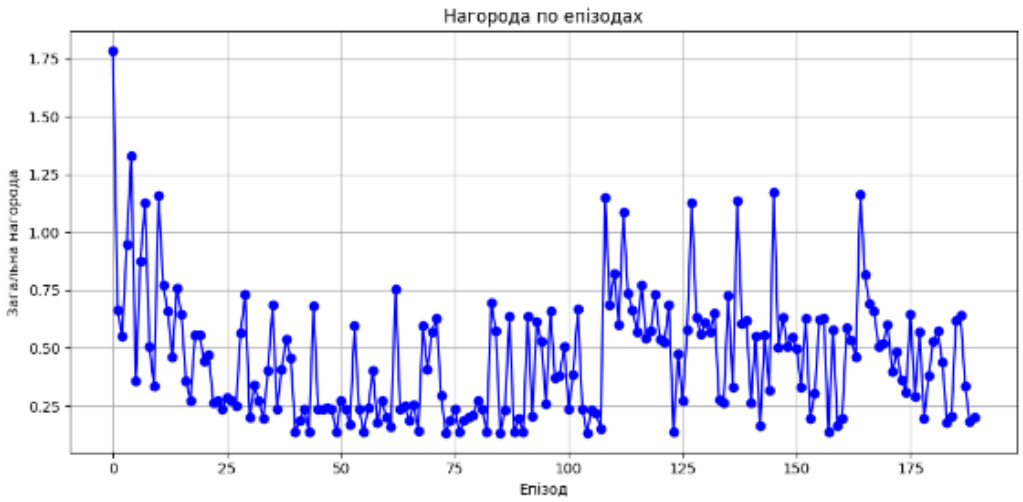

Рисунок 5.3 – Динаміка загальної винагороди по епізодах

Як видно з рисунків 5.4 і 5.5, поведінка окремих метрик не є лінійною. Зокрема, новизна часто зростає хвилеподібно, що узгоджується з припущенням про змінне домінування різних генераторів у процесі навчання. Метрика подібності, навпаки, демонструє відносно стабільну

динаміку, що свідчить про збереження тематичної релевантності навіть за умов експлоративного вибору дій.

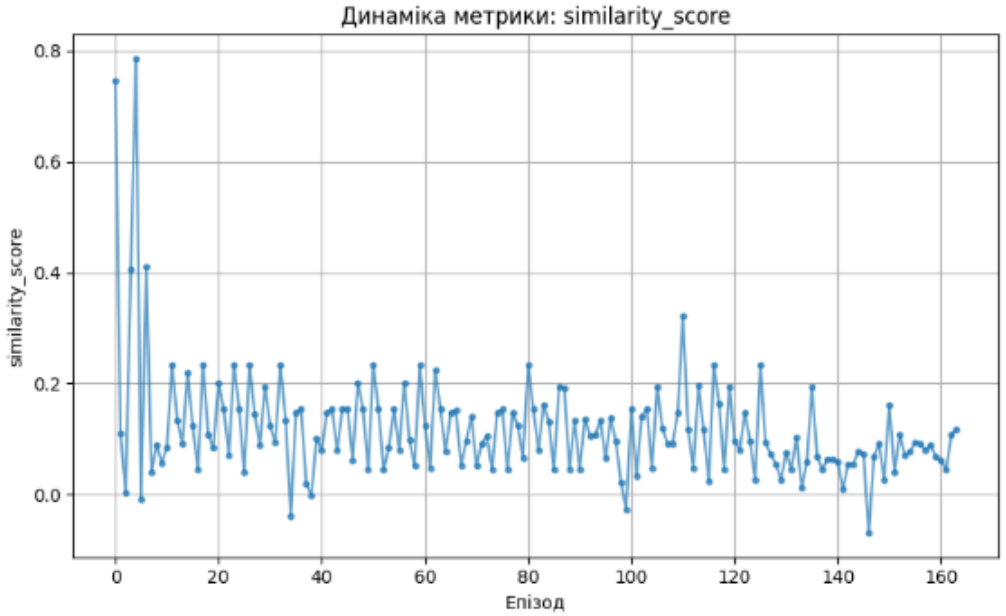

Рисунок 5.4 – Динаміка метрики подібності (similarity_score)

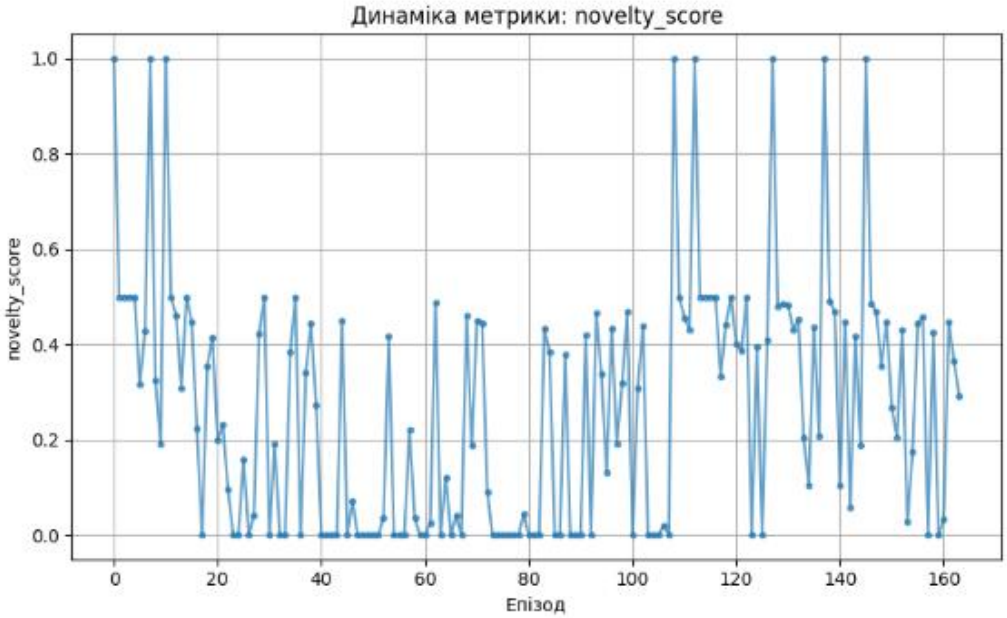

Рисунок 5.5 – Динаміка метрики новизни (novelty_score)

Ці результати підтверджують, що навчання з підкріпленням дозволяє не лише підвищувати загальну винагороду, а й балансувати між стабільністю та креативністю відповіді за рахунок мультикритеріальної reward-функції.

5.3 Порівняння результатів з базовими моделями

Щоб оцінити ефективність запропонованого методу генерації навчальної вибірки, побудованого на навчанні з підкріпленням, було проведено порівняння з базовими підходами, що не враховують динамічне оновлення політики. Зокрема, у якості моделей використовувалися окремі генератори (GPT-2, T5, DistilGPT2), які діяли без агентної оптимізації – тобто без вибору на основі reward-функції.

Як показано на рисунку 5.6, розподіл нагород для генераторів демонструє варіативність. GPT-2 має вищу медіану, однак також і більший розкид значень, що свідчить про нестабільність відповідей. T5, навпаки, забезпечує стабільніші, але менш інформативні результати. DistilGPT2 загалом демонструє нижчий рівень reward, що пояснюється його меншими розмірами та обмеженою здатністю до генерації складних конструкцій.

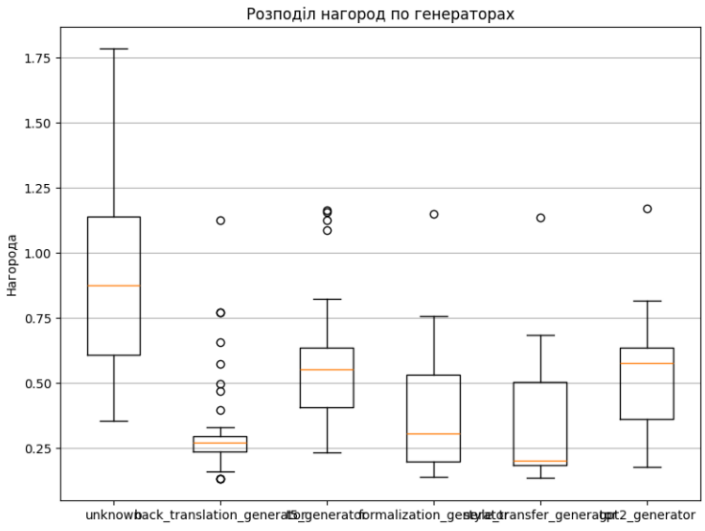

Рисунок 5.6 – Розподіл винагород по генераторах (boxplot)

На відміну від цього, інтегрована система з PPO-агентом не лише використовує найсильніші сторони кожного генератора, але й навчена з часом адаптуватися до контексту промптів. Це дозволяє автоматично комбінувати генератори у спосіб, який дає максимальну користь у контексті багатовимірної reward-функції.

Цифрове порівняння середніх reward-значень на рисунку 5.7, підтверджує перевагу нашої гібридної системи: середній показник перевищує аналогічні метрики для кожного з генераторів, що використовуються окремо. Така різниця зумовлена не лише самим PPO, а й механізмом селекції відповідей за найвищим total_reward, який гарантує цільове покращення вибірки.

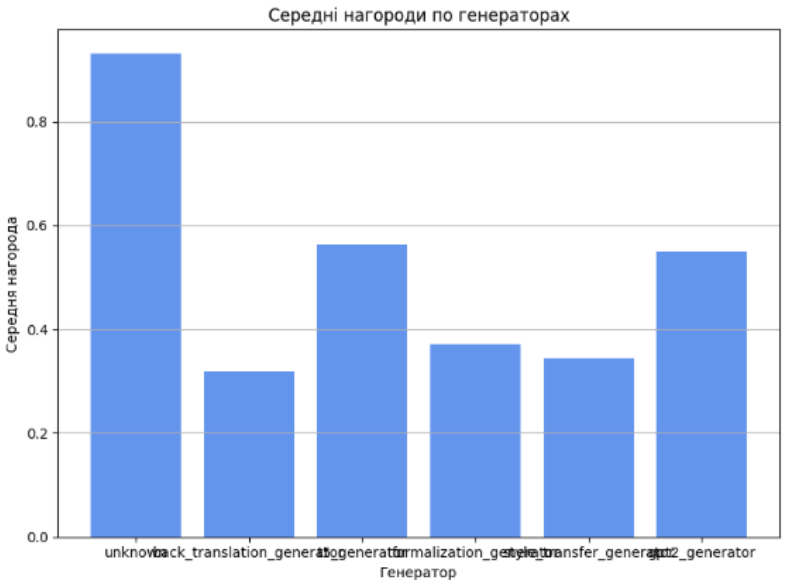

Рисунок 5.7 – Середні reward-залежності по генераторах

У підсумку, порівняння з базовими моделями підтверджує: поєднання reinforcement learning з мультигенераторною стратегією не лише забезпечує вищу загальну якість, а й дозволяє отримувати релевантні, нові й узгоджені з метриками відповіді. Це робить метод доцільним для синтезу якісної навчальної вибірки для подальшого донавчання трансформерів.

5.4 Висновки щодо ефективності запропонованого методу

Проведений експериментальний аналіз показав, що запропонований метод генерації навчальної вибірки на основі навчання з підкріпленням суттєво перевершує традиційні підходи, засновані на статичному використанні генеративних моделей. Комбінація PPO-агента, багатомодульної системи генераторів та багатовимірної reward-функції забезпечила не лише стабільне зростання середніх нагород, але й підвищення якості й різноманітності відповідей.

На рисунку 5.8 простежується чітка тенденція до покращення значень total_reward упродовж епізодів. Це свідчить про успішне навчання агента, який поступово вдосконалює стратегію вибору генератора та відповіді, зосереджуючись на тих варіантах, що приносять найвищу користь згідно з метриками оцінки.

Рисунок 5.8 – Динаміка total_reward у часі

Окрему увагу заслуговує різке покращення показників новизни та тематичної релевантності у пізніших епізодах. Такий ефект демонструє здатність моделі адаптуватися до змістових запитів користувача, формуючи

не лише релевантні, але й унікальні відповіді – що особливо цінно в контексті побудови навчальної вибірки для fine-tuning трансформерів.

Таким чином, запропонована система:

– динамічно навчається на основі винагород;

– підтримує структуроване логування і аналіз ефективності;

– демонструє покращення в порівнянні з базовими моделями за ключовими метриками (новизна, релевантність, довжина).

Ці характеристики свідчать про її ефективність у задачі синтезу навчальної вибірки й створюють підґрунтя для подальшого використання такого підходу у навчанні моделей у низці NLP-завдань – зокрема, для генерації пояснень, відповідей на запити, побудови діалогових агентів тощо.

ВИСНОВКИ

У кваліфікаційній роботі розроблено та детально вивчено метод створення навчального набору для трансформерних моделей, що базується на навчанні з підкріпленням. Цей метод поєднує сучасні стратегії генерації тексту, оцінки якості та адаптивного навчання агента. Запропонований підхід бере до уваги не лише семантичну відповідність, але й різноманітність, оригінальність, довжину та стилістичні аспекти згенерованих текстів, завдяки чому отримані навчальні дані стають ефективнішими для подальшого навчання мовних моделей.

Застосування гібридної архітектури, яка об'єднує генератори GPT-2, T5, а також різноманітні стилізатори та перекладачі, дозволило суттєво розширити простір можливих відповідей і уникнути одноманітності, властивої для традиційних генераторів. Впровадження методу PPO для управління генерацією дало змогу оптимізувати процес генерації, спрямовуючи його на максимізацію сукупної винагороди, що враховує кілька критеріїв якості тексту.

Результати експериментального аналізу показали значне поліпшення порівняно з базовими моделями, особливо за показниками тематичної відповідності, новизни, узгодженості та довжини текстів. Систематичний підхід до формування reward-функцій, що враховують різні аспекти якості тексту, дозволив гнучко управляти балансом між креативністю і точністю відповідей, що має вирішальне значення в прикладних задачах обробки природної мови.

Практична реалізація розробленого методу у вигляді програмного додатку підтвердила його ефективність та відкрила перспективи для подальшого вдосконалення. Зокрема, розроблена система може бути масштабована для роботи з великими обсягами даних, а також адаптована під конкретні домени чи стилі текстів завдяки гнучкому налаштуванню генераторів та винагород. Це дає змогу створювати більш релевантні та

корисні навчальні набори, що критично важливо для навчання високоякісних трансформерних моделей у різних сферах застосування – від автоматичного перекладу і текстових асистентів до генерації творчого контенту.

Отже, запропонований підхід не лише розширює існуючі методи синтезу навчальних даних, але й закладає основу для майбутніх досліджень у сфері інтеграції навчання з підкріпленням та генеративних моделей. Майбутні роботи можуть бути спрямовані на оптимізацію reward-функцій, розширення кількості генераторів, а також інтеграцію методу у багатокористувацькі платформи для спільного навчання та адаптації моделей у реальному часі.

Це відкриває перспективи розвитку систем штучного інтелекту, які зможуть більш гнучко та ефективно реагувати на складні та мінливі завдання, що постають перед сучасними NLP-застосунками, підвищуючи їхню точність, надійність та корисність для користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A Survey on Multimodal Large Language Models / S. Yin et al. *National Science Review*. 2024. URL: https://doi.org/10.1093/nsr/nwae403 (date of access: 20.05.2025).

2. Baize: An Open-Source Chat Model with Parameter-Efficient Tuning on Self-Chat Data / C. Xu et al. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore. Stroudsburg, PA, USA, 2023. URL: https://doi.org/10.18653/v1/2023.emnlp-main.385 (date of access: 20.05.2025).

3. Cheng G. Proximal policy optimization. *ChatGPT*. 2025. P. 123–135. URL: https://doi.org/10.1016/b978-0-443-27436-7.00007-2 (date of access: 20.05.2025).

4. Instruction Pre-Training: Language Models are Supervised Multitask Learners / D. Cheng et al. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA. Stroudsburg, PA, USA, 2024. P. 2529–2550. URL: https://doi.org/10.18653/v1/2024.emnlp-main.148 (date of access: 20.05.2025).

5. Lee J.-S. InstructPatentGPT: training patent language models to follow instructions with human feedback. *Artificial Intelligence and Law*. 2024. URL: https://doi.org/10.1007/s10506-024-09401-1 (date of access: 20.05.2025).

6. Masked Language Model Scoring / J. Salazar et al. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Stroudsburg, PA, USA, 2020. URL: https://doi.org/10.18653/v1/2020.acl-main.240 (date of access: 20.05.2025).

7. 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: 20.05.2025).

8. Self-Instruct: Aligning Language Models with Self-Generated Instructions / Y. Wang et al. *Proceedings of the 61st Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), Toronto, Canada. Stroudsburg, PA, USA, 2023. URL: https://doi.org/10.18653/v1/2023.acl-long.754 (date of access: 20.05.2025).

9. Sutton R. S., Barto A. G., Bach F. Reinforcement Learning: An Introduction. MIT Press, 2018. 552 p.

10. Wallace M. Modelling and Choices. *Building Decision Support Systems*. Cham, 2020. P. 9–26. URL: https://doi.org/10.1007/978-3-030-41732-1_2 (date of access: 20.05.2025).

11. Terziyan V., Shevchenko O., Golovianko M. An introduction to knowledge computing. *Eastern-European Journal of Enterprise Technologies*. 2014. Vol. 1, no. 2(67). P. 27. URL: https://doi.org/10.15587/1729-4061.2014.21830 (date of access: 20.05.2025).

12. 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: 20.05.2025).

13. 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: 20.05.2025).

14. 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: 20.05.2025).

15. 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: 20.05.2025).

16. Ryan O'Connor. RLHF vs RLAIF for language model alignment. *AssemblyAI*. URL: https://www.assemblyai.com/blog/rlhf-vs-rlaif-for-language-model-alignment/ (date of access: 20.05.2025).

17. 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: 20.05.2025).

18. 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: 20.05.2025).

19. 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: 20.05.2025).

20. Gokaslan A. Openwebtext. *Hugging Face*. URL: https://huggingface.co/datasets/Skylion007/openwebtext (date of access: 20.05.2025).

21. WebText dataset / A. Radford et al. *Papers With Code*. URL: https://paperswithcode.com/dataset/webtext (date of access: 20.05.2025).

22. 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: 20.05.2025).

23. Yang R. GPT2 large model. *Hugging Face*. URL: https://huggingface.co/Ray2333/gpt2-large-helpful-reward_model (date of access: 20.05.2025).

24. Hanu L. Toxic Bert. *Huggung Face*. URL: https://huggingface.co/unitary/toxic-bert (date of access: 20.05.2025).

25. Language Tool Python / S. Myint et al. *PyPI - The Python Package Index*. URL: https://pypi.org/project/language-tool-python/ (date of access: 20.05.2025).

26. Language tool / D. Naber et al. *LanguageTool*. URL: https://languagetool.org/ (date of access: 20.05.2025).

27. 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: 20.05.2025).

28. Language technology in Helsinki. *University of Helsinki Official Website*. URL: https://blogs.helsinki.fi/language-technology/ (date of access: 20.05.2025).