

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Кваліфікаційна наукова
праця на правах рукопису

ЗДЕБСЬКИЙ ПЕТРО ВАСИЛЬОВИЧ

УДК 004.8

**ДИСЕРТАЦІЯ
МЕТОДИ ТА ЗАСОБИ БАГАТОЕТАПНОЇ ГЕНЕРАЦІЇ СЕМАНТИЧНО
КОРЕКТНИХ ТЕКСТІВ**

124 – Системний аналіз
Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей,
результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Здебський П. В.

Науковий керівник – Берко Андрій Юліанович

д.т.н., професор

Львів - 2024

АНОТАЦІЯ

Здебський П. В. Методи та засоби багатоетапної генерації семантично коректних текстів. – Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 124 “Системний аналіз” – Національний університет “Львівська політехніка”, Міністерство освіти і науки України, Львів, 2024.

Зміст анотації. Дисертація присвячена розробці методів та засобів багатоетапної генерації семантично коректних текстів за допомогою великих мовних моделей.

На сьогоднішній день залишається невирішеною задача узгодження мовних моделей, або у широкому розумінні проблема контролю над штучним інтелектом. Вона полягає у тому, що моделі формально вирішують задачу, але не так, які очікують від них дослідники. Також залишається проблема із непослідовністю мовних моделей, тобто вони можуть генерувати висловлювання, що суперечать за змістом.

Було поставлено задачу покращити консистентність знань великих мовних моделей через повторну перевірку згенерованого тексту, що має на меті покращити узгодження та зменшити кількість суперечливих висловлювань мовних моделей.

Для вирішення поставленої задачі було вирішено використовувати логічні правила для збору тренувальних даних і на них дотреноувати модель, бо знання, консистентність яких ми прагнемо покращити, моделі вивчають із тренувальних даних. Моделі на вхід передається питання природною мовою з очікуванням отримати приклад, що слідує певному правилу логіки (наприклад імплікації). Після цього модель автоматичним способом визначається у яких випадках модель конфліктує із своїми попередніми висловлюваннями. Потім, на таких конфліктуючих висловлюваннях дотреноується модель, щоб зменшити ймовірність таких помилок. Нових фактів модель не вивчає, а єдині нові знання б передавались моделі, це певні закони логіки, які представлені більш явно ніж у її початкових тренувальних даних.

Пізніше замість обширного “слідuje певному правилу логіки” було використано “слідuje запиту користувача”. Тобто користувач робить запит, щоб згенерувати текст за певним правилом чи інструкцією, а та ж модель після генерування тексту перевіряє чи він не суперечить запиту користувача. Можна сказати, що тут використовується другий закон логіки, що два протилежні судження не можуть бути істинними одночасно. Тобто якщо модель згенерувала текст за запитом і при генеруванні вважає його правильним, а при повторній перевірці неправильним, то тут і виникає суперечність. Потім зразки таких суперечливих відповідей додаються до набору прикладів поведінки, щоб модель навчилася не допускати таких помилок. Також замість дотреновування моделі було вирішено використовувати конструювання промπτу із вбудовуванням у нього нових прикладів поведінки.

Для перевірки такого підходу вибрано імплікацію між двома висловлюваннями. У цій роботі імплікація береться не у строго у математичному значенні, а у контексті задачі логічного виведення (Natural Language Inference), тобто що із першого висловлювання випливає друге. Наприклад, модель може допустити помилку і сказати, що із висловлювання “Об’єкт є студентом” можна зробити висновок, що “Об’єкт вивчає фізику”, хоча те, що об’єкт є студентом не обов’язково означає, що він вивчає фізику.

У процесі дослідження було протестовано такий підхід і вирішено додати крок перевірки відкидання згенерованого тексту, якщо він не відповідає запиту користувача. Тобто спочатку генерується текст на основі великої мовної моделі із вбудованими новими прикладами поведінки у промπτ, а після цього відбувається перевірка згенерованого тексту. Генерація тексту відбувається доки та сама модель не скаже, що згенерований текст відповідає тій задачі, яку перед нею ставив користувач. Та модель, що перевіряє з точки зору архітектури і тренувальних даних є такою самою моделлю, що генерує текст. Єдиною відмінністю є те, що їй ставиться задача класифікації, а не генерації тексту. Це реалізується через зміну промπτу.

Наукова новизна отриманих результатів:

1. Вперше розроблено модель процесу генерування текстів, яка використовує конструювання промπτу для повторної перевірки згенерованого тексту, що дало можливість підвищити рівень семантичної коректності результату.
2. Вперше розроблено метод генерування тексту, який включає велику мовну модель, що використовується для генерування тексту, у якості дискримінатора для перевірки узгодженості результатів із запитом користувача, що дало змогу підвищити точність генерування.
3. Удосконалено метод верифікації згенерованого тексту, що на відміну від відомих методів не вимагає зворотного зв'язку людини, що дало змогу повністю автоматизувати процес генерування.

У **першому розділі** розглянуто задачу логічного виведення та аргументовано її складність для перевірки точності великих мовних моделей.

Досліджено транзитивне відношення у контексті задачі логічного виведення та наборів даних для неї. Аналіз літературних джерел свідчить, що для деяких задач опрацювання природної мови можна досягти високої точності за допомогою простих евристик.

Проаналізовано закони формальної логіки у контексті опрацювання природної мови, а також досліджено проблему логічних помилок у мовних моделях та достовірність їх результатів. Розглянуто вимоги до наборів даних для опрацювання природної мови та стратегії вибору досліджень у галузі штучного інтелекту.

Після аналізу була обрана стратегія використання попередньо натренованих великих мовних моделей. Виявлено, що нейронні мережі з архітектурою Трансформер та великою кількістю параметрів показують найкращі результати для опрацюванні природної мови.

У **другому розділі** було досліджено основні способи представлення тексту у числовому форматі. Такі як торба слів, підхід на основі TF-IDF, word2vec, та використання останнього шару нейронних мереж.

Обґрунтовано корисність використання великих мовних моделей у якості судді для підтримки якості дискусій. Запропоновано розглядати задачу як

узгодження моделей із потребами корисувачів, окрім як покращення консистентності знань великих мовних моделей.

Виявлено, що для вирішення задачі узгодження використовується Proximal Policy оптимізація (PPO) та навчання із підкріпленням на основі зворотного зв'язку людини (RLHF). Також було здійснено формальний опис запропонованого підходу.

У **третьому розділі** перевірено на практиці здатність мовної моделі GPT-4 робити логічні висновки та її здатність перевіряти згенерований текст у контексті задачі логічного виведення.

Була запропонована метрика для оцінки впевненості в правильності відповідей великих мовних моделей. Описано алгоритм використання великої мовної моделі для перевірки її відповідей.

Також були представлені діаграми варіантів використання, послідовності, діяльності, класів та розгортання для двох алгоритмів генерування тексту: використання дискримінатора для поповнення тренувального набору даних та використання дискримінатора для відсіювання неправильних відповідей. На основі цих двох алгоритмів сформульовано метод генерування текстового контенту.

У **четвертому розділі** було представлено постановку початкового експерименту з використанням 80 прикладів і випадковим розбиттям на дві групи для тестування базової моделі GPT-4 та її версії із розширеним промптом. Після цього експерименту було вирішено розширити обсяг тестового набору даних і виділити окремий набір для тестування всіх моделей, щоб імітувати реальні умови використання моделі.

Були реалізовані два підходи: один із використанням дискримінатора для поповнення тренувального набору даних, та інший із використанням дискримінатора для відсіювання неправильних відповідей.

Проведено порівняння точності обох підходів, і запропоновано використовувати метод, що включає обидва алгоритми генерування тексту послідовно через його кращу точність у реальних умовах використання. Також був проведений описовий аналіз тестового набору даних та згенерованого тексту.

Розроблено прототип програмного забезпечення, яке використовує велику мовну модель для перевірки тексту, що дозволило збільшити якість тексту, що генерується. Це досягнуто за рахунок використання моделі штучного інтелекту у якості дискримінатора для розмічування згенерованого тексту.

Розроблене програмне забезпечення буде особливо корисним у задачах пов'язаних із генеруванням текстового контенту, у яких є багато різних варіантів правильно згенерованого тексту, але легко перевірити його правильність. Наприклад, якщо користувач хоче, щоб діалогова система відповідала у офіційно-діловому, а не розмовному стилі. Є багато можливих відповідей в процесі діалогу, але можна однозначно сказати чи текст згенерований у офіційно-діловому стилі, чи ні. Прикладами задач, у яких запропонований підхід буде менш корисним є інформаційний пошук (information retrieval), та автоматизоване реферування (automatic text summarization). Це пов'язане з тим, що інформаційний пошук має лише одну правильну відповідь, а автоматизоване реферування не має однозначного критерію правильності.

Ключові слова: gpt-4, генерація тексту, великі мовні моделі, задача узгодження, задача логічного виведення, штучний інтелект, опрацювання природної мови.

ABSTRACT

Zdebskyi P. V. Methods and means of multistep generation of semantically correct texts. – Qualifying scientific work on the rights of manuscript. Dissertation for the degree of Doctor of Philosophy degree in specialty 124 “System Analysis” - Lviv Polytechnic National University, Ministry of Education and Science of Ukraine, Lviv, 2024.

Abstract content. The dissertation is devoted to the development of methods and means of multistep generation of semantically correct texts using large language models.

Currently the problem of aligning language models, or in a broad sense the problem of control over artificial intelligence, remains unsolved. It consists in the fact that the models formally solve the problem, but not in the way that researchers expect from them. There is also the problem of inconsistency of language models, they can generate statements that are inconsistent in content.

The goal was to improve the consistency of the knowledge of large language models through revalidation of the generated text, which aims to improve alignment and reduce the number of conflicting statements of language models.

To solve the given problem, it was decided to use logical rules to collect training data and train the model on them, because the knowledge, the consistency of which we seek to improve, is learned by models from training data. The model is given a question in natural language as an input, with the expectation of receiving an example that follows a certain rule of logic (for example, an implication). After that, the model automatically determines in which cases the model conflicts with its previous statements. Then, the model is trained on such conflicting statements to reduce the probability of such errors. The model does not learn new facts, and the only new knowledge that would be transmitted to the model is certain laws of logic, which are presented more clearly than in its initial training data.

Later, instead of the general "follows a certain rule of logic" was used a rule "follows the user's request". That is, the user makes a request to generate text according to a certain rule or instruction, and the same model, after generating the text, checks whether it does not contradict the user's request. We can say that the second law of logic

is used here, that two opposite judgments cannot be true at the same time. That is, if the model generated the text on request and during generation considers it correct, and when rechecked it is incorrect, then a contradiction arises here. Samples of such conflicting responses are then added to the set of examples of model behaviour so that the model learns to avoid making such mistakes. Also, instead of retraining the model, it was decided to use prompt engineering and augmenting it with new examples of behaviour.

An implication between two statements is chosen to test this approach. In this work, implication is not taken in a strictly mathematical sense, but in the context of the problem of logical deduction (Natural Language Inference), that is, that the second statement follows from the first statement. For example, the model might make a mistake and say that the statement "The object is a student" can infer that "The object studies physics", although the fact that the object is a student does not necessarily mean that it studies physics.

In the research process, such an approach was tested and it was decided to add a step of checking and discarding generated text if it's not aligned with user intent. That is, first a text is generated based on a large language model with embedded new examples of behavior in the prompt, and then the generated text is checked. Text generation takes place until the same model says that the generated text corresponds to the task that the user set before it. The model that verifies in terms of architecture and training data is the same model that generates text. The only difference is that it is tasked with classification rather than text generation. This is implemented by changing the prompt.

Scientific novelty of the obtained results:

1. For the first time, a model of the process of generating texts was developed, which uses the construction of a prompt to re-check the generated text, which made it possible to increase the level of semantic correctness of the result.
2. For the first time, a text generation method has been developed that includes a large language model used for text generation as a discriminator to

check the consistency of the results with the user's request, which has made it possible to increase the accuracy of the generation.

3. The method of verifying the generated text has been improved, which, unlike known methods, does not require human feedback, which made it possible to fully automate the generation process.

In **the first chapter**, the problem of Natural Language Inference is considered and its complexity is argued for checking the accuracy of large language models.

The transitive relation is studied in the context of the problem of logical deduction and data sets for it. The analysis of literary sources shows that for some tasks of natural language processing, high accuracy can be achieved with the help of simple heuristics.

The laws of formal logic in the context of natural language processing were analyzed, as well as the problem of logical errors in language models and the reliability of their results were investigated. Data set requirements for natural language processing and research selection strategies in the field of artificial intelligence are considered.

After the analysis, a strategy of using pre-trained large language models was chosen. It was found that neural networks with the Transformer architecture and a large number of parameters show the best results for natural language processing.

In **the second chapter**, the main ways of representing text in numerical format were explored. Such as bag of words, TF-IDF-based approach, word2vec, and using last layer of neural networks.

The usefulness of using large language models as a judge to support the quality of discussions is substantiated. It is proposed to consider the task as aligning models with the needs of users, in addition to improving the consistency of knowledge of large language models.

Proximal Policy Optimization (PPO) and Reinforcement Learning from Human Feedback (RLHF) are found to be used to solve the alignment problem. A formal description of the proposed approach was also carried out.

In **the third chapter**, the ability of the GPT-4 language model to make logical inferences and its ability to verify the generated text in the context of the logical inference task are tested in practice.

A metric was proposed to evaluate the confidence in the correctness of the answers of large language models. An algorithm for using a large language model to check its answers is described.

Use case, sequence, activity, class, and deployment diagrams were also presented for two text generation algorithms: using a discriminator to populate the set of model behaviour and using a discriminator to filter out incorrect responses. Based on these two algorithms, a text content generation method is formulated.

In **the fourth chapter**, an initial experiment setup using 80 examples and randomly divided into two groups was presented to test the basic GPT-4 model and its version with augmented prompt. After this experiment, it was decided to expand the scope of the test data set and allocate a separate set for testing all models in order to simulate the real conditions of using the model.

Two approaches were implemented: one using a discriminator to populate the set of model behaviour, and another using a discriminator to filter out incorrect responses.

A comparison of the accuracy of both approaches is made, and a method using both text generation algorithms sequentially is proposed because of its better accuracy in real-world use. A descriptive analysis of the test data set and the generated text was also performed.

A software prototype was developed that uses a large language model for text validation, which allowed to increase the quality of the generated text. This was achieved by using an artificial intelligence model as a discriminator to label generated text.

The developed software will be especially useful in tasks related to the generation of textual content, where there are many different options for correctly generated text, but it is easy to verify its correctness. For example, if the user wants the dialog system to respond in a formal business style, rather than a conversational style. There are many possible answers in the dialogue process, but you can definitely say whether the text is generated in an official business style or not. Examples of tasks in which the proposed approach will be less useful are information retrieval and automatic text summarization. This is due to the fact that information search has only one correct answer, and automated referencing does not have an unambiguous criterion of correctness.

Keywords: gpt-4, text generation, large language models, alignment problem, natural language inference, artificial intelligence, natural language processing.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Статті у наукових фахових виданнях України:

1. Здебський П. В., Берко А. Ю. Метод покращення якості генерування тексту за рахунок повторного передавання згенерованого тексту на модель. *Вісник Хмельницького національного університету. Серія: Технічні науки*. 2024. № 1 (331). С. 259–263.
2. Здебський П. В., Берко А. Ю. Перевірка тексту після генерації за допомогою великих мовних моделей для фільтрування неправильних відповідей. *Наукові праці Вінницького національного технічного університету*. 2024. № 1.
3. Берко А. Ю., Здебський П. В., Висоцька В. А. Метод формування набору даних для перевірки якості вивчення мовними моделями транзитивного відношення у контексті задачі логічного висновку. *Вісник Національного університету “Львівська політехніка”. Серія: Інформаційні системи та мережі*. 2024. Вип. 14. С. 46–60.
4. Здебський П. В., Берко А. Ю., Чирун Л. В. Інформаційна система видобування інформації з відкритих WEB-ресурсів. *Вісник Національного університету “Львівська політехніка”. Серія: Інформаційні системи та мережі*. 2022. Вип. 12. С. 141–168.

Статті у виданнях інших держав:

1. Zdebskyi P., Berko A., Vysotska V. Investigation of transitivity relation in natural language inference. *CEUR Workshop Proceedings*. 2023. Vol. 3396 : Computational linguistics and intelligent systems 2023 : proc. of the 7th Intern. conf. on computational linguistics and intelligent systems. Vol. II: Computational linguistics workshop, Kharkiv, Ukraine, Apr. 20-21, 2023. P. 334–345.
2. Zdebskyi P., Lytvyn V., Burov Y., Rybchak Z., Kravets P., Lozynska O., Holoshchuk R., Kubinska S., Dmytriv A. Intelligent system for semantically similar sentences identification and generation based on machine learning methods. *CEUR Workshop Proceedings*. 2020. Vol. 2604 : Proc. of the 4th

Intern. conf. on computational linguistics and intelligent systems COLINS 2020, Lviv, Ukraine, Apr. 23-24, 2020. Vol. I : main conf. P. 317–346.

3. Zdebskyi P., Vysotska V., Peleshchak R., Peleshchak I., Demchuk A., Krylyshyn M. An application development for recognizing of view in order to control the mouse pointer. *CEUR Workshop Proceedings*. 2019. Vol. 2386 : Workshop proc. of the 8th Intern. conf. on "Mathematics. Information technologies. Education" MoMLeT&DS 2019, Shatsk, Ukraine, 2-4 June 2019. P. 55–74.

Матеріали конференцій:

1. Zdebskyi P., Berko A., Vysotska V., Voloshyn S., Chyrun S., Schuchmann V. Framework for improving the effectiveness of discussions at English-language articles analysis. *Advanced information and communication technologies* : proc. of the 5th IEEE Intern. conf., Lviv, Ukraine, Nov. 21–25, 2023. 2023. P. 127–132.
2. Zdebskyi P., Berko A., Vysotska V. Information technology for finding answers to questions from open web resources. *International Scientific and Technical Conference on Computer Sciences and Information Technologies*. 2023. 18th IEEE Intern. conf. on computer science and information technologies CSIT 2023, Lviv, Ukraine, 19-21 Oct. 2023.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ВИМІРЮВАННЯ, СКОРОЧЕНЬ І ТЕРМІНІВ

- Задача логічного виведення – Natural Language Inference (NLI), Recognising Textual Entailment (RTE).
- Промпт - prompt.
- Конструювання промпту (передавання інструкцій на вхід, конструювання підказок) – prompt engineering.
- Велика мовна модель – large language model (LLM).
- PPO – Proximal Policy оптимізація (Proximal Policy Optimization)
- RLHF – навчання із підкріпленням на основі зворотного зв'язку людини (Reinforcement Learning from Human Feedback).
- Механізм уваги – attention.
- CNN – згорткові нейронні мережі.
- RNN – рекурентні нейронні мережі.
- LSTM – архітектура довгої короткочасної пам'яті (Long short-term memory).
- GRU – вентильні рекурентні вузли (Gated recurrent units).
- Передавальне навчання – transfer learning.
- BERT – Bidirectional Encoder Representations from Transformers.
- GPT – Generative Pre-trained Transformer.
- Тонке настроювання (дотреновування) – fine-tuning.
- T5 – Text-to-Text Transfer Transformer.
- Модель торби слів – bag-of-words.
- Стратегія (лінія) поведінки – policy.
- Точність – accuracy.
- Влучність – precision.
- Повнота – recall.

ЗМІСТ

ВСТУП.....	19	
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРНИХ ТА ІНШИХ ДЖЕРЕЛ	23	
1.1. Проблематика семантичного аналізу тексту	23	
1.2. Приклади проблем мовленнєвих моделей.....	23	
1.2.1. Ідентифікація та генерування семантично подібних речень на основі методів машинного навчання	23	
1.2.2. Дослідження транзитивного відношення у задачі логічного виведення	25	
1.2.3. Інформаційна система пошуку відповідей на запитання з відкритих web-ресурсів.....	26	
1.3. Філософське підґрунтя підходів для семантичного аналізу тексту	27	
1.3.1. Проблема логіки мовленнєвих моделей	27	
1.3.2. Достовірність результатів великих мовних моделей	28	
1.4. Вимоги до наборів даних для опрацювання природної мови	29	
1.5. Огляд логіки в контексті моделей опрацювання природної мови	30	
1.6. Стратегії вибору досліджень у галузі штучного інтелекту	39	
1.7. Нейронні мережі для опрацювання природної мови	41	
1.8. Рекурентні нейронні мережі та архітектура довга короткочасна пам'ять...44		
1.9. Аналіз моделей, що використовують механізм уваги	45	
1.10. Аналіз великих натренованих моделей для опрацювання природної мови	47	
Висновки до розділу 1.....	50	
РОЗДІЛ 2. ЗАСТОСУВАННЯ БАГАТОЕТАПНИХ ПІДХОДІВ ДЛЯ	ГЕНЕРУВАННЯ ТЕКСТОВОГО КОНТЕНТУ	51

2.1. Способи представлення тексту	51
2.2. Спосіб використання великих мовних моделей	52
2.2.1. Загальний опис логіки як способу мислення	52
2.2.2. Аргументація проблематики використання мовних моделей у дискусіях	53
2.2.3. Метод використання мовленнєвої моделі для підтримки якості дискусій	56
2.3. Проблема узгодження мовних моделей із потребами користувачів	62
2.4. Навчання з підкріпленням на основі зворотного зв'язку людини та Proximal Policy оптимізація	67
2.5. Тренування моделей, що допомагають людям оцінювати точність у RLHF	70
2.6. Покращення вирішення математичних задач мовними моделями за допомогою нагляду за процесом	72
2.7. Опис запропонованої моделі процесу генерування текстового контенту ...	73
2.7.1. Використання дискримінатора для наповнення набору прикладів поведінки	73
2.7.2. Використання дискримінатора для відсіювання неправильних відповідей	78
2.7.3. Комбінований підхід, що включає використання дискримінатора для наповнення набору прикладів поведінки і відсіювання неправильних відповідей	79
Висновки до розділу 2	80
РОЗДІЛ 3. АЛГОРИТМИ ТА МЕТОДИ БАГАТОЕТАПНОГО ГЕНЕРУВАННЯ ТЕКСТОВОГО КОНТЕНТУ	81
3.1. Аналіз якості логічного мислення GPT-4	81

3.2. Метрика для визначення впевненості у правильності відповіді мовної моделі GPT-4	83
3.3. Підхід покращення консистентності мовних моделей через використання логічного правила імплікації	84
3.4. Опис підходу з використанням моделі для перевірки своїх відповідей.....	86
3.5. Опис алгоритмів генерування тексту та запропонованого методу	89
3.5.1. Діаграма варіантів використання	90
3.5.2. Діаграма послідовності.....	91
3.5.3. Діаграма діяльності	93
3.5.4. Діаграма класів	96
3.5.5. Діаграма розгортання	98
3.5.6. Опис запропонованого методу генерування текстового контенту.....	99
Висновки до розділу 3.....	101
РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ БАГАТОЕТАПНОГО ГЕНЕРУВАННЯ СЕМАНТИЧНО КОРЕКТНИХ ТЕКСТІВ ТА АНАЛІЗ РЕЗУЛЬТАТІВ	102
4.1. Постановка початкового експерименту	102
4.2. Реалізація остаточного методу генерування тексту	109
4.2.1. Програмна реалізація запропонованих алгоритмів генерування тексту	110
4.2.2. Аналіз результатів та точності запропонованих підходів.....	114
4.3. Описовий аналіз тестового набору даних і згенерованого тексту	121
Висновки до розділу 4.....	125
ВИСНОВКИ.....	126

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ129

ВСТУП

Актуальність дослідження. Задача узгодження великих мовних моделей, або проблема контролю штучного інтелекту, є особливо актуальною сьогодні у зв'язку з тим, що вони стають більш здібними у вирішенні різного роду задач. Здібність розв'язування складних задач призводить до того, що моделі здатні “обійти” задачу. Тобто формально вони розв'язують поставлену перед ними задачу, але не так, як від них очікують дослідники. Прикладом у контексті робототехніки може бути спроба робота обманути оцінювача, розташувавши маніпулятор між камерою та об'єктом, щоб вдати, що він його бере.

Проблема узгодженості виникає через обмеженість дослідників у формулюванні тих складних задач, які сучасні мовні моделі можуть вирішувати. Упустивши часткові випадки використання при постановці задачі може призвести до того, що моделі штучного інтелекту вироблять поведінку, щоб обманювати оцінювачів, а не виконувати завдання. Також ефективність таких алгоритмів залежить від інтуїції оцінювача щодо того, яка поведінка вважається правильною.

Ця проблема є небезпечною, при використанні системи у реальних умовах де помилка є особливо чутливою для кінцевого користувача, бо важко контролювати текст, що генерує модель і немає гарантій стосовно того, що текст буде співпадати із очікуваннями користувача.

Наприклад, текст, створений моделлю GPT-3, іноді може містити недостовірну, токсичну або шкідливу інформацію. В такому випадку модель можна назвати неузгодженою із потребами користувача. Великі мовні моделі часто стикаються з явищем, що називається галюцинаціями. Це означає, що вони можуть видавати неправдиві відповіді або вигадувати факти. Це може призвести, наприклад, до генерації неправдивих новин, навіть якщо модель була натренована генерувати лише достовірні результати.

Провідна компанія у цій сфері OpenAI визначає цей напрям як один із головних. Вони використовують запитання, які вони отримали від користувачів їхнього API, щоб тренувати свої моделі, що краще узгоджуються із реальними запитамися користувачів.

Для цього вони використовують навчання з підкріпленням на основі зворотного зв'язку людини. Він полягає у тому, що люди наводять приклади очікуваної поведінки і ранжують відповіді від найбільш до найменш підходящого. Після цього ці дані використовуються для дотреновування моделей, які називають InstructGPT. Вони використовуються по замовчуванню у їхньому API, бо було виявлено, що ці моделі при меншій кількості параметрів показують кращі результати, рідше придумують факти і є менш токсичними.

Мета дослідження: узгодження великих мовних моделей із потребами користувачів.

Об'єкт дослідження: процеси генерування текстового контенту за допомогою великих мовних моделей.

Предмет дослідження: методи повторної перевірки текстового контенту згенерованого великими мовними моделями.

Завдання дослідження:

1. Провести аналіз підходів для опрацюванні природної мови та обґрунтувати стратегію досліджень.
2. Розробити модель багатоетапного генерування текстового контенту на основі великих мовних моделей.
3. Розробити алгоритми і метод реалізації моделі багатоетапного генерування текстового контенту.
4. Розробити архітектуру та здійснити програмну реалізацію системи багатоетапного генерування семантично коректних текстів.
5. На основі експериментальних досліджень провести аналіз точності запропонованого методу генерування текстового контенту.

Методи дослідження. Для реалізації генератора, імітатора, та дискримінатора було використано метод підвищення точності великих мовних моделей — конструювання промπτу (prompt engineering). Для програмної реалізації системи було використано методи об'єктно-орієнтованого програмування. Для перевірки точності було використано метод на основі притриманого набору даних (holdout dataset).

Наукова новизна отриманих результатів. Вперше розроблено модель процесу генерування текстів, яка використовує конструювання промпту для повторної перевірки згенерованого тексту, що дало можливість підвищити рівень семантичної коректності результату. Вперше розроблено метод генерування тексту, який включає велику мовну модель, що використовується для генерування тексту, у якості дискримінатора для перевірки узгодженості результатів із запитом користувача, що дало змогу підвищити точність генерування. Удосконалено метод верифікації згенерованого тексту, що на відміну від відомих методів не вимагає зворотного зв'язку людини, що дало змогу повністю автоматизувати процес генерування.

Особистий внесок здобувача. Основні положення та результати дисертаційної роботи одержані автором самостійно. Особисто здобувачеві належать наступні наукові результати: розроблено модель процесу багатоетапного генерування тексту, розроблено метод генерування тексту з використанням дискримінатора для перевірки узгодженості результатів із запитом користувача, проведено експериментальне дослідження точності запропонованого методу у порівнянні із звичайним використанням великої мовної моделі.

Апробація результатів дисертації. Основні наукові та практичні результати роботи доповідались та обговорювались на таких міжнародних конференціях:

- Computational linguistics and intelligent systems 2023: proc. of the 7th Intern. conf. on computational linguistics and intelligent systems. Vol. II: Computational linguistics workshop, Kharkiv, Ukraine, Apr. 20-21, 2023.
- Advanced information and communication technologies : proc. of the 5th IEEE Intern. conf., Lviv, Ukraine, Nov. 21–25, 2023.

Також, результати дисертаційних досліджень регулярно доповідалися на наукових семінарах кафедри “Інформаційні системи та мережі” Національного університету “Львівська політехніка”.

Практичне значення отриманих результатів. Запропонований метод буде особливо корисним у задачах, що пов'язані з генеруванням тексту, де існує багато варіантів правильно згенерованого тексту і його правильність легко перевірити. Наприклад, якщо користувач бажає, щоб чат-бот відповідав у офіційно-діловому, а

не розмовному стилі. У таких випадках існує багато можливих відповідей під час діалогу, але можна чітко визначити, чи згенерований текст має офіційно-діловий стиль.

Прикладами задач, де запропонований методу буде менш корисним, є, наприклад, інформаційний пошук (information retrieval) і автоматизоване реферування (automatic text summarization). Це пов'язано з тим, що перша має лише одну правильну відповідь, тоді як у другій відсутній однозначний критерій правильності.

Іншим прикладом задачі, у якій запропонований метод буде корисним, є задача логічного виведення, бо є багато можливих висловлювань, що впливають із заданого користувачем, але якщо висловлювання не обов'язково впливає із заданого, або йому суперечить, то це легко визначити. На цій задачі було перевірено запропонований метод, але він може бути використаний до будь-яких інших задач пов'язаних із генеруванням текстового контенту.

Структура та обсяг дисертації. Дисертація складається із вступу, чотирьох розділів, висновків, списку використаних джерел із 98 найменувань. Повний обсяг дисертації складає 138 сторінок, основний зміст викладено на 128 сторінках, де наведено 42 рисунки та 8 таблиць.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРНИХ ТА ІНШИХ ДЖЕРЕЛ

1.1. Проблематика семантичного аналізу тексту

Семантика - це розділ мовознавства, метою якого є дослідження значення мови [1]. Семантика має справу зі значенням речень і слів [2, 3]. У контексті даної роботи семантично коректним є такий згенерований текст, що з точки зору змісту підходить під те формулювання задачі, що описав користувач природною мовою. Тобто, текст повинен бути раціональним у контексті запиту користувача.

1.2. Приклади проблем мовленнєвих моделей

1.2.1. Ідентифікація та генерування семантично подібних речень на основі методів машинного навчання

Формально задача ідентифікації семантично подібних речень може розглядатись як завдання “Recognising Textual Entailment”. The Recognising Textual Entailment (RTE) задача розглядається як розпізнавання двох фрагментів тексту, чи значення одного можна вивести з іншого. Ця задача не є прив’язаною до домену і запропонована для розпізнавання мінливості смислового вираження, які зазвичай потрібні у багатьох задачах [4].

Фундаментальний феномен природної мови це різноманітність семантичних виражень, в якій те саме значення може бути виражено, або логічно виведено з різних текстів. Цей феномен може розглядатись як проблема мовної двозначності, в якій формуються зв’язки багато до багатьох між мовними виразами і значеннями [4].

Текстове відношення логічного виведення між двома текстами: Т (текст) and Н (гіпотеза) представляє фундаментальний феномен природної мови. Це позначається як $T \rightarrow H$ і означає, що значення Н можна логічно вивести із Т [5].

Це відношення напрямлене бо значення одного виразу можна зазвичай логічно вивести з іншого в свою чергу логічне виведення в іншу сторону менш очевидне [6]. Текстове логічне виведення залежить від контексту, нетранзитивне, і немонотонне [7].

Згідно з фразовою структурою граматики речення складається з іменної фрази та дієслівної фрази. Схему представлено на рис. 1.1.

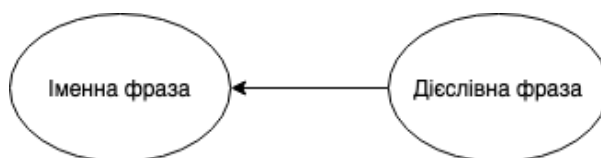


Рис. 1.1 Схema взаємозв'язків згідно з фразовою структурою речення у англійській мові

Для задачі логічного виведення популярним набором даних є MultiNLI. Його було використано для тренування і тестування системи у цій роботі. З нього було вибрано підмножину набору даних розміром сто тисяч прикладів. Двадцять п'ять відсотків даних було виділено на тестування системи решту виділено на тренування.

У роботі [8] наведені наступні чисельні характеристики. Середнє арифметичне для класів “similar”, “neutral” та “contradiction” дорівнює 0.914, 0.885 та 0.89 відповідно (рис. 1.2). Медіана для “similar”, “neutral” та “contradiction” дорівнює 0.928, 0.898 та 0.904 відповідно. Середнє квадратичне відхилення рівне 0.062, 0.068 та 0.07 відповідно. Можемо бачити, що хоча найбільше середнє арифметичне і медіана є у класу “similar”, проте різниця між значеннями різних класів не достатньо велика для того, щоб їх можна було легко класифікувати використовуючи цю ознаку. Враховуючи розподіли різних класів можемо зробити висновок, що ці класи не є чітко розділимі при використанні лише значень косинуса подібності між векторами.

Використавши косинус кута між векторами для тренування лінійної моделі класифікації точність на яку можемо розраховувати це 0.4. Це низька точність, що доводить складність задачі. Тобто задача логічного виведення, інші назви “Recognising Textual Entailment” та “Natural Language Inference”, є досить складною, щоб її можна було використовувати в якості задачі для перевірки точності великих мовних моделей [8].

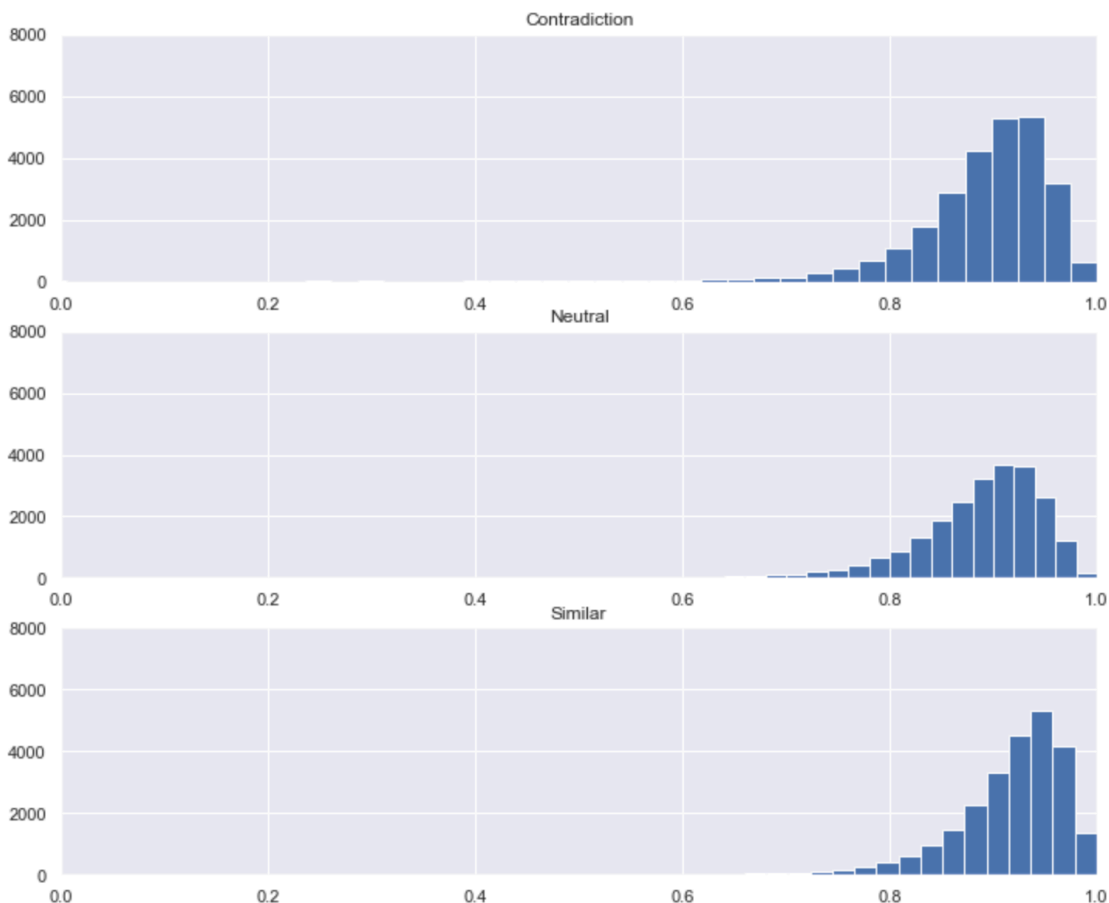


Рис. 1.2. Гістограми для косинусів подібності по кожному із класів

1.2.2. Дослідження транзитивного відношення у задачі логічного виведення

Задача логічного виведення (NLI) є хорошим орієнтиром для дослідження розуміння природної мови (NLU), де два речення передаються моделі для ідентифікації зв'язку між ними, вибравши один із 3 варіантів: логічний наслідок, нейтральний і протиріччя. Успіх у ній не вимагає складних навичок машинного навчання, а радше здатність розуміти значення речень, лексичну та композиційну семантику, а також такі явища, як квантифікація, час, переконання, модальність, а також лексичну та синтаксичну неоднозначність. Дослідження спирається виключно на набір даних MultiNLI (рис. 1.3), який вважається найбільшим набором даних для логічного виведення природної мови та підтримує кілька мов. MultiNLI складається з пар речень, анотованих текстовою інформацією, і охоплює різні жанри тексту [9].

Передумова	Гіпотеза	Клас
The new rights are nice enough	Everyone really likes the newest benefits	neutral
This site includes a list of all award winners and a searchable database of Government Executive articles.	The Government Executive articles housed on the website are not able to be searched.	contradiction
uh i don't know i i have mixed emotions about him uh sometimes i like him but at the same times i love to	i like him for the most part, but would still enjoy seeing someone beat him.	entailment
yeah i i think my favorite restaurant is always been the one closest you know the closest as long as it's it's	My favorite restaurants are always at least a hundred miles away from my house.	contradiction
i don't know um do you do a lot of camping	I know exactly.	contradiction
well that would be a help i wish they would do that here we have got so little landfill space left that we're g	We have plenty of space in the landfill.	contradiction
yeah i know and i did that all through college and it worked too	I did that all through college but it never worked	contradiction
Calcutta seems to be the only other production center having any pretensions to artistic creativity at all, bu	Most of Mrinal Sen's work can be found in European collections.	neutral
If that investor were willing to pay extra for the security of limited downside, she could buy put options with	The strike price could be \$8.	contradiction
3) Dare you rise to the occasion, like Raskolnikov, and reject the petty rules that govern lesser men?	Would you rise up and defeat all evil lords in the town?	neutral
The most important directions are simply up and up leads eventually to the cathedral and fortress commar	Go downwards to one of the gates, all of which will lead you into the cathedral.	contradiction
The bhakti movement of the Tamils brought a new warmth to the hitherto rigid Brahmanic ritual of Hinduism	The Tamils' bhakti movement froze the previously warm ritual of Hinduism.	contradiction
see now in a situation like that the boys are only sixteen years old and they were sexually involved with her	Everyone involved was the same age.	contradiction
You and your friends are not welcome here, said Severn.	Severn said the people were not welcome there.	entailment
um-hum um-hum yeah well uh i can see you know it's it's it's it's kind of funny because we it seems like w	We don't loan a lot of money.	neutral
i'm not sure what the overnight low was	I don't know how cold it got last night.	entailment
so i have to find a way to supplement that	I need a way to add something extra.	entailment
the hologram makes up all these things and uh i mean sometimes sometimes it's funny sometimes it's no	Sometimes it is amusing to see what the hologram creates.	entailment
5 The share of gross national saving used to replace depreciated capital has increased over the past 40 ye	Gross national saving was highest this year.	neutral
So far, however, the number of mail pieces lost to alternative bill-paying methods is too small to have any r	The amount of lost mail is huge and really impacts mail volume	contradiction
you want to punch the button and go	You don't want to push the button lightly, but rather punch it hard.	neutral
Buffet and a la carte available.	It has a buffet.	entailment
Blue says Blumenthal claimed Clinton had told him that Lewinsky had made unwanted sexual advances.	Clinton said that Monica Lewinsky made unwanted sexual advances during her time as a journalist in the White House	neutral
We also have found that leading organizations strive to ensure that their core processes efficiently and effe	Leading organizations want to be sure their employees are safe.	neutral
oh uh-huh well no they wouldn't would they no	No, they wouldn't go there.	neutral

Рис. 1.3. Приклад зразків MultiNLI (<https://cims.nyu.edu/~sbowman/multinli>)

Це дослідження спирається на застосуванні таких інструментів машинного навчання як трансформери, механізм уваги (attention), моделей BERT (Bidirectional Encoder Representations from Transformers) і RoBERTa та набору даних MultiNLI. Воно показало, що модель у транзитивному відношенні робить помилки лише там, де вона робить їх і при звичайні класифікації. Тобто транзитивність не ускладнює задачу для моделі [10].

1.2.3. Інформаційна система пошуку відповідей на запитання з відкритих web-ресурсів

Метою була розробка системи відповідей на питання, що стосуються найвищого ступеня порівняння, на основі відкритих веб-ресурсів. Приклади таких питань: “Які найпопулярніші авто 2023?”, “Який найкращий роман?”. Система реалізована на Python. Використовувалися додаткові бібліотеки, такі як Selenium для веб-збирання та NLTK для вилучення ключових слів. Результатом роботи системи є ранжований список елементів із кількістю повторень. Рейтинг базується на популярності товару, а точніше, на кількості веб-ресурсів, на яких він з'явився. Вона відрізняється від класичних QA систем тим, що зазвичай відповідає на однозначні питання. Натомість реалізована система агрегує думки з різних

ресурсів, щоб відповісти на суб'єктивні запитання. Для цього завдання немає наборів даних, тому систему було перевірено на створених вручну зразках запитань. Враховуючи наведене вище обмеження, точність системи, виражена в оцінці F1, близька до 0,8. У майбутньому можна створити веб-сайт, який збирає дані з багатьох веб-ресурсів, призначених для ранжування елементів із різних доменів.

Для цього завдання немає набору даних, але, виходячи з даних відкритих веб-ресурсів, найкращими IDE для Python були вибрані: Pycharm, Visual Studio Code, Jupyter, Spyder і Atom. Інші популярні IDE та редактори коду: Idle, Thonny, Pydev, Sublime text, Vim, Wing, Eric тощо.

Порогове значення кількості веб-сайтів, які шукала система, дорівнює 50. Після пошуку 50-ти перших веб-сайтів за запитом “Найкраща IDE для Python” системі вдалося знайти всі популярні IDE. Із результатів роботи системи можна зробити висновок, що найбільш популярні варіанти відповідей близькі до істини. Тобто перші відповіді – це дійсно середовища розробки, і PyCharm є найпопулярнішим середовищем розробки для Python. Четвертий варіант відповіді було слово “код”, це можна пояснити тим, що одним з найкращих середовищ розробки є Visual Studio Code і через проблему з об'єднанням частин власного імені виникли проблеми. Інші галасливі слова з'явилися в результаті через те, що для фільтрації використовували лише 50 сайтів. Точність моделі можна покращити, використовуючи великий корпус тексту для фільтрації таких слів, як “топ”, “навчання” тощо.

З цього можемо зробити висновок, що деякі задачі, як от “Найкраща IDE для Python”, можуть бути досить якісно вирішені без використання великих мовних моделей [11].

1.3. Філософське підґрунтя підходів для семантичного аналізу тексту

1.3.1. Проблема логіки мовленнєвих моделей

Модель дає статистично найбільш ймовірну відповідь [12]. Схоже дають відповіді і люди, але у людей є апіорне розуміння логіки, а статистичні моделі не мають явних ознак дотримання логіки [13, 14]. Наприклад, мовна модель Roberta

від Фейсбук інколи робить логічні помилки у прикладах з транзитивністю. Це було одним із стимулів до написання цієї дисертації, бо у великих мовних моделях є статистичні знання про мову і світ, але не має строгих апріорних знань, як наприклад знання законів логіки і слідуванні їм. За рахунок цього модель знання моделі не є консистентними і вона може сама собі суперечити. Початковою ідеєю було навчити велику мовну модель законам логіки, так як вони є апріорними і вона не вивчить їх із тексту [15]. Однією із проблем може бути якраз те, що вони є апріорними і їх неможливо буде задати.

1.3.2. Достовірність результатів великих мовних моделей

Обмеженість всіх мовних моделей це достовірність результатів [16, 17]. Наприклад модель ChatGPT генерує дуже реалістичний текст, може спілкуватись схоже до справжнього співрозмовника, але помиляється якщо запитувати питання що мають лише одну правильну відповідь. Наприклад ChatGPT дає неправильну відповідь на питання “Яка найменша країна Південної Америки?”, якою є Суринам. Хоча модель часто наводить і реальні факти. В тому що модель робить помилки немає нічого дивного, бо і люди помиляються, але цікаво яка причина цих помилок, і чи вона така сама як і у людей. Однією із причин може бути недостатня кількість даних по цьому питанню у тренувальних даних моделі. У тренувальних даних можуть бути упередження, через спосіб збору цих даних, важливо це розуміти, щоб розуміти обмеження моделі. Також може бути проблема обмеженої пам’яті моделі, що не може все запам’ятати з тренувальних даних, подібно до людини.

Дослідження достовірності відповідей сучасних мовленнєвих моделей є зараз важливою задачею у даній області, і через те, що їх натреновано на текстових даних написаних людьми і ціллю таких моделей є поводитись схоже до людини, то вони часто використовують розмиті поняття замість точних чи наукових. Чим більше модель сумнівається у своїй відповіді, тим легше буде її переконати у протилежній думці. Тобто можна висунути гіпотезу, що простота переконання у протилежному факті може слугувати метрикою впевненості моделі у своїй відповіді стосовно цього факту.

1.4. Вимоги до наборів даних для опрацювання природної мови

Корпус – це не просто збірка текстів. Швидше, корпус прагне представити мову або якусь частину мови [18]. Таким чином, відповідний дизайн для корпусу залежить від того, що він має представляти. Репрезентативність корпусу, у свою чергу, визначає види дослідницьких питань, які можуть бути розглянуті, та можливість узагальнення результатів дослідження [19]. Наприклад, корпус, що складається в основному з новинних репортажів, не дозволить загальне дослідження варіантів англійської мови. Подібним чином дослідження, засноване на корпусі, що містить один тип розмови – наприклад, розмови між підлітками – не можна узагальнити на розмову в цілому. Таким чином, незалежно від того, чи ви розробляєте власний корпус, обираєте корпус для використання в дослідженні або читаєте роботу інших, що базується на корпусі, питання репрезентативності в дизайні корпусу є вирішальними [19].

Важливо заздалегідь усвідомити, що представлення мови – або навіть частини мови – є проблематичним завданням. Ми не знаємо повного обсягу варіацій у мовах або всіх контекстуальних змінних, які необхідно охопити, щоб охопити всі варіації в текстах. Однак увага до певних питань гарантує, що корпус буде максимально репрезентативним, враховуючи наші поточні знання мови.

Основним питанням у дослідженнях корпусу є визначення розміру корпусу [20]. Корпус має бути достатньо великим, щоб забезпечити наявність достатньої кількості зразків спеціалізованої лексики для аналізу чи вибору. Для високочастотних слів корпуси не повинні бути дуже великими, оскільки слова зустрічаються часто. Подібним чином слова, тісно пов'язані зі спеціалізованою сферою, також мають тенденцію зустрічатися досить часто. У сантехніці, наприклад, дуже часто вживане слово в письмових текстах - труба. Низькочастотні слова є більш проблематичними, оскільки вони зустрічаються не так часто.

Репрезентативність корпусу є важливим питанням у дослідженнях на основі корпусу, і воно має ряд елементів. Одне питання щодо репрезентативності полягає в тому, чи представляє корпус той тип письма, читання чи мультимедійного “тексту”, з яким мали б працювати студенти англійської мови спеціального вжитку.

Різні підходи до систем класифікації в розробці та аналізі корпусів можуть створити проблеми для порівняння результатів різних досліджень. Рішення повинні прийматися на принциповій основі щодо того, чи підходить певна галузь дослідження до тієї чи іншої області. Важливий момент щодо аналізу корпусів, що корпуси можуть лише свідчити про те, що можливо, а не про те, що неможливо [20].

1.5. Огляд логіки в контексті моделей опрацювання природної мови

Логічний закон це суттєвий та необхідний зв'язок між логічними формами у контексті міркувань. Є чотири основні закони логіки:

- Закон тотожності
- Закон протиріччя
- Закон виключення третього
- Закон достатньої підстави.

Прийнято називати закони у цьому порядку. Тобто інколи закон тотожності називають першим законом логіки, закон протиріччя другим і так далі. Надалі у цій роботі також інколи будемо послуговуватись такою нумерацією законів логіки.

Арістотель каже про закон тотожності таким чином, що не можна нічого мислити, якщо не мислити постійно одне і те ж. Тобто думка повинна зберігати свій зміст протягом міркування. Неможна один раз використовувати поняття у одному значенні, а потім у іншому. Для цього перш ніж щось обговорювати, варто чітко визначити поняття, що будуть обговорюватись. Це дозволить уникнути різних трактувань одного і того ж поняття різними учасниками дискусії. Тобто кожна думка повинна бути тотожною самій собі, звідси і походить назва закону.

Є три види помилок, що порушують закон тотожності:

- Еквівокація
- Амфіболія
- Логомахія

Еквівокація виникає коли слово описує два різних поняття. Тобто у двох понять таке саме звучання, від цього походить і назва. Наприклад, “6 і 5 є парним і непарним” та “6 і 5 є дев'ять” звідси, користуючись властивістю транзитивності,

можна зробити помилковий висновок що “9 є парним і непарним”. Проблеми виникає, бо у першому випадку “і” вживається у значенні сполучника, а у другому у значенні математичної операції додавання. Транзитивність це така властивість властивість тотожних понять, якщо поняття “А” тотожне з “Б”, а “Б” з поняттям “В”, то поняття “А” тотожне з “В”.

Амфіболія стосується не конкретних слів, а речень, які можна трактувати по різному. Наприклад, у деяких випадках кома поставлена і іншому місці в реченні повністю змінює його зміст, або якщо речення вживається в різних контекстах, то воно також може мати різні значення.

Логомахія це різне трактування словосполучень. Тобто всі ці види помилок виникають на різних рівнях текст, на рівні речення, словосполучення та окремих слів.

Закон протиріччя полягає у тому, що не можуть бути істинними одночасно два судження, у яких одне стверджує щось про якусь ознаку предмета, а інше стверджує несумісну ознаку у тому самому відношенні. Прикладом можуть бути судження “Його знайомий перебуває у Києві” та “Його приятель перебуває у Львові”. Якщо ознаки є несумісними і виявилось, що одне із суджень істинне, то це означає, що друге хибне. Тобто, якщо виявилось, що знайомий перебуває у Києві, то він не може також бути у Львові у тому самому значенні. При тому, якщо одне судження виявилось хибою, то це ще не означає, що друге істинне. У наведеному прикладі, знайомий може бути у будь-якому іншому місті окрім цих двох, тому якщо він не є у одному з них не означає, що він є у іншому.

Не варто плутати несумісні ознаки предметів та протилежні судження. Несумісні ознаки означають, що вони не можуть існувати одночасно у тому самому відношенні до предмета. При цьому це не означає, що вони є протилежними. Протилежними будуть судження лише, якщо одне можна виразити через інше із приставкою “не”. До вищезгаданого прикладу “Його знайомий перебуває у Києві” можна навести такий варіант протилежного судження “Його знайомий не перебуває у Києві”. Тобто, тут наведено не просто несумісну ознаку, а й протилежну. У такому випадку, якщо одне судження хибне, то інше вірне. Тобто протилежні судження є підмножиною несумісних (рис. 1.4), а закон

протиріччя говорить не лише про протилежні, а про ширший клас суджень, які називаються несумісними.



Рис. 1.4. Зображення протилежних суджень як підмножини несумісних

Закон виключення третього каже, що одне із двох суджень де одне заперечує інше є істинним, а інше хибним. У цьому законі на відміну від закону протиріччя йдеться не про несумісні судження, а саме про протилежні. Тобто якщо у предмета описується якась несумісна ознака у двох судженнях, то це покриває закон протиріччя, який лише каже, що вони не можуть бути двоє істинними одночасно. При цьому закон виключення третього каже, що протилежні судження повинні обов'язково мати якесь одне хибне, а інше істинне.

Четвертим і останнім із основних законів формальної логіки є закон достатньої підстави. Він каже, що жодне твердження не можна вважати істинним поки не буде достатньої підстави так вважати. Можна сказати, ще так: кожна істинна має достатню підставу. Достатньою підставою є така, що форма поєднання істинних суджень змушує визнавати істинність судження, що обґрунтовується. Тобто щоб мати достатню підставу у нас повинні бути раніше доведені істинні судження і форма їх поєднання у такому вигляді, що це змушує нас визнавати істинність судження, що обґрунтовується.

Проаналізувавши ці основні чотири закони логіки можемо сказати, що кожен з них вносить певну гіпотезу про наше мислення і світ. Їх істинність

ґрунтується на істинності понять які вони вводять. Вони є тавтологічними і їх можна сприймати як форму представлення цих гіпотез про онтологію світу, знання та мислення. Перший закон вводить гіпотезу, що кожен предмет має якісь ознаки. Тобто, що об'єкт мови має певні способи використання і у різних способах використання відіграє іншу роль. Ця думка тісно переплітається із філософією Людвіга Вітгенштейна, бо він казав, що значення слова це спосіб його використання і ввів поняття мовних ігор.

Другий закон, закон протиріччя, вводить поняття сумісності ознак. Тобто, що є певні сумісні ознаки предмета, та несумісні ознаки. За означенням несумісні ознаки не можуть бути одночасно застосовані до предмета у тому самому відношенні, тому це закон випливає суто з означення несумісності ознак, яке використовується у даному законі.

Третій закон, закон виключення третього, вводить оператор “не”. Тобто, певні поняття мисляться не просто як несумісні, а як протилежні. За означенням протилежності, судження повинні мати одне істинне, а друге хибне. Цей закон виділяє із несумісних ознак підмножину протилежних.

Четвертий закон, закон достатньої підстави, вводить поняття форми поєднання суджень, для отримання нових. Ця форма чітко не визначається у даному законі, а лише висувається гіпотеза, що така форма присутня у нашому мисленні.

Логіка досліджує принципи мислення через мову, а не прямо [21]. Основною метою логіки є визначення логічних форм у мові. Визначення значень для символів у знакових системах базується на конкретних правилах, які називаються семантичними. У мові семантичні правила не є чітко визначеними. Зв'язок між символом та його сенсом формуються в процесі людської комунікації та залежать від численних факторів. За рахунок цього значення слова у природній мові визначається способом його використання. До такого висновку прийшов австро-англійський філософ Людвіг Вітгенштейн [22].

Логіка в основному працює з розповідними реченнями. У логіці є поняття “висловлювання”, яке собою являє підмножини розповідних речень, які можуть бути істинними, або хибними за значенням.

Логічні сполучники фіксують зв'язки між висловлюваннями [23]. У логіці для таких сполучників введені відповідні поняття, що називають операторами:

- І (кон'юнкція)
- Або (диз'юнкція)
- Якщо і тільки якщо, то (еквіваленція)
- Ні (заперечення)
- Якщо, то (імплікація)

Заперечення це такий оператор, який із будь-якого істинного висловлювання робить хибне, і навпаки. Кон'юнкція істинна тільки коли істинні два операнда. Диз'юнкція коли хоча б один з операндів істинний. Прикладом диз'юнкції може бути висловлювання “він досягнув гарних результатів, або завдяки старанності, або таланту”.

Імплікація у свою чергу є оператором, у якому другий операнд обов'язково істинний, якщо істинний перший. У мові імплікацію між А та Б представляють такими словами: Б тоді, коли А; Б, якщо А; для Б достатньо А; для А необхідно Б; якщо А, то Б. Імплікація хибна тільки коли другий операнд хибний, а перший істинний. У всіх інших комбінація істинності та хибності операндів, імплікація є істинною.

Один із способів задати імплікацію у природній мові це висловлювання у такому вигляді “Якщо А є <підклас об'єктів>, то А є <клас об'єктів>”. Наприклад, “Якщо А є метал, то А є провідником струму”. Замість А можна підставляти назву будь-якого матеріалу, імплікація завжди буде працювати. При цьому, якщо підставимо поняття, яке не є матеріалом, то з точки природної мови отримаємо абсурдне речення. Наприклад, якщо замість А підставимо “книга”, або якесь абстрактне поняття типу “багатство”. До слова “багатство” ми не можемо застосувати поняття металевості, тому такий поділ мови на логічні структури вимагає розуміння контекстів і способів використання слів. Це споріднене із думкою Людвіга Вітгенштейна, що якщо не можна про щось сказати чітко, то про це краще мовчати. Тобто у певних випадках поєднання слів не буде нести якогось раціонального змісту, можемо сказати, не буде приносити користі. У такому випадку, той, хто буде застосовувати такі розбиття речень на певні логічні форми

повинен буде використовувати замість змінних, поняття із класу об'єктів, які вимагають інші слова з речення. Тобто він повинен буде розуміти способи використання наявних слів, щоб замість змінної підібрати слово, чи поняття, яке буде мати раціональний зміст. Це може слугувати певною критикою такого підходу, а саме підходу розбиття речень на певні логічні форми. Бо якщо ми говоримо, що ми можемо виділити певні логічні форми, то вартувало би мати правило підбору понять у ці логічні форми. Таке правило не можемо сформулювати, що підкреслює складність визначення значень слів як чогось відмінного від способу їх використання. Схожим чином і у першому законі логіки, законі тотожності, поняття повинні братись у той сам час, та у тому самому відношенні. Але час та відношення ми чітко визначити не можемо, бо відношення характеризується способом використання.

Якщо подивитись на імплікацію із точки зору множин, то перший властивість у першому операдні імплікації є підмножиною властивості другого у відношенні, чи контексті, що визначені у даному висловлюванні. У прикладі наведеному вище “метал” є підмножиною “провідник струму”. Тобто всі метали провідники струму, але не всі провідники струму є металами, наприклад вуглець за певних умов. Також можемо подати це таким чином, що не існує металу, який не є провідником струму. Візьмемо інший приклад “Об'єкт є книгою на англійській мові” та “Об'єкт використовує латиницю”. Між цими двома висловлюваннями є відношення імплікації. Тут “є книгою на англійській мові” є підмножиною “використовує латиницю”.

Для практичних цілей мови перше висловлювання імплікації повинно бути істинним, бо в іншому випадку можемо отримати висловлювання “Якщо Париж столиця Німеччини, то Дніпро впадає у Чорне море” [23]. Це висловлювання не порушує імплікацію, але має мало змісту у реальному світі, бо другу його частину можна робити як істинною так і хибною.

Є поняття формальної імплікації. Формальна імплікація це такий вид імплікації, у якому операнди мають між собою раціональний зв'язок [23]. Тобто вони збігаються за формою, від цього пішла і назва такої імплікації. У прикладі вище із Парижем та Дніпром наведено приклад імплікації, яка не є формальною, бо

ці висловлювання говорять про різні особливості об'єктів обговорення. Перше про столицю, а друге про річку у різних відношеннях. Ця імплікація є правильною, але не є формальною. У формальній імплікації обидва операнди мають певний спільний контекст. Якщо говорити термінами Людвіга Вітгенштейна, то вони належать до однієї мовної гри. А якщо виривати висловлювання із мовної гри до якої вони належать, то вони втрачають зміст у природній мові. Прикладом формальної імплікації може бути “Якщо А є бактерією, то А є одноклітинним організмом”.

У цьому випадку обидва операнди імплікації взяті у однаковому контексті біології. Тому таке їх поєднання має зміст у природній мові. Бертран Рассел пропонував використовувати такий вид імплікації щоб позначати закони природи.

Отже, різниця між імплікацією у логіці у широкому сенсі (йдеться не лише про формальну) і імплікацією, яку ми використовуємо в природній мові у тому, що у природній мові ми стараємось передати певний смисловий зв'язок між висловлюваннями, а у логіці ми фіксуємо хибність імплікації тільки тоді, якщо перший операнд істинний, а другий хибний.

Таблиця 1.1

Таблиця істинності для імплікації

A	B	Імплікація між A та B
1	1	1
1	0	0
0	1	1
0	0	1

Імплікація тісно пов'язана з поняттям достатньої підстави. У випадку імплікації перший операнд є достатньою підставою для другого. Тобто у виразі “А імплікація B”, істинність А є достатньою підставою для істинності B. Можна сказати, що достатня підстава та імплікація відображають одне і теж, просто визначені у різних контекстах. Імплікація визначена у контексті логічних

операцій, а достатня підстава у контексті ознак об'єктів обговорення. Достатньою є підстава, яка обов'язково обумовлює певний наслідок. Цей наслідок якраз і можна зобразити у вигляді імплікації.

Не варто плутати поняття достатньої підстави із поняттям необхідної підстави. Необхідною називається підстава певного явища, відсутність якої зумовлює відсутність цього явища. Прикладом достатньої, але не необхідної умови для істинності висловлювання "А є автомобілем" може бути "А визнали автомобілем року". Тобто, щоб визнати що А автомобіль, достатньо знати, що його визнали автомобілем року, але не обов'язково бути визнаним автомобілем року, щоб вважатись автомобілем. Прикладом необхідної, але не достатньої підстави істинності того ж висловлювання про автомобіль є "А має двигун". Адже двигун є складовою частиною (необхідною) будь-якого автомобіля, але при цьому окрім двигуна автомобіль потребує інших складових, наприклад трансмісія.

Достатні та необхідні підстави існують на різних якісних рівнях, або ж на різних рівнях абстракції. Тобто у випадку достатніх підстав для об'єкту також обов'язково повинні виконуватись необхідні підстави. В іншому випадку він не буде сам собою, бо за означенням ці підстави є необхідними. Тобто необхідні підстави тісно переплетені із поняттям об'єкту і його структурними особливостями. Достатні підстави говорять про об'єкт із більшого рівня абстрактності. Зв'язок між достатніми і необхідними умовами наведено на рис. 1.5. Візьмемо попередній приклад "А визнали автомобілем року". У цьому висловлюванні йдеться про певну категоризацію об'єкта без розглядання його структурних складових (двигун, трансмісія і тд.), які є необхідними для нього. При цьому при переході із одного більш абстрактного ніж структурний способу до іншого ми не виходимо за межі мовної гри. Тобто і в першому і в другому випадку під автомобілем мали на увазі певну марку автомобілів.

Еквіваленція це такий оператор який буде істинним лише якщо два операнда будуть одночасно або істинними, або хибними. Еквіваленцію можна виразити через імплікацію. Вона по суті є імплікацією у обидві сторони, від першого операнда до другого і навпаки.

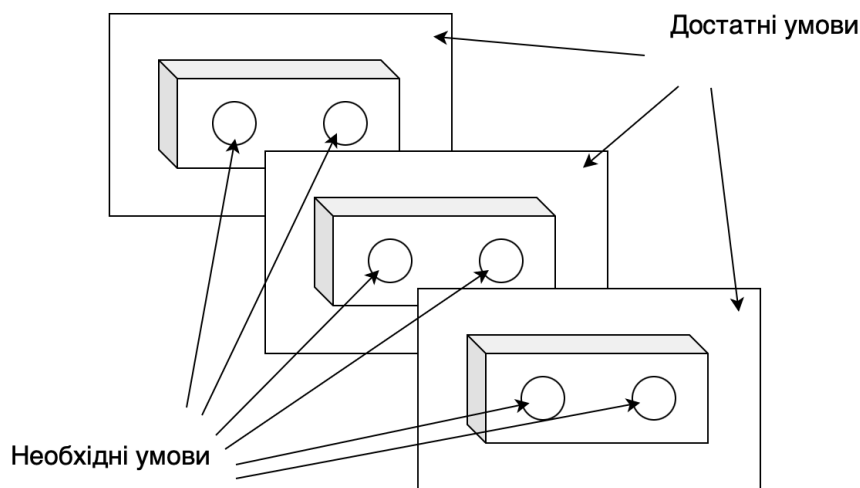


Рис. 1.5. Зв'язок між достатніми і необхідними умовами

Окрім логічних операторів є ще також поняття кванторів, поняття описових виразів, таких як предикати та терми. Предикат описує певну властивість або приналежність об'єкта. Наприклад “авто їде” можна зобразити через предикат так $P(\text{авто})$, де P - предикат. Якщо у попередньому виразі замінимо слово “авто” на змінну “ x ”, то вираз буде більш узагальненим і буде виглядати так $P(x)$. Він буде зображати процес їзди, де замість “ x ” можна підставляти об'єкти з такою властивістю. У даному випадку “ x ” є термом. Терми виступають логічним підметом у висловлюванні, а предикати логічним присудком. Логічний підмет позначає об'єкт висловлювання, а логічний присудок позначає що стверджується про об'єкт. Є два види кванторів:

- Квантор загальності
- Квантор існування.

Квантор загальності використовують коли потрібно зробити узагальнення. У природній мові йому відповідають слова “усі”, “кожен”, “будь-який” і так далі. Наприклад, “усі планети є кулястої форми”. Квантор існування у свою чергу зображає часткові випадки. У природній мові йому відповідають слова “існує”, “іноді”, “деякі”, “кілька” тощо. Прикладом може бути висловлювання “Деякі планети мають атмосферу”.

Якщо розглядати ці квантори у контексті множин, то квантор загальності показує відношення підпорядкування нашого об'єкту по відношенні до більш загальної множини об'єктів, а квантор існування описує існування певної підмножини об'єктів у нашій множині. Візуально залежність між цими кванторами зображено на рис. 1.6.

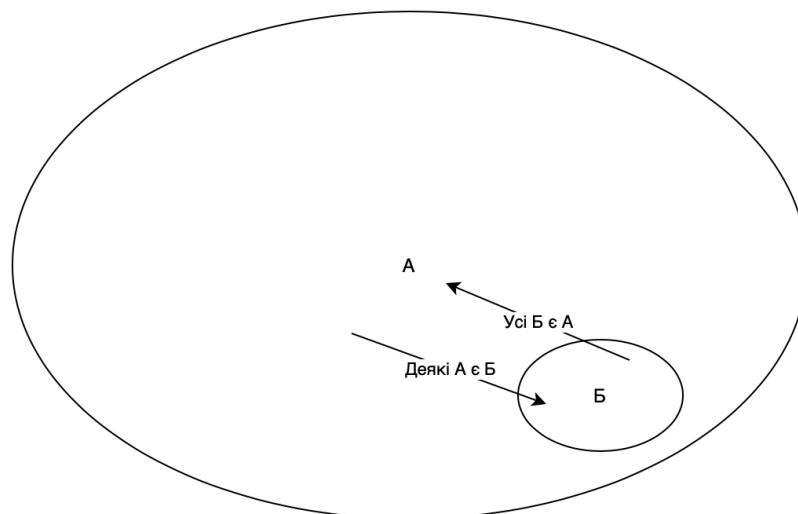


Рис. 1.6. Зв'язок між квантором загальності та існування

1.6. Стратегії вибору досліджень у галузі штучного інтелекту

Зараз моделі із величезною кількістю параметрів показують найкращі результати. Зазвичай моделі мають просту ціль (objective), наприклад передбачити наступне слово у тексті, але натреновані на величезних наборах даних. Основні дослідження часто зводяться до алгоритмів роботи із великими даними, розпаралелення тощо. Тому у дослідників штучного інтелекту виникає проблема у виборі досліджень, бо важко конкурувати із великими компаніями, що володіють великими можливостями у обчислювальних ресурсах. Ще десять років, якщо у вас був непоганий комп'ютер та доступ до Інтернету, то ви могли конкурувати із найкращими дослідниками. Інноваційні статті часто писали одна або дві людини, які проводили всі експерименти на своїх звичайних робочих комп'ютерах.

Завдяки глибокому навчанню стало зрозуміло, що масштабування працює. Від конкурсів ImageNet і їх різноманітних переможців до GPT-4 ми бачимо, що більше даних і більше обчислень, то кількісно, а часто навіть якісно, кращі

результати. Тобто кількість переростає у якість. Звичайно, також є вдосконалення алгоритмів навчання та архітектур мереж, але ці вдосконалення дійсно корисні лише в контексті великих даних. Опрацювання такої кількості даних неможливо досягти академічним дослідникам. Прості методи, які добре масштабуються, завжди перемагають коли стає доступним більше обчислювальних ресурсів. Помітно, що розрив між обсягом обчислень, доступним звичайним дослідникам, і обсягом, доступним для збереження конкурентоспроможності, зростає з кожним роком.

Один із способів конкурувати із великими корпораціями це спробувати масштабувати модель ще більше. Це коштовний підхід, адже він буде вимагати купівлю обчислювальних ресурсів у хмарних середовищах. А також перший експеримент не завжди вдалий, тому можливо буде проводити декілька, що в свою чергу ще збільшує затратність такого підходу.

Наступний спосіб протилежний попередньому. Тобто можна спробувати вирішувати спрощені задачі. Наприклад замість справжнього авто використовувати ігрову симуляцію із спрощеним середовищем. Якщо модель буде показувати себе краще у такому контексті ніж великі моделі, то у деяких випадках можна очікувати покращення і на реальних задачах, якщо її змасштабувати і натренувати на більшій кількості даних.

Інша річ, яку можна зробити з загальнодоступними попередньо навченими моделями це проаналізувати їх. Хоча це не може безпосередньо сприяти новим можливостям, це все одно може сприяти науковому прогресу. Нинішній стан речей такий, що у нас є чудові моделі для генерації тексту та зображень у відкритому доступі, але ми не дуже добре їх розуміємо. Наприклад, архітектура трансформер не є інтуїтивно зрозумілою для нас, і масштаб даних, на яких навчаються ці моделі, сам по собі є для нас незбагненим. Можна розробляти візуалізації та концептуальні підходи, що допоможуть нам їх зрозуміти. Також можна досліджувати обмеження таких моделей, пробуючи їх “зламати”.

Також можна працювати над маленькими моделями, які не потребують великих обчислювальних ресурсів. Вони є корисним для задач, які потребують щоб модель використовувалась часто (в онлайн режимі). Тобто вони потребують

великої швидкості отримання результатів моделі, і в такому випадку моделі із малою кількістю параметрів є необхідністю. Також інколи є потреба, щоб моделі були розгорнуті на пристроях користувача, наприклад якщо вони будуть застосовуватись без доступу до інтернету. Ці пристрої зазвичай мають значні обмеження по обчислювальних ресурсах і пам'яті, тому тут малі моделі також будуть доречними.

Іншою стратегією може бути розробка спеціалізованих моделей для якогось окремого домену, або вирішувати широкі задачі, але використанням великих натренованих моделей [24]. І саме використання великих натренованих моделей і є стратегією, що була вибрана у цій роботі.

1.7. Нейронні мережі для опрацювання природної мови

Опрацювання природної мови (NLP) є однією з найбільш активних і динамічних галузей у сфері штучного інтелекту [25]. У центрі цієї революції лежать нейронні мережі, які стали основою для численних проривів у розумінні мови машинами. Історично методи опрацювання природної мови базувалися на складних лінгвістичних правилах та статистичних моделях. Але глибокі нейронні мережі змінили гру [26].

Для розпізнавання зображень популярними були типи ознак, які використовуються в алгоритмі Віоли-Джонса [27]. Вони були сформовані вручну, але пізніше із використанням згорткових нейронних мереж (CNN) їх навчилися вивчати автоматично. Також і в опрацюванні природної мови раніше було популярним використання евристик, але зараз їх не потрібно формувати вручну, бо алгоритми машинного навчання можуть їх вивчити автоматично.

Нейронні мережі відрізняються своєю архітектурою, яка включає в себе кількість шарів, типи нейронів та їхні зв'язки. Центральною одиницею є нейрон, який отримує вхідні дані, обробляє їх і передає далі. Дані пропускаються через ваги, які адаптуються під час навчання.

Одношарові мережі, такі як перцептрони, є найпростішими і використовуються для лінійно роздільних задач. Однак вони обмежені в своїх можливостях апроксимації складних функцій [28]. Тому були розроблені

багатошарові перцептрони, які мають один або кілька прихованих шарів між вхідним і вихідним. Це дозволяє моделі бути універсальним апроксиматором, що може апроксимувати будь-які функції [29].

Застосування нейронних мереж до опрацювання природної мови вимагало створення спеціалізованих архітектур. Рекурентні нейронні мережі (RNN) були розроблені для опрацювання послідовностей даних, зокрема тексту. Вони мають пам'ять, яка зберігає інформацію з попередніх кроків, що дозволяє розглядати контекст слова у реченні [30]. Однак рекурентні нейронні мережі мають проблеми з довготривалими залежностями. Для рішення цієї проблеми були введені архітектури довгої короткочасної пам'яті (LSTM, Long Short-Term Memory) та GRU, які дозволяють інформації ефективно "проходити" через мережу без затухань [30].

Сьогоднішні підходи до опрацювання природної полягають у використанні машинного навчання для вивчення зв'язків між словами та їхньої семантики. Рекурентні нейронні мережі були одними з перших, що демонстрували значний успіх у задачах перетворення вхідної послідовності у вихідну послідовність, або ж seq2seq моделях [30]. Потім, LSTM вирішила основні проблеми рекурентний нейронних мереж, зокрема проблему затухання градієнта.

Згодом були розроблені GRU, які спрощують LSTM, зберігаючи їхню ефективність. Однак справжнім проривом стала архітектура трансформера, яка використовує механізми уваги для зосередження на важливих словах у великих текстових послідовностях. Моделі, такі як BERT, GPT і T5, базуються на трансформерах та демонструють надзвичайні результати у різноманітних задачах NLP.

Згорткові нейронні мережі, відомі своєю ефективністю в області обробки зображень, також виявилися корисними для опрацювання природної. Вони можуть вивчати локальні залежності в тексті, такі як біграми або триграми, і їх часто комбінують з RNN для оптимальних результатів [31].

Завдяки глибокому навчанню маємо змогу обробляти текст на безпрецедентно високому рівні. Це дає змогу виконувати задачі, такі як машинний

переклад, класифікація текстів, генерація мови, розпізнавати іменовані сутності та визначати сентимент з високою точністю.

Тим не менш, існують і виклики. Полісемія, коли одне слово має декілька значень, є великою проблемою для нейронних мереж. Крім того, моделі досі важко розпізнають іронію та сарказм. Хоча нейронні мережі вивчають відносини між словами, їх "розуміння" тексту є досить поверхневим.

Завдяки гнучкості архітектур нейронних мереж можливо їх адаптувати до конкретних завдань, змінюючи розмір мережі, кількість шарів або типи нейронів. Переклад, розпізнавання голосу, класифікація текстів та відповіді на питання - лише декілька з них.

Однією з великих переваг нейронних мереж є їхня здатність адаптуватися до нових мов та діалектів. Це зокрема корисно для мов, якими говорить невелика кількість людей. Передавальне навчання (transfer learning), коли модель навчається на одному наборі даних і адаптується до іншого, також є ключовим аспектом сучасного NLP.

Великі нейронні моделі, як-от GPT-3, можуть мати мільярди параметрів. Це призводить до питань про вартість навчання, екологічний вплив та доступність для дослідників із обмеженими ресурсами. З іншого боку, моделі зменшеного розміру, такі як DistilBERT, намагаються зберегти потужність оригіналу, але при менших обчислювальних витратах [32].

Слід враховувати, що більший розмір мережі може призвести до перенавчання. Тому важливо використовувати методи регуляризації, такі як випадкове видалення нейронів (dropout).

Загалом, нейронні мережі для опрацювання природної мови продовжують революціонізувати наш підхід до машинного розуміння мови. І хоча існують виклики, перспективи виглядають багатообіцяюче. В майбутньому очікується поява нових архітектур, покращення існуючих методів та розширення можливостей застосування нейронних мереж у різних галузях. Вони стали вирішальним інструментом у NLP завдяки їхній здатності адаптуватися до складних завдань опрацювання мови. Їх архітектура, що постійно

вдосконалюється, і здатність вивчати великі об'єми даних роблять їх незамінними у сфері штучного інтелекту.

1.8. Рекурентні нейронні мережі та архітектура довга короткочасна пам'ять

Рекурентні нейронні мережі представляють собою клас нейронних мереж, основне призначення яких – обробка послідовностей. Вони володіють унікальною особливістю: можливістю зберігати інформацію з попередніх кроків, що робить їх особливо корисними для аналізу даних, які мають послідовний характер, таких як текст або часові ряди.

Основна ідея рекурентних нейронних мереж полягає у використанні попереднього стану при обчисленні поточного стану. Тобто вони пам'ятають інформацію з попередніх кроків і використовують її для отримання поточного виводу. Проте рекурентні нейронні мережі мають свої обмеження. Однією з найбільших проблем є їхнє обмежене вміння моделювати довготривалі залежності в даних. Це означає, що коли між двома подіями в послідовності є велика “відстань”, рекурентна нейронна мережа може втратити здатність визначити зв'язок між ними. Для подолання цієї проблеми були розроблені архітектури довгої короткочасної пам'яті, або LSTM. Ці мережі є варіантом рекурентних, але вони мають специфічну структуру, що допомагає їм зберігати інформацію протягом довгих періодів часу [30].

Блок LSTM має елементи, які дозволяють їй визначати, коли зберігати, видаляти або читати інформацію. Ці елементи називаються "вентиллями". Є три основних типи вентилів: вхідний вентиль, вентиль забування та вихідний вентиль.

Вхідний вентиль контролює, коли і яку інформацію слід вводити в клітину пам'яті. Вентиль забування вирішує, коли відкидати інформацію, що вже не потрібна. Вихідний вентиль контролює, яка інформація буде передана на наступний крок [30].

Ця структура вентилів дозволяє LSTM зберігати інформацію протягом тривалого часу, не втрачаючи її, як це може статися в традиційних рекурентних

мереж. Вони показали відмінні результати в багатьох завданнях опрацювання природної мови, таких як переклад, розпізнавання мови та генерація тексту [33].

Окрім LSTM, існують інші варіанти рекурентних нейронних мереж, такі як вентильні рекурентні вузли, або GRU (Gated Recurrent Units), які також мають структури вентилів, але менш складні за LSTM.

Загалом, як рекурентні нейронні мережі, так і LSTM відіграють ключову роль у сучасних системах опрацювання природної мови. Їх здатність моделювати послідовності робить їх величезно цінними для вивчення мовних структур. Ці мережі працюють особливо добре, коли їх комбінують з іншими типами нейронних мереж, такими як згорткові мережі, для створення більш точних систем опрацювання природної мови.

1.9. Аналіз моделей, що використовують механізм уваги

Моделі, що використовують механізм уваги, вважаються однією з найбільш революційних ідей в галузі опрацювання природної мови за останні роки. Ця концепція була вперше запропонована для покращення роботи нейронних мереж у завданнях машинного перекладу. Суть механізму полягає в тому, щоб надавати більше ваги певним частинам вхідних даних під час генерації вихідних даних [34].

При перекладі довгих речень не всі слова вихідної мови однаково корисні для перекладу на цільову мову. Механізм уваги дозволяє моделі "зосереджувати" свою увагу на тих словах вхідного тексту, які є найбільш релевантними для поточного слова в перекладі. Ця ідея швидко стала популярною і була адаптована для багатьох інших завдань опрацювання природної мови, таких як аналіз настрою, генерація тексту та інші. Основна перевага механізму уваги полягає в його здатності динамічно адаптуватися до контексту [34].

Також такі моделі зазвичай вимагають менше параметрів та тренувальних даних, щоб досягти високої якості результатів порівняно з традиційними архітектурами, такими як RNN або LSTM. Однак, необхідно зазначити, що цей механізм не є панацеєю. У деяких випадках його застосування може бути надмірно складним або навіть контрпродуктивним.

Архітектура трансформер, яка базується на механізмі уваги, стала основою для багатьох сучасних моделей опрацювання природної мови [35], таких як BERT, GPT, T5 та інші. Її головна особливість - це відмова від рекурентності в користь паралельної обробки даних. Механізм уваги дозволяє трансформер моделям ефективно обробляти довгі послідовності даних, оскільки вони можуть "зосереджувати" увагу на релевантних частинах вхідного тексту без необхідності послідовно проходити через кожне слово або символ. На рис. 1.7 зображено як змінювались підходи до опрацювання природної мови.

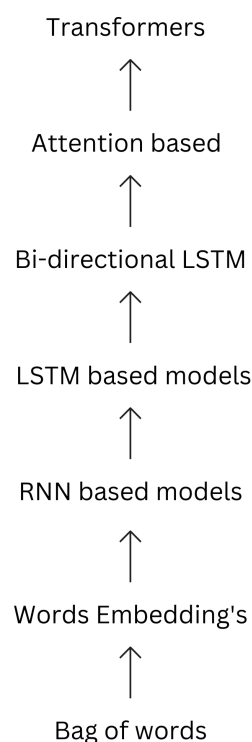


Рис. 1.7. Еволюція підходів опрацювання тексту

BERT є однією з найвідоміших моделей, заснованих на цьому механізмі. Ця модель була спроектована для генерації більш універсальних векторних представлень слова, враховуючи його контекст з обох боків. Це досягається завдяки використанню двонаправленого механізму уваги [36]. Така структура дозволила BERT встановлювати нові стандарти якості в багатьох завданнях опрацювання природної мови. Наприклад, в завданнях класифікації тексту, витягання інформації, а також відповіді на питання. Хоча механізм уваги значно

покращує роботу моделей, він може стати причиною певних проблем. Однією з таких проблем є “затухання уваги”, коли модель приділяє занадто багато уваги менш важливим частинам тексту. Також, такі моделі можуть мати тенденцію до перенавчання на конкретних наборах даних, якщо не використовуються належні регуляризаційні техніки. Особливо це стає проблемою для дуже глибоких моделей з великою кількістю параметрів. GPT - інший відомий представник на основі механізму уваги. Відмінність GPT полягає в тому, що він використовує однонаправлену структуру та зосереджений на генерації тексту.

1.10. Аналіз великих натренованих моделей для опрацювання природної мови

Останніми роками великі натреновані моделі стали домінуючими в галузі опрацювання природної мови [37]. Зокрема, моделі, як-от BERT, GPT-4 та T5, демонструють хороші результати в багатьох завданнях.

BERT, розроблений командою вчених з Google, став проривом в сфері NLP завдяки своєму методу двонаправленого тренування на основі архітектури трансформера. Його архітектура забезпечує здатність розглядати контекст слова з обох боків [36].

До BERT більшість моделей аналізували текст зліва направо або справа наліво. Це обмеження вдалося обійти в BERT, що дало йому змогу краще розуміти контекст. Основна ідея BERT полягає в тому, що він намагається передбачити пропущені слова в реченні, використовуючи контекст з обох сторін. Тренування BERT вимагає великих обчислювальних потужностей. Це пояснюється великою кількістю параметрів у моделі. Але після тренування, використання BERT для того, щоб робити передбачення (inference) не вимагає багато ресурсів.

GPT-4, створена OpenAI, представляє собою модель для генерації тексту. Підхід до тренування, заснований на прогнозуванні наступного слова в послідовності. За своєю архітектурою GPT-4 є трансформером, яка використовує механізм уваги для забезпечення високої якості генерування тексту [38]. Ця архітектура дозволила досягти значного прогресу у різних завданнях опрацювання природної мови. Однією з особливостей GPT-4 є її розмір. З кожною новою

ітерацією моделі GPT стають все більшими, і GPT-4 не є винятком. Це забезпечує моделі можливість вивчати величезні обсяги інформації. GPT-4 часто використовують для тонкого настроювання (fine-tuning). Завдяки цьому, модель може бути адаптована до конкретних завдань, які можуть відрізнятися від первинних завдань, що використовувались при тренуванні моделі [39]. GPT-4 була натренована на різноманітних джерелах даних, що забезпечило їй широкий спектр знань з різних областей. Спроби застосування GPT-4 до конкретних завдань виявили, що модель може бути надзвичайно корисною в ряді сценаріїв. Вона може бути використана для служб підтримки клієнтів, допомагаючи відповідати на запитання в реальному часі. Однак, існують і задачі, де GPT-4 може зіткнутися з труднощами. Наприклад, деякі складні проблеми потребують глибокого аналізу, або критичного мислення, тобто здібностей, які наразі відсутні в моделях подібного роду.

Модель T5 приймає текст на вхід і “перетворює” його на вихідний текст. Її ідея полягає в тому, щоб виконувати будь-яке завдання NLP як завдання перетворення тексту: вхідний текст перетворюється на вихідний текст. Центральною концепцією архітектури T5 є ідея “текст як вхід, текст як вихід” [40]. Замість того, щоб створювати специфічні моделі для кожного завдання, T5 може бути налаштований на будь-яке завдання, переформулювавши його як задачу перекладу. Архітектурно, T5 базується на архітектурі трансформера, що є основою для багатьох сучасних моделей опрацювання природної мови [40].

Сучасні моделі опрацювання природної мови мають величезну кількість параметрів. Тобто підходи полягають не у тому, що модель має складні цілі (objective) чи архітектуру. Вони мають досить просту ціль, як наприклад передбачення наступного слова, і архітектуру яку легко розпаралелити [41]. Тобто зараз прийшли до висновку, що навіть прості задачі такі як передбачення наступного слова є достатньо складними і вимагають розуміння семантики мови. Тому їх використовують як ціль для великих мовних моделей під час тренування і розробляють архітектури, що дозволяють паралельні обчислення, щоб можна було “прогнати” через модель багато даних. Такі великі розміри моделі призводить до покращення якості на багатьох завданнях опрацювання природної мови, але також

створює нові виклики. Зокрема, великі моделі потребують значних обчислювальних ресурсів як для тренування, так і для розгортання. Це може обмежити доступ до них для дослідників із обмеженими ресурсами. Крім того, великі моделі можуть є менш стійкими до перенавчання. Це змушує вчених застосовувати різноманітні методи регуляризації для вирішення цієї проблеми. Існує також питання етики та відповідальності у використанні таких моделей. GPT-4, наприклад, може генерувати текст, який виглядає переконливим, але може бути неточним або навіть шкідливим. Зрештою, хоча великі натреновані моделі демонструють вражаючі результати, їх ефективність та практичність залежить від конкретного застосування та обставин.

Враховуючи вищезазначене, основною перевагою є те, що модель, навчена на одному наборі даних, може ефективно застосовуватися для різних завдань без потреби в її повторному тренуванні. Така здатність виявляється особливо корисною у ситуаціях, коли доступний набір даних є обмеженим. Використання попередньо натренованої моделі може допомогти уникнути складних етапів тренування та оптимізації.

Виникають питання пов'язані з інтерпретованістю таких моделей. Через їхню складність та великий розмір, вони можуть виявитися “чорними ящиками”, в яких важко зрозуміти, як саме приймаються рішення [42]. Це стає проблемою, особливо у застосуваннях, де потрібна прозорість та зрозумілість моделі. Великі моделі можуть переймати упередження з даних, на яких вони були навчені. Це може призвести до неточних або упереджених висновків, коли модель використовується в реальному світі. Серед інших викликів великі витрати на ресурси. Не лише тренування, але і розгортання таких моделей вимагає значної обчислювальної потужності.

На рис. 1.8 зображено як росла кількість параметрів з часом. У моделі GPT-1, що була опублікована у червні 2018 року, є 117 мільйонів параметрів, а у GPT-3, що була опублікована у червні 2020 року, вже аж 175 мільярдів параметрів. Тобто за 2 роки розмір моделі зріс у 1000 раз.

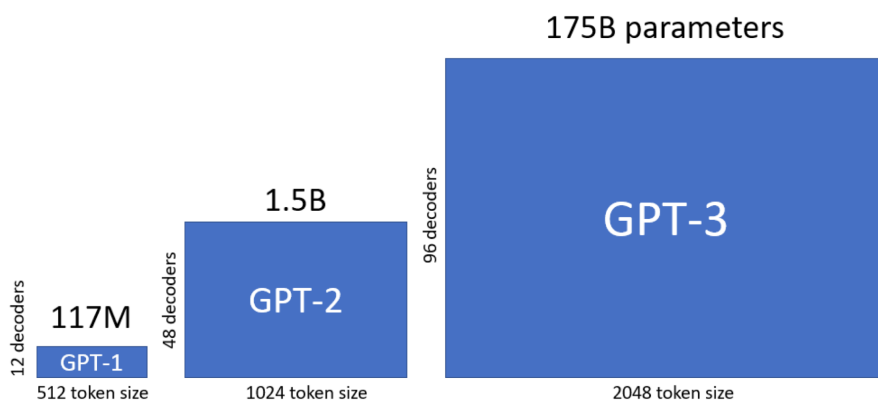


Рис. 1.8. Ріст кількості параметрів моделей GPT із роками

Висновки до розділу 1

Було аргументовано значення слова “семантика” у контексті даної роботи. Розглянуто задачу логічного виведення та аргументовано її складність для перевірки точності великих мовних моделей.

Досліджено транзитивне відношення у контексті задачі логічного виведення та набори даних для цієї задачі. Для деяких задач опрацювання природної мови хорошої точності можна досягти, використовуючи прості евристики.

Проаналізовано зв'язок між логікою та опрацюванням природної мови. А також досліджено проблематику логічних помилок мовних моделей та достовірність їх результатів.

Проаналізовано вимоги до наборів даних для опрацювання природної мови та стратегії вибору досліджень у галузі штучного інтелекту. В результаті аналізу було обрано стратегію використання попередньо натренованих великих мовних моделей, замість зміни архітектури моделей опрацювання природної мови.

Виявлено, що наразі нейронні мережі із архітектурою Трансформер і великою кількістю параметрів показують найкращі результати для опрацюванні природної мови.

РОЗДІЛ 2. ЗАСТОСУВАННЯ БАГАТОЕТАПНИХ ПІДХОДІВ ДЛЯ ГЕНЕРУВАННЯ ТЕКСТОВОГО КОНТЕНТУ

2.1. Способи представлення тексту

Природна мова є дискретною та розрідженою, і тому є складним джерелом даних [43-45]. Щоб текст можна було використовувати як вхідні дані, він спочатку має бути перетворений у відповідне представлення, яке зазвичай є вектором характеристик тексту, отже, вектором чисел. Векторні представлення тексту можна побудувати багатьма різними способами.

Найпростішим представленням тексту є модель торби слів (bag-of-words). У такому підході спочатку визначається словник слів. Зазвичай в упорядкований словник входять усі, або найчастіше вживані слова із документів. Після формування словника відбувається підрахунок, кожного слова для документу який потрібно представити у вигляді вектора [46]. Упорядкований набір підрахунків вживання кожного слова і є векторним представленням документу [47]. Інколи замість підрахунків використовують бінарні значення. Тобто “1”, якщо слово зустрічається в документі і “0” якщо ні. Цей спосіб представлення слів називається торбою слів, бо у ньому не враховується порядок слів, але лише їх наявність, або кількість.

Більш складним є підхід на основі статистичного показника TF-IDF. У ньому після формування упорядкованого словника, обчислюється значення показника TF-IDF для кожного слова. Значення TF-IDF, або значимість слова буде пропорційна кількості його вживань у документі, і обернено пропорційна кількості вживань цього слова у інших документах корпусу текстів [48]. TF-IDF складається із двох частин TF та IDF, які обчислюються за окремими формулами. TF визначає важливість слова у документі і є відношенням кількості цього слова у документі до кількості усіх слів у ньому. IDF зменшує значимість, або вагу широкоживаних слів і є логарифмом відношення кількості документів до кількості документів, у яких зустрічається слово [48]. Кінцевий показник TF-IDF є добутком TF та IDF. В результаті велику значимість, або вагу отримують слова що часто зустрічаються в документі, але рідко в інших документах. Можна сказати,

що ці слова будуть ключовими у цьому документі. Цей підхід на відміну від попереднього краще працює якщо потрібно зробити представлення більшої кількості тексту, наприклад документу, але якщо потрібно представити речення, то обидва дадуть схожі результати [49, 50].

Ще одним популярним підходом представлення тексту у вигляді вектора є алгоритм word2vec. Він використовує нейронні мережі із одним прихованим шаром, які навчені передбачати слово із контексту слів, або контекст за заданим словом [51]. Це дві різні архітектури, які в результаті кожне слово представляють у вигляді вектора. Цей вектор отримується взявши ваги прихованого шару. Тобто, як це часто буває, модель використовується не для тієї ж задачі для якої тренувалась. Задача слугує лише способом отримати ваги, мережі для представлення слова у векторному вигляді. При цьому над векторами можна здійснювати математичні операції [51]. Наприклад, часто для представлення речення знаходять суму всіх векторів слів. У такому підході не враховується порядок слів [52, 53], тому зараз використовують більш складні підходи на основі архітектури Transformer і беруть останній шар таких мереж як представлення цілих речень чи уривків тексту [54].

2.2. Спосіб використання великих мовних моделей

2.2.1. Загальний опис логіки як способу мислення

Логіка є однією із найдавніших галузей наукового знання. Цікаво, що саме логіку Арістотеля було першою відновлено після падіння античної цивілізації. В часи Середньовіччя було негативне ставлення до античної науки, але вона почала визнаватись саме з дослідження праць Арістотеля [23]. Пізніше у часи Відродження знову починають використовуватись логічні методи античної науки і з цього починається вся наука Нового часу разом з філософією Декарта та інших філософів.

Арістотель створив логіку як засіб боротьби із софістикою і захисту істини. Вона й надалі використовується у цих цілях. У часи Середньовіччя схоласти зробили вклад у логіку. Пізніше важливою фігурою був Ф. Бекон, а далі Г. Лейбніц, із якого почалась сучасна логіка [23].

Через дивовижні успіхи сучасної логіки закрадається думка про те, що традиційна логіка є непотрібною. Але люди, як і раніше, продовжують міркувати, спростовувати та доводити користуючись природною мовою. А для цього найефективнішим апаратом є апарат формальної, або традиційної логіки.

Різницею між традиційною і сучасною логікою є те, що перша аналізує форму мислення, а друга смисловий зміст висловлювань [23]. У традиційній логіці не так важлива конкретика мови, а скоріше абстрактні закони мислення, способи умовиводу і так далі. А у сучасній логіці досліджується як функціонує мова. Тобто перша переплетена із мисленням, а друга із мовою.

Мислення людини не є хаотичним, воно підлягає певним законам. Ці закони є законами логіки [55, 56]. Вони відображають певний сутнісний і неминучий зв'язок між думками. Закони логіки дозволяють підтримувати продуктивну дискусію та запобігають порушенню її принципів. Вони це реалізують через запобігання порушення принципів мислення та допомагаючи будувати доведення [56-58]. Дотримання законів логіки сприяє виникненню послідовності та зв'язності у розмові. У дискусії також потрібно забезпечити дотримання семантичної відповідності понять та етичних норм спілкування.

2.2.2. Аргументація проблематики використання мовних моделей у дискусіях

У дискусії потрібний суддя, бо у іншому вона часто є непродуктивною [57]. Тобто суддя повинен підтримувати якість дискусії, слідкуючи за правилами. Часто учасники починають полеміку, тобто стараються переконати опонента у своїй думці. Хоча ціллю дискусії має бути знаходження істини, а не переконання суперника у своїй правоті. Навіть якщо стараються переконати у своїй думці не завжди це відбувається “законними” методами. Наприклад, учасник може просто відкидати аргументи опонента, або відмовлятися у аргументації своїх думок. Повинні бути правла, але навіть якщо вони будуть, учасники зможуть їх порушувати і потрібно буде доводити іншим учасникам, що відбулося порушення правил.

Наявні платформи на яких люди дискутують не сприяють продуктивним дискусіям. Проаналізуємо дві популярні платформи у яких часто відбуваються дискусії: Реддіт та Твіттер.

Недоліком обидвох платформ є те, що той хто швидше відповідь має більший шанс отримати реакції. Це стимулює людей слідкувати за публікаціями, щоб раніше прокоментувати, але вносить упередження у процес дискусії.

Твіттер це соціальна мережа. Перевагою Твіттера є те, що є обмеження кількості символів повідомлення. Це заставляє учасників дискусії стисло формулювати думки і подавати по одному аргументу. Це обмеженням розміру повідомлення запобігає проблемі бомбардування аргументами. (Можна використати модель, щоб перевіряла чи дійсно наведено один аргумент) Іншою перевагою є те, що кількість підписників часто корелює із експертністю. Тому те, що їхні аргументи будуть отримувати більше позитивних оцінок буде корисним, бо часто їхні автори часто мають більше експертності. Але недоліком є те, що експертність не завжди відображається кількістю підписників. Останньою перевагою Твіттера є community note, тобто користувачі можуть робити факт-чекінг висловлювань.

Твіттер має декілька недоліків. Перший, більш популярні люди мають більшу вагу, бо їх більше читають і відповідно більше людей поставлять “Подобається” під їхнім коментарем. Тобто оцінюється не якість аргументу, а також популярність людини [59]. Другим вагомим недоліком є те, що Твіттер має лише кнопку “Подобається”, але немає кнопки “Не подобається”. За рахунок цього весь негатив відображається у коментарях і це стимулює токсичне середовище [59-60]. Ця проблема підвищує трафік Твіттеру і це може бути причиною чому її не виправляють.

Реддіт це форум. Перевагою Реддіту є неупередженість до особи, що залишила коментар. Принцип content-first, тобто те, що все залежить від змісту повідомлення [61-62]. Також перевагою, є симетрична система “Подобається та “Не подобається”. Тобто можна поставити і “Подобається”, і “Не подобається”, і вони мають однакову вагу. Останньою перевагою є те, що він ховає думки з

більшою кількістю оцінок “Не подобається” ніж “Подобається”. Недоліком Реддиту, є відсутність будь-яких обмежень довжини повідомлень.

Тобто обидві платформи мають мінімальні інструменти підтримки продуктивної дискусії. Наприклад, Реддіт часто ховає непродуктивні думки, а Твіттер стимулює стисло висловлювати по одному аргументу. Цього є недостатньо для продуктивної дискусії, через це нова платформа для дискусій буде корисною.

Нова платформа для дискусій буде включати систему для підвищення якості дискусій і мовленнєву модель у якості рефері.

Платформа буде мати такі особливості: дискусія відбувається почергово, аргументи записуються в блоки, один блок має лише один аргумент, на бінарні питання дозволяє лише варіанти так/ні, на модель будуть подаватись судження із змінними/плейсхолдерами замість реальних фраз і тд.

У випадку дискусій двох людей потрібен зовнішній спостерігач, що зможе їх розсудити. У соціальних мережах і форумах у якості рефері виступають інші користувачі які ставлять оцінки “Подобається” та “Не подобається”, але не завжди це ефективно.

У запропонованій платформі можна виключити фактор інших користувачів, бо можна у якості рефері використати мовленнєву модель. Проте основна логіка аргументації лежить на учасниках дискусії, а модель слугує лише рефері, який фінально затверджує результати. Щоб у моделі не було упередження, перед подачою тексту на модель, вона буде на неї подаватись із питанням замінити конкретику на заповнювач (placeholder).

Для сучасних мовленнєвих моделей задачею є поводитись схоже до людини, тому вони часто використовують розмиті поняття. Вони можуть вирішувати математичні задачі, але допускають помилок коли потрібно з використанням логіки опрацювати текст [63].

Наприклад, модель не може підтримувати філософську дискусію, бо основним інструментом філософії є логіка за допомогою якої поєднують факти. Модель може слугувати довідником і витягувати з пам'яті факти, які вона вивчила з тренувальних даних. Також у моделі є “розуміння” контекстів слів, що дозволяє казати, що вона знає значення тих слів, бо з точки зору філософії Людвіга

Вітгенштейна значення слова це спосіб його використання [64]. При цьому моделі важко дотримуватись логіки і підтримувати складну філософську дискусію.

Через це було прийнято рішення спростити задачу для моделі і логіку аргументації перенести на учасників.

Є схожі підходи, які дозволяють знаходити логічні помилки, але відмінність запропонованого полягає в тому, що основним способом довести неправоту суперника це показати неконсистентність, чи суперечливість його поглядів та картини світу. Зазвичай ціллю є знайти ті висловлювання, у яких ймовірно допущено одну із типових помилок у дискусіях. Прикладом таких помилок може бути конспірологічні теорії. Сама по собі помилка не є строго логічною, а лише прикладом помилки, типової для дискусій. Строгою логічною помилкою у цьому випадку може бути 4 закон логіки, що вимагає достатньої підстави. Тому для багатьох учасників дискусії сам факт, що система буде підозрювати їхнє висловлення у конспірології не буде строгим доказом неправдивості цього висловлювання. Проте аргумент, що немає достатньої підстави вважати висловлювання істинним, бо воно не аргументоване, для учасників дискусії буде переконливим, бо за філософією Іммануїла Канта закони логіки є апіорними [15].

В цьому контексті знаходження суперечностей між висловлюваннями буде важливою задачею для моделі опрацювання природної мови.

2.2.3. Метод використання мовленнєвої моделі для підтримки якості дискусій

Основний підхід запропонованого підходу схожий до “зведення до абсурду” (ad absurdum). У зведенні до абсурду, для підтвердження істинності свого судження показується, що у протилежному випадку отримуємо логічну суперечність. Цей підхід є продовженням 2 закону логіки, що два протилежних судження не можуть бути істинними одночасно. Різницею між цим підходом та підходу “зведення до абсурду” є те, що використовуються аргументи опонента і до абсурду зводиться його картина світу. Тобто ціллю є пошук суперечностей у поглядах опонента, а не наведення нового прикладу, як це робиться у підході “зведення до абсурду” [65].

Також цей підхід схожий до підходу певних бойових мистецтв, у яких зброя противника використовується проти нього. Схожістю є те, що у запропонованому підході аргументація будується на розкритті картини світу опонента, щоб можна було підсвітити у ній суперечності. Тобто щоб перемогти опонента у дискусії використовується його ж картина світу.

Мовленнєва модель буде виконувати роль судді. Для того, щоб вона якісно працювала потрібно спростити їй задачу, бо було виявлено, що вона допускає помилки, відповідає надто обширно і розмито [65]. Наприклад, навіть якщо є суперечності у судженнях, модель може їх не помітити, а замість того відповісти, що певний аргумент є недостатньою підставою для висновку, базуючись на досвіді про конкретні події, що вивчені із тренувальних даних. При тому, якщо замінити конкретні події, країни на змінні А, Б, В, то модель каже, що немає однозначної відповіді і потрібно враховувати контекст.

Для початку ця платформа може буде використана для окремих дискусій між двома людьми. Причому одне і те ж питання може бути обговорене різними людьми у різних дискусіях. Але коли набереться база дискусій можна буде створити також базу знань. У цій базі знань будуть агреговані популярні питання і аргументи які їх стосуються. Тобто кожен зможе глянути які аргументи підтверджують, чи спростовують певну думку.

Для забезпечення неупередженості формулювань які передаються на модель, платформа буде дозволяти обмежені елементи інтерфейсу. Тобто формулювання природною мовою буде лише у блоках, а решта дій буде робитись за допомогою елементів інтерфейсу платформи. Наприклад, після завершення написання тексту у блоці, текст кожного цього блоку буде передаватись на модель із запитанням “Чи є текст питанням із можливими відповідями Так/Ні? Дай коротку відповідь, так, чи ні.”. І якщо модель відповість так, то для учасника дискусії, що буде відповідати на запитання появляться кнопки “Так” і “Ні”. Це забезпечить конкретні відповіді замість розмитих. Далі, такі блоки можна буде автоматично опрацьовувати правилами класичної логіки. При цьому в учасників дискусії можна буде уточнювати питання, якщо воно не достатньо чітко сформульовано.

Є 4 основні закони формальної логіки, або класичної логіки [23]. Вони разом із підходом зведення до абсурду є основним інструментарієм платформи. Для підтримки підходу зведення до абсурду перевіряти чи правильно проведені аналогії. Перші три закони логіки сформулював Арістотель. Четвертий сформулював Лейбніц, але такі ж думки висловлював і Паскаль [23].

Перший закон логіки це закон тотожності. Він полягає у підміні понять. Питання до моделі: “Чи у даному судженні поняття вживається у іншому значенні ніж вказано у визначенні?”, “Чи у даному випадку відбувається підміна понять, якщо враховувати початкове визначення?”. На початку дискусії і протягом неї потрібно питаннями до опонента уточнювати поняття і вимагати вказати визначення до них. Також до понять, які використовуються для пояснення інших понять.

Другий закон логіки це закон суперечності. Він полягає у тому, що не два протилежні судження не можуть бути одночасно істинними. Питання до моделі: “Чи можна судження А виразити, як не Б?”, “Чи можна судження Б виразити, як не А?”. Якщо вони протилежні і обидва учасник вважає істинним, то можна констатувати логічну помилку.

Третій закон логіки це закон виключення третього. Він полягає у тому, що кожне судження може бути або істинним, або хибним. Класична логіка є бінарною і працює лише із такими судженнями. У іншому випадку не можна буде застосовувати закони логіки. Є і види логіки, які не вимагають бінарної відповіді, але вони не застосовуються у цій роботі. Давати учасниками на інтерфейсі лише відповіді так, або ні. Питання до моделі: “Чи дане питання вимагає відповіді Так, або Ні?”.

Четвертий закон логіки це закон достатньої підстави. Він полягає у тому, що думка може бути істинною тільки тоді, коли вона обґрунтована. Тобто повинні бути наведені аргументи, що її підтверджують. Питання до моделі: “Чи у даному судженні є відмова від аргументації своєї думки?”. Це досить розмитий закон у якому важко контролювати чи наведених аргументів є достатньо.

Через те, що основною метою є продемонструвати неконсистентність/ суперечливість теорії опонента, дискусія має бути наочною. Тобто чітко повинно

бути видно аргументи та зв'язки між ними. Для зображення цього підходять графові структури. Тобто дискусія буде зображатиме графом, де аргументи є його вершинами, а ребрами буде позначатись перехід від одного аргументу до іншого.

Для реалізації закону тотожності до кожного аргументу можна буде задати питання, щоб уточнити значення його понять.

Щоб реалізувати закон суперечності буде опція на інтерфейсі, щоб позначити два судження як ті, що конфліктують між собою. Вони будуть з'єднуватись між собою ребром унікального кольору з написом суперечність.

Для реалізації закону виключення третього, учасники дискусії зможуть створювати бінарні питання. Тобто питання які мають відповідь так, або ні. На такі питання не можна буде відповідати розлого, а лише вибрати одну з двох опцій на інтерфейсі: так, ні.

Реалізацією закону достатньої аргументації, слугує сама форма дискусії у вигляді графу. Бо у такому вигляді можна чітко проводити аргументацію, поєднуючи одну вершину з іншою.

Загалом платформа без використання мовної моделі буде корисна для дискусій. Бо ціллю дискусії є досягти істини, тому учасники не будуть зловживати можливістю завести дискусію в тупик. Наприклад, не будуть уникати аргументації. У дискусіях основною проблемою є саме відсутність наочності, через це учасники можуть не бачити недоліків своєї теорії.

Полеміка це вид суперечки, у якому на відміну від дискусії кожен учасник хоче переконати опонента у своїй правоті, а не досягти істини. У такому випадку корисною буде мовна модель, яка буде виступати рефері і підтверджувати дії учасників суперечки.

У двійковій, або булевій логіці є такі базові операції: кон'юнкція, диз'юнкція та заперечення. Вторинними є імплікація, виключна диз'юнкція та еквівалентність. У булевій логіці є такі властивості логічних операцій: комутативність, ідемпотентність, асоціативність, дистрибутивність, закон де Моргана, поглинання та інші. Ці правила рідко порушуються у дискусії, бо самі по собі є складнішими а учасники рідко вдаються до такої складної аргументації, тому вони не використовуються у даній роботі. Наприклад, візьмемо правило

дистрибутивності $A \text{ і } (B \text{ або } V) = (A \text{ і } B) \text{ або } (A \text{ і } V)$, де A , B , та V аргументи. У дискусіях зазвичай не використовують такі складні структури для аргументації, але це може бути потенційною ідеєю для покращення системи.

Тобто особливостями системи будуть: відображення дискусії у вигляді графу, можливість задати уточнюючі питання, можливість задати бінарне питання (знаючи чітку відповідь на нього можна буде якісніше знаходити суперечності), наочний інтерфейс позначення суперечностей між відповідями на бінарні запитання.

Основною функцією графу є наочність дискусії. Графова структура також дозволить задавати питання, що стосуються декількох аргументів. Наприклад, можна запитати “чи підтримуєш закон A ?” і пізніше “чи підтримуєш закон B ?”. У випадку, якщо на одне буде відповідь “так”, а на інше “ні”, то можна буде з’єднати ці два із третім у якому запитати “ви вибірково підтримуєте закони?”. Якщо на третє відповідь буде “ні”, то можна судити про логічну суперечність. Платформи Твіттер і Реддіт дозволяють цитувати попередні коментарі, що дозволить досягти схожого ефекту, але у графі можна буде, читаючи оригінальний аргумент, зразу перейти по ребру графа і побачити з яким він конфліктує.

У Реддіт читання коментарів побудоване як обхід графу у глибину, а у Твіттер як обхід у ширину. Обхід у глибину дозволяє зручно зануритись у обговорення однієї лінії дискусії, а у ширину наочно побачити всі аргументи якими підкріплюється дана теза. Кожен з них має свої переваги і недоліки, але для комфорту і наочності дискусії повинна бути можливість робити як і перший так і другий. Тому зображення у вигляді графу буде корисним.

Можливість задати уточнюючі питання є і у Реддіт та Твіттер, але по суті вони є семантично іншими і повинні виділятися на інтерфейсі як щось додаткове, або автоматично приховуватись. Тому у переписках даних соціальних мережах можна загубитись, бо не зрозуміло де уточнення, де аргумент, а де бінарне питання.

Можливість задати бінарне питання є і у згаданих соціальних мережах, але не факт що відповідь буде бінарною. Інший учасник дискусії може вдатись до розмитої відповіді, що порушить 3 закон логіки про виключення третього.

Також суперечність між судженнями можна вказати у цих соціальних мережах, але її не можна буде так чітко визначити як у випадку коли у вас є чіткі відповіді на бінарні запитання.

На рис. 2.1 зображено діаграму варіантів використання, як формально зображає основний функціонал платформи [65] у якій можна застосувати велику мовну модель у якості судді.

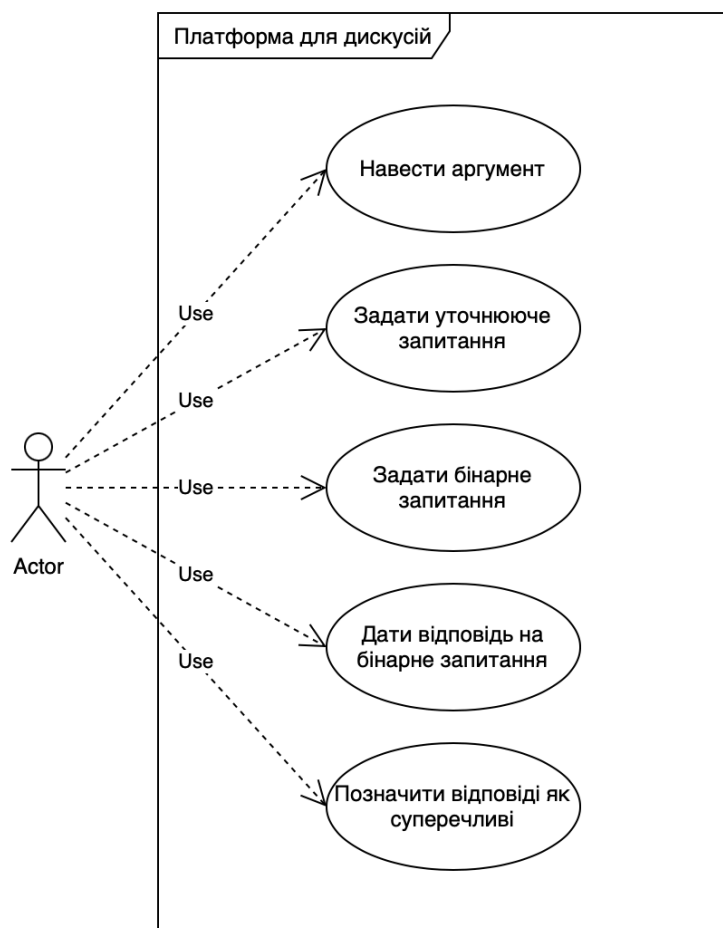


Рис. 2.1. Діаграма варіантів використання платформи, що використовує велику мовну модель у якості судді

Розмитістю понять, або невпевненість моделі можна перевірити спробувавши переконати її у протилежному, а потім спробувати вказати на істинність її початкової відповіді. Якщо модель буде міняти свою думку, то означає що вона погано вивчила ці поняття. Прикладом можуть слугувати логічні задачі. При тому арифметику модель вивчила якісно, тому її не вдається

переконати у тому, що наприклад “ $2+2=5$ ”. І у випадку із цим виразом модель не міняє своєї думки незалежно скільки разів намагатись її переконати.

Для покращення якості моделі можна сказати згенерувати певний текст, а потім старатись застосувати логічні правила, наприклад дистрибутивності, щоб перевірити логіку висловлювань. Після цього на прикладах де модель помилилась можна буде її дотренувати.

Тобто запропоновано приклад використання великої мовної моделі у контексті платформи для дискусій. Під час дискусій на інтернет форумах часто важко прийти до висновку хто правий, бо кожна сторона не хоче визнавати себе неправою. Суддею у таких дискусіях може бути модель аналізу природомовного тексту, яка буде знаходити логічні помилки, робити сумаризацію та інше. Така мовленнєва модель буде корисною, бо Реддіт та Твіттер часто не сприяють продуктивній дискусії. Реддіт це форум, а Твіттер соціальна мережа, і жодна з цих платформ не є націлена суто на дискусії. Модель яка зможе аналізувати текст потенційно зможе покращити якість дискусій.

2.3. Проблема узгодження мовних моделей із потребами користувачів

Проблема узгодження (alignment) із потребами користувачів, або проблема контролю штучного інтелекту, є актуальною проблемою з якою стикаються дослідники при роботі із великими моделями опрацювання тексту [66]. Ця проблема виникає через те, що великі мовні моделі зазвичай натреновані для передбачення наступного слова, а не для того, щоб відповідь співпадала з більш конкретними очікуваннями користувача. Наприклад текст, який генерує модель GPT-3 інколи є неправдивим, токсичним, або відображає шкідливу поведінку [67]. Тобто ці моделі не узгоджені із користувачем, бо вони не відповідають його конкретним потребам. Це може нести велику небезпеку при використанні таких моделей у реальних умовах. Крім цього великі мовні моделі мають так звані галюцинації. Це коли вони генерують неправдиві відповіді і придумують факти. Це у свою чергу може призвести, наприклад, до генерування неправдивих новин, навіть якщо модель натреновано генерувати лише достовірні результати.

Найбільші мовні моделі наразі є натреновані компанією OpenAI. Ця компанія використовує навчання з підкріпленням на основі зворотного зв'язку людини (reinforcement learning from human feedback, RLHF) для вирішення проблеми узгодження [68]. Для цього вони використовують запитання, що були задані справжніми користувачами їхнього API. Люди, що розмічають дані, наводять приклади очікуваної поведінки моделі і ранжують декілька відповідей моделі від більш підходящого, до менш підходящого. Після цього вони використовують ці дані для дотреновування (fine-tuning) моделі.

Результуючі моделі мають назву InstructGPT і вони краще слідують інструкціям користувача. Також вони рідше придумують факти і є невелике зменшення токсичності. Люди, що розмічали дані, надали перевагу моделі InstructGPT порівняно із GPT-3, хоча вона мала в 100 раз менше параметрів. Ці моделі зараз є моделями за замовчуванням у їхньому API. Їх тренування зайняло близько 20 тисяч годин людського зворотнього зв'язку [69].

Загалом, OpenAI вважають, що дотреновування мовних моделей із зворотнім зв'язком від людей є потужним засобом покращення надійності результатів великих мовних моделей [70].

Моделі InstructGPT поводять себе краще ніж базові моделі і ті, що використовували підхід навчання із учителем. Також вони показали себе краще ніж моделі, яким було на вхід передані приклади (prompted model). Цікаво, що якість моделей, яким на вхід були передані приклади, стрімко росте з ростом кількості параметрів моделі. Хоча моделі натреновані із підходом навчання з учителем, тобто натреновані на розмічених даних, показують себе краще ніж моделі, яким було передано приклади на вхід, розрив між їхньою точністю зменшується коли кількість параметрів наближається до 175 мільярдів (рис. 2.2).

Для навчання моделей InstructGPT основною технікою є навчання з підкріпленням на основі зворотного зв'язку людини. Ця техніка використовує людські вподобання як сигнал винагороди для точного налаштування моделей [68]. Важливо, що проблеми з безпекою та узгодженням із бажаною поведінкою є складними та суб'єктивними, і не повністю враховуються простими

автоматичними показниками, тому зворотній зв'язок людини вирішує цю проблему.

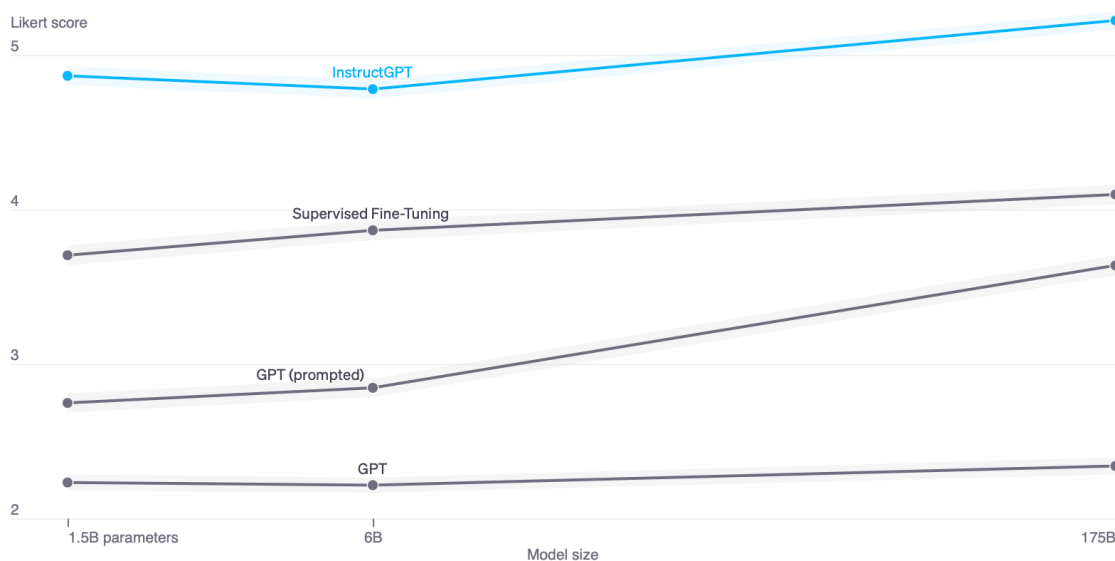


Рис. 2.2. Порівняння InstructGPT з іншими моделями

Спочатку збирається набір даних із запитів, що робили звичайні користувачі API OpenAI, і він використовується для тренування моделі на розмічених даних із підходом тренування з учителем. Далі збирається набір даних порівнянь, розмічених людьми, між двома виходами моделі на більшому наборі із запитів до API. Потім тренується модель винагороди (RM) на цьому наборі даних, щоб передбачити, якому результату віддадуть перевагу люди, що розмічали дані. В кінці, використовується RM як функція винагороди для дотреновування лінії поведінки (policy) моделі GPT-3, щоб максимізувати цю винагороду за допомогою алгоритму PPO (рис.2.3).

Один із способів представлення цього процесу полягає в тому, що він “розблоковує” можливості, які вже були в GPT-3, але їх було важко виявити лише за допомогою конструювання промпту. Тому, що така процедура навчання має обмежені можливості навчити модель новим можливостям порівняно з тими, що вивчаються під час попереднього навчання, оскільки вона використовує менше 2% обчислень і даних порівняно із попереднім навчанням моделі [70].

Обмеження цього підходу полягає в тому, що він запроваджує “податок на узгодження”: узгодження моделей лише з клієнтськими завданнями може погіршити їх продуктивність у деяких інших академічних завданнях NLP. Це небажано, оскільки, якщо такі методи узгодження погіршують моделі на типових завданнях, що використовуються для тестування моделей, то менш імовірно, що вони будуть використані на практиці. Проте, було виявлено, що якість моделей на типових завданнях, що використовуються для перевірки, не погіршилась, а узгодження із очікуваною поведінкою моделі покращилось [70].

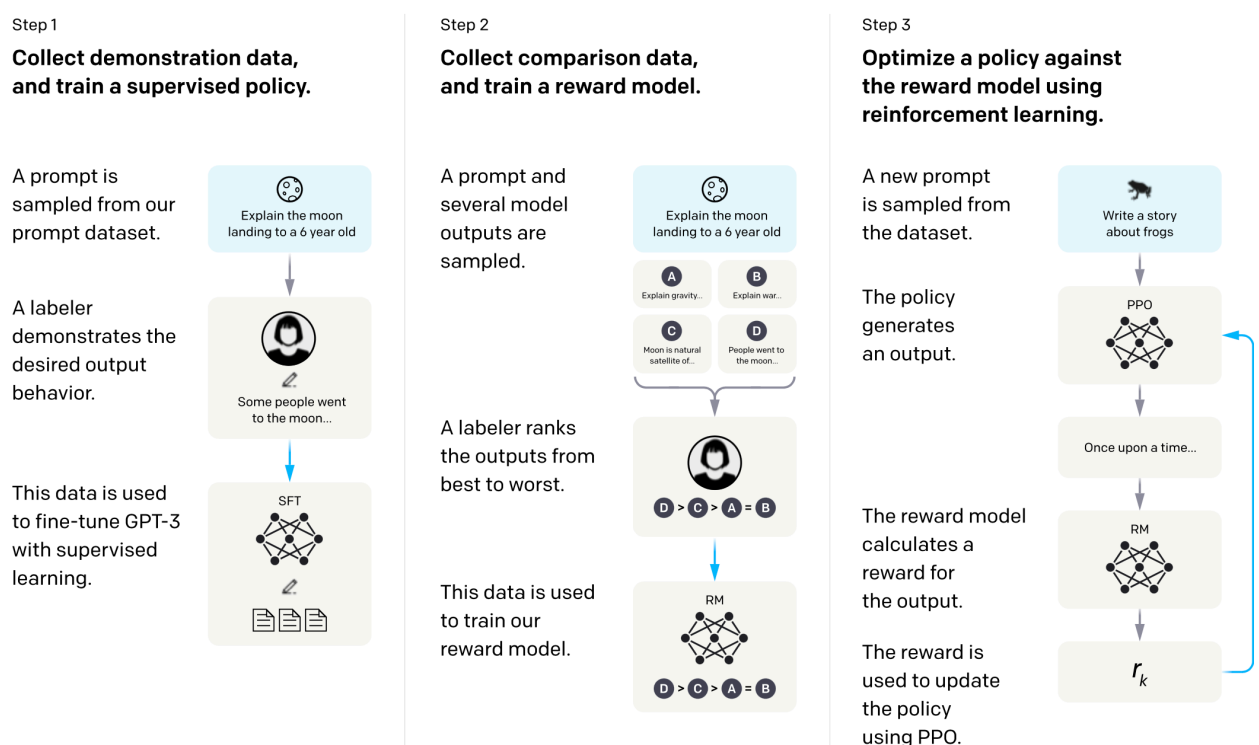


Рис. 2.3. Опис алгоритму для навчання моделі InstructGPT

Також було виявлено, що дотреновування моделей навіть на невеликих наборах даних значно покращує її відповіді у тому значенні, що поведінка таких моделей стає ближчою до очікуваної [71]. Чим більша мовна модель, тим таких підхід із дотреновуванням є більш ефективним. OpenAI у своєму дослідженні вибрали категорії, які мають прямий вплив на благополуччя людини, і описали бажану поведінку в кожній категорії в основному на основі законодавства США та міжнародного законодавства про права людини та західних соціальних рухів за

рівність людей, таких як рух за громадянські права США. Прикладами бажаних відповідей є такі, що протистоять насильству, не діагностують захворювання та не призначають лікування, протистоять підриву демократичних процесів тощо. Далі було створено набір даних із 80 зразками тексту; кожен зразок був у форматі запитання-відповідь і містив від 40 до 340 слів [71]. Для розуміння масштабу цей набір даних становив приблизно 120 КБ, приблизно 0,000000211% навчальних даних GPT-3. Потім стандартними способами дотренували модель (fine-tuning) GPT-3 (починаючи з версій на 125 мільйонів параметрів і до версій із 175 мільярдів параметрів) на цьому наборі даних. Також було дотреновано інші моделі на схожому наборі даних, але без інструкцій, що визначають очікувану поведінку. Такі моделі було натреновано, щоб сформувати контрольну групу для статистичного тестування.

При аналізі точності моделі у слідкуванні бажаній поведінці було виявлено, що модель дотренована на наборі даних із менше ніж 100 прикладами показує себе значно краще. Ця різниця є статистично значущою [71]. На рис. 2.4 зображено блакитною лінією таку модель, а також базову модель червоною і контрольну групу зеленою. Також видно, що із збільшенням розміру моделі росте і різниця у якості порівняно із базовою. Після цього дослідження OpenAI сформулювали наступні питання, щодо майбутніх досліджень:

- З ким слід проконсультуватися під час розробки набору даних?
- Хто несе відповідальність, коли користувач отримує результат, який не відповідає його власним цінностям?
- Як це дослідження застосувати до неанглійських мов і генеративних моделей поза мовою, таких як зображення, відео чи аудіо?
- Наскільки надійна ця методологія для реальних запитів?

На рис. 2.4 наведено порівняння точності по п'ятибальній шкалі моделі, що дотреноувалась на 100 прикладах (блакитний) з базовою моделлю (червоний) і дотренованою на випадкових прикладах (зелений). З ростом кількості параметрів моделі збільшується розрив між базовою і дотренованою.

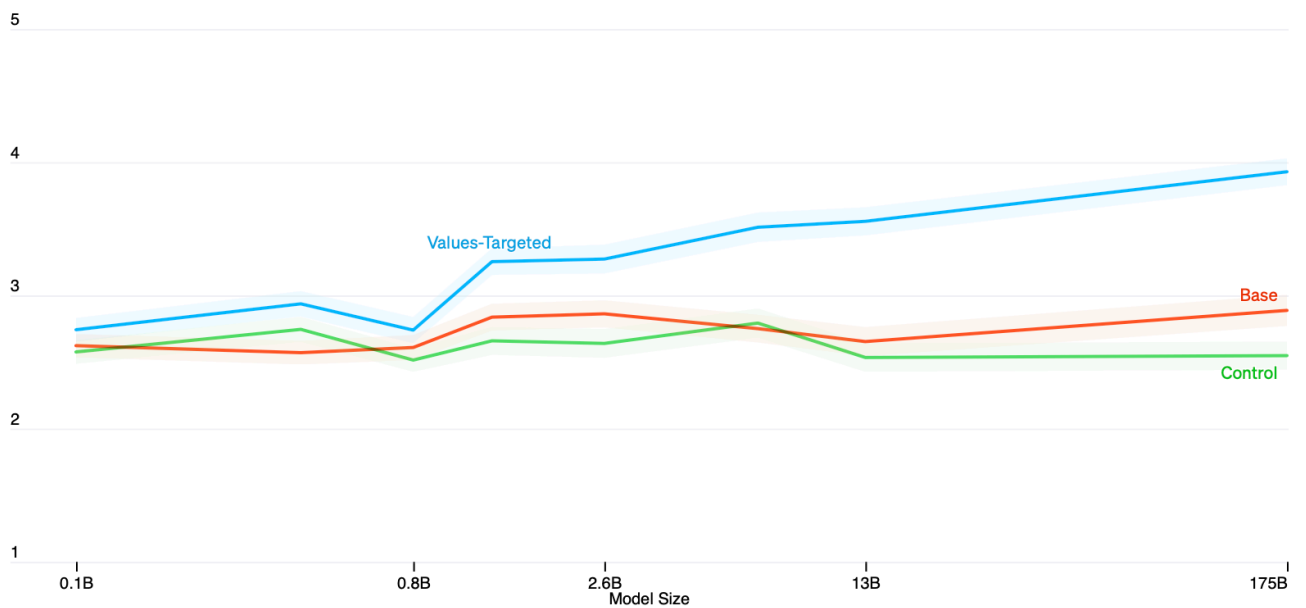


Рис. 2.4. Порівняння точності моделі, що дотреноувалась на 100 прикладах

2.4. Навчання з підкріпленням на основі зворотного зв'язку людини та Proximal Policy оптимізація

Цільові функції написані людьми можуть бути занадто простими, або може бути неправильно визначено складну ціль. Це може призвести до небажаної поведінки моделей. Тому актуальним є підхід, у якому модель визначає, що є бажаним результатом, отримавши від людей інформацію про те, яка з двох запропонованих моделей поведінки краща.

Навчання із підкріпленням за відгуками людей це алгоритм навчання, який використовує невелику кількість відгуків людини для вирішення сучасних задач навчання з підкріпленням (RL). Системи машинного навчання із зворотним зв'язком людини досліджувалися раніше, але цей підхід дає змогу працювати над набагато складнішими завданнями. Наприклад, йому було потрібно 900 біт зворотного зв'язку від оцінювача, щоб навчитися виконувати зворотне сальто у середовищі OpenAI Gym під назвою “Norper” [68]. Це просте завдання, яке легко оцінити, але складно визначити. Загальний процес навчання зображено на рис. 2.5.

Спочатку агент штучного інтелекту починає діяти випадковим чином у середовищі. Періодично людині передають два відеозаписи його поведінки, і

людина вирішує, який із двох роликів ближче до досягнення її мети — у цьому випадку сальто назад. Штучний інтелект поступово створює модель мети завдання, знаходячи функцію винагороди, яка найкраще пояснює судження людини [68, 72]. Потім він використовує RL, щоб дізнатися, як досягти цієї мети. У міру того, як його поведінка покращується, агент продовжує запитувати відгук у людини про пари дій, де він найбільш не впевнений щодо того, яка краща, і далі вдосконалює своє розуміння цілі.

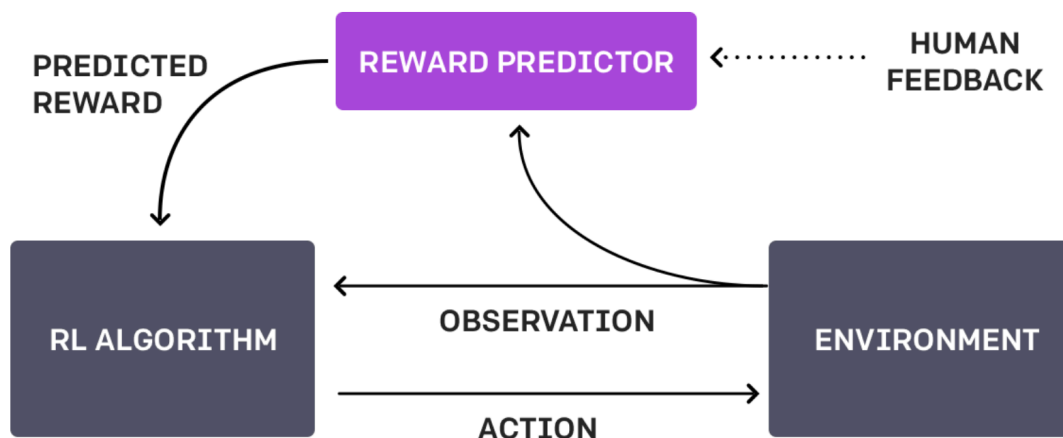


Рис. 2.5. Опис навчання з підкріпленням за відгуками людей

Цей підхід демонструє багатообіцяючу ефективність, бо як було сказано раніше, для виконання зворотного сальто у середовищі OpenAI Gym відео потрібно менше 1000 біт людського відгуку. Це зайняло менше години часу людини-оцінювача, тоді як у фоновому режимі було накопичено близько 70 годин загального досвіду (змодельованого набагато швидше, ніж у реальному часі).

Цей метод тестували на низці завдань із симуляції робототехніки та доменів Atari (без доступу до функції винагороди, тобто в Atari без доступу до результатів гри). Агенти можуть вчитися на відгуках людей, щоб досягти високої, а часом і надлюдської продуктивності в багатьох ігрових середовищах, на яких вони тестувались [68].

Немає необхідності, щоб зворотний зв'язок узгоджувався зі звичайною функцією винагороди в середовищі. Тобто агенту не обов'язково просто

максимізувати стандартну функцію винагороди в середовищі. Навчання на основі зворотного зв'язку ефективніше, ніж навчання з підкріпленням зі звичайною функцією винагороди, тому що людина формує винагороду краще, ніж той, хто написав винагороду для середовища [72, 73]. Тобто бали які отримує гравець це стандартна винагорода написана розробниками гри, а якщо винагороду формує людина, то вона може вказувати на частково хороші дії, тобто ті які зараз не дають балів, але можуть дати їх потенційно.

Ефективність такого алгоритму залежить лише від інтуїції оцінювача щодо того, яка поведінка виглядає правильною, тому, якщо людина погано розуміє завдання, вона може не надати корисного відгуку. Відповідно, у деяких доменах таких підхід може призвести до того, що агенти вироблять поведінку, щоб обманювати оцінювачів, а не виконувати завдання [68, 74]. Наприклад, робот, який замість того, щоб брати предмети розташовує свій маніпулятор між камерою та об'єктом так, що здавалося, що він його бере. Щоб такого не траплялося потрібно додавати додаткову розмітку, що допоможе оцінювачам розуміти, чи дійсно виконується завдання.

У OpenAI знадобилося дві години, щоб створити нашу власну функцію винагороди, щоб змусити агента виконувати сальто назад у середовищі OpenAI Gym. І хоча агенту це вдається, це сальто є менш елегантним візуально, ніж те, яке було навчено за допомогою зворотного зв'язку людини. Тому можна зробити висновок, що є багато випадків, коли людський відгук може дозволити визначити конкретну ціль більш інтуїтивно та швидко, ніж це можливо, формуючи мету вручну [68].

Інший алгоритм, що використовується OpenAI для навчання із підкріпленням називається Proximal Policy оптимізація (PPO). Він став стандартним алгоритмом навчання підкріплення в OpenAI через його простоту використання та високу продуктивність [75].

Policy Gradient методи є фундаментальними для останніх проривів у використанні глибоких нейронних мереж, від відеоігор до 3D-пересування. Але отримати хороші результати за допомогою цих методів складно, оскільки вони чутливі до вибору розміру кроку. Надто малий, і прогрес занадто повільний;

занадто великий, і сигнал переповнений шумом, або можна побачити катастрофічні падіння продуктивності [75]. Вони також часто мають дуже низьку ефективність з точки зору кількості даних для тренування, виконуючи мільйони (або мільярди) кроків, щоб вивчити прості завдання. Дослідники намагалися усунути ці недоліки за допомогою таких підходів, як TRPO та ACER, обмежуючи чи якимось іншим чином оптимізуючи розмір оновлення стратегії. Проте, цих методів є свої компроміси та недоліки.

За допомогою навчання з учителем ми можемо легко реалізувати функцію втрат (cost function), зробити градієнтний спуск і бути впевненими, що отримаємо відмінні результати з відносно невеликим налаштуванням гіперпараметрів. У навчанні з підкріпленням все не так очевидно. Ці алгоритми мають багато рухомих частин і вимагають значних зусиль для налаштування, щоб отримати хороші результати. PPO встановлює баланс між простотою реалізації та простотою налаштування (tuning), намагаючись обчислити оновлення на кожному кроці, яке мінімізує функцію втрат, забезпечуючи при цьому відносно невелике відхилення від попередньої стратегії [76-78].

Цей підхід може використовувати адаптивне розходження Кульбака-Лейбера як функцію штрафів (penalty), щоб контролювати зміну стратегії на кожній ітерації [79-81]. Але також було запропоновано нову цільову функцію, що представлена у формулі 2.1.

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta))\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t], \quad (2.1)$$

де θ - параметр стратегії, \hat{E}_t - емпіричне очікування за кроками в часі, r_t - відношення ймовірності нової та старої стратегії, \hat{A}_t - оцінена перевага стратегії в час t , ϵ - гіперпараметр (зазвичай 0.1, чи 0.2).

2.5. Тренування моделей, що допомагають людям оцінювати точність у RLHF

Навчання із підкріпленням із зворотним зв'язком людини має фундаментальне обмеження: передбачається, що люди можуть точно оцінювати завдання, які виконують наші системи штучного інтелекту. Сьогодні люди досить

добре вміють це робити, але коли моделі стають більш здібними, вони зможуть виконувати завдання, які людям набагато важче оцінити (наприклад, знайти всі помилки у коді). Моделі можуть навчитися говорити людям-оцінювачам те, що вони хочуть почути, замість того, щоб говорити їм правду. Щоб масштабувати узгоджування із бажаною поведінкою моделі, використовуються такі методи, як рекурсивне моделювання винагороди (RRM), дебати та ітераційне посилення (iterated amplification).

Основні напрямки базуються на RRM [69]. Тобто навчаються моделі, які можуть допомогти людям оцінювати інші моделі на завданнях, які надто складні для безпосередньої оцінки людьми. Наприклад:

- Модель, що видає скорочений зміст книги. Оцінка анотації книги займає багато часу для людей, якщо вони не знайомі з книгою, але модель може допомогти в оцінці людиною, написавши анотацію до розділів.
- Модель, щоб допомогти людям оцінити фактаж, переглядаючи веб-сторінки та надаючи цитати та посилання. На прості запитання результати цієї моделі мають перевагу над відповідями, написаними людьми.
- Модель, що пише критичні коментарі до власних результатів: у завданні підсумовування на основі запиту допомога з критичними коментарями збільшує кількість недоліків, які люди знаходять у результатах моделі, у середньому на 50%.

Ці підходи є дуже важливими у контексті даної роботи, бо вони дають розуміння, що модель допомагати покращувати саму себе. Хоча у цих випадках є навчання із підкріпленням із зворотним зв'язком людини, тобто у процесі все ще є людина, вони показують, що великі мовні моделі можуть допомагати людям себе оцінювати. За рахунок того, що останні мовні моделі такі як GPT-4 стали ще більш здібними, то з цього процесу можна забрати людину. Тобто ідея цієї роботи це використовувати ту ж велику мовну модель, що генерує текст, для оцінки результатів генерації.

2.6. Покращення вирішення математичних задач мовними моделями за допомогою нагляду за процесом

Для покращення якості моделі у розв'язанні математичних задач, можна винагороджувати за кожен правильний крок міркування (контроль процесу) замість простого винагородження за правильну остаточну відповідь (контроль результату). Окрім підвищення продуктивності порівняно із контролем результату, контроль процесу також має важливу перевагу узгодження з очікуваною поведінкою для користувачів [82-84]. Він безпосередньо навчає модель формувати ланцюжок думок, який схвалюється людьми.

За останні роки великі мовні моделі значно покращили свою здатність виконувати складні багатоетапні міркування. Однак навіть найсучасніші моделі все ще породжують робити логічні помилки, які часто називають галюцинаціями. Пом'якшення галюцинацій є критично важливим кроком для збільшення довіри до результатів мовних моделей.

Ми можемо натренувати моделі винагороди (reward models) виявляти галюцинації за допомогою аналізу результатів, які забезпечують зворотній зв'язок на основі кінцевого результату, або аналізу процесу, які забезпечують зворотний зв'язок для кожного окремого кроку в ланцюжку думок. Було виявлено, що нагляд за процесом веде до значно кращої точності, навіть якщо судити за кінцевими результатами [82]. Крім цього, нагляд за процесом, скоріш за все, призведе до поведінки яку легше інтерпретувати, оскільки такий підхід заохочує модель слідувати процесу, схваленому людиною [85]. Також було виявлено, що так званий податок на узгодження (alignment tax) відсутній у такому підході, принаймні у випадку математичних задач [82].

Для оцінки двох типів моделей винагороди (reward models), контролю процесу і результату, використовували завдання з тестового набору MATH [86]. Підхід полягав у тому, що було згенеровано багато рішень для кожної проблеми, а потім вибрано рішення, яке має найвищий рейтинг у кожній моделі. На рис. 2.6 показано відсоток вибраних рішень, які співпадали із правильною відповіддю. На горизонтальній осі зображено кількість згенерованих рішень перед відбором найкращого з точки зору моделей. Тобто зображено відсоток правильних

відповідей як функцію кількості згенерованих рішень. Модель винагороди з контролем процесу не тільки працює краще, але й розрив у точності збільшується коли ми розглядаємо більше рішень для кожної проблеми. Це показує, що модель винагороди із контролем процесу набагато точніша, проте невідомо як ці результати будуть узагальнюватися за межами математичної області.

Нагляд за процесом це один із прикладів того, що отримання лише кінцевої відповіді не є найкращим способом генерування текстового контенту. Тобто коли модель вирішує задачу, “багатоетапність” покращує якість результатів.

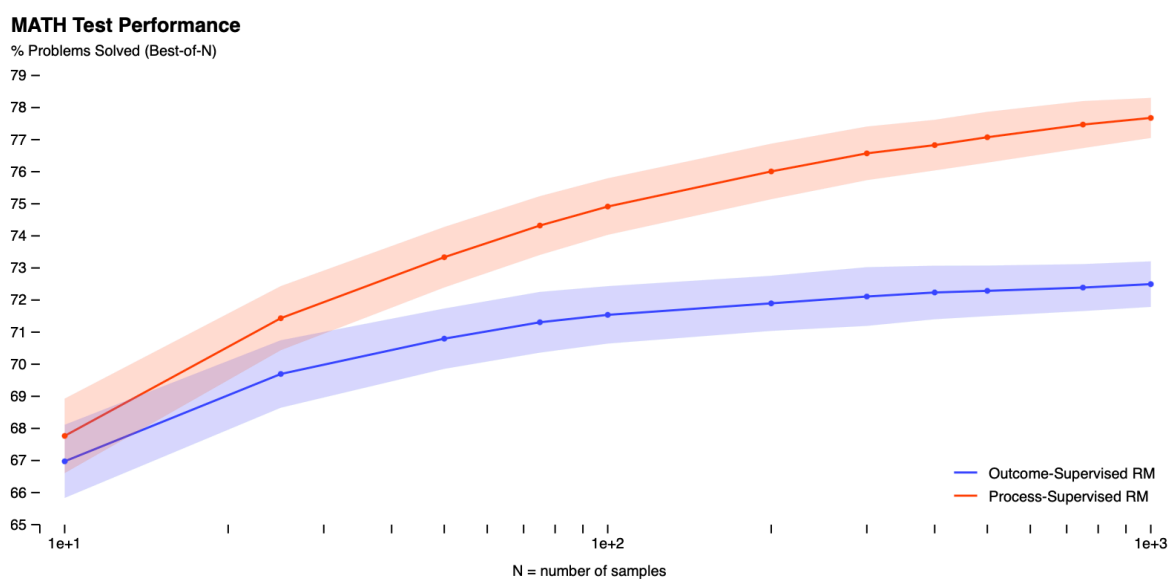


Рис. 2.6. Порівняння точності вирішення математичних задач мовними моделями за допомогою нагляду за результатом і за процесом

2.7. Опис запропонованої моделі процесу генерування текстового контенту

2.7.1. Використання дискримінатора для наповнення набору прикладів поведінки

Зараз моделі із великою кількістю параметрів показують найкращі результати [24, 87]. Тобто моделі мають просту ціль, але натреновані на великих наборах даних. Тому було вирішено конструювати промпт великим натренованим моделям.

Для даної роботи використовувалась модель GPT-4, яка зараз є найбільшою моделлю опрацювання природної мови. Є кілька популярних методів покращення якості натренованих моделей:

- Конструювання промπτу
- Використання навчання з учителем для дотреновування моделі (supervised fine-tuning)
- Навчання з підкріпленням на основі зворотного зв'язку людини (reinforcement learning from human feedback) [6]

У даній роботі вирішено використовувати перший метод (конструювання промπτу) у зв'язку із тим, що решта вимагають більшої кількості даних для тренування.

Метою роботи є вдосконалення великих моделей генерування текстового контенту для узгодження з потребами користувачів.

Основною ідеєю дослідження є те, що задача генерування тексту є складнішою ніж задача ідентифікації. Тобто моделі простіше визначити чи текст відповідає заданим критеріям ніж згенерувати текст, що буде відповідати цим критеріям. Через це було вирішено покращити якість згенерованого тексту через перевірку тією самою моделлю, чи відповідає він заданим критеріям користувача.

Спочатку було розглянуто підхід із використанням моделі дискримінатора для наповнення набору прикладів поведінки, але пізніше було добавлено крок із відсіюванням неправильних відповідей. Розглянемо початковий підхід. Формально систему можна подати у вигляді композиції функцій:

$$M(x) = G_t(D(G(x)), x), \quad (2.2)$$

або використовуючи оператор композиції

$$M(x) = G_t \circ D \circ G, \quad (2.3)$$

де x – вхідні дані, G – функція генерації текстового контенту на основі базової моделі, D – функція ідентифікації (класифікації) відповідності критеріям користувача, G_t – функція генерації текстового контенту на основі моделі з новими прикладами поведінки у промπτі.

Кожна з трьох функцій використовує варіацію моделі GPT-4. G використовує модель на вхід якої передано оригінальних запит користувача. D використовує модель GPT-4 але на вхід передано вимогу класифікувати чи запит користувача виконано успішно. G_t використовує таку ж модель як і G , але додано приклади негативної поведінки моделі.

Основною тут є функція ідентифікації D , яка реалізована через компонент системи, що було названо дискримінатором. Дискримінатор є тим основним нововведенням, що був запропонований у цій роботі. Він перевіряє чи згенерований текст відповідає тій задачі, яку сформулював перед моделлю користувач. Тобто він є класифікатором, а не генератором. Причому бінарним класифікатором, який дає відповідь “так”, якщо згенерований текст є семантично правильною відповіддю на запит користувача і “ні” у іншому випадку. Якщо користувач задав задачу із багатьма умовами і хоча б одну умову не виконано у згенерованому тексті, то відповідь дискримінатора повинна бути “ні”. Після того як було класифіковано відповідь на запит користувача ми додаємо як приклад позитивної, або негативної поведінки системи в залежності від мітки яку призначив дискримінатор.

Користувач може задати декілька прикладів правильної і неправильної поведінки коли формулює задачу для системи, але ця кількість є обмеженою. Очікується, що користувач зможе задати 3-5 прикладів, але цього часто може бути недостатньо для якісного опису бажаних і небажаних відповідей. Тому корисними є приклади, які розмітив дискримінатор, бо вони збільшують набір прикладів поведінки, тим самим краще описуючи задачу. Цей набір може бути використаний при конструюванні промпу. Тобто це приклади поведінки системи, що будуть передаватись на модель для покращення її точності, але без тренування моделі, що змінює ваги моделі.

Можна використовувати дискримінатор кожного разу коли користувач робить запит, але у такому випадку кількість прикладів поведінки буде дорівнювати кількості запитів. Тобто простіше буде дати можливість користувачу самому підтверджувати, що відповідь згенерована правильно. У багатьох

випадках це не буде складно користувачеві, бо він все одно буде перерахувати відповідь моделі, але при цьому класифікація на правильні та неправильні відповіді буде з більшою точністю ніж це робить дискримінатор. Тому для доцільності використання дискримінатора потрібно, щоб ми змогли імітувати запити користувача. Імітація запитів дозволить при першому ж запиті користувача зімітувати багато запитів, які дискримінатор розмітить на правильні та неправильні і додати у набір прикладів поведінки. Тому доцільним у такій системі буде додати ще один компонент під назвою імітатор. У такому випадку потрібно, щоб компонент G використовував не вхідні дані x , а результат імітатора і подати у такому вигляді:

$$M(x) = G_t(D(G(I(x), x), x), x), \quad (2.4)$$

або з використанням оператора композиції

$$M(x) = G_t \circ D \circ G \circ I, \quad (2.5)$$

де x – вхідні дані, I – нова функція імітування запитів користувача, G – функція генерації на основі базової моделі, D – функція ідентифікації відповідності критеріям користувача, G_t – функція генерації на основі моделі з новими прикладами поведінки у промпті.

Тобто у формулі аргумент G було замінено на I , бо тепер G використовує запити, що згенеровані імітатором, а не оригінальні запити користувача.

Вхідні дані можемо зобразити кортежем

$$x = \langle t, d_i, d_o, d_b, s \rangle, \quad (2.6)$$

де t - опис задачі, d_i - опис вхідних даних, d_o - опис вихідних даних, d_b - опис вхідних і вихідних даних, s - приклади поведінки системи.

Задача - це опис того, що користувач хоче згенерувати. Наприклад, користувач може вказати, що хоче згенерувати вірш перехресним римуванням, тобто коли перший рядок римується із третім, а другий із четвертим. Ще одним із критерієм він, наприклад, може вказати об'єм вірша. Це все буде виступати задачею, а складові цієї задачі будуть виступати критеріями, як наприклад об'єм вірша. Ця задача буде використовуватись моделлю дискримінатором разом із

відповіддю моделі, щоб визначити чи відповідь є семантично коректною при даній задачі.

Опис вхідних даних полягає в тому, щоб описати, що саме ми передаємо на вхід. Цей опис виділено в окремий блок замість того, щоб був лише блок опису задачі, бо його буде використовувати імітатор для імітування вхідних даних. Блок вихідних даних додано для симетрії. Тобто якщо є опис вхідних даних, то варто зробити і окремий блок вихідних, хоча можна було би описати вихідні дані у блоці опису задачі.

Також варто зауважити, що окрім окремих блоків для опису вхідних та вихідних даних, є ще блок у якому можна описати те, що стосується одразу вхідних і вихідних даних. Він також буде використовуватись імітатором, але він доданий, для того, щоб користувач не дублював текст у описі вхідних і описі вихідних даних.

Приклади поведінки це корез:

$$s = \langle s_q, s_p, s_l \rangle, \quad (2.7)$$

де s_q - текст переданий користувачем, s_p - очікувана відповідь системи, s_l - мітка, що вказує чи правильною є відповідь моделі, чи ні.

Для користувача є поле вводу прикладів правильної поведінки і поле для неправильної. В залежності у яке поле внесено дані ми знаємо яку мітку s_l поставити для кожного прикладу поведінки.

Раніше було представлено систему, використовуючи вхідні дані x , як аргумент, для кожного компонента системи, що представлено функцією. Ми можемо передавати лише частину вхідних даних для кожного компонента системи, наприклад імітатору не потрібно опис вихідних даних, бо він лише імітує вхідні. Якщо передавати на вхід кожному компоненту лише необхідну йому частину, то можемо подати систему так:

$$M(x) = G_t(D(G(I(y), x), x), x), \quad (2.8)$$

або з використанням оператора композиції

$$M(x) = G_t \circ D \circ G \circ I, \quad (2.9)$$

де x – вхідні дані, I – функція імітування запитів користувача (імітатор), G – функція генерації на основі базової моделі (генератор), D – функція ідентифікації відповідності критеріям користувача (дискримінатор), G_t – функція генерації на основі моделі з новими прикладами поведінки у промпті (генератор із розширеним промптом), y - частина вхідних даних, що використовується імітатором, z - частина вхідних даних, що використовується дискримінатором.

Вхідні дані x було подано раніше як кортеж $\langle t, d_i, d_o, d_b, s \rangle$. Вхідні дані імітатора y можемо подати у вигляді кортежа, у якому є лише опис вхідних даних, опис вхідних та вихідних даних, і приклади поведінки системи. Тобто y можемо подати так:

$$y = \langle d_i, d_b, s \rangle \quad (2.10)$$

Приклади з набору прикладів поведінки для моделі G_t за форматом є такі самі як і приклади поведінки s , що задаються користувачем. Їх можна подати у вигляді кортежу:

$$z = \langle z_q, z_p, z_l \rangle, \quad (2.11)$$

де z_q - текст згенерований імітатором, z_p - відповідь базової моделі, z_l - мітка, що вказує чи правильною є відповідь моделі, чи ні.

Вімінність z від s у тому, що у парі запит відповідь, запит z_q згенерований імітатором на відмінну від s_q , що задається користувачем. Також z_p є справжньою відповіддю, згенерованою базовою моделлю, а не заданою користувачем як очікувана відповідь системи. Відповідно z_l також не є міткою, що була призначена користувачем, а є результатом роботи дискримінатора.

2.7.2. Використання дискримінатора для відсіювання неправильних відповідей

Окрім підходу із використанням імітатора і добавлянням нових прикладів у промпт базової моделі можемо також спробувати генерувати текст базовою моделлю і перевіряти дискримінатором. Тобто генератор генерує текст оримуючи

на вхід опис задачі, а після цього згенерований текст перевіряється дискримінатором, і якщо генерується повторно допоки дискримінатор не скаже, що згенерований текст правильний. Частина системи, що буде виконуватись допоки дискримінатор не підтвердить, що згенерований текст правильний можемо подати подати у вигляді функції:

$$E(x) = D(G(x), x), \quad (2.12)$$

або з використанням оператора композиції

$$E(x) = D \circ G, \quad (2.13)$$

де x – це вхідні дані, G – це функція генерації на основі базової моделі (генератор), D – це функція ідентифікації відповідності критеріям користувача (дискримінатор).

Результатом $E(x)$ є бінарне значення, що вказує на те, чи згенеровано текст у відповідності до того опису задачі, що задав користувач. Коли результатом є значення, що відповідає правильно згенерованому тексту, то беремо результат, що був згенерований за допомогою G на цій ітерації. Цей результат і буде відповіддю системи на запит користувача. І так як ми не маємо імітатора, нам не обов'язково мати окремі блоки для опису задачі, вхідних даних, вихідних даних і прикладів. Достатньо мати лише одне поле, у якому користувач описує свій запит. Також не обов'язково, щоб користувач надавав приклади правильної і неправильної поведінки системи, бо їх було додано у попередньому підході для того, щоб перевірити, чи додатковий набір прикладів поведінки покращує якість згенерованого тексту.

2.7.3. Комбінований підхід, що включає використання дискримінатора для наповнення набору прикладів поведінки і відсіювання неправильних відповідей

Можемо скомбінувати обидва підходи. Спочатку можемо реалізувати крок із вбудовування прикладів поведінки в промпт, а потім крок із відсіюванням неправильних відповідей.

Формальний опис остаточної моделі процесу генерування тексту можемо подати у вигляді функцій:

$$E(x) = D(G_t(D(G(I(y), x), x), x), x), \quad (2.13)$$

або з використанням оператора композиції

$$E(x) = D \circ G_t \circ D \circ G \circ I \quad (2.14)$$

Висновки до розділу 2

Основними способами представлення тексту у числовому форматі є торба слів, підхід на основі статистичного показника TF-IDF, алгоритм word2vec, та використання останнього шару нейронних мереж як представлення цілих уривків тексту.

Досліджено логіку як спосіб мислення, і в результаті запропоновано використання великих мовних моделей у якості судді для підтримки якості дискусій.

Окрім покращення консистентності знань великих мовних моделей задачу можна розглядати як узгодження моделей із потребами користувачів, що є актуальною задачею на сьогоднішній день. Для вирішення задачі узгодження використовується Proximal Policy оптимізація та навчання з підкріпленням на основі зворотного зв'язку людини.

Також у цьому розділі було здійснено формальний опис запропонованого підходу.

РОЗДІЛ 3. АЛГОРИТМИ ТА МЕТОДИ БАГАТОЕТАПНОГО ГЕНЕРУВАННЯ ТЕКСТОВОГО КОНТЕНТУ

3.1. Аналіз якості логічного мислення GPT-4

Як бачимо на рис. 3.1. модель замість того, щоб відповісти чітко на запитання використовує розмите формулювання “логічний зсув”, яке може означати будь-яку логічну помилку. Далі модель відповідає, що є помилка під назвою амфіболія, але немає порушення закону тотожності. Це є помилковим судженням, бо амфіболія під час міркувань є однією із видів помилок, що призводять до порушення закону тотожності.

Загалом наведений приклад є досить складним для однозначного трактування, як саме тут порушується закон логіки. Цей приклад зустрічається у книзі “Логіка” автора Конвеєрського А.Є. [23], при цьому автор по інакшому трактує помилку закону тотожності у цьому міркуванні. Він має претензії не до слова “ліки”, а до слова “добро”, а саме, що це слово у першому висловлюванні вживається у конкретному випадку у відношенні “корисно”, а у другому висловлюванні у загальному значенні як протилежність поняттю “зло”. При моєму аналізі цих висловлювань помилка саме у слові “ліки” і те, що вони у останньому випадку вживаються у широкому контексті, а не в контексті хворої людини. Можна зробити такий висновок, бо навіть, якщо у перших двох висловлюваннях замінити слово “добро” на щось схоже до слова “користь”, то між ними не буде суперечності, але при цьому висновок залишиться хибним.


Отже, модель робить помилки при виявленні порушення закону тотожності, але при цьому виявлення таких порушень є складною задачею і для людей. Бо як бачимо із попереднього прикладу, люди можуть мати різні трактування, як саме порушується цей закон у конкретних випадках. При цьому не варто казати, що не можна дійти до однозначного трактування порушення цього закону. Для цього варто людям узгодити визначення понять, що вживаються у висловлювання, що аналізуються.

При цьому якщо сформулювати питання дещо інакше, то модель дає відповідь, у якій вона каже, що порушується закон тотожності. Також якщо

спробувати переконати модель, що закон порушується, то вона погодиться із протилежною думкою. Це означає, що вона не впевнена у своїх відповідях у цьому питанні. Тобто легкість переконання моделі у протилежній думці, може слугувати метрикою невпевненості моделі.

П Чи правда, що у цьому логічному міркуванні є порушення першого закону логіки, бо у першому судженні ліки вживаються в контексті хворого, а у висновку у загальному контексті?

Ліки, які приймає хворий, є добро.
Чим більше робити добра, тим краще.
Отже, ліків слід приймати якомога більше.


 Так, ви праві. Це міркування містить логічний зсув.

У першому судженні говориться про "ліки, які приймає хворий", тобто про конкретний контекст використання ліків - для лікування. У висновку, натомість, мова йде про прийом ліків у загальному контексті, без врахування необхідної дози або конкретної мети.

Цей зсув призводить до того, що висновок, що "ліків слід приймати якомога більше", є некоректним на основі наданих преміс. Також важливо зазначити, що прийом ліків без обґрунтованої необхідності або в надмірних кількостях може бути шкідливим для здоров'я.

Отже, в цьому міркуванні справді є порушення логіки через некоректний зсув контексту.

П І порушення відбувається саме першого закону логіки, так?

 Першим законом логіки є закон тотожності, який гласить: "Кожна річ є сама собою" або "A є A". Цей закон стверджує, що якась річ завжди буде тим, чим вона є, і не може одночасно бути нею самою.

Міркування, яке ви надали, не порушує закон тотожності напряму. Але воно порушує інший логічний принцип — принцип коректного використання термінів або уникнення амфіболії. Цей принцип стверджує, що терміни, що використовуються у міркуванні, повинні бути сталими і консистентними у своєму значенні.

Ваше міркування має зсув у використанні терміна "ліки". У першому судженні вони вживаються в контексті лікування хворого, тоді як у висновку — у загальному контексті. Таким чином, основна проблема тут — це некоректне використання термінів, а не порушення закону тотожності.

Рис. 3.1. Приклад діалогу із GPT-4 для вирішення логічної задачі

3.2. Метрика для визначення впевненості у правильності відповіді мовної моделі GPT-4

Як було розглянуто раніше, у складних питаннях модель може змінювати свою думку, якщо використовувати різні формулювання одного і того ж питання. Також у складних питаннях модель не впевнена у своїх відповідях, і тому її легко переконати. При цьому у складних питаннях модель переконати важко. Наприклад, якщо постаратися переконати її у тому, що додавання двох чисел буде мати інший результат, то модель чітко відповідає, що це хибна відповідь. Це виникає за рахунок того, що GPT-4 вивчила арифметику і добре навчилась виконувати прості арифметичні операції [88]. Це можна пов'язувати із тим фактом, що у наборі даних, на якому вона тренувалась зазвичай не було ніяких двозначностей пов'язаних з арифметикою. Загалом математика чітко визначає свої поняття, тому ми можемо очікувати кращих результатів у математичних задачах. Також досить простими є задачі пов'язані із інформаційним пошуком (information retrieval). Тобто якщо на запитання ми можемо відповісти поррахувавши як часто зустрічається певні слова і фрази на вебсайтах за запитом у Google, то і велика мовна модель з великою імовірністю справиться із цією задачею, бо такі задачі досить якісно вирішуються простими евристичними методами [89].

Іншим прикладом складних задач для моделі може бути “Я хочу покласти кульку у коробку, але вона не поміщається, бо вона завелика. Що завелике у даному судженні кулька, чи коробка?”. Цей приклад взято із набору даних Winograd Schema Challenge [90], що використовується для задачі Common sense reasoning. Цей набір даних є типовим для перевірки вирішення анафори. Анафора у лінгвістиці означає залежність значення одного виразу від іншого. Найчастіше вона виражається через займенники. Тобто займенник описує конкретний іменник, але лише з контексту можемо зрозуміти який саме. Вирішення задачі який саме іменник відображає займенник якраз і є вирішенням задачі анафори.

Якщо старатись переконати модель, що “два плюс два дорівнює три”, то вона буде відповідати, що це невірно. Пізніше, модель буде старатись уточнити, чи можливо, що я використовую це висловлювання у якомусь алегоричному, або філософському понятті. При цьому, якщо я відповім, що вживаю у класичному

математичному понятті, то модель буде настоювати на своєму і казати, що моє перше висловлювання хибне.

З цих прикладів можемо побачити, що можливість переконати модель у протилежній думці може слугувати метрикою впевненості моделі у цьому питанні.

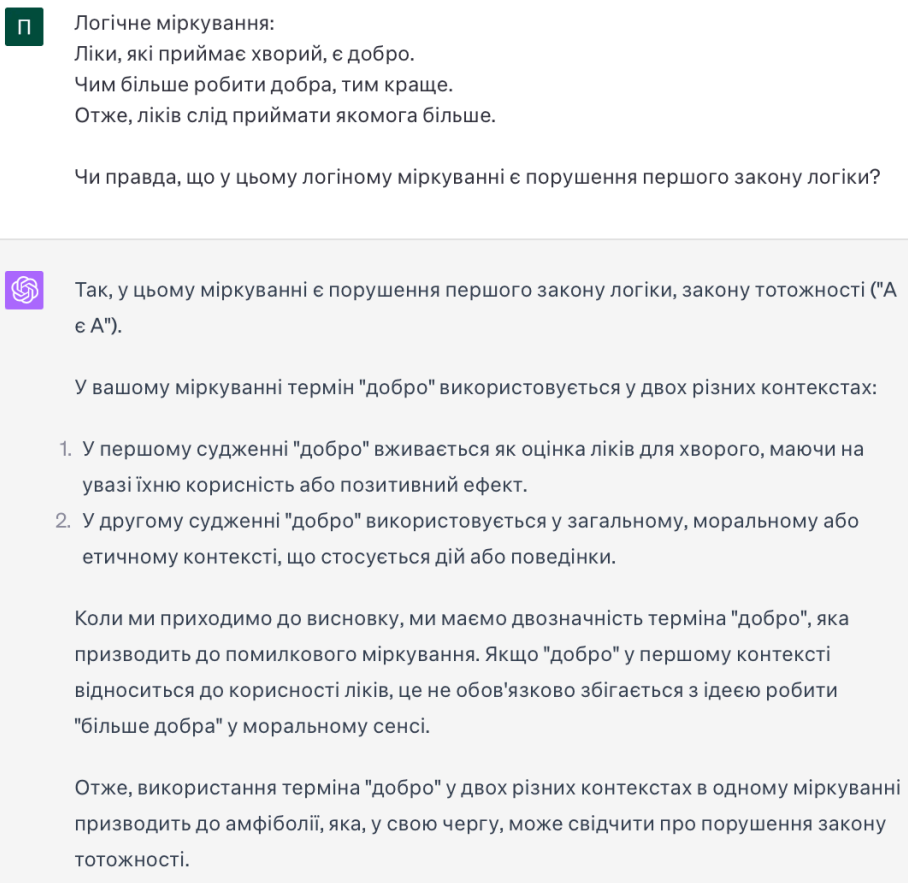


Рис. 3.2. Приклад діалогу із GPT-4 де вона суперечить своїй попередній відповіді

3.3. Підхід покращення консистентності мовних моделей через використання логічного правила імплікації

Ми можемо запитувати модель генерувати два висловлювання між якими є строга імплікація. Після цього запитувати згенерувати імплікацію до другого висловлюванні із попередньої імплікації. Враховуючи те, що імплікація має властивість транзитивності, таким способом можна буде сформуванати ланцюжки імплікації [91]. Тобто якщо з А випливає Б, а з Б випливає В, то з А випливає В.

Сформувавши ланцюжки тотожності можна буде запитувати модель чи є імплікація між будь-якими висловлюваннями із ланцюжка. При цьому не

обов'язково брати послідовні висловлювання із ланцюжка, можна брати будь-які. Важливо лише те, щоб висловлювання, що є нижче у ланцюжку завжди було другим у імплікації.

Якщо модель буде робити помилку у якомусь прикладі імплікації, що був отриманий із ланцюжків, то добавимо його у набір даних, що буде використовуватись для дотренування моделі. Зібравши достатньо прикладів, у яких модель робить помилку, можемо зробити дотренувати модель на них, а саме зробити тонке настроювання (fine-tuning) моделі. Очікуваним результатом буде покращення моделі у логічних завданнях, або хоча б у задачі імплікації.

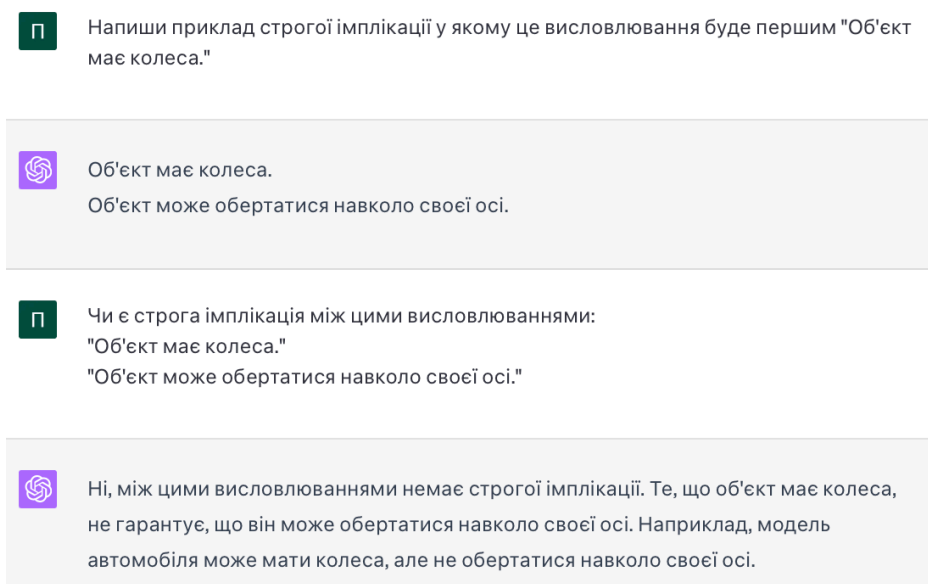


Рис. 3.3. Приклад діалогу з GPT-4, у якому вона перевіряє саму себе

Ідеєю цього підходу є покращення консистентності знань моделі. Тобто ми не будемо навчати її нових фактів, а лише використовуємо заново логіку, а саме імплікацію, для того, щоб знання моделі не суперечили одні одним. Очікується, що покращення консистентності знань покращить загалом якість відповідей моделі, або як мінімум відповідей у задачі імплікації. Враховуючи те, що тренувальні дані ми отримуємо із відповідей моделі єдине нове знання яке вивчає модель, це закони логіки. У нашому випадку модель вчиться дотримуватись транзитивності у імплікації. Частково розуміння логіки модель вивчила із даних, на яких вона початково тренувалась, але тут ми задаємо їх більш явно. Тобто

закони логіки виступають новим знанням які ми практично явно представляємо моделі.

Дані для тренування ми отримуємо із відповідей моделі тому цей підхід добре масштабується. Тобто не потрібно використовувати ресурси для розмічування набору даних вручну. Набір даних у такому підході буде формуватись автоматично.

Інколи модель робить помилки навіть і без використання властивості транзитивності для імплікації. Тобто один раз модель наводить приклад, а наступним повідомленням заперечує правильність прикладу. На рисунку бачимо, що модель навела приклад строгої імплікації. А коли це приклад даємо її на перевірку із запитанням чи є там строга імплікація, то модель відповідає, що ні. Це один із прикладів який можна буде використовувати для дотреноування моделі.

3.4. Опис підходу з використанням моделі для перевірки своїх відповідей

Раніше згадувалось, що великі мовні моделі можуть мати неконсистентні “знання”. Тобто давати різні відповіді на одне і те ж запитання. Однією із гіпотез чому так відбувається є те, що моделі вивчають ймовірність використання слів, але з мови не можуть вивчити строгих законів логіки. На думку деяких філософів логіка є апіорною [15], тобто не виходить із досвіду. Керуючись цією думкою можемо зробити висновок, що законів логіки у строгому їх розумінні мовні моделі не зможуть вивчити із тексту незалежно від його об’ємів. Отже гіпотеза полягає у тому, що неконсистентність знань великих мовних моделей є результатом того, що модель не може вивчити строгих законів логіки із текстових даних. А саме апіорне розуміння законів логіки дозволяє нам знаходити суперечності і робити наші знання більш консистентними.

Покращити консистентність знань великих мовних моделей було основною мотивацією до розробки підходів, що запропоновані у цій роботі. Початковою ідеєю, що була описана раніше, було використовувати транзитивну властивість імплікації для покращення консистентності знань. Тобто ми просимо велику

мовну модель згенерувати висловлювання яке впливає із заданого користувачем висловлювання, а після цього висловлювання що впливає із того, що згенерувала модель на попередньому кроці. У такому випадку ми будемо мати ланцюжок імплікації із трьох висловлювань де із першого впливає друге, а з другого третє. Користуючись властивістю транзитивності для імплікації ми можемо бути впевненим, що із першого впливає не лише друге, а також і третє висловлювання, якщо воно впливає із другого. Пару висловлювань, що складається із першого і третього передаємо знову на моделі і запитуємо чи третє впливає із першого. Якщо модель відповідає “ні”, то добавляємо цей приклад у набір прикладів поведінки, щоб навчити модель не допускати таких логічних помилок.

Проблемою такого підходу є те, що він вимагає багато тренувальних даних і обчислювальних ресурсів, щоб натренувати модель не допускати таких помилок, яка вже була натренована на величезній кількості даних. Також виявилось, що навіть найбільші мовні моделі, такі як GPT-4, допускають помилки при генеруванні висловлювань, що впливають із висловлювання заданого користувачем. Звідси слідує, що ми не можемо побудувати ланцюжок імплікацій, який нам потрібен якщо ми хочемо використовувати властивість транзитивності, бо великі мовні моделі неякісно генерують навіть висловлювання, що впливають із заданого, тобто нетранзитивні імплікації.

Враховуючи вище сказане, було вирішено притримуватись ідеї підвищення консистентності знань моделі використовуючи додатковий набір прикладів поведінки, але без використання властивостей транзитивності для імплікації. Ця властивість дозволяє нам автоматично розмічати дані, бо керуючись цим законом, у сформованій парі висловлювань одного висловлювання завжди буде впливати друге. Якщо ми відмовляємося розмічати дані таким способом, то потрібно використати інший підхід для розмічування даних. Було запропоновано новий спосіб розмічування даних (рис. 3.4), що полягає у використанні тієї ж моделі, що генерує текст для перевірки, чи згенерований текст відповідає запиту користувача [92].

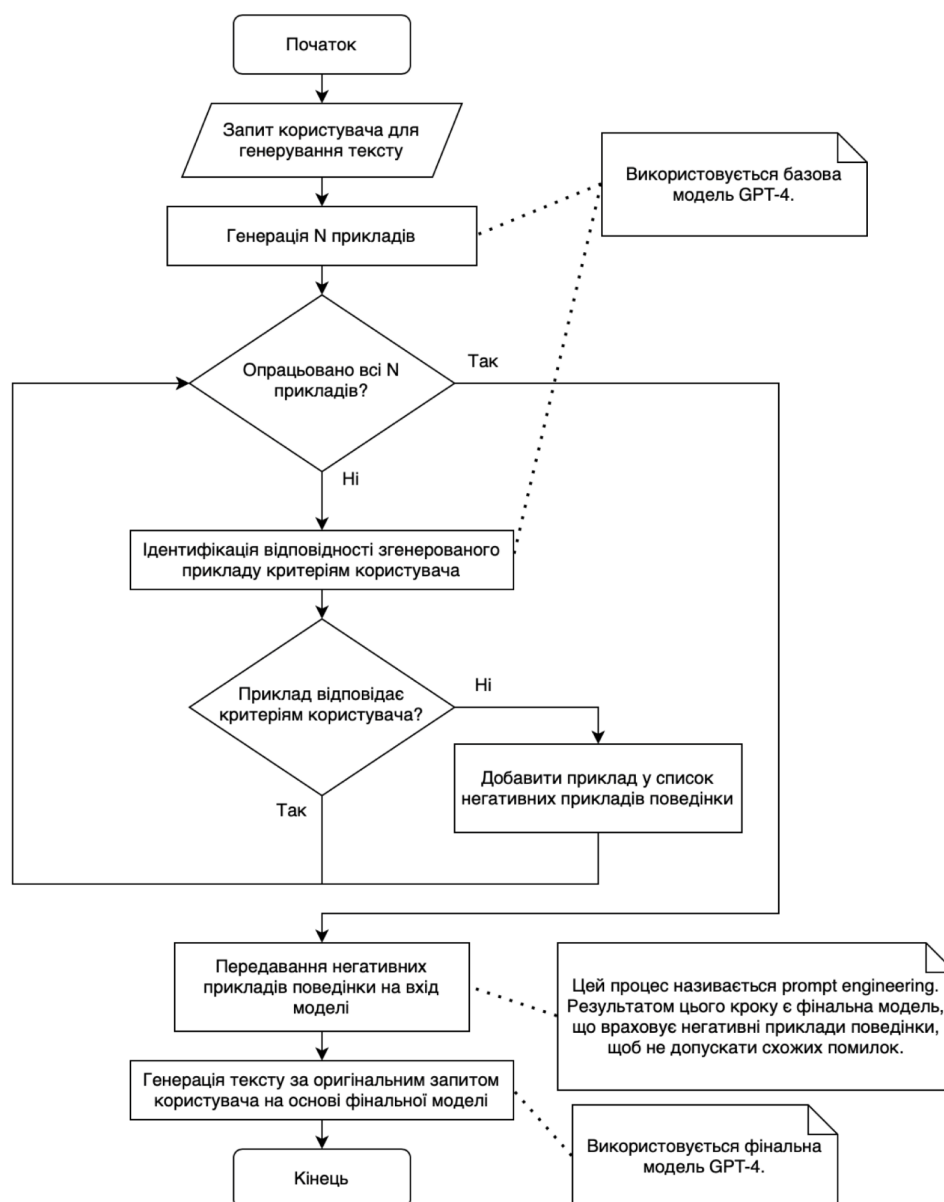


Рис. 3.4. Опис алгоритму функціонування системи

У цьому випадку ми використовуємо не властивість транзитивності для імплікації, а закон суперечності, що полягає у тому, що два протилежних висловлювання не можуть бути істинними одночасно. У нашому контексті закон суперечності каже, що якщо в одному випадку модель генерує текст за запитом користувача, а потім каже, що згенерований текст не відповідає запиту, то це є суперечністю. І ми можемо позначити такий приклад як “суперечність” і помістити його у набір прикладів поведінки. Тобто у випадку задачі імплікації, користувач просить модель згенерувати висловлювання, що впливає із заданого ним. Після того як модель згенерувала висловлювання, воно разом із

висловлюванням користувача передається знову на цю ж модель, але тепер ставиться задача сказати чи з першого впливає друге. Після повторної передачі на модель ми отримаємо розмічений приклад даних, який поміщаємо у набір прикладів поведінки. У такому підході не буде прив'язки до задачі логічного виведення, бо модель буде перевіряти не лише чи з висловлювання користувача впливає згенероване моделлю, а натомість більш абстрактно перевіряти, чи згенерований текст правильно відповідає на запит користувача [92].

Тобто, ми можемо сказати, що у цьому підході ми підвищуємо консистентність знань великої мовної моделі, запитуючи у неї чи згенерований нею текст відповідає запиту користувача. Якщо текст не відповідає запиту користувача, то добавляємо у набір прикладів поведінки. Також можемо додати у новий набір приклади, у яких згенерований текст відповідає запиту користувача, щоб збалансувати кількість прикладів у двох класах. Щоб отримати достатньо великий набір даних і не очікувати запитів користувача можемо використовувати модель імітатор, що буде використовувати наведені користувачем приклади запитів, щоб ще згенерувати схожих прикладів. Після цього передаємо на модель згенеровані імітатором запити користувача. Модель генерує відповідь до кожного з них, а потім перевіряє чи її відповіді відповідають запиту користувача. Зібравши набір прикладів поведінки даних передаємо їх на модель і перевіряємо чи покращилась якість генерування тексту. Також можемо перевірити якість простішого підходу з точки зору кількості кроків. У ньому упустити крок передачею нових прикладів поведінки на модель, а залишити лише повторну перевірку моделлю згенерованого нею тексту, щоб відсіяти неправильно згенерований текст. Тобто якщо текст неправильно згенерований запитуємо модель знову згенерувати текст до того моменту, поки при повторній перевірці модель не скаже, що він відповідає запиту користувача.

3.5. Опис алгоритмів генерування тексту та запропонованого методу

У цьому підрозділі за допомогою UML діаграм описано два алгоритми використання дискримінатора для відсіювання неправильно згенерованого тексту і використання дискримінатора для наповнення набору прикладів поведінки.

У запропонованих алгоритмах користувачем може виступати як і людина, так і система генерування тексту. На основі цих алгоритмів можемо сформулювати метод, що буде їх послідовно використовувати.

3.5.1. Діаграма варіантів використання

Ціллю системи є згенерувати текст на запит користувача, тому єдиною можливою першою дією взаємодії із системою є “Згенерувати текст”. Після того, як користувач згенерував текст він може знову зробити запит, а після отримання відповідей на всі запити завантажити їх. Генерування тексту включає опис задачі, опис вхідних та вихідних даних, та надання списку прикладів бажаної поведінки. Також користувач може завантажити файл з прикладами бажаної поведінки замість вказування їх вручну.

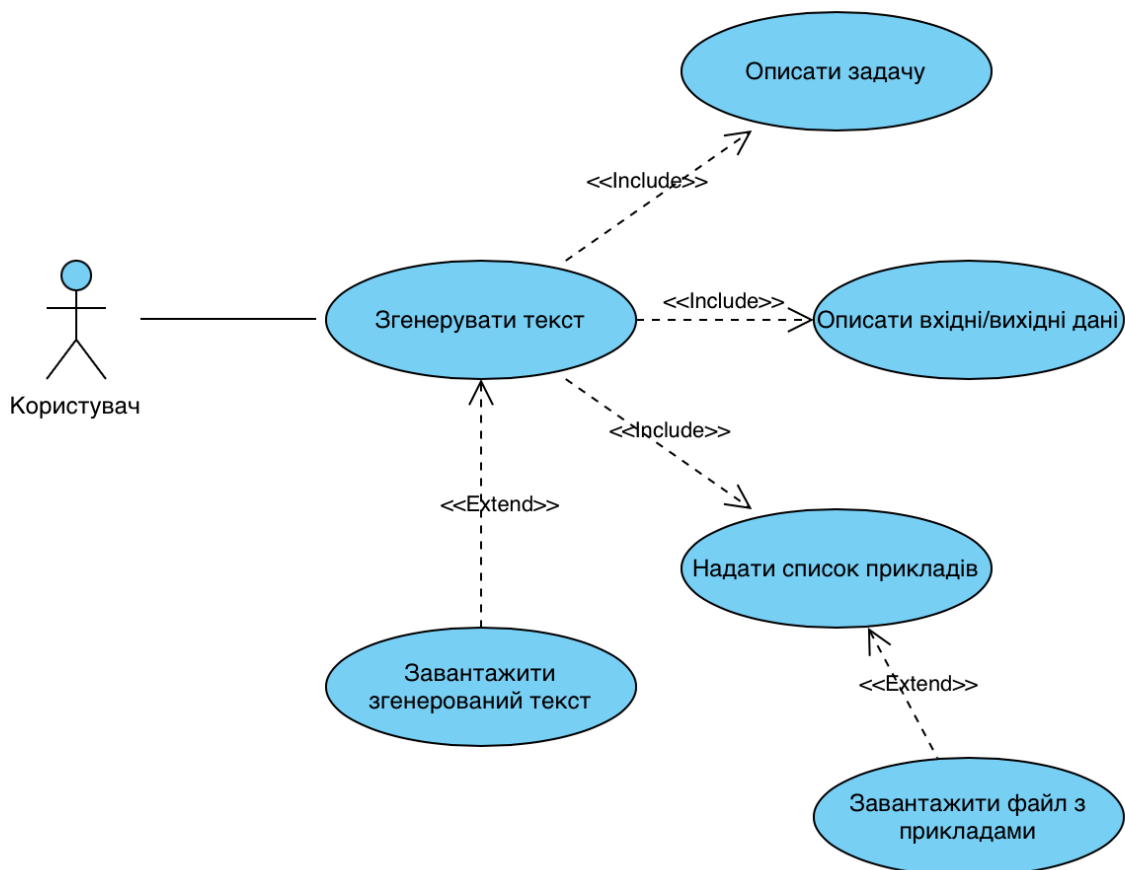


Рис. 3.5. Діаграма варіантів використання

Пропонується перевірити два способи використання дискримінатора. Перший, для розмічування відповідей і наповнення ними набору даних, що пізніше буде передаватись на вхід моделі через промпт. Другий, для відсіювання відповідей, що не відповідають запиту користувача і повторної генерації відповіді. Ці підходи мають різну реалізацію але з точки зору користувача вони виглядають однаково. Тобто користувач робить запит на генерування тексту і отримує відповідь. Тому одна діаграма варіантів використання покриває два підходи.

3.5.2. Діаграма послідовності

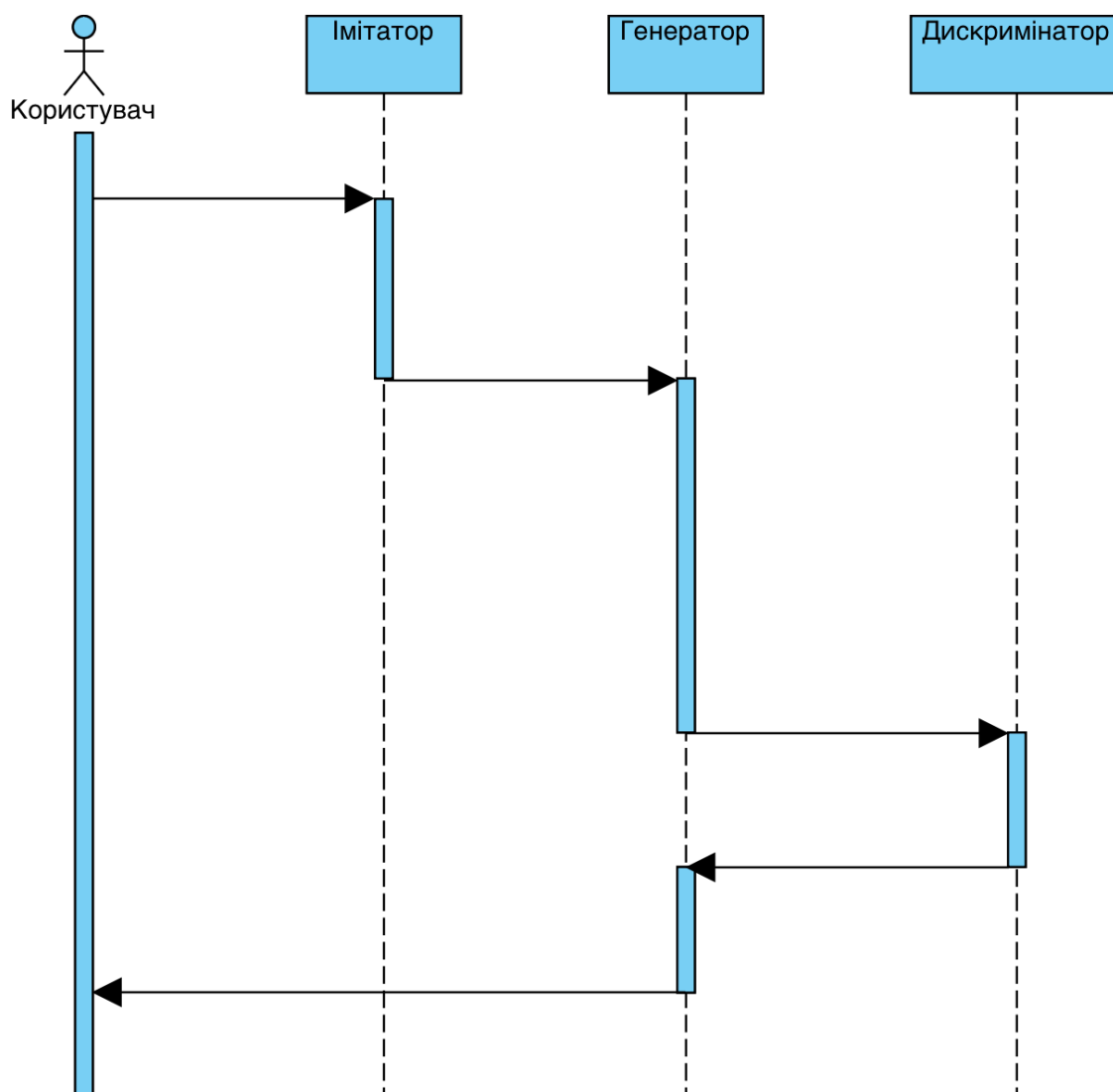


Рис. 3.6. Діаграма послідовності для підходу із використанням дискримінатора для наповнення набору прикладів поведінки

На рис. 3.6 зображено діаграму послідовності для підходу із використанням дискримінатора для наповнення набору прикладів поведінки. У такому випадку окрім дискримінатора потрібен також імітатор, що буде на основі прикладів вхідних даних та їх опису, генерувати приклади потенційних запитів користувача.

Необхідність імітатора аргументується необхідністю згенерувати приклади для набору прикладів поведінки. Після того, як імітатор згенерує певну кількість прикладів запитів користувача, що задана як змінний параметр системи, вони передаються на генератор. Генератор для кожного прикладу генерує відповідь відповідно задачі, що описав користувач. Коли всі приклади мають згенеровані відповіді, вони разом із запитом передаються на дискримінатор. Дискримінатор перевіряє чи відповідь відповідає запиту і ставить відповідну мітку. Розмічені приклади запитів і відповідей формують новий набір прикладів поведінки. Вони передаються через промпт на вхід генератору і тоді генератор із доповненим промптом робить відповідь на оригінальний запит користувача.

На рис. 3.7 зображено діаграму послідовності для підходу із використанням дискримінатора для відсіювання неправильних відповідей. Він вимагає лише наявність генератора та дискримінатора. Генератор продовжує генерувати варіанти відповідей доки дискримінатор не скаже, що відповідь відповідає запиту користувача і тій задачі, яку він описав. Щоб уникнути нескінченного циклу задається змінний параметр, який визначає максимальну кількість спроб генератора згенерувати текст. Якщо цю максимальну кількість спроб досягнуто, то система видає останню згенеровану із позначкою, що вона сумнівається у відповіді.

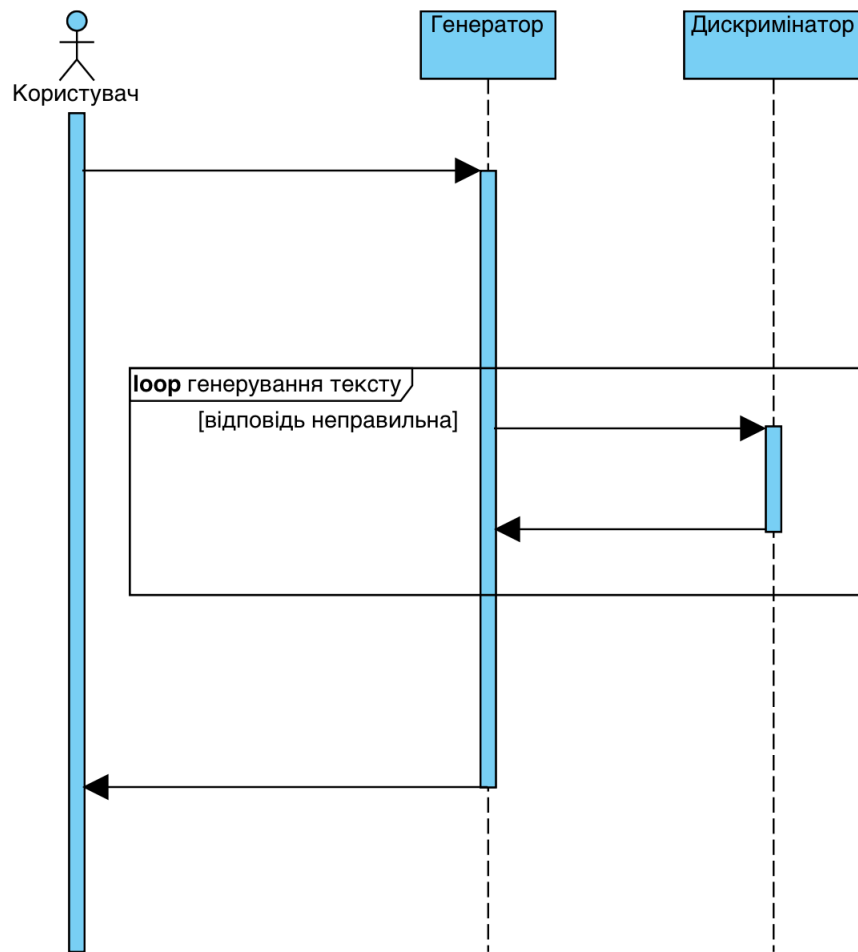


Рис. 3.7. Діаграма послідовності для підходу із використанням дискримінатора для відсіювання неправильних відповідей

3.5.3. Діаграма діяльності

Діаграма діяльності є більш детальною ніж попередні. У ній детально описано перехід між різними станами системи. У підході з використанням дискримінатора для наповнення набору прикладів поведінки (рис. 3.8) перший стан у який переходить система після зчитування запиту користувача є генерування прикладів потенційних наступних запитів користувача. Якщо їх згенеровано достатньо відповідно до заданого параметра у системі, то вона переходить до генерування відповідей на кожен із цих прикладів. Після цього дискримінатор класифікує кожен приклад і вони добавляються у набір прикладів поведінки. Наприкінці цей набір вбудовується у промпт генератора і він дає

відповідь на оригінальний запит користувача. Після цього результати видаються користувачеві.

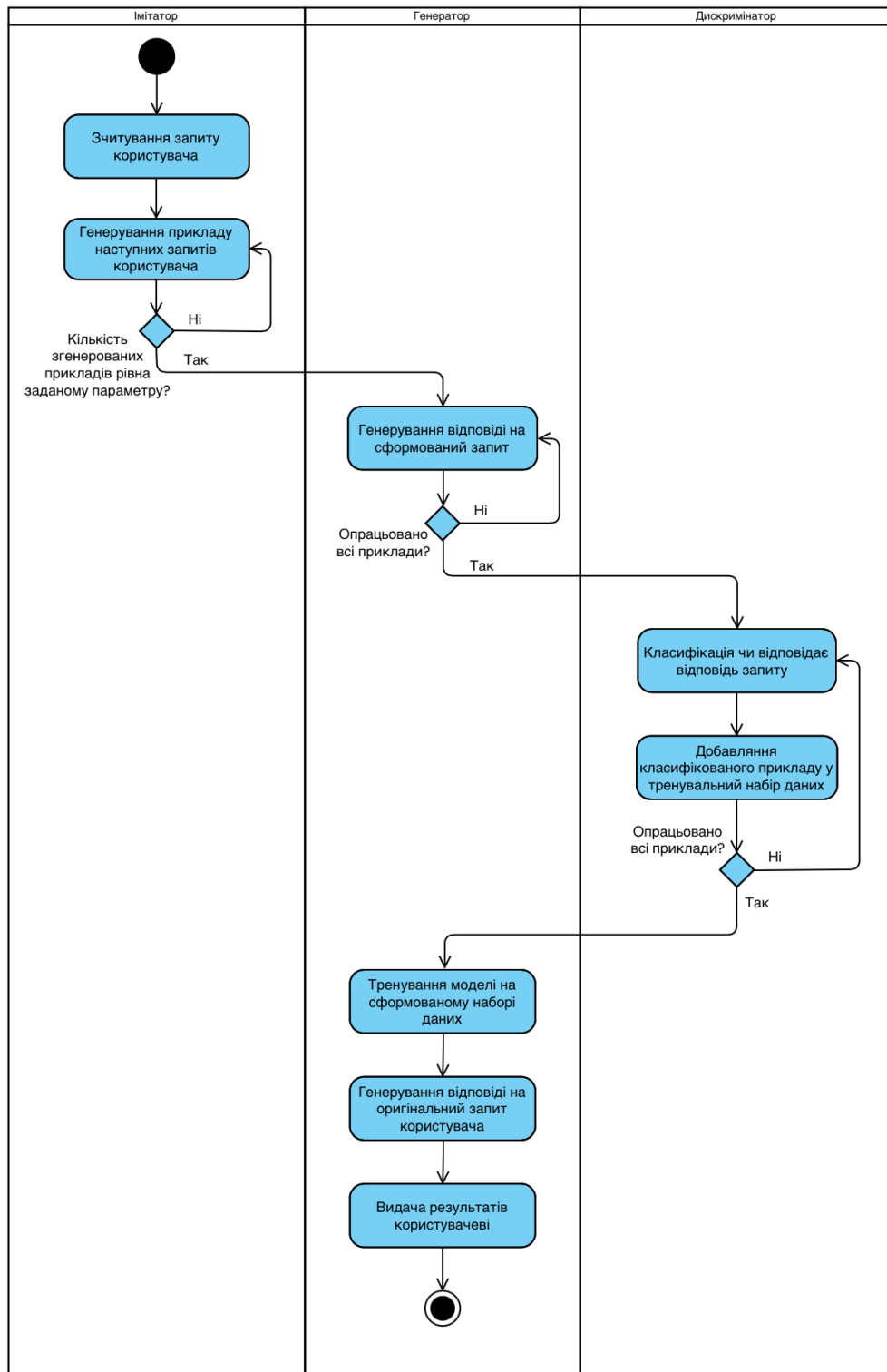


Рис. 3.8. Діаграма діяльності для підходу із використанням дискримінатора для наповнення набору прикладів поведінки

У підході з використанням дискримінатора для відсіювання неправильних відповідей (рис. 3.9) першим станом після зчитування запиту є генерування відповіді на запит. Після цього вона перевіряється дискримінатором на відповідність запиту користувача і якщо вона правильна, то зразу видається користувачеві. Якщо відповідь неправильна, то вона система повторно переходить у стан генерування відповіді і генерує наступний варіант відповіді. Цикл між генератором і дискримінатором повторюється допоки відповідь не буде класифікована як правильна дискримінатором, або не буде перевищено ліміт.

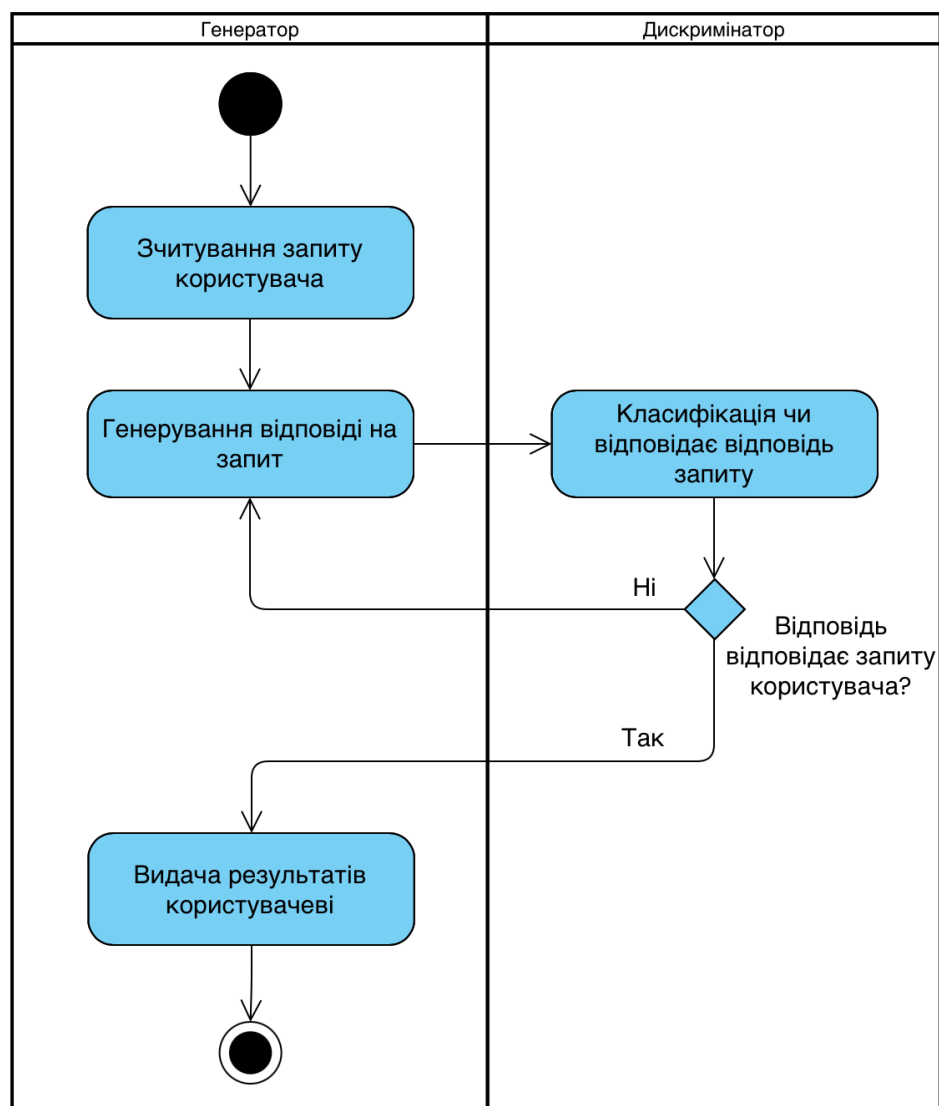


Рис. 3.9. Діаграма діяльності для підходу із використанням дискримінатора для відсіювання неправильних відповідей

3.5.4. Діаграма класів

Основним класом є клас `TextGenerationEngine`. Він відповідає за зчитування запитів користувача і видачу йому відповідей.

У підході із використанням дискримінатора для наповнення набору прикладів поведінки (рис. 3.10), метод `get_response` відповідає за надання відповіді і реалізує логіку взаємодії імітатора, генератора, та дискримінатора, які реалізовані такими класами як `Imitator`, `Generator`, та `Discriminator` відповідно. `TextGenerationEngine` працює з інтерфейсом `TextGenerator`, який вже реалізують конкретні класи.

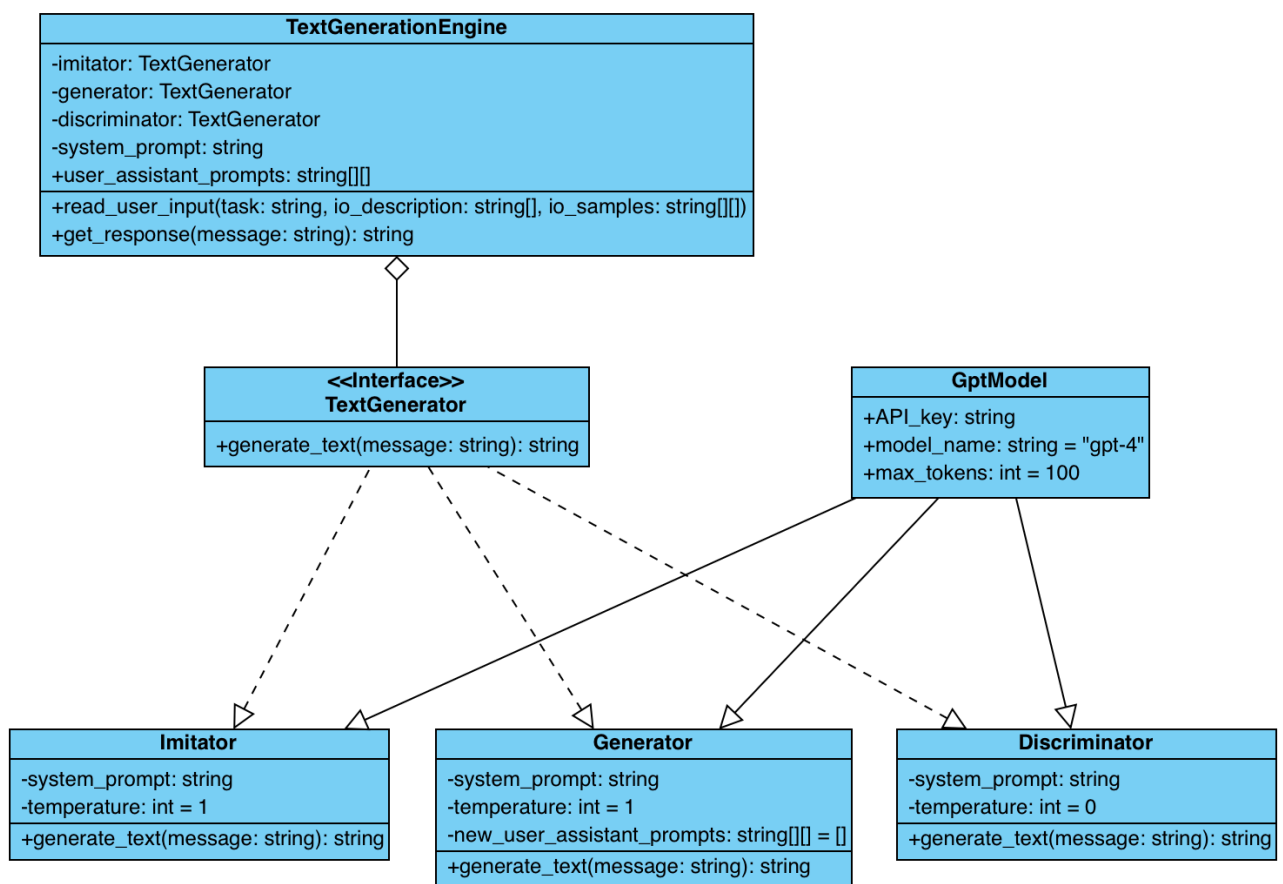


Рис. 3.10. Діаграма класів для підходу із використанням дискримінатора для наповнення набору прикладів поведінки

Вимогою для реалізації цього інтерфейсу є лише реалізація методу `generate_text`, тобто він є абстракцією для будь-якого генератора тексту. У класах `Imitator`, `Generator`, та `Discriminator` атрибут `temperature` відповідає за те, наскільки

творчими і різноманітними будуть відповіді моделі. У Imitator та Generator його значення рівне одиниці, а у Discriminator рівне нулю, бо він повинен бути консистентним у своїх відповідях, тобто у його відповідях менший рівень випадковості. Кожен має свій `system_prompt`, але у класі Generator також є атрибут `new_user_assistant_prompts`, що зберігає згенеровані приклади поведінки системи, які будуть вбудовуватись у промпт. Клас `GptModel` наслідується класами Imitator, Generator, та Discriminator. Він зберігає спільні для них атрибути, як наприклад максимальну кількість токенів.

На відміну від попереднього підходу, у підході із використанням дискримінатора для відсіювання неправильних відповідей (рис. 3.11), немає класу Imitator, а також атрибуту `new_user_assistant_prompts` у класі Generator, бо йому не потрібно вбудовувати у промпт нові приклади поведінки.

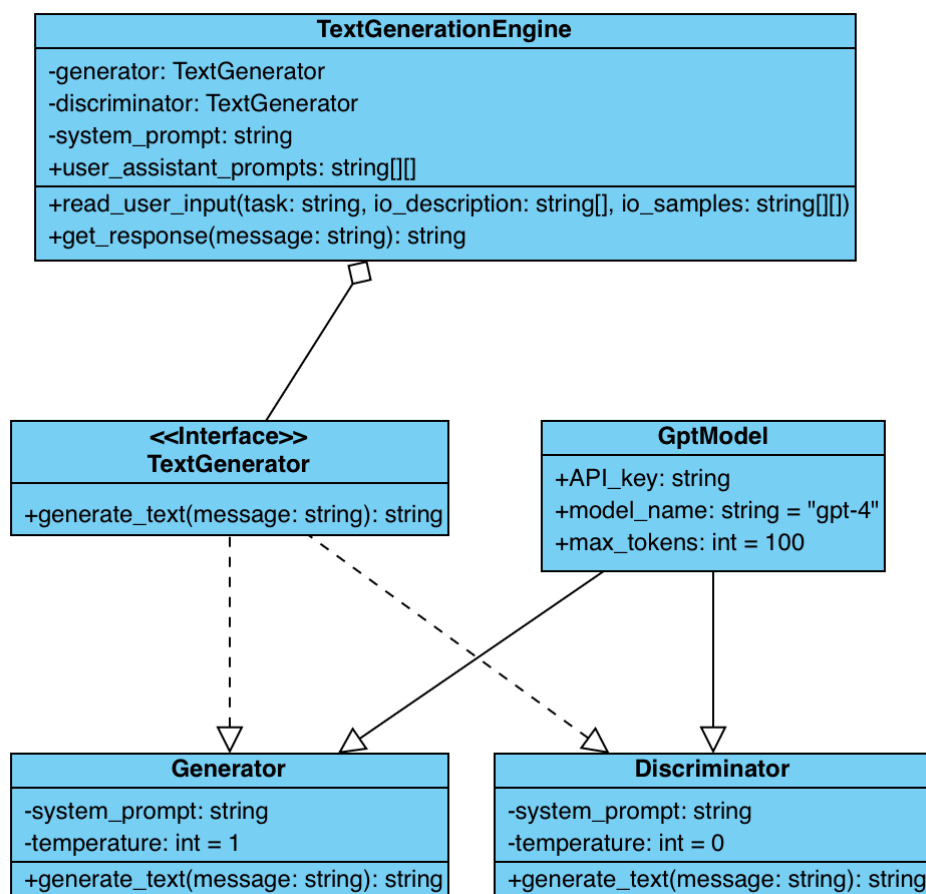


Рис. 3.11. Діаграма класів для підходу із використанням дискримінатора для відсіювання неправильних відповідей

Відповідно метод `get_response` реалізує іншу логіку взаємодії генератора, та дискримінатора, які реалізовані такими класами `Generator` та `Discriminator`. Він реалізує цикл у якому генератор здійснює генерацію відповідей, а дискримінатор їх перевіряє. Цикл завершується, якщо при перевірці відповідь була класифікована як правильна, або був перевищений ліміт генерацій.

3.5.5. Діаграма розгортання

У діаграмі розгортання зображено обчислювальні вузли, а також компоненти, що виконуються у цих вузлах. У обидвох підходах у окремих вузлах розділено компонент опрацювання запитів і компоненти, що генерують текст. При цьому кожен компонент, що генерує текст також знаходиться у окремому обчислювальному вузлі для того, щоб можна було розпаралелити опрацювання, якщо буде багато запитів на систему. У підході із використанням дискримінатора для наповнення набору прикладів поведінки (рис. 3.12) обчислювальний вузол системи також зберігає набір прикладів поведінки, який є артефактом, що є результатом роботи дискримінатора.

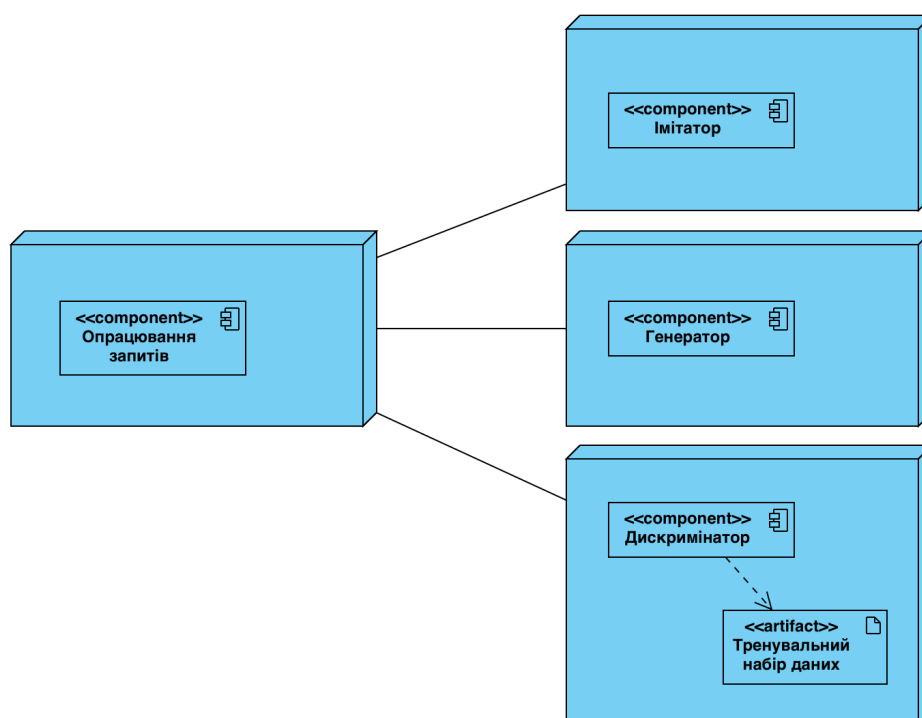


Рис. 3.12. Діаграма розгортання для підходу із використанням дискримінатора для наповнення набору прикладів поведінки

Схожою є діаграма розгортання у підході із використанням дискримінатора для відсіювання неправильних відповідей, але у ній немає ні набору прикладів поведінки, ні компоненту імітатора, як зображено на рис. 3.13.

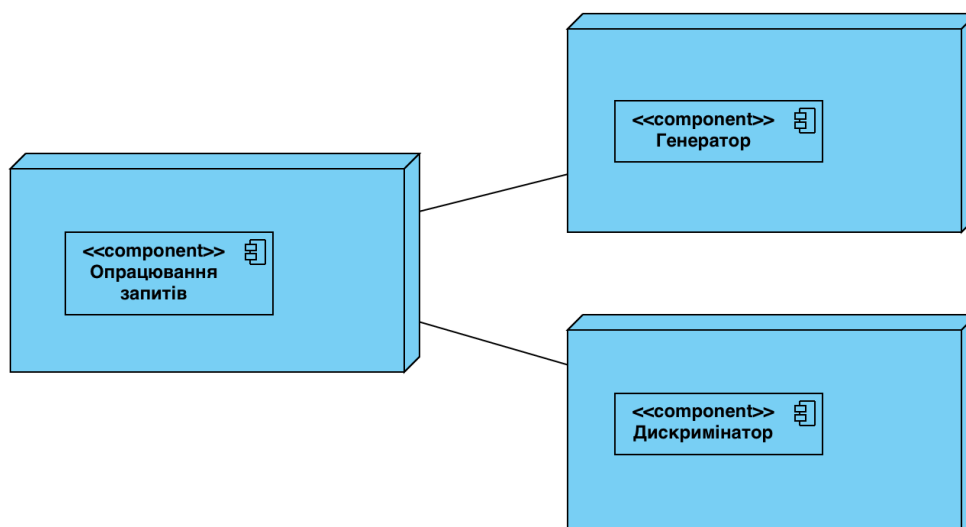


Рис. 3.13. Діаграма розгортання для підходу із використанням дискримінатора для відсіювання неправильних відповідей

3.5.6. Опис запропонованого методу генерування текстового контенту

Алгоритм генерування тексту, що використовує дискримінатор для формування набору прикладів поведінки включає такі базові кроки:

- Генерація тексту на основі базової моделі;
- Ідентифікація відповідності критеріям користувача;
- Вбудовування у промпт розмічених прикладів поведінки з попереднього кроку;
- Генерація тексту на основі фінальної моделі.

Алгоритм генерування тексту, що використовує дискримінатор для відсіювання неправильних відповідей включає такі базові кроки:

- Генерація тексту;
- Перевірка відповідності згенерованого тексту запиту користувача;
- Повторна генерація тексту, якщо текст згенеровано неправильно.

На основі цих двох алгоритмів генерування тексту можемо сформуванати загальний метод, що використовує вбудовування у промпт прикладів поведінки, а після цього відсіювання неправильних відповідей.

Загальний опис методу:

1. Користувач робить запит x .
2. Модель I (імітатор) генерує N прикладів запитів користувача. За замовчуванням $N = 40$.
3. Модель G (базовий генератор) генерує відповідь до кожного із N прикладів.
4. Модель D (дискримінатор) класифікує кожен із N прикладів на відповідність запиту користувача x .
5. Розмічені приклади за допомогою моделі D зберігаються.
6. Створюється копія моделі G , якій у промпт вбудовують розмічені приклади. Копія G із вбудованими у промпт прикладами поведінки утворює модель G_t (генератор із роширеним промптом), що враховує ці приклади для генерації, на відміну від оригінальної моделі G .
7. Модель G_t використовується для відповіді на оригінальний запит користувача x .
8. Модель D перевіряє чи згенерований текст відповідає запиту користувача x .
9. Якщо на попередньому кроці відповідь моделі класифікувалась як неправильна, то відбувається повторна генерація тексту за допомогою моделі G_t (тобто повертаємось на крок 7).
10. Повторне генерування тексту відбувається допоки модель D не класифікує відповідь як правильну, або не буде перевищено ліміт кількості генерацій L . За замовчуванням $L = 3$.
11. Після цього останній варіант відповіді видається користувачу. Якщо цикл генерування завершився через перевищення ліміту кількості генерацій L , то відповідь помічаєть міткою, щоб показати користувачу, що система не впевнена у цій відповіді.

Висновки до розділу 3

Перевірено на практиці якість логічного мислення великої мовної моделі GPT-4 та можливість цієї моделі себе перевіряти у контексті задачі логічного виведення.

Запропоновано метрику для визначення впевненості у правильності відповіді великих мовних моделей.

Описано алгоритм використання великої мовної моделі для перевірки своїх відповідей. Наведено діаграми варіантів використання, послідовності, діяльності, класів, розгортання для підходу із використанням дискримінатора для наповнення набору прикладів поведінки, та підходу із використанням дискримінатора для відсіювання неправильних відповідей.

РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ БАГАТОЕТАПНОГО ГЕНЕРУВАННЯ СЕМАНТИЧНО КОРЕКТНИХ ТЕКСТІВ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1. Постановка початкового експерименту

Раніше задача яку ми розв'язуємо розглядалась з точки зору покращення консистентності знань моделі, але ми також можемо її розглядати з точки зору проблеми узгодження систем штучного інтелекту. Важливість розгляду із цієї точки зору полягає у тому, що це зараз дуже активна галузь досліджень у сфері штучного інтелекту. Зростаюча популярність великих мовних моделей підкреслила потребу їх узгодження із потребами користувача. Задача узгодження є однією із найважливіших підзадач безпеки штучного інтелекту. Деякі дослідники штучного інтелекту стверджують, що у майбутньому ця проблема буде ще більш нагальною, через те, що системи будуть більш потужними і в свою чергу зможуть краще знаходити обхідні шляхи досягнення поставлених перед ними задач. Зараз ці проблеми виникають у комерційних продуктах пов'язаних із великими мовними моделями, рекомендаційними системами, автономними транспортними засобами тощо.

Задача узгодження систем штучного інтелекту полягає у секривуванні систем до цілей, уподобань, та етичних принципів людини. Система вважається узгодженою, якщо вона досягає намічених цілей, і неузгодженою, якщо вона переслідує певні цілі, які не були заплановані [66]. Проблема узгодження полягає у складності опису універсальної бажаної поведінки, через це розробники таких систем часто описують проміжні спрощені цілі. Прикладом може бути отримання зворотного відгуку від людини. Але такий підхід може створювати лазівки і винагороджувати систему за те, що вона імітує бажану поведінку. Системи можуть навчитись досягати проміжних цілей, при цьому не досягаючи бажаної кінцевих цілей. Такі неузгоджені системи можуть завдати шкоди при використанні у реальних умовах.

Із ростом розміру моделей опрацювання природної мови більш очевидною стає проблему узгодження цих моделей [93, 94]. Ця задача є однією із підзадач

безпеки штучного інтелекту. У даному розділі описано реалізацію методів багатоетапного генерування текстів з використанням великих мовних моделей для узгодження згенерованого тексту із запитом користувача.

Для покращення точності великих мовних моделей використовують різні стратегії. Розробник GPT-4, компанія OpenAI, вказує розбиття складних задач на простіші як одну із стратегій конструювання пром프트 для покращення результатів. Наприклад одна модель класифікує запит користувача, а інша пропонує вирішення його проблеми. Ще одна стратегія що пропонується це дати час на роздуми. Наприклад, модель витягнула із книги всі абзаци що стосуються певної теми, але після цього можна її запитати щоб вона перевірила чи ніякі абзаци вона не забула.

Зараз активно ведуться дослідження, що пов'язані із багатоетапною генерацією тексту [70]. Прикладом може бути робота Large Language Models are Zero-Shot Reasoners [95] у якій запропоновано запитати модель описати кроки для вирішення задачі. Після того як вона згенерує кроки, їх разом із оригінальною задачею передають цій є моделі на вхід щоб вона дала остаточну відповідь, як зображено на рис. 4.1.

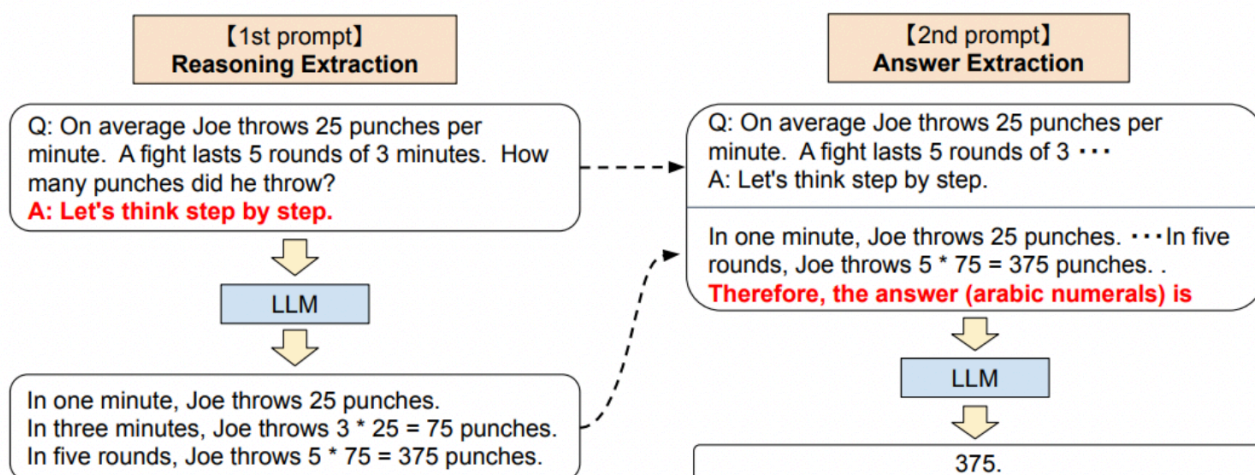


Рис. 4.1. Запитування великої мовної моделі описати кроки для знаходження відповіді на запитання

У роботі Bootstrapping Reasoning With Reasoning [96] також запропоновано генерувати покрокове вирішення задачі, а потім добавляти його до тренувального набору даних (рис. 4.2). Добавляться будуть лише ті покрокові вирішення які

ведуть до правильної відповіді, правильність можна перевірити, бо працюємо із розміченими даними. Після доповнення тренувального набору новими даними ми ще раз проходимося по прикладах, для яких було неправильно згенеровано покрокове вирішення.

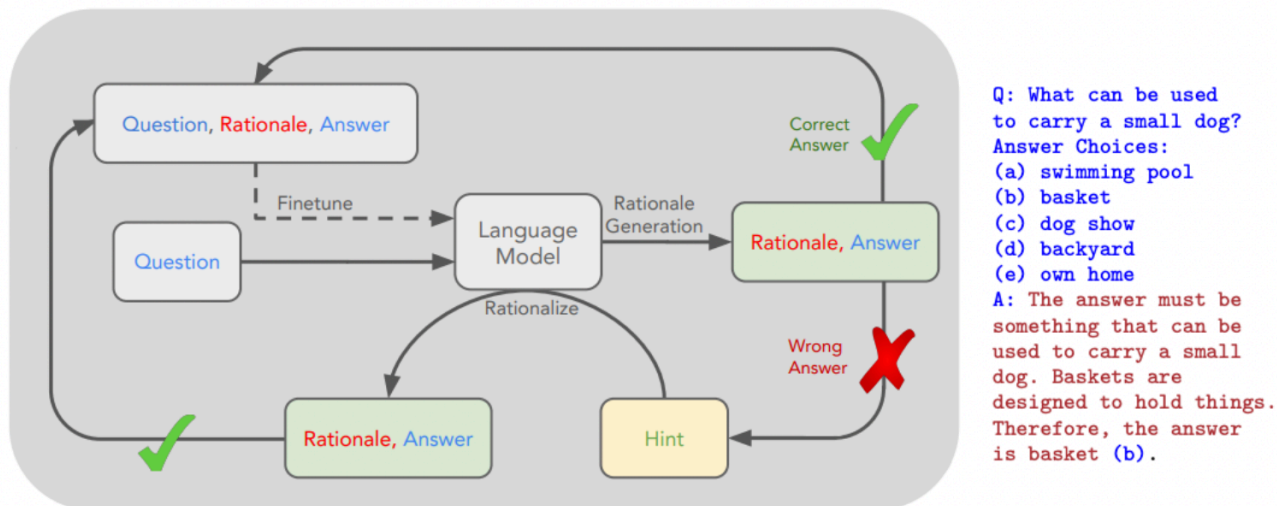


Рис. 4.2. Наповнення тренувального набору даних правильними покроковими відповідями на запитання

Ще одним прикладом може бути робота Self-Consistency Improves Chain of Thought Reasoning in Language Models [97] де модель генерує багато відповідей на одне питання, а кінцевою відповіддю вважається та, яка найчастіше зустрічається (рис. 4.3).

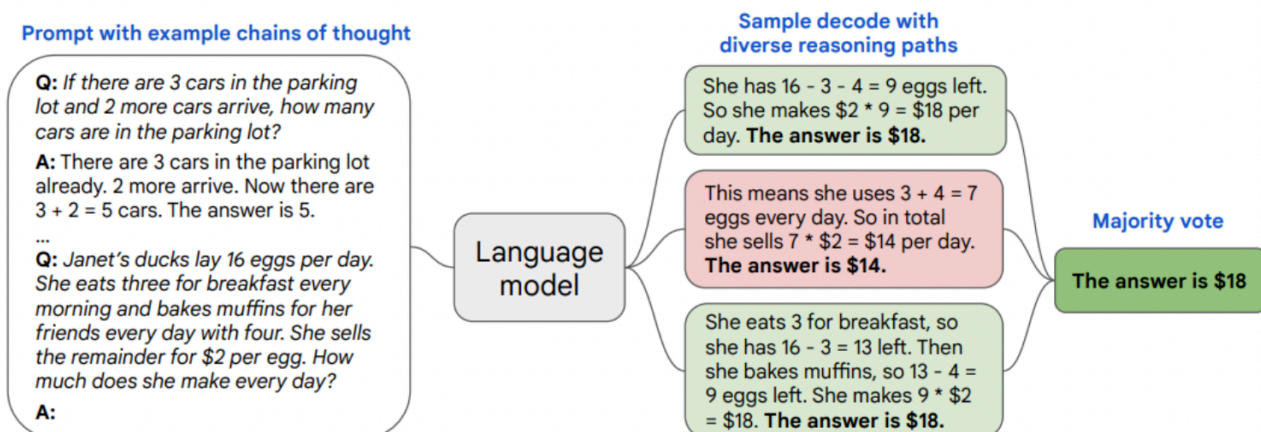


Рис. 4.3. Генерування декількох відповідей і вибір найбільш популярної

У роботі Training Verifiers to Solve Math Word Problems [84] тренується модель верифікатор, який буде перевіряти генератор (рис. 4.4). Схожий підхід було використано і у цій роботі, але запропонований підхід не вимагає розмічених даних.

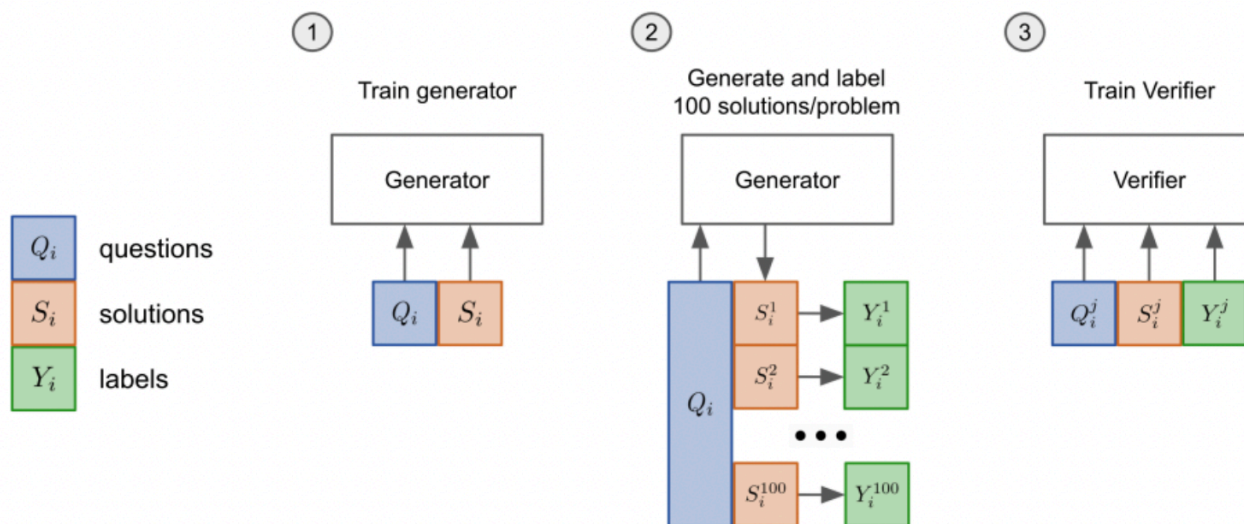


Рис. 4.4. Використання верифікатора для перевірки генератора для знаходження відповіді на запитання на розмічених даних

Прикладом великої мовної моделі для розробки якої була задача узгодження із потребами користувачів є InstructGPT. Це версія моделі GPT, але дотренована з використанням підходу Reinforcement Learning from Human Feedback. Ця модель розроблена у 2022 році. Це показує що навчання моделей відповідати очікуванням користувача зараз є активною галуззю досліджень. Працівники OpenAI що займаються розміткою даних віддають перевагу InstructGPT порівняно із GPT-3 (175B), незважаючи на те, що вона мають більш ніж у 100 разів менше параметрів. З цього можна зробити висновок, що тренуючи модель генерувати текст, що краще відповідає очікуванням користувача ми можемо отримати кращі результати ніж тренуючи більшу модель для передбачення наступного слова, що є типовою ціллю (objective) для статистичних моделей мови. Це показує перспективність такого підходу для покращення результатів моделі.

Для перевірки якості запропоновано підходу було обрано просту задачу генерування пари речень між якими є строга імплікація. Користувач передає

речення моделі, а модель генерує інше, яке завжди має бути істинним, якщо перше є істинним. "Об'єкт може літати" і "Об'єкт має крила" - це пара, яка не має імплікації, оскільки, наприклад, повітряна куля може літати без крил. Процес постановки експерименту візуально цей зображено на рис. 4.5.

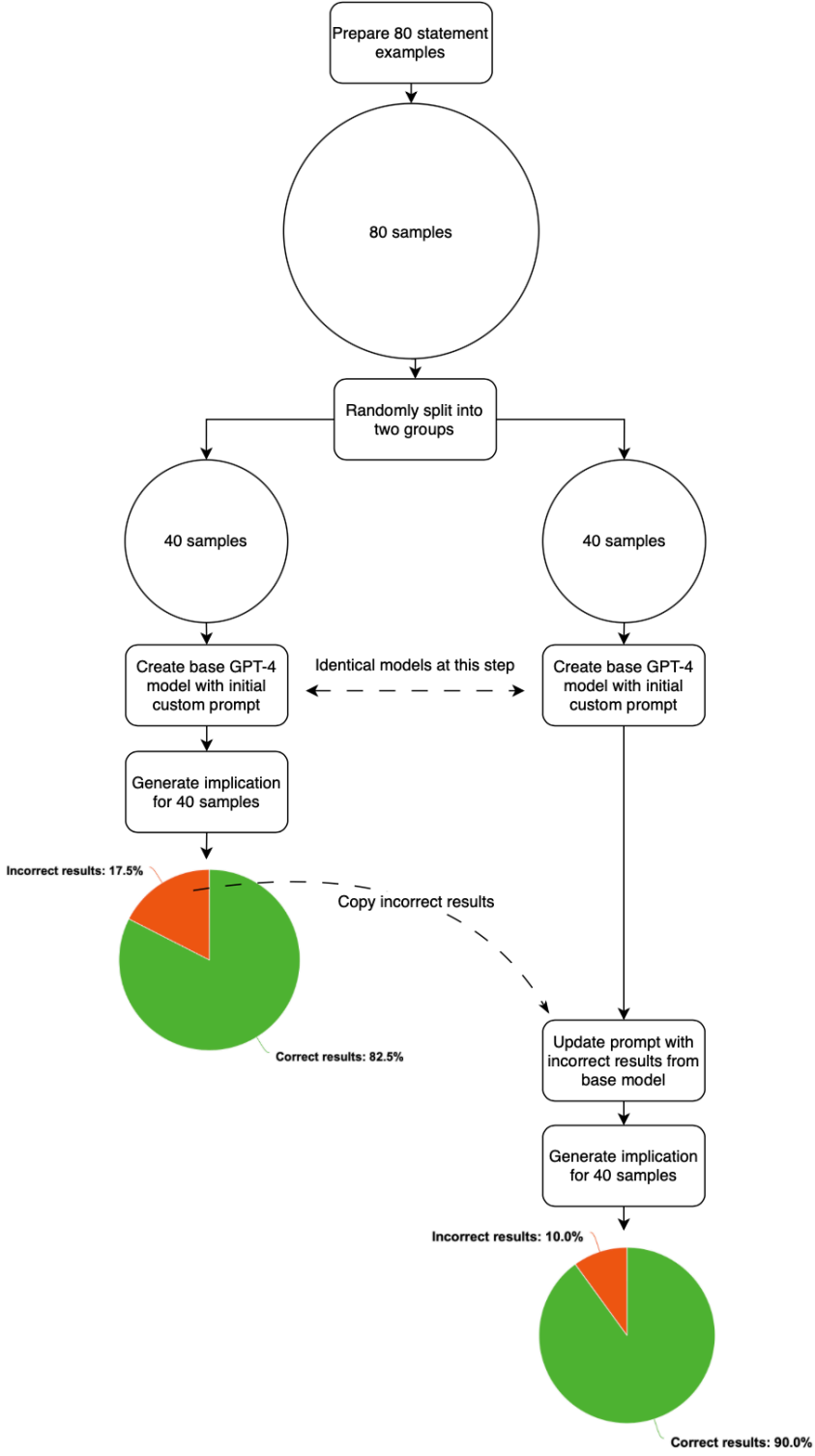


Рис. 4.5. Візуальне представлення постановки початкового експерименту

Для експерименту було використано 80 зразків речень (деякі з них створені за допомогою моделі GPT, а деякі створені вручну). Після цього було випадково вибрано половину речень та передано в модель GPT-4 для генерування речень між якими є імплікація. Маючи результати цього кроку обраховується точність базової моделі.

Потім іншу модель GPT-4 попросили класифікувати, чи є імплікація між парами речень із попереднього кроку. Пари, які були класифіковані як “не імплікація”, були додані як приклади негативної поведінки для третьої моделі, яка є дублікатом базової моделі генерації. Фінальна модель потім тестується на другій половині 80 зразків речень. Це буде точністю фінальної моделі.

На рис. 4.6 можемо бачити, що запропонована модель, у якій ми добавляли неправильні генерації як зразки негативних результатів, підняла точність генерації тексту із 82.5% до 90%. Точність вимірювалась на 40 прикладах для кожної моделі. Метрикою точності рахувалась як відношення правильних генерацій до всіх генерацій.

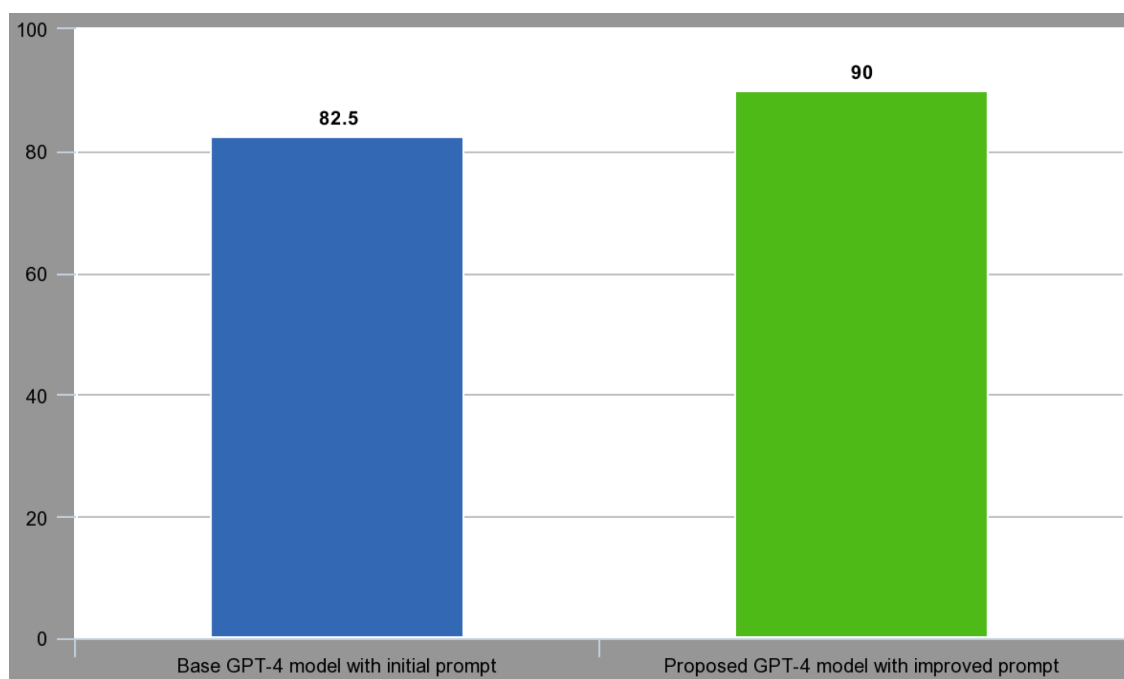


Рис. 4.6. Порівняння точності базової і запропонованої моделі

Запропонований метод покращення якості генерації тексту буде корисним у випадках, коли користувач не може надати моделі достатньо прикладів, щоб

описати бажану поведінку моделі для генерації тексту. Було використано конструювання промπτу, але для покращення результатів моделі можна використати навчання з учителем для дотреновування моделі та навчання з підкріпленням на основі зворотного зв'язку людини.

Модель було перевірено на спрощеному завданні генерації речень між якими є імплікація. Хоча його було перевірено на конкретному спрощеному завданні, такий підхід може допомогти будувати моделі генерації тексту, які краще відповідають уподобанням користувача. Для того щоб використовувати запропонований підхід потрібно сформулювати задачу у такому форматі щоб задачею моделі було генерувати текст який відповідає певному критерію. Прикладом можуть бути діалогові системи. Наприклад користувач хоче, щоб відповіді були офіційною, а не розмовною мовою і звучали професійно. Критерієм тут виступає те, чи текст звучить офіційно чи є у розмовному стилі. Проблема в тому, що користувачеві буде важко згенерувати 50-100 зразків, які можуть знадобитися моделі для розуміння нюансів. Тож замість цього ми можемо попросити модель створити професійний текст, а потім попросити її визначити, чи звучить він професійно. І якщо згенерований текст має звучати професійно, але модель визначає його як непрофесійний, ми можемо додати це до навчальних даних як приклад непрофесійного тексту.

Результати показують даний підхід покращує якість генерації тексту навіть при невеликій кількості даних які передавались на вхід моделі. Можемо бачити перспективність цього підходу, бо точність для задачі генерування пар речень між якими є імплікація зріс із 82.5% до 90%. Проте на цьому етапі його перевірено на невеликій кількості даних і пізніше буде описано як здійснювалась перевірка на більшій кількості даних та зі зміною постановки експерименту.

Отже, було запропоновано метод покращення якості генерування тексту великими мовними моделями на прикладі моделі GPT-4. Запропоновано ітеративний метод для узгодження згенерованого тексту із запитом користувача шляхом вбудовування у промπτ моделі прикладів, на яких вона допускає помилки. Воно відбувається автоматично з передачею на вхід моделі прикладів, у яких була допущена помилка для повторного опрацювання. У порівнянні з оригінальною

базовою моделлю, запропонований підхід демонструє суттєві покращення, збільшуючи точність (accuracy) з 82.5 до 90. Запропонований метод під час експериментів показав перспективність для практичного застосування у реальних задачах генерації тексту.

4.2. Реалізація остаточного методу генерування тексту

В даний час особливо актуальною є задача узгодження великих мовних моделей. При цьому моделі настільки здібні, що можуть вирішити багато різних завдань, використовуючи підхід zero-shot. Але оскільки вони стали більш здібними, вони знаходять обхідні шляхи для вирішення завдань не так, як очікують дослідники. Це особливо небезпечно у при використанні на справжніх задачах, оскільки важко контролювати вихід моделі, яка була навчена бути універсальною. Пропонується використовувати одну і ту ж модель кілька разів в різній формі з метою покращення якості згенерованого тексту.

Подальший розвиток отримав метод підвищення точності моделей генерації текстового контенту. Це дозволяє користувачеві не наводити десятки прикладів бажаної та небажаної поведінки моделі, оскільки сама модель може робити це автоматично. Тобто, на відміну від звичайних методів підвищення точності моделі, які вимагають навчального набору моделей, запропонований підхід включає етап ідентифікації. В результаті ідентифікації ми отримуємо набір прикладів, на яких модель автоматично “навчається” і тим самим підвищує свою точність.

Для покращення якості згенерованого тексту використовується багатоетапний підхід, що включає модель дискримінатор для перевірки результатів генерації. Тобто після того як основна модель згенерує текст, він передаються на вхід іншій моделі, яка від основної відрізняється промптом. Ця інша модель дискримінатор визначає чи відповідає результат оригінальному запиту користувача.

У цьому розділі буде реалізовано два алгоритми і метод для генерування тексту, що використовує ці два алгоритми послідовно. Перший алгоритм більш складний, він окрім дискримінатора також використовує модель імітатора. Процес

вимагає, щоб модель імітатора генерувала зразки введених користувачем даних, потім генератор генерував текст відповіді для кожного зразка, після чого дискримінатор перевіряв згенеровані результати та додавав їх до набору прикладів поведінки. Це підвищило точність з 56% до 66% у задачі логічного виведення.

Другий алгоритм використовує модель дискримінатор для перевірки результатів моделі генератора та запитує повторно створити текст, якщо результати не відповідають критеріям користувача. За допомогою цього підходу вдалося видалити всі неправильні генерації, але за рахунок позначення третини правильних як неправильні.

Тобто, розглядається два способи використання дискримінатора, щоб дослідити чи повторне передавання на вхід великій мовній моделі згенерованих прикладів може покращити якість генерування. Перший для розмічування результатів генерації для збільшення набору прикладів поведінки для основної моделі і другий для відсіювання неправильно згенерованого тексту. У першому на вхід передаються приклади у яких модель допустила помилку, але вказується, що це приклади негативної поведінки, для того, щоб модель більше не допускала схожих помилок. Процес є повністю автоматичним, бо розміткою займається дискримінатор. Також розглядається можливість використовувати обидва способи послідовно для генерування тексту.

4.2.1. Програмна реалізація запропонованих алгоритмів генерування тексту

Користувач просить модель згенерувати текст, який буде відповідати певним правилам, і надає кілька прикладів. Модель використовує введені користувачем дані для генерування відповіді, але перед її відображенням модель перевіряє, чи згенерований текст відповідає правилам, що задані користувачем. Якщо ні, він додається як зразок негативної поведінки. Щоб перевірити цей підхід, було обрано просте завдання генерувати імплікацію з речення. Архітектуру системи представлено у таблиці 4.1.

Після проведення початкового експерименту було вирішено змінити підхід. У попередньому підході набір висловлювань ділився на дві групи, але було

вирішено виділити окремий набір даних для тестування всіх моделей, бо в такому випадку ми імітуємо поведінку моделі справжніх умовах використання. Тепер модель буде самостійно генерувати собі приклади запитів користувача. Через те, що попередньому підході ми давали додатковий набір даних моделі порівняно з базовою, то якість могла покращуватись через більшу кількість даних, а не те, що модель здатна себе покращити [98].

Таблиця 4.1

Компоненти системи

Назва компонента архітектури системи	Призначення
get_baseline_model_response	Отримання результатів базової моделі GPT-4.
get_classification_model_response	Класифікація результатів базової моделі на істинні та хибні (хибні результати використовуються для покращення моделі).
get_trained_model_response	Отримання результатів покращеної моделі.

Два запропоновані алгоритми покращують якість текстових генеративних моделей. Перший (рис. 4.7) полягає в тому, що модель імітатор генерує певну кількість потенційних запитів користувача. Потім до кожного запиту генерується відповідь. Після цього кожну відповідь перевіряє модель дискримінатор і з відповідною міткою передає його у набір прикладів поведінки. Модель із додаванням цих прикладів поведінки у промпт генерує текст по оригінальному запиту користувача. А другий (рис. 4.8), у тому, що після генерації модель дискримінатор відкидає текст, якщо він не відповідає оригінальному запиту

користувача. Текст генерується декілька разів поки дискримінатор не підтвердить, що текст правильний.

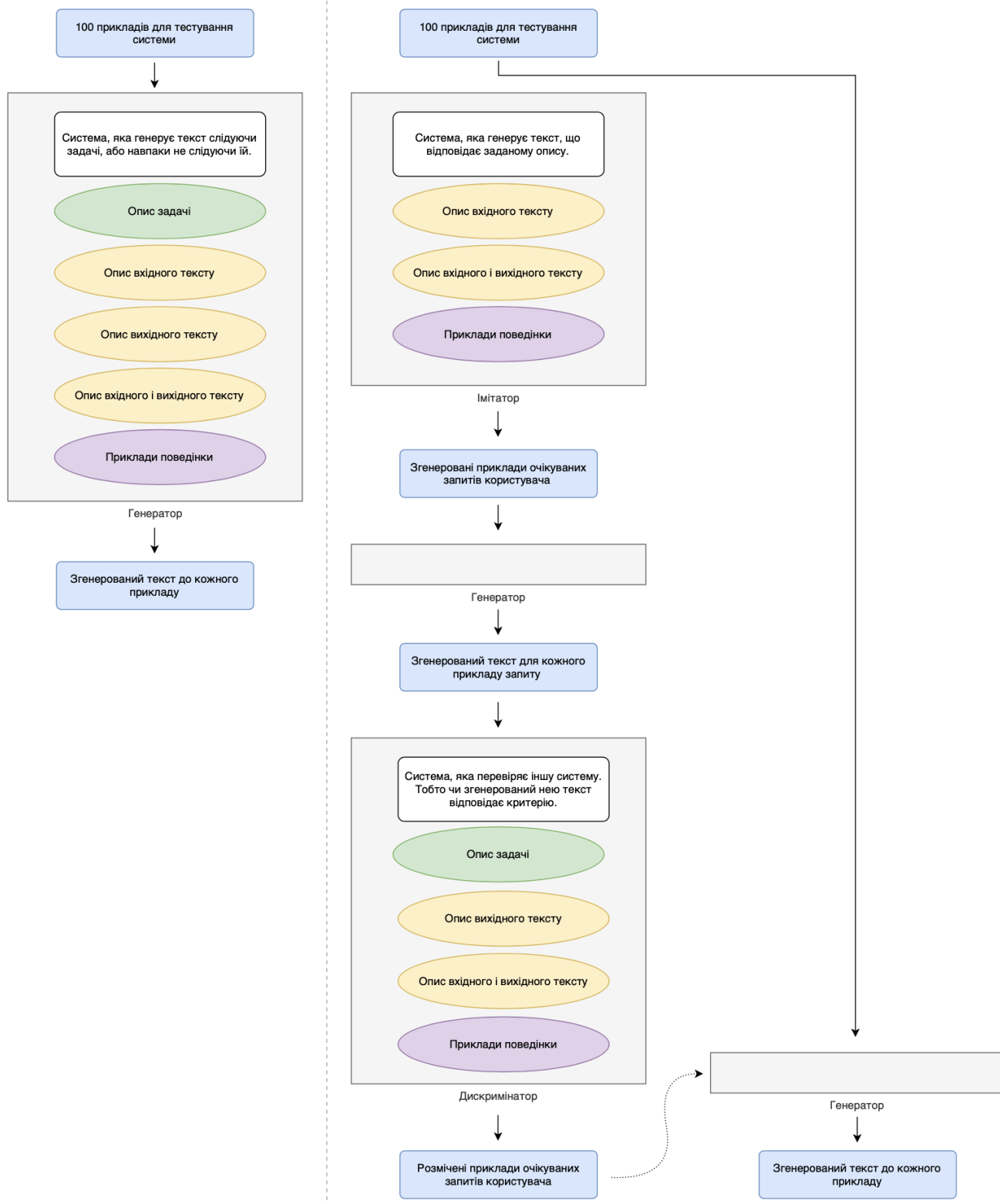


Рис. 4.7. Використанням імітатора та дискримінатора для збільшення набору прикладів поведінки

Для того щоб використовувати запропоновані підходи потрібно сформулювати задачу, описати вимоги до вхідного і вихідного тексту, надати декілька прикладів правильної і неправильної поведінки системи. Другий алгоритм генерування тексту дає значне покращення результатів і може бути використана для будь-якої задачі пов'язаної з генерацією тексту. Перший має більш вузьке застосування. Особливо коли немає ресурсів підготувати для моделі більше декількох прикладів і якщо легше сформулювати задачу показуючи неправильні приклади.



Рис. 4.8. Використанням дискримінатора для перевірки згенерованого тексту

Обидва алгоритми генерування тексту використовують дискримінатор для оцінки результатів після того, як основна модель згенерує текст. Реалізуємо метод, що буде використовувати два алгоритми, згенерувавши текст першим алгоритмом, а потім передавши його результат на вхід другому. Це дало можливість покращити точність генерування тексту на прикладі задачі логічного виведення [98].

4.2.2. Аналіз результатів та точності запропонованих підходів

У підході із додаванням до промпту моделі нових прикладах точність зросла із 56% до 66%. Метрика точності рахувалась як відношення правильних генерацій до всіх генерацій на 100 прикладах. Є дві, що обмежують приріст точності. Це мультиплікація помилки та збільшення те, що кількості прикладів недостатньо значне. Але більше прикладів, більший вплив помилок у новому наборі даних і більша вартість, що вимагає fine-tuning (який є обмежений у GPT-4). Варто звернути увагу, що запропоновані підходи тестувались на задачі логічного виведення. Прикладом є запит “Об’єкт має всі кути 90 градусів”, базова модель відповіла “Об’єкт є квадратом”, а запропонована “Об’єкт є прямокутником”.

Підхід із моделлю дискримінатором для перевірки згенерованого тексту більш простий з точки зору кількості кроків, який також тестувався на 100 прикладах з тією ж метрикою. Результат тестування це 60 правильних класифікацій “так”, 28 правильних класифікацій “ні”, 12 неправильних класифікацій “так”, та 0 неправильних класифікацій “ні”. Тобто модель дискримінатор правильно відкине близько 28% відповідей згенерованих системою, бо класифікує їх як ті, що не відповідають критерію користувача. Із тих, що модель відкинула, всі будуть відкинуті правильно. Тобто модель дискримінатор не відкинула жодної правильної генерації, але з тих що не відкинула, правильними були близько 83% (60/72) згенерованих прикладів. Прикладом пари речень, які відсіяла модель дискримінатор є запит “Об’єкт вивчає фізику” і відповідь базової моделі “Об’єкт є студентом”, бо не тільки студенти вивчають фізику.

Прикладами згенерованих висловлювань, які відкинув дискримінатор, є також пара запит-відповідь “Об’єкт має крила” та “Об’єкт може літати”, або пара “Об’єкт вивчає фізику” та “Об’єкт є студентом”. Вони були правильно відкинуті, бо не всі об’єкти, що мають крила можуть літати, наприклад повітряна куля. Також не всі хто вивчають фізику є студентами.

Матрицю невідповідності зображено у таблиці 4.2. У нашому випадку позитивний стан це коли дискримінатор класифікував, що відповідь є правильною, а негативний коли класифікував, що відповідь неправильна. При цьому, якщо він класифікував як правильну і вона дійсно правильна, то цей приклад відносимо до істинно позитивних (ІП), а якщо якщо вона насправді неправильна, то відносимо до хибно позитивних (ХП). Якщо дискримінатор класифікував, що відповідь неправильна і вона дійсно неправильна, то відносимо цей приклад до істинно негативних (ІН), а якщо вона насправді правильна, то до хибно негативних (ХН).

Таблиця 4.2

Матриця невідповідностей для дискримінатора, якщо йому передаємо текст згенерований базовою моделлю GPT-4

		Справжній клас	
		Позитивний стан	Негативний стан
Прогнозований клас	Позитивний стан	ІП = 60	ХП = 12
	Негативний стан	ХН = 0	ІН = 28

Тобто, якщо обраховуватись метрики точності цього підходу, то повнота (recall) буде рівна 1, а влучність (precision) буде дорівнювати близько 0.83 (60/72). Цей підхід буде кращим при застосуванні у реальних умовах, бо буде мати високу влучність. Це забезпечується тим, що у прикладах в яких система не впевнена

буде позначка, що вона має сумніви стосовно правильності цієї відповіді. Це дозволить користувачеві змінити запит і згенерувати іншу відповідь. Точність (accuracy) буде рівна 0.6, або 60%. У попередньому підході із вбудовуванням у промпт моделі нових прикладів поведінки ми рахували лише точність, бо там немає бінарної класифікації і відповідно прогнозованого негативного стану. Якщо порівнювати точність обидвох підходів, то попередній має більшу, а саме 66%, порівняно із точністю цього підходу 60%. Але якщо користувач відкине всі згенеровані приклади, в яких система сумнівається, то перевага є у другого підходу, бо впевненість у правильному результаті стає рівною 83%. У реальних умовах застосування акцент робиться на влучність (precision), бо зазвичай користувач зможе перегенерувати приклади, в яких система сумнівається, зробивши запит повторно, або змінивши запит. Якщо користувач має можливість це зробити, то його впевненість в результатах буде значно більшою (83%) порівняно із першим підходом у якому впевненість у правильності результатів буде 66%. Тому у реальних умовах застосування другий підхід має значні переваги.

Також було перевірено точність підходу що послідовно використовує два попередні. Тобто спочатку використовуємо дискримінатор розмічування згенерованих відповідей, які потім вбудовуємо у промпт. Після цього модель з розширеним промптом генерує відповідь, перевіряється дискримінатором і генерує текст повторно, якщо відповідь неправильна.

Точність першої частини цього підходу рівна 0.66, як зазначалося раніше. Це більше ніж базова GPT-4 модель, яка мала точність 0.55. Друга частина цього підходу перевіряє згенерований текст на попередньому кроці і робить запит на повторну генерацію. За рахунок того, що на вхід дискримінатору передаємо текст, згенерований із більшою точністю порівняно із базовою GPT-4 моделлю, зростає і точність такого комбінованого підходу. У таблиці 4.3 зображена матриця невідповідностей для дискримінатора, якщо йому передаємо текст згенерований моделлю GPT-4 із розширеним промптом. Повнота буде рівна 1, бо немає жодного хибно негативного прикладу. Влучність буде дорівнювати близько 0.86 (60/72). Точність буде рівною 0.62.

Матриця невідповідностей для дискримінатора у комбінованому підході

		Справжній клас	
		Позитивний стан	Негативний стан
Прогнозований клас	Позитивний стан	ІП = 62	ХП = 10
	Негативний стан	ХН = 0	ІН = 28

На рис. 4.9 зображено порівняння точності базової моделі і запропонованих підходів. Можемо зробити висновок, що комбінований підхід має переваги порівняно із іншими, бо якщо користувач зможе перегенерувати приклади, в яких система сумнівається, то з використанням цього підходу його впевненість в результатах буде близько 86%.

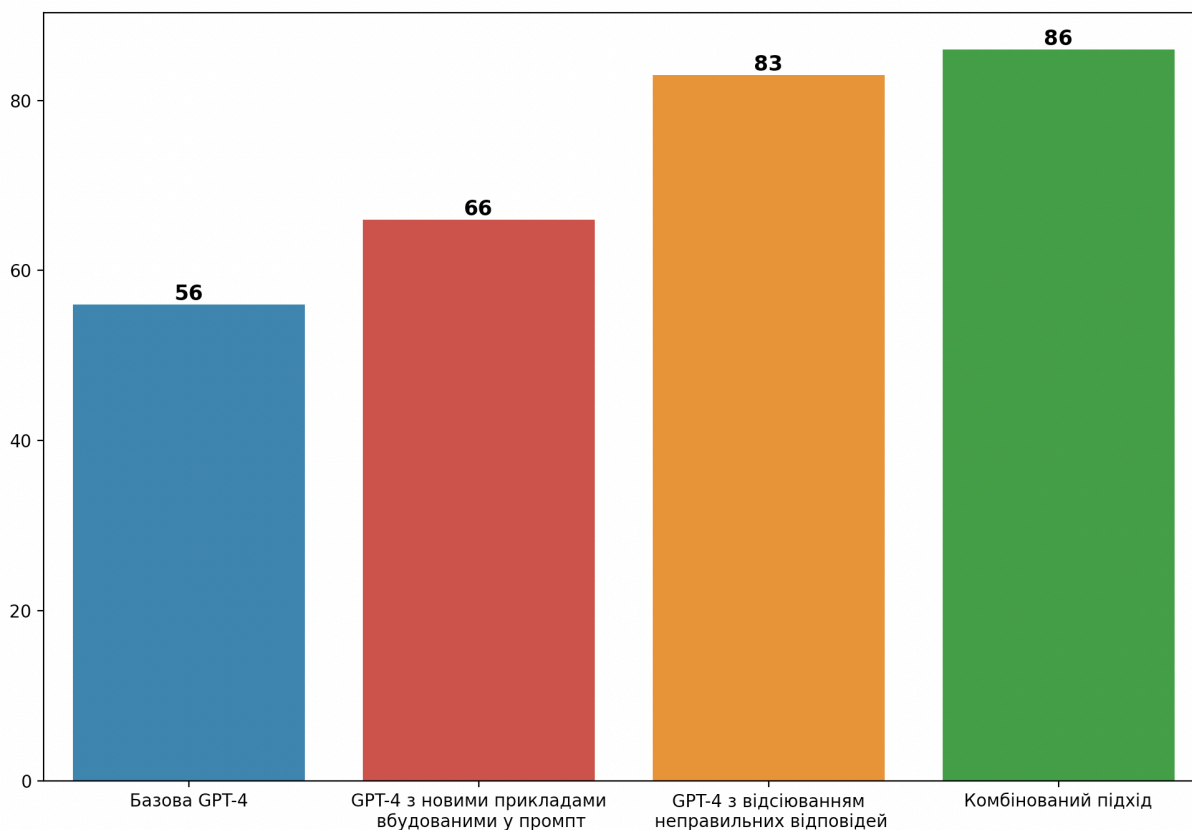


Рис. 4.9. Порівняння точності базової моделі та запропонованих підходів

Для перевірки згенерованого тексту було задано параметр максимальної кількості варіантів відповідей рівний 3. Тобто генератор генерує відповіді допоки дискримінатор не класифікує відповідь як правильну, але генерує максимум 3 рази. Якщо на третій раз дискримінатор також класифікує відповідь як неправильну, то вона видається користувачу, але вказується, що система не впевнена в її правильності. На рис. 4.10 показано скільки спроб робив генератор для генерування відповіді.

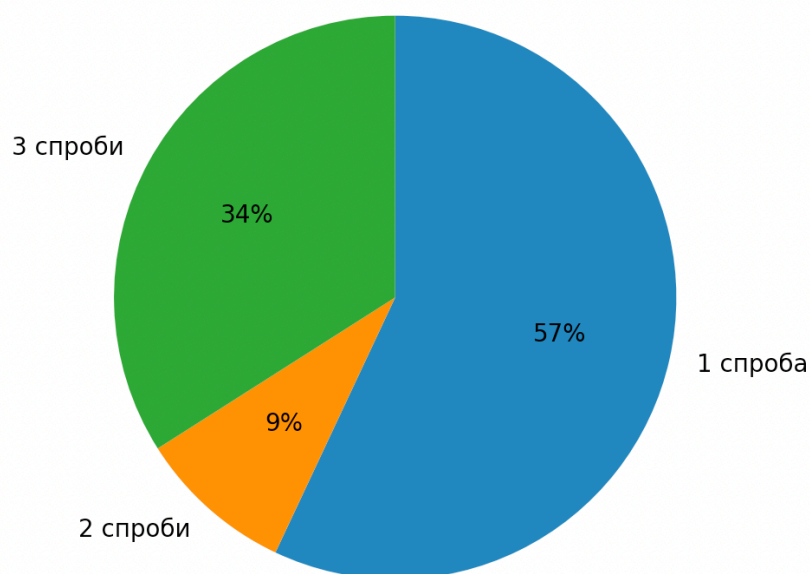


Рис. 4.10. Діаграма кількості спроб для генерування відповіді з використанням дискримінатора для перевірки згенерованого тексту

Якщо побудувати кругову діаграму (рис. 4.11.) для кількості спроб для правильно згенерованих прикладів (ПП), то можемо побачити, що абсолютна більшість правильних прикладів згенеровані з першого разу. З першої спроби було згенеровано 80% правильних прикладів, а з третьої лише 8%. Тобто ліміт у 3 спроби є виправданим, бо збільшуючи його ми не отримаємо набагато більше правильно згенерованих прикладів. При цьому, збільшуючи ліміт, буде зростати час опрацювання запиту та його вартість, якщо будуть використовуватись платні натреновані великі мовні моделі.

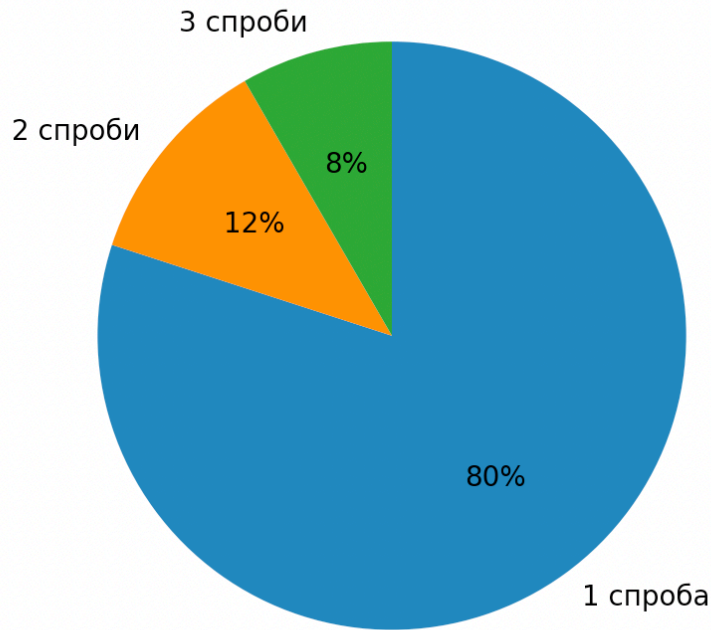


Рис. 4.11. Діаграма кількості спроб для правильно згенерованих прикладів з використанням дискримінатора для перевірки згенерованого тексту

Приклад необроблених результатів дискримінатора зображено на рис. 4.12. Кожен рядок — це окрема пара тверджень і мітка, яка вказує, чи є зв'язок між ними. Горизонтальна лінія «|» є роздільником. Перше висловлювання у кожному рядку визначається користувачем, другий є висловлюванням, яке на думку моделі, впливає з першого. Після цього модель дискримінатор просить визначити, чи є логічний зв'язок між парою речень, тому мітка також генерується моделлю. Пари тверджень потім додаються до нового набору прикладів поведінки остаточної моделі.

Загалом результати показують запропоновані підходи покращують якість генерації тексту. Особливо значущі покращення якості генерації дає більш простий підхід з точки зору кількості кроків з перевіркою згенерованого тексту і повторній генерації, якщо він не відповідає запиту користувача. Складніший підхід також показує кращі результати ніж базова модель і буде особливо корисним у випадках, коли користувач не може надати моделі достатньо прикладів, щоб описати бажану поведінку моделі для генерації тексту.

Певну схожість дані підходи мають із моделями GAN. А саме те, що вони також використовують дискримінатор. Причиною використання дискримінатора є те, що у задачах, що вирішуються, простіше перевірити результат генерації на правильність ніж надати моделі всі можливі результати генерації.

Об'єкт має крила. | Об'єкт може літати. | ні
Об'єкт має здатність мислити або обробляти інформацію. | Об'єкт є істотою або комп'ютером. | так
Об'єкт містить англійські слова. | Об'єкт може бути використаний для вивчення англійської мови. | так
Об'єкт може бути використаний для вивчення англійської мови. | Об'єкт містить англійські слова або фрази. | так
Об'єкт є твариною. | Об'єкт є живим істотою. | так
Об'єкт може плавати у воді. | Об'єкт не втопить у воді. | так
Об'єкт здатен говорити людською мовою. | Об'єкт є розумним істотою. | так
Об'єкт є раритетним автомобілем. | Об'єкт має високу колекційну вартість. | так
Об'єкт експонується у музеї. | Об'єкт є предметом мистецтва або історії. | так
Об'єкт покритий шерстю. | Об'єкт може допомогти зігрітися в холодну погоду. | ні
Об'єкт складається з металу. | Об'єкт проводить електрику. | так
Об'єкт має шість сторін. | Об'єкт є кубом. | так
Об'єкт знає правила дорожнього руху. | Об'єкт має водійський дозвіл. | ні
Об'єкт містить англійські слова або фрази. | Об'єкт може бути використаний для вивчення англійської мови. | так
Об'єкт має високу колекційну вартість. | Об'єкт може бути предметом аукціону. | так
Об'єкт проводить електрику. | Об'єкт може бути визначений як провідник. | так
Об'єкт може показувати погоду. | Об'єкт є метеостанцією. | ні
Об'єкт – це щоденник. | В об'єкті можна робити записи. | так
Об'єкт розміщений у кінотеатрі. | Об'єкт може бути пов'язаний з просуванням фільмів. | так
Об'єкт живе в пустелі. | Об'єкт може витримати високі температури. | так
Об'єкт складається з снігу. | Об'єкт може розтанути при підвищенні температури. | так
Об'єкт не згорить при високих температурах. | Об'єкт є вогнетривким. | так
Об'єкт має доступ до Google Play Store. | Об'єкт є пристроєм на базі Android. | так
Об'єкт є китом. | Об'єкт є морським ссавцем. | так

Рис. 4.12. Приклад необроблених результатів дискримінатора

При аналізі результатів було виявлено, що запропоновані підходи стаються також обійти задачу, тому мають схильність до перефразування і більшого узагальнення. Тобто, замість того щоб згенерувати висловлювання, що впливає із заданого інколи перефразується оригінальне висловлювання. Наприклад, висловлювання “Об'єкт складається із дерева” і “Об'єкт є виробом з дерева” можна сприймати як перефразування, бо більш коректним другим висловлюванням, що впливає з першого було б, наприклад, “Об'єкт складається із природного матеріалу рослинного походження”. Це не суперечить умові задачі у строгому розумінні, але не є бажаною поведінкою для кінцевого користувача. Прикладом більшого узагальнення може бути висловлювання “Об'єкт є прямокутником”, яке було відповіддю на запит “Об'єкт має всі кути 90 градусів”, при тому що базова модель відповіла “Об'єкт є квадратом”. Тобто покращений підхід дав відповідь “прямокутник” замість “квадрат”, що є більш загальною фігурою із точки зору математики.

формулювання слова “об’єкт” у інших відмінках (таблиця 4.5). Також значно частіше вживається дієслово “є”.

Таблиця 4.4

Частота вживання слів у наборі даних для тестування

Слово	Кількість вживань
об’єкт	98
є	20
має	17
знає	4
містить	4
складається	4
знаходиться	4
книгою	3
використаний	3
мовою	3

Таблиця 4.5

Частота вживання слів у тексті згенерованому базовою моделлю

Слово	Кількість вживань
об’єкт	96
є	59
має	6
книгою	4
людиною	3
об’єкті	2
знаходиться	2
англійською	2
мовою	2
англійською	2

Із таблиці нижче видно, що у тексті, що згенерований із вбудовуванням у промпт нових прикладів поведінки також дещо частіше вживаються формулювання слова “об’єкт” у інших відмінках і значно частіше вживається дієслово “є”.

Таблиця 4.6

Частота вживання слів у тексті згенерованому за допомогою підходу із наповненням набору прикладів поведінки дискримінатором

Слово	Кількість вживань
об’єкт	96
є	75
джерелом	3
мовою	3
твариною	3
пристроєм	3
частиною	3
знаходиться	3
має	3
об’єкті	2

Як видно у таблиці 4.7, у тексті, що у тексті, згенерованому за допомогою підходу із відсіюванням результатів дискримінатором спостерігається схожа поведінка як у попередніх двох.

Зобразивши кількість вживань 15-ти найчастіше вживаніших слів у кожному наборі даних (рис. 4.15.), можемо побачити, що є певні схожості між всім згенерованим текстом, якщо порівнювати його із тим, що використовувався для тестування. На рис. 4.15. зображено діаграму частоти вживання найпопулярніших слів у тестовому наборі даних, тексті згенерованому базовою моделлю, тексті згенерованому моделлю із промптом, розширеним новими прикладами поведінки, та у тексті згенерованим з використанням дискримінатора для відсіювання

відповідно. По горизонталі зображено порядковий номер найчастіше вживаного слова у кожному наборі даних (вони різні у різних наборах даних, окрім декількох перших), а по вертикалі скільки раз воно вживалось у відповідному наборі даних.

У тестовому наборі даних є значний спад у другому слові “є”, яке вживається майже так само часто як і третє “має”. Далі слова мають малу і майже рівномірну частоту вживання.

Таблиця 4.7

Частота вживання слів у тексті згенерованому за допомогою підходу із відсіюванням результатів дискримінатором

Слово	Кількість вживань
об’єкт	97
є	53
має	7
книгою	4
освіту	3
підручником	2
об’єктом	2
людиною	2
твариною	2
базі	2

У згенерованому тексті значний спад є лише після другого слова “є”. Тобто це слово значно частіше вживається у згенерованому тексті. Це може бути пояснене тим, що простіше згенерувати висловлювання, які вказують чим об’єкт є, ніж що він має, чи з чого складається. При цьому модель, якій вбудовувались у промпт нові приклади поведінки, навіть ще частіше використовує це дієслово ніж інші. Можемо зробити висновок, що вона старається знайти оптимальні формулювання, що будуть вирішувати задачу жертвуючи різноманітністю формулювань.

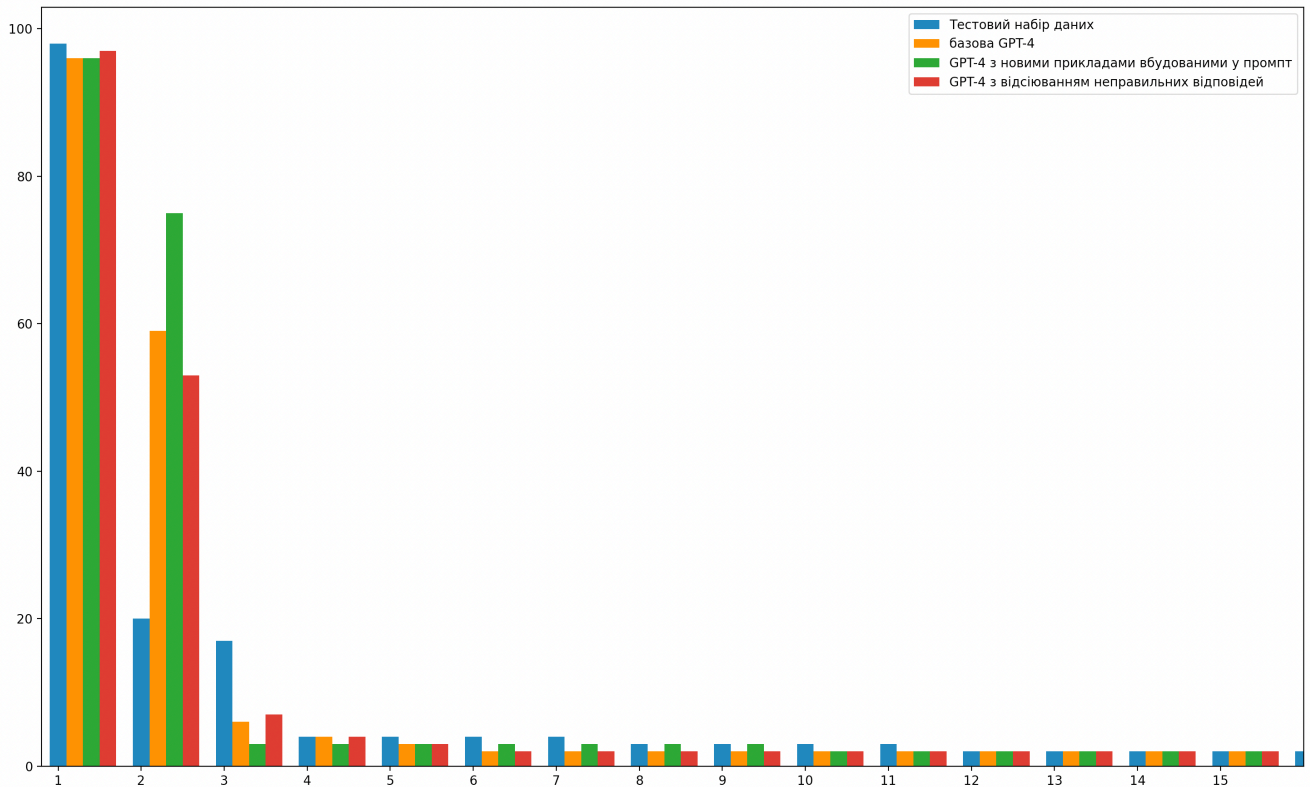


Рис. 4.15. Діаграма частоти вживання найпопулярніших слів наборах даних

Висновки до розділу 4

Описано постановку початкового експерименту із використанням 80-ти прикладів і рандомізованим розбиттям на дві групи для перевірки базової моделі GPT-4 та моделі із промптом, розширеною новими прикладами поведінки. Після проведення початкового експерименту було вирішено збільшити набір даних для тестування, а також виділити окремий набір даних для тестування всіх моделей, бо в такому випадку ми імітуємо поведінку моделі справжніх умовах використання.

Було реалізовано підхід із використанням дискримінатора для наповнення набору прикладів поведінки, а також підхід із використанням дискримінатора для відсіювання неправильних відповідей. Здійснено порівняння точності обидвох підходів та запропоновано метод, що використовує ці обидва підходи до генерування тексту послідовно через кращу точність у справжніх умовах використання. Також було здійснено описовий аналіз тестового набору даних і згенерованого тексту.

ВИСНОВКИ

У дисертаційній роботі розв'язано актуальне науково-прикладне завдання розроблення методів та засобів багатоетапного генерування семантично коректних текстів. Дослідження, що були проведені в рамках даної дисертаційної роботи, мали на меті дослідити та вирішити проблеми пов'язані із консистентністю знань великих мовних моделей та узгодженням таких моделей із запитам користувачів. Було розроблено метод з використанням дискримінатора для перевірки згенерованого тексту, мета якого покращити якість згенерованого тексту за допомогою великих мовних моделей.

Було виявлено, що великі мовні моделі у задачі логічного виведення мають проблеми із транзитивністю. Тобто, якщо з висловлювання А випливає висловлювання Б, а з Б випливає В, то з А повинно випливати В. У зв'язку з тим, що імплікація має властивість транзитивності, то ця властивість повинна виконуватись у задачі логічного виведення, але інколи великі мовні моделі допускають в ній помилки.

Тому було вирішено сформувати із існуючих наборів даних ланцюжки імплікації і дотренувати на них модель. В процесі роботи над цим було виявлено, що з існуючих наборів даних неможливо сформувати достатній набір даних для дотреновування моделі. Також, під час тестування на невеликому наборі даних виявлено, що модель допускала помилки тільки у тих прикладах, у яких допускала помилки в оригінальному наборі даних.

Через розвиток генеративних мовних моделей появилась можливість не використовувати існуючі набори даних, а згенерувати свій. Тобто згенерувати свій набір даних із ланцюжками імплікацій. Це реалізується через передавання відповіді моделі повторно їй на вхід, запитуючи згенерувати висловлювання, що впливає із заданого. В процесі формування такого набору даних було виявлено, що модель допускає помилки генеруючи висловлювання, що впливають із заданого. Через це було вирішено після генерування тексту перевіряти, чи відповідає він запиту користувача.

Загалом перевірка, чи згенерований текст відповідає запиту користувача може бути узагальнена до інших задач, але було вирішено протестувати

запропонований підхід на задачі логічного виведення. Тобто у запропонованому підході модель генерує текст, а потім він разом із запитом користувача передається їй на вхід, і її задачею є класифікувати, чи відповідає згенерований текст запиту, чи ні. Розмічені приклади після класифікації добавляються у набір прикладів поведінки, які вбудовуються у промпт моделі.

Це дозволило покращити точність у задачі логічного виведення, але в кінцевому методі було вирішено об'єднати обидва алгоритми генерування тексту, бо такий підхід показував ще кращі показники. Тобто в остаточному підході у промпт моделі вбудовуються нові приклади поведінки, після чого вона генерує відповідь на запит користувача. Наступним кроком відбувається класифікація, чи вона відповідає запиту користувача. І якщо відповідь не є семантично правильною у контексті запиту користувача, то відповідь генерується повторно. Кроки генерації і класифікації повторюються доти, доки відповідь не буде класифікована як правильна, або не буде перевищено ліміт генерацій.

Під час дослідження вирішено такі завдання та отримано такі результати:

1. Здійснено аналіз підходів для опрацюванні природної мови та обґрунтовано стратегію досліджень. У зв'язку із тим, що попередньо натреновані моделі показують зараз найкращі результати, було вирішено використовувати конструювання промпту великих мовних моделей замість зміни архітектури чи тренування моделі.
2. Розроблено формальний опис моделі процесу генерування текстового контенту.
3. Розроблено метод, що включає два алгоритми генерування тексту. Перший алгоритм полягає у використанні дискримінатора для формування набору прикладів поведінки, що буде вбудовуватись у промпт моделі. Другий полягає у використанні дискримінатора для відсіювання неправильних відповідей. Під час реалізації було обрано конструювання промпту для покращення точності генерування тексту, у зв'язку з тим, що є декілька обмежень попередньо натренованої моделі, що використовувалась. Перше обмеження полягає у тому, що на даний

момент дотреновування моделі GPT-4 є недоступним. Друге, в у тому, що інші методи вимагають більшої кількості даних для тренування.

4. Здійснено програмну реалізацію системи багатоетапного генерування семантично коректних текстів.
5. На основі експериментальних досліджень було проведено аналіз точності запропонованого методу. Було вирішено використовувати дискримінатор для наповнення набору прикладів поведінки, що будуть вбудовуватись у промпт, а також для відсіювання неправильно згенерованих відповідей, бо такий комбінований метод показував кращі результати, і дасть можливість відкинути приклади, в яких система сумнівається.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AHD Staff (2022). "Semantics". American Heritage Dictionary. Harper Collins.
2. Saeed, John I. (2009). *Semantics* (3rd ed.). Wiley-Blackwell. ISBN 978-1-405-15639-4.
3. Crimmins, Mark (1998). "Semantics". *Routledge Encyclopedia of Philosophy*. Routledge. doi:10.4324/9780415249126-U036-1. ISBN 978-0-415-25069-6.
4. Google AI blog, <https://ai.googleblog.com/2018/05/advances-in-semantic-textual-similarity.html>.
5. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. ArXiv. /abs/1310.4546
6. Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning Semantic Textual Similarity from Conversations. In *Proceedings of the Third Workshop on Representation Learning for NLP*, pages 164–174, Melbourne, Australia. Association for Computational Linguistics.
7. Lebet, R., & Collobert, R. (2014). Word Embeddings through Hellinger PCA. In S. Wintner, S. Goldwater, & S. Riezler (Eds.), *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 482–490). Stroudsburg, PA, USA: Association for Computational Linguistics.
8. P. Zdebskyi, V. Lytvyn, Y. Burov, Z. Rybchak, P. Kravets, O. Lozynska, R. Holoshchuk, S. Kubinska, A. Dmytriv. Intelligent System for Semantically Similar Sentences Identification and Generation Based on Machine Learning Methods. *CEUR Workshop Proceedings*. 2020. Vol. 2604. P. 317–346.
9. Williams, A., Nangia, N., & Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1112–1122). Stroudsburg, PA, USA: Association for Computational Linguistics.

10. P. Zdebskyi, A. Berko, V. Vysotska. Investigation of Transitivity Relation in Natural Language Inference. CEUR Workshop Proceedings. 2023. Vol. 3396. P. 334–345.
11. P. Zdebskyi, A. Berko, V. Vysotska. Information Technology for Finding Answers to Questions from Open Web Resources. International Scientific and Technical Conference “Computer Sciences and Information Technologies”. 2023. IEEE 18th International Conference on Computer Science and Information Technologies CSIT 2023, Lviv, Ukraine, 19-21 Oct. 2023. P. 1–7.
12. Huang, J., & Chang, K. C. (2022). Towards Reasoning in Large Language Models: A Survey. ArXiv. /abs/2212.10403
13. Sheng, Y., Li, L., Wang, Y., & Zeng, D. (2024). Integrating language models with symbolic formulas for first-order logic reasoning. ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE.
14. Richards, D., & Hamburger, H. (2017). Logic and language models for Computer Science. World Scientific Publishing Company.
15. Boghossian, P., & Peacocke, C. (Eds.). (2000). New Essays on the A Priori. Oxford, England: Clarendon Press.
16. Raj, H., Rosati, D., & Majumdar, S. (2022). Measuring Reliability of Large Language Models through Semantic Consistency. ArXiv. /abs/2211.05853
17. Zhong, L., & Wang, Z. (2023). Can ChatGPT replace StackOverflow? A Study on Robustness and Reliability of Large Language Model Code Generation. ArXiv. /abs/2308.10335
18. Atkins, Sue & Clear, Jeremy & Ostler, Nicholas. (1992). Corpus Design Criteria. *Literary and Linguistic Computing*. 7. 1-16. 10.1093/lc/7.1.1.
19. Biber, D. (1994). Representativeness in Corpus Design. In *Current Issues in Computational Linguistics: In Honour of Don Walker* (pp. 377–407). Dordrecht: Springer Netherlands.
20. Kennedy, G. (2014). *An Introduction to Corpus Linguistics*. Routledge.
21. van Benthem, Johan, & Ter Meulen, A. G. (Eds.). (1996). *Handbook of logic and language*. Oxford, England: North-Holland.

22. Flew, A., & Ryle, G. (1993). *Essays on logic and language: First series/second series* (Antony Flew, Ed.). London, England: Gregg Revivals.
23. Конверський А.Є. *Логіка: Підручник. – 2-ге вид. Виправлене.* К.: ВПЦ “Київський університет”, Філософський факультет. 2017.
24. Togelius, J., & Yannakakis, G. N. (2023). *Choose Your Weapon: Survival Strategies for Depressed AI Academics.* ArXiv. /abs/2304.06035
25. Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). *Natural language processing: state of the art, current trends and challenges.* *Multimedia Tools and Applications*, 82(3), 3713–3744.
26. Cios, K. J. (2018). *Deep neural networks—A brief history.* In *Advances in Data Analysis with Computational Intelligence Methods* (pp. 183–200). Cham: Springer International Publishing.
27. P. Zdebskyi, V. Vysotska, R. Peleshchak, I. Peleshchak, A. Demchuk, M. Krylyshyn *An application development for recognizing of view in order to control the mouse pointer.* *CEUR Workshop Proceedings.* 2019. Vol. 2386. P. 55–74.
28. Chui, C. K., Li, X., & Mhaskar, H. N. (1996). *Limitations of the approximation capabilities of neural networks with one hidden layer.* *Advances in Computational Mathematics*, 5(1), 233–243.
29. Lu, Y., & Lu, J. (2020). *A universal approximation theorem of deep neural networks for expressing probability distributions.* *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 3094–3105. Red Hook, NY, USA: Curran Associates Inc.
30. Sherstinsky, A. (2020). *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network.* *Physica D. Nonlinear Phenomena*, 404(132306), 132306.
31. Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2016). *Very deep convolutional networks for natural language processing.* *arXiv preprint arXiv:1606.01781*, 2(1).
32. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter.* ArXiv. /abs/1910.01108

33. Smagulova, K., & James, A. P. (2019). A survey on LSTM memristive neural network architectures and applications. *The European Physical Journal. Special Topics*, 228(10), 2313–2324.
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *ArXiv. /abs/1706.03762*
35. Singh, S., & Mahmood, A. (2021). The NLP cookbook: Modern recipes for transformer based deep learning architectures. *IEEE Access: Practical Innovations, Open Solutions*, 9, 68675–68702.
36. Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv. /abs/1810.04805*
37. Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... & Wen, J. R. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.
38. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., . . . Zoph, B. (2023). GPT-4 Technical Report. *ArXiv. /abs/2303.08774*
39. Church, K. W., Chen, Z., & Ma, Y. (2021). Emerging trends: A gentle introduction to fine-tuning. *Natural Language Engineering*, 27(6), 763–778.
40. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *ArXiv. /abs/1910.10683*
41. Young, T., Hazarika, D., Poria, S., & Cambria, E. (2017). Recent Trends in Deep Learning Based Natural Language Processing. *ArXiv. /abs/1708.02709*
42. Chakraborty, S., Tomsett, R., Raghavendra, R., Harborne, D., Alzantot, M., Cerutti, F., Gurram, P. (2017). Interpretability of deep learning models: A survey of results. 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI). IEEE.
43. Mufwene, S. S., Coupé, C., & Pellegrino, F. (2017). Complexity in Language: A Multifaceted Phenomenon. In S. S. Mufwene, C. Coupé, & F. Pellegrino

- (Eds.), *Complexity in Language: Developmental and Evolutionary Perspectives* (pp. 1–29). chapter, Cambridge: Cambridge University Press.
44. Binder, P.-M., & Smith, K. (Eds.). (2013). *The language phenomenon: Human communication from milliseconds to Millennia* (2013th ed.). Berlin, Germany: Springer.
45. Dahl, O. (2004). *The growth and maintenance of linguistic complexity*. Amsterdam, Netherlands: John Benjamins Publishing.
46. Zhang, Y., Jin, R., & Zhou, Z.-H. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1–4), 43–52.
47. Qader, W. A., Ameen, M. M., & Ahmed, B. I. (2019). An overview of bag of words; Importance, implementation, applications, and challenges. 2019 International Engineering Conference (IEC). IEEE.
48. Aizawa, A. (2003). An information-theoretic perspective of tf-idf measures. *Information Processing & Management*, 39(1), 45–65.
49. Ramos, J.E. (2003). Using TF-IDF to Determine Word Relevance in Document Queries.
50. Qaiser, S., & Ali, R. (2018). Text mining: Use of TF-IDF to examine the relevance of words to documents. *International Journal of Computer Applications*, 181(1), 25–29.
51. Church, K. W. (2017). Word2Vec. *Natural Language Engineering*, 23(1), 155–162.
52. Rong, X. (2014). Word2vec Parameter Learning Explained. ArXiv. /abs/1411.2738
53. Ma, L., & Zhang, Y. (2015). Using Word2Vec to process big text data. 2015 IEEE International Conference on Big Data (Big Data), 2895–2897. IEEE.
54. Li, B., Zhou, H., He, J., Wang, M., Yang, Y., & Li, L. (2020). On the sentence embeddings from pre-trained language models. In B. Webber, T. Cohn, Y. He, & Y. Liu (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 9119–9130). Stroudsburg, PA, USA: Association for Computational Linguistics.
55. Stephen Read. (1995). *Thinking about logic: An introduction to the philosophy of logic*. London, England: Oxford University Press.

56. Henle, M. (1962). On the relation between logic and thinking. *Psychological review*, 69(4), 366.
57. Kelley, D. (2013). *The art of reasoning: An introduction to logic and critical thinking* (4th ed.). New York, NY: WW Norton.
58. Logic models used to enhance critical thinking. (2006). *The Journal of Nursing Education*, 45(6), 220–227.
59. Alba-Cepero, J., Cabezas-Puerto, M., López-Batista, V. F., & Moreno-Montero, Á. M. (2022, July). Bias Analysis on Twitter. In *International Conference on Disruptive Technologies, Tech Ethics and Artificial Intelligence* (pp. 131-142). Cham: Springer International Publishing.
60. Lopez, P. (2023). Power and resistance in the twitter bias discourse. *2023 ACM Conference on Fairness, Accountability, and Transparency*. New York, NY, USA: ACM.
61. Ferrer, X., Van Nuenen, T., Such, J. M., & Criado, N. (2020). Discovering and Categorising Language Biases in Reddit. *ArXiv*. /abs/2008.02754
62. Glenski, M., & Weninger, T. (2017, July). Predicting user-interactions on reddit. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017* (pp. 609-612).
63. Hadi, M. U., Tashi, Q. al, Qureshi, R., Shah, A., Muneer, A., Irfan, M., Mirjalili, S. (2023). A survey on Large Language Models: Applications, challenges, limitations, and practical usage.
64. Gubelmann, R. (2023). A Loosely Wittgensteinian Conception of the Linguistic Understanding of Large Language Models like BERT, GPT-3, and ChatGPT. *Grazer Philosophische Studien*, 99(4), 485-523.
65. P. Zdebskyi, A. Berko, V. Vysotska, S. Voloshyn, S. Chyrun, V. Schuchmann Framework for Improving the Effectiveness of Discussions at English-Language Articles Analysis. *International Scientific and Technical Conference “Advanced Information and Communication Technologies”*. 2023. *IEEE 5th International Conference on Advanced Information and Communication Technologies AICT 2023*, Lviv, Ukraine, 21-25 Nov. 2023. P. 127–132.

66. Ji, J., Qiu, T., Chen, B., Zhang, B., Lou, H., Wang, K., Duan, Y., He, Z., Zhou, J., Zhang, Z., Zeng, F., Ng, K. Y., Dai, J., Pan, X., Lei, Y., Xu, H., Tse, B., Fu, J., McAleer, S., Gao, W. (2023). AI Alignment: A Comprehensive Survey. ArXiv. /abs/2310.19852
67. Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big?. Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. New York, NY, USA: ACM.
68. Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. ArXiv. /abs/1706.03741
69. Our approach to alignment research: <https://openai.com/blog/our-approach-to-alignment-research>
70. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R. (2022). Training language models to follow instructions with human feedback. ArXiv. /abs/2203.02155
71. Solaiman, I., & Dennison, C. (2021). Process for Adapting Language Models to Society (PALMS) with Values-Targeted Datasets. ArXiv. /abs/2106.10328
72. Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., & Thomaz, A. L. (2013). Policy shaping: Integrating human feedback with reinforcement learning. Advances in neural information processing systems, 26.
73. Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., Elhage, N., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Kaplan, J. (2022). Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. ArXiv. /abs/2204.05862
74. Arumugam, D., Lee, J. K., Saskin, S., & Littman, M. L. (2019). Deep Reinforcement Learning from Policy-Dependent Human Feedback. ArXiv. /abs/1902.04257
75. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. ArXiv. /abs/1707.06347

76. Duan, Y., Chen, X., Houthoofd, R., Schulman, J., & Abbeel, P. (2016). Benchmarking Deep Reinforcement Learning for Continuous Control. ArXiv. /abs/1604.06778
77. Kakade, S., & Langford, J. (2002). Approximately optimal approximate reinforcement learning. Proceedings of the Nineteenth International Conference on Machine Learning, 267–274. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
78. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533.
79. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. ArXiv. /abs/1602.01783
80. Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015). Trust Region Policy Optimization. ArXiv. /abs/1502.05477
81. Szita, I., & Lőrincz, A. (2006). Learning Tetris using the noisy cross-entropy method. Neural Computation, 18(12), pp. 2936–2941.
82. Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., & Cobbe, K. (2023). Let's Verify Step by Step. ArXiv. /abs/2305.20050
83. Askell, A., Bai, Y., Chen, A., Drain, D., Ganguli, D., Henighan, T., Jones, A., Joseph, N., Mann, B., DasSarma, N., Elhage, N., Hernandez, D., Kernion, J., Ndousse, K., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Kaplan, J. (2021). A General Language Assistant as a Laboratory for Alignment. ArXiv. /abs/2112.00861
84. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021). Training Verifiers to Solve Math Word Problems. ArXiv. /abs/2110.14168
85. Creswell, A., Shanahan, M., & Higgins, I. (2022). Selection-Inference: Exploiting Large Language Models for Interpretable Logical Reasoning. ArXiv. /abs/2205.09712

86. Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., & Steinhardt, J. (2021). Measuring Mathematical Problem Solving With the MATH Dataset. ArXiv. /abs/2103.03874
87. Ignat, O., Jin, Z., Abzaliev, A., Biester, L., Castro, S., Deng, N., Gao, X., Gunal, A., He, J., Kazemi, A., Khalifa, M., Koh, N., Lee, A., Liu, S., Min, D. J., Mori, S., Nwatu, J., Shen, S., Wang, Z., Mihalcea, R. (2023). Has It All Been Solved? Open NLP Research Questions Not Solved by Large Language Models. ArXiv. /abs/2305.12544
88. Frieder, S., Pinchetti, L., Griffiths, R. R., Salvatori, T., Lukasiewicz, T., Petersen, P., & Berner, J. (2024). Mathematical capabilities of chatgpt. *Advances in Neural Information Processing Systems*, 36.
89. Здебський П., Берко А., Чирун Л. Інформаційна система видобування інформації з відкритих WEB-ресурсів. Вісник Національного університету “Львівська політехніка”. Серія: “Інформаційні системи та мережі”. 2022. Вип. 12. С. 141–168.
90. Levesque, H., Davis, E., & Morgenstern, L. (2012). The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.
91. Берко А., Здебський П., Висоцька В. Метод формування набору даних для перевірки якості вивчення мовними моделями транзитивного відношення у контексті задачі логічного висновку. Вісник Національного університету “Львівська політехніка”. Серія: “Інформаційні системи та мережі”. 2023. Вип. 14. С. 46–60.
92. Здебський П., Берко А. Метод покращення якості генерування тексту за рахунок повторного передавання згенерованого тексту на модель. Вісник Хмельницького національного університету. Серія: “Технічні науки”. 2024. Вип. 1, т. 331. С. 259–263.
93. Ngo, R., Chan, L., & Mindermann, S. (2022). The Alignment Problem from a Deep Learning Perspective. ArXiv. /abs/2209.00626
94. Langosco, L., Koch, J., Sharkey, L., Pfau, J., Orseau, L., & Krueger, D. Goal Misgeneralization in Deep Reinforcement Learning. *Proceedings of the 39th*

International Conference on Machine Learning. International Conference on Machine Learning. PMLR. pp. 12004–12019.

95. Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large Language Models are Zero-Shot Reasoners. ArXiv. /abs/2205.11916
96. Zelikman, E., Wu, Y., Mu, J., & Goodman, N. D. (2022). STaR: Bootstrapping Reasoning With Reasoning. ArXiv. /abs/2203.14465
97. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2022). Self-Consistency Improves Chain of Thought Reasoning in Language Models. ArXiv. /abs/2203.11171
98. Здебський П., Берко А. Перевірка тексту після генерації за допомогою великих мовних моделей для фільтрування неправильних відповідей. Наукові праці Вінницького національного технічного університету. 2024. Вип. 1.