

СИСТЕМА НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
(КПІ ім. Ігоря Сікорського)

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

«До захисту допущено»

В. о. завідувачки кафедри БМК

_____ Світлана АЛХІМОВА

“___” червня 2025р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Комп’ютерні технології в біології та медицині»
спеціальності 122 «Комп’ютерні науки»

на тему: Інформаційна система детекції галюцинацій у результатах
діагностичних висновків генеративного штучного інтелекту

Виконав: студент IV курсу, групи БС-11

БУТКО ВІТАЛІЙ ДМИТРОВИЧ

(прізвище, ім’я, по батькові)



(підпис)

Наковий керівник: к.т.н., доцент каф. БМК

Павлов Володимир Анатолійович

(посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові)



(підпис)

Консультант з розділів дипломної роботи:

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ім’я, по батькові)

(підпис)

Рецензент: к.т.н., доцент каф. біомедичної інженерії

Богомолов Микола Федорович

(посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові)

(підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів без
відповідних посилань.

Студент
(підпис)



Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет біомедичної інженерії
Кафедра біомедичної кібернетики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні технології в біології та медицині»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри БМК

_____ Світлана АЛХІМОВА

«__»__ __ 2025 р.

ЗАВДАННЯ
на дипломну роботу студенту

БУТКУ ВІТАЛІЮ ДМИТРОВИЧУ

(прізвище, ім'я, по батькові)

1. Тема роботи _____ **Інформаційна система детекції галюцинацій у**
результатах діагностичних висновків генеративного штучного інтелекту

Керівник роботи

Павлов Володимир Анатолійович, к.т.н., доцент каф. БМК

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_26_»_травня_ 2025 р. №_1758-с_

22-23 травня 2025р.

2. Термін подання студентом роботи

3. Вихідні дані до роботи: *Відкриті дані медичних аналізів пацієнтів з їх інтерпретаціями. Пре треновані базові великі мовні моделі (Llama 3.2 8b та подібні).*

4. Зміст роботи: *аналітичний огляд методів досліджень та засобів протидії галюцинаціям у великих мовних моделях, аналіз та вибір оптимального метода детекції галюцинацій, програмна реалізація, висновки, список використаних літературних джерел.*

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): *14 слайдів, 22 рисунки, 0 таблиць.*

6. Дата видачі завдання **06 квітня 2025 року.**

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримати завдання за темою ДР на практику	До 24.12.2024р.	<i>виконано</i>
2	Переддипломна практика. Виконання практичної частини ДР та додаткових розділів	1-4 тиждень за графіком практики	<i>виконано</i>
3	Виконання основних розділів ДР (Вступ, Аналітичний огляд існуючих методів детекції галюцинацій, Засоби реалізації інформаційної системи детекції галюцинацій, Обґрунтування вибору програмного середовища, Програмна реалізація та методика роботи інформаційної системи детекції галюцинацій у великих мовних моделях)	Протягом усієї практики	<i>виконано</i>
4	Перевірка ДР науковим керівником	до 20.05.2025	<i>виконано</i>
5	Подання в електронному вигляді ДР на перевірку нормоконтролера та плагіат (StrikePlagiarism.com). Доопрацювання ДР	Не пізніше 23.05.2025	<i>виконано</i>
6	Отримати відгук від керівника ДР	Не пізніше 05.06.2025	<i>виконано</i>
7	Надати пакету документів по ДР та супровідних до неї документів на засідання кафедри		<i>виконано</i>
8	Подання ДР рецензенту. Отримання рецензії.	09.06 – 13.06.2025	<i>виконано</i>
9	Захист ДР в ЕК	16.06 - 21.06.2025	

Студент



(підпис)

Віталій БУТКО

(ім'я, ПРІЗВИЩЕ)

Керівник ДР



(підпис)

Володимир ПАВЛОВ

(ім'я, ПРІЗВИЩЕ)

Нормоконтролер

(підпис)

Галина КОРНІЄНКО

(ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота за темою «Інформаційна система детекції галюцинацій у результатах діагностичних висновків генеративного штучного інтелекту» виконана студентом кафедри біомедичної кібернетики ФБМІ Бутком Віталієм Дмитровичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та складається зі: вступу; 4 розділів (аналітичний огляд існуючих методів детекції галюцинацій, засоби реалізації інформаційної системи детекції галюцинацій, обґрунтування вибору програмного середовища, програмна реалізація та методика роботи інформаційної системи детекції галюцинацій у великих мовних моделях), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 34 джерела. Загальний обсяг роботи 84 сторінки.

Актуальність теми полягає в необхідності створення ефективних методів детекції галюцинацій у медичних діагностичних висновках, оскільки ШІ присутній в усіх сферах нашого життя. Медичний домен створює умови, в яких ціна помилки є надзвичайно високою, а в існуючих системах методів перевірки галюцинацій або взагалі немає, або вони малоефективні. Тому розробка ефективної системи контролю правдивості відповідей LLM стає критично важливою для безпечного впровадження асистованих рішень у сферу охорони здоров'я.

Мета і завдання роботи. Метою роботи є підвищення точності та надійності формування діагностичних висновків у медицині за допомогою впровадження системи протидії галюцинаціям у результатах генеративного штучного інтелекту. Це дозволить досягти високих показників достовірності автоматизованих рішень та мінімізувати ризики виникнення помилкових або хибних висновків у клінічній практиці.

Сформульовано наступні задачі:

1. Провести аналіз сучасних методів детекції галюцинацій в LLM;
2. Вивчити теоретичний матеріал щодо основних методів детекції галюцинацій;
3. Знайти та підготувати набір даних з клінічними даними пацієнтів.

4. Практично реалізувати 3 методи детекції галюцинацій.
5. Розробити та реалізувати власну покращену версію методу аналізу активацій.
6. Розробити систему тестування методів детекції галюцинацій.
7. Протестувати реалізовані методи та зробити висновки.
8. Розробити графічний інтерфейс для продукту.

Використані методи. В процесі проведеного дослідження було визначено та обґрунтовано використання: мови програмування Python як оптимального вибору завдяки першокласній підтримці бібліотеки Transformers, потужній ML/NLP екосистемі та швидкості розробки; ключових бібліотек, таких як accelerate, instructor, torch, transformers, pandas, openai, numpy для основної функціональності системи та допоміжних бібліотек colorlog, datasets, ipykernel, matplotlib, openpyxl, pydantic-settings, python-frontmatter, weave, streamlit, jinja2 для розширення можливостей; моделі Mistral-Small-24B-Instruct-2501 з архітектурою трансформер, що містить 24 мільярди параметрів та демонструє оптимальний баланс між обчислювальною ефективністю та якістю генерації відповідей; хмарного сервісу RunPod з конфігурацією 4 × NVIDIA RTX 4080 SUPER графічних прискорювачі, 32 CPU ядра, 166 GB RAM для розгортання та використання моделі; моделі GPT-4.1 як критика для перевірки наявності галюцинацій у відповідях LLM; спеціалізованого датасету MSDiagnosis, який є репрезентативною вибіркою реальних клінічних випадків у педіатричній практиці.

Отримані результати. Результати були отримані на підставі дослідження системи детекції галюцинацій у медичних текстах, згенерованих великими мовними моделями. Було розроблено і навчено спеціалізований класифікатор HallucinationDetector, що базується на аналізі внутрішніх станів мовної моделі, який продемонстрував найвищу точність (83.3%) та F1-score (84.7%) серед усіх досліджених методів. Реалізовано та протестовано два альтернативні методи виявлення галюцинацій: аналіз logprobs з найменшою латентністю (12 мс) та SelfCheckGPT-NLI з можливістю роботи з моделями як з "чорним ящиком". Створено інтерактивний веб-додаток на базі Streamlit для демонстрації роботи

системи детекції галюцинацій. Отриманий результат може мати практичне застосування в системах підтримки прийняття медичних рішень для забезпечення достовірності автоматично згенерованих відповідей, особливо в медичному контексті, де ціна помилки надзвичайно висока, а достовірність інформації є критичною для здоров'я пацієнтів.

Публікації. Не заплановано.

Ключові слова. Великі мовні моделі, детекція галюцинацій, внутрішні активації LLM, лог-ймовірності, SelfCheckGPT-NLI.

Бібліографічний опис ДР

Бутко В. Д. Інформаційна система детекції галюцинацій у результатах діагностичних висновків генеративного штучного інтелекту : дипломна роб. бакалавра : 122 Комп'ютері науки / Бутко Віталій Дмитрович. – Київ, 2025. – 84 с.

ABSTRACT

Diploma thesis on the topic "Information System for Hallucination Detection in Diagnostic Conclusions Generated by Artificial Intelligence" completed by the student of the Department of Biomedical Cybernetics of FBME *Butko Vitaliy Dmytrovych in specialty 122 "Computer Science" under the educational-professional program "Computer Technologies in Biology and Medicine" and *consists of: introduction; 4 chapters (analytical review of existing hallucination detection methods, implementation tools for the hallucination detection information system, justification of the software environment selection, software implementation and methodology of the hallucination detection information system in large language models), conclusions for each of these chapters; general conclusions; list of references, which includes 34 sources. The total volume of the work is 84 pages.

Relevance of the topic. The relevance of the topic lies in the necessity to create effective methods for detecting hallucinations in medical diagnostic conclusions, as AI is present in all spheres of our lives. The medical domain creates conditions in which the cost of an error is extremely high, and in existing systems, methods for verifying hallucinations are either completely absent or ineffective. Therefore, the development of an effective system for controlling the truthfulness of LLM responses becomes critically important for the safe implementation of assisted solutions in healthcare.

The purpose and objectives of the work. The purpose of the work is to increase the accuracy and reliability of diagnostic conclusions in medicine through the implementation of a system to counter hallucinations in the results of generative artificial intelligence. This will allow achieving high indicators of reliability in automated decisions and minimizing the risks of erroneous or false conclusions in clinical practice.

The following tasks were formulated:

1. Conduct an analysis of modern methods of hallucination detection in LLM;
2. Study theoretical material on the main methods of hallucination detection;
3. Find and prepare a dataset with clinical patient data.
4. Practically implement 3 methods of hallucination detection.
5. Develop and implement an improved version of the activation analysis method.

6. Develop a system for testing hallucination detection methods.
7. Test the implemented methods and draw conclusions.
8. Develop a graphical interface for the product.

Methods used. In the process of the conducted research, the following were determined and justified: the use of Python programming language as an optimal choice due to first-class support for the Transformers library, powerful ML/NLP ecosystem, and development speed; key libraries such as accelerate, instructor, torch, transformers, pandas, openai, numpy for the main functionality of the system and auxiliary libraries colorlog, datasets, ipykernel, matplotlib, openpyxl, pydantic-settings, python-frontmatter, weave, streamlit, jinja2 for expanding capabilities; the Mistral-Small-24B-Instruct-2501 model with transformer architecture, containing 24 billion parameters and demonstrating an optimal balance between computational efficiency and response generation quality; the RunPod cloud service with a configuration of $4 \times$ NVIDIA RTX 4080 SUPER graphics accelerators, 32 CPU cores, 166 GB RAM for deploying and using the model; the GPT-4.1 model as a critic for checking for hallucinations in LLM responses; the specialized MSDiagnosis dataset, which is a representative sample of real clinical cases in pediatric practice.

Results. The results were obtained based on the study of a hallucination detection system in medical texts generated by large language models. A specialized classifier HallucinationDetector was developed and trained, based on the analysis of internal states of the language model, which demonstrated the highest accuracy (83.3%) and F1-score (84.7%) among all the methods studied. Two alternative methods of hallucination detection were implemented and tested: logprobs analysis with the lowest latency (12 ms) and SelfCheckGPT-NLI with the ability to work with models as a "black box". An interactive web application based on Streamlit was created to demonstrate the operation of the hallucination detection system. The obtained result can have practical application in medical decision support systems to ensure the reliability of automatically generated responses, especially in the medical context, where the cost of an error is extremely high, and the reliability of information is critical for patient health.

Publications. Not planned.

Keywords. Large language models, hallucination detection, internal activations of LLM, log-probabilities, SelfCheckGPT-NLI.

ЗМІСТ

ВСТУП.....	13
РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ДЕТЕКЦІЇ ГАЛЮЦІНАЦІЙ.....	16
1.1. Ймовірнісні та лог-пробабілістичні методи.....	16
1.2. Методи на основі багаторазової генерації та самоперевірки	17
1.3. Аналіз внутрішніх станів моделей	17
1.4. Методи з використанням зовнішніх джерел знань.....	18
1.5. Гібридні та спеціалізовані підходи	19
Висновки до розділу 1	20
РОЗДІЛ 2 ЗАСОБИ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЕТЕКЦІЇ ГАЛЮЦІНАЦІЙ	21
2.1 Великі мовні моделі та їх застосування в медицині	21
2.1.1. Архітектура та принципи роботи LLM	21
2.1.2. Особливості генерації тексту в LLM	22
2.1.3. Застосування LLM у медичній галузі.....	22
2.2 Галюцинації в LLM та їх прояв у медичному контексті	23
2.2.1. Механізми виникнення галюцинацій	23
2.2.2. Типи галюцинацій у медичних інтерпретаціях	25
2.3. Структура та характеристики медичного набору даних.....	26
2.3.1 Структурні компоненти MSDiagnosis.....	27
2.3.2 Типовий фрагмент датасету.....	28
2.3.3 Статистичні характеристики та доменна репрезентативність....	28
2.3.4 Обмеження та практична цінність	29
2.4 LLM як критик.....	29
2.5 Ймовірнісно-логарифмічні та ентропійні методи	31
2.5.1 Логарифмічні ймовірності.....	31
2.5.2 Основи логітів та softmax перетворення	32
2.5.3 Класичні ймовірнісні метрики та їх застосування.....	33
2.5.4 Висновки та практичні аспекти.....	34
2.6 Методи на основі самоперевірки.....	34
2.6.1 Загальна концепція методу.....	34
2.6.2 Математична модель та оцінка консистентності.....	35

	11
2.6.3 Реалізація та алгоритмічні аспекти	36
2.6.4 Переваги та обмеження	37
2.6.5 Висновки та практичні аспекти.....	37
2.7 Методи на основі аналізу внутрішніх станів моделі	38
2.7.1 Загальна концепція методу.....	38
2.7.2 Створення датасету.....	38
2.7.3 Математична модель та архітектура класифікатора	39
2.7.4 Переваги та обмеження.....	39
Висновки до розділу 2	40
РОЗДІЛ 3 ОБГРУНТУВАННЯ ВИБОРУ ПРОГРАМНОГО СЕРЕДОВИЩА	41
3.1 Вибір мови програмування.....	41
3.2 Середовище розробки	42
3.3 Вибір та завантаження LLM	44
3.4 Вибір середовища для розгортання моделі.....	46
3.5 Вибір моделі-критика	47
Висновок до розділу 3	48
РОЗДІЛ 4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА МЕТОДИКА РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЕТЕКЦІЇ ГАЛЮЦІНАЦІЙ У ВЕЛИКИХ МОВНИХ МОДЕЛЯХ.....	49
4.1 Вибір та підготовка даних	49
4.1.1 Обґрунтування вибору датасету <i>MSDiagnosis</i>	49
4.1.2 Структура даних для експериментів.....	50
4.1.3 Переклад даних.....	51
4.2 Постановка задачі та методологія тестування	53
4.2.1 Формалізація задачі	53
4.2.2 Методологія тестування.....	54
4.3 Аналіз лог-ймовірностей	56
4.3.1 Отримання лог-ймовірностей	56
4.3.2 Детекція галюцинацій.....	56
4.4 SelfCheck GPT	58
4.4.1 Практична реалізація.....	58
4.4.2 Проблеми та обмеження	59
4.5 Метод на основі аналізу внутрішніх активацій	62
4.5.1 Порівняння з оригінальною статтею <i>MIND</i>	62

	12
4.5.2 Процес створення датасету	64
4.5.3 Тренування класифікатора	68
4.6 Порівняння методів детекції галюцинацій	73
4.7 Інтерфейс програми	74
4.8 Розрахунок економічного ефекту диплому	76
Висновок до розділу 4	76
ЗАГАЛЬНІ ВИСНОВКИ	78
Результати роботи	78
Обговорення результатів роботи.....	78
Висновки.....	79
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ	81

ВСТУП

Актуальність теми полягає в необхідності створення ефективних методів детекції галюцинацій у медичних діагностичних висновках, оскільки ШІ присутній в усіх сферах нашого життя. Медичний домен створює умови, в яких ціна помилки є надзвичайно високою, а в існуючих системах методів перевірки галюцинацій або взагалі немає, або вони малоефективні. Тому розробка ефективної системи контролю правдивості відповідей LLM стає критично важливою для безпечного впровадження асистованих рішень у сферу охорони здоров'я.

Мета і завдання роботи. Метою роботи є підвищення точності та надійності формування діагностичних висновків у медицині за допомогою впровадження системи протидії галюцинаціям у результатах генеративного штучного інтелекту. Це дозволить досягти високих показників достовірності автоматизованих рішень та мінімізувати ризики виникнення помилкових або хибних висновків у клінічній практиці.

Сформульовано наступні задачі:

1. Провести аналіз сучасних методів детекції галюцинацій в LLM;
2. Вивчити теоретичний матеріал щодо основних методів детекції галюцинацій;
3. Знайти та підготувати набір даних з клінічними даними пацієнтів.
4. Практично реалізувати 3 методи детекції галюцинацій.
5. Розробити та реалізувати власну покращену версію методу аналізу активацій.
6. Розробити систему тестування методів детекції галюцинацій.
7. Протестувати реалізовані методи та зробити висновки.
8. Розробити графічний інтерфейс для продукту.

Використані методи. В процесі проведеного дослідження було визначено та обґрунтовано використання: мови програмування Python як оптимального вибору завдяки першокласній підтримці бібліотеки Transformers, потужній ML/NLP екосистемі та швидкості розробки; ключових бібліотек, таких як

accelerate, instructor, torch, transformers, pandas, openai, numpy для основної функціональності системи та допоміжних бібліотек colorlog, datasets, ipykernel, matplotlib, openpyxl, pydantic-settings, python-frontmatter, weave, streamlit, jinja2 для розширення можливостей; моделі Mistral-Small-24B-Instruct-2501 з архітектурою трансформер, що містить 24 мільярди параметрів та демонструє оптимальний баланс між обчислювальною ефективністю та якістю генерації відповідей; хмарного сервісу RunPod з конфігурацією 4 × NVIDIA RTX 4080 SUPER графічних прискорювачі, 32 CPU ядра, 166 GB RAM для розгортання та використання моделі; моделі GPT-4.1 як критика для перевірки наявності галюцинацій у відповідях LLM; спеціалізованого датасету MSDiagnosis, який є репрезентативною вибіркою реальних клінічних випадків у педіатричній практиці.

Отримані результати. Результати були отримані на підставі дослідження системи детекції галюцинацій у медичних текстах, згенерованих великими мовними моделями. Було розроблено і навчено спеціалізований класифікатор HallucinationDetector, що базується на аналізі внутрішніх станів мовної моделі, який продемонстрував найвищу точність (83.3%) та F1-score (84.7%) серед усіх досліджених методів. Реалізовано та протестовано два альтернативні методи виявлення галюцинацій: аналіз logprobs з найменшою латентністю (12 мс) та SelfCheckGPT-NLI з можливістю роботи з моделями як з "чорним ящиком". Створено інтерактивний веб-додаток на базі Streamlit для демонстрації роботи системи детекції галюцинацій. Отриманий результат може мати практичне застосування в системах підтримки прийняття медичних рішень для забезпечення достовірності автоматично згенерованих відповідей, особливо в медичному контексті, де ціна помилки надзвичайно висока, а достовірність інформації є критичною для здоров'я пацієнтів.

Публікації. Не заплановано

Структура роботи. Дипломна робота за темою «Система детекції галюцинацій у результатах діагностичних висновків генеративного штучного інтелекту» виконана студентом Бутком Віталієм Дмитровичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині», побудована за класичним типом та викладена на

84 сторінках машинописного тексту. Вона складається з: вступу; 4 розділів (аналітичний огляд існуючих методів детекції галюцинацій, засоби реалізації інформаційної системи детекції галюцинацій, обґрунтування вибору програмного середовища, програмна реалізація та методика роботи інформаційної системи детекції галюцинацій у великих мовних моделях), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 34 джерела. В роботі представлено 25 рисунків.

РОЗДІЛ 1

АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ДЕТЕКЦІЇ ГАЛЮЦІНАЦІЙ

У цьому розділі узагальнюються основні сучасні підходи до детекції галюцинацій, систематизовані за ключовими методологічними напрямками.

1.1. Ймовірнісні та лог-пробабілістичні методи

Методи цієї категорії ґрунтуються на аналізі впевненості самої моделі, що відображається у ймовірностях чи логарифмічних оцінках, які модель призначає згенерованим токенам. Основна гіпотеза полягає у тому, що під час галюцинації LLM демонструє нижчу впевненість у своїх відповідях, тому низькі значення ймовірності токенів можуть сигналізувати про можливу недостовірність інформації.

Зокрема, підхід Seq-Logprob використовує довжино-нормалізовану послідовність лог-імовірностей для кожного слова у відповіді. Емпіричні дослідження показали, що у випадках, коли модель «галюцинує», її оцінки впевненості суттєво знижуються, що дозволяє використовувати Seq-Logprob як ефективний евристичний індикатор якості генерації, особливо у задачах машинного перекладу та сумаризації [\[23\]](#).

Більш сучасні підходи передбачають використання надбудов над стандартними лог-імовірностями. Наприклад, HalluDetect, розроблений дослідниками з Університету Бейлора, базується на навчанні двох простих класифікаторів, які використовують лише чотири числові характеристики, отримані з ймовірнісних розподілів токенів та словникових структур. Такий підхід дозволяє досягати показників, що перевищують сучасні бенчмарки за точністю та продуктивністю, при мінімальних обчислювальних витратах [\[9\]](#).

1.2. Методи на основі багаторазової генерації та самоперевірки

Друга група методів полягає у створенні кількох відповідей на один і той самий запит та порівнянні їхньої смислової узгодженості. Основна інтуїція полягає у тому, що якщо LLM дійсно «розуміє» предмет запиту, більшість її відповідей будуть схожими; якщо ж модель галюцинує, різні генерації, як правило, починають суперечити одна одній.

Мабуть, найвідомішим та найвживанішим методом цієї категорії є SelfCheckGPT – метод, який жодним чином не залежить від зовнішніх баз знань і може застосовуватись навіть для закритих пропрієтарних LLM (наприклад, Claude). SelfCheckGPT генерує декілька відповідей на той самий запит (з різними параметрами), обчислює ступінь їхньої схожості за допомогою метрик типу BERTScore або NLI, після чого на основі низької консистентності робить висновок про можливу галюцинацію. Дослідження показують, що цей підхід перевищує точність базових методів на рівні окремих речень, однак його недоліком є суттєве зростання обчислювальних витрат, що ускладнює використання у реальному часі [\[21\]\[8\]](#).

Подальший розвиток ідей самоперевірки відображає підхід Multiple-Choice Question Answering Generation (MQAG), у якому для кожного згенерованого фрагмента формулюються множинні питання, відповіді на які аналізуються щодо послідовності та відповідності початковому змісту. Цей підхід підвищує точність, але ще більше збільшує обсяг обчислень, що обмежує його застосування у високонавантажених системах [\[22\]](#).

1.3. Аналіз внутрішніх станів моделей

Методи аналізу прихованих станів LLM спрямовані на дослідження проміжних представлень (hidden states), які формуються у процесі генерації, без необхідності багаторазового запуску моделі чи залучення зовнішніх знань. Цей напрямок базується на припущенні, що у структурі векторів прихованих станів відображається як змістовна наповненість, так і впевненість моделі.

INSIDE – INternal States for hallucIDE detection – пропонує використовувати щільну семантичну інформацію, що зберігається у прихованих шарах, для оцінки ймовірності галюцинацій. В межах INSIDE запропоновано метрику EigenScore, яка вимірює консистентність/диверсифікацію у векторному просторі генерацій на основі власних значень коваріаційної матриці відповіді. Дослідники також впровадили техніку обрізання екстремальних активацій у прихованих станах, що дозволяє знизити ймовірність надмірної впевненості та, відповідно, покращити виявлення галюцинацій, які супроводжуються завищеною впевненістю моделі [10].

Важливим кроком у цьому напрямку є впровадження Semantic Entropy Probes (SEPs), які дозволяють оцінювати семантичну ентропію лише на основі одного проходу по прихованих станах, без додаткових генерацій, що значно підвищує ефективність аналізу у реальному часі. SEPs демонструють високу продуктивність і здатність до генералізації на нових, раніше не бачених даних, що робить їх перспективними для практичного застосування [6].

Окремо варто виділити MIND – повністю ненаглядовий (unsupervised) фреймворк, який навчається на прихованих станах LLM і дозволяє проводити детекцію галюцинацій у реальному часі, без ручного маркування прикладів. MIND показав перевагу над сучасними аналогами на бенчмарку HELM завдяки високій продуктивності та здатності працювати у режимі online-моніторингу [14].

1.4. Методи з використанням зовнішніх джерел знань

Інтеграція зовнішніх джерел знань із генеративними можливостями LLM дає змогу проводити квазі-фактчекінг шляхом порівняння відповідей моделі з достовірними джерелами даних. RAG-архітектури дозволяють компенсувати притаманні LLM обмеження у знаннях за рахунок залучення актуальної інформації з зовнішніх баз.

Яскравим прикладом сучасного впровадження RAG є інтеграція з платформою Vertex AI від Google Cloud. Такий підхід поєднує механізми інформаційного пошуку (retrieval) із генерацією, що забезпечує доступ до реальних даних у режимі онлайн. Основні елементи RAG включають дроблення текстів на

сміслові блоки для полегшення пошуку, розширення запитів для підвищення релевантності та використання гібридних стратегій пошуку (ключові слова + семантичний пошук). Практичні кейси показують, що застосування RAG у промислових середовищах дозволяє суттєво зменшити частоту галюцинацій і підвищити довіру користувачів до LLM, проте потребує додаткової інфраструктури для якісного управління знаннями [34].

1.5. Гібридні та спеціалізовані підходи

Оскільки жодна з категорій не є універсальною, окремі методи комбінують декілька стратегій або адаптуються для конкретних доменів.

Наприклад, HaloScore пропонує фреймворк, що працює з неанотованими генераціями LLM, автоматично оцінюючи правдивість за допомогою спеціального membership score. Перевага підходу полягає у відсутності потреби у додаткових людських анотаціях, що забезпечує гнучкість і масштабованість для використання у нестандартних середовищах. HaloScore перевершує сучасні аналоги за точністю виявлення галюцинацій у великих обсягах неструктурованих даних [11].

Окремої уваги заслуговує гібридний підхід SLM-LLM, який комбінує швидкість і простоту невеликих мовних моделей (SLM) із глибиною аналізу LLM. Спочатку SLM виступає класифікатором для попереднього відбору підозрілих фрагментів, а потім LLM генерує детальні пояснення, що дозволяє досягти балансу між продуктивністю, інтерпретованістю та затримкою – особливо важливо для задач реального часу [18].

Для специфічних галузей, таких як електронна комерція, розробляються спеціалізовані рішення. Наприклад, двофазна система для перевірки галюцинацій у товарних описах, де перший етап зосереджений на виявленні кандидатів до галюцинацій, а другий використовує LLM для уточнення й підвищення точності. Ця комбінація демонструє високу ефективність і дозволяє підлаштовувати підхід до особливостей конкретної галузі [16].

Висновки до розділу 1

У даному розділі було проведено аналітичний огляд існуючих методів детекції галюцинацій у великих мовних моделях. Розглянуті підходи можна класифікувати за п'ятьма основними напрямками: лог-пробабілістичні методи, методи на основі багаторазової генерації та самоперевірки, аналіз внутрішніх станів моделей, методи з використанням зовнішніх джерел знань, а також гібридні та спеціалізовані підходи.

Всі методи використовують різні механізми та ідеї, тому їх використання обумовлено багатьма критеріями та станом системи. Лог-пробабілістичні методи забезпечують простоту реалізації та низькі обчислювальні витрати, що робить їх привабливими для систем з обмеженими ресурсами, проте вони не нададуть високої точності детекції.

Методи на основі багаторазової генерації, зокрема SelfCheckGPT, демонструють високу точність (у деяких випадках), але вимагають значних обчислювальних ресурсів через необхідність генерації кількох відповідей на один запит. Цей підхід ефективний для офлайн-перевірки якості відповідей, проте має обмеження в системах реального часу.

Аналіз внутрішніх станів моделей, на мою думку, є найцікавішим, оскільки дозволяє виявити тонкі патерни в поведінці моделі під час галюцинацій без додаткових генерацій. Такі методи як INSIDE та MIND забезпечують хороший баланс між продуктивністю та точністю, але вимагають доступу до внутрішньої структури моделі, що може бути проблематичним для закритих комерційних LLM.

RAG-архітектури, не лише виявляють галюцинації, але й надають інструменти для їх запобігання. Вони особливо ефективні для задач, де фактологічна точність є критичною, однак вимагають підтримки та оновлення бази знань, що може бути складно в динамічних доменах.

Про гібридні методи можна говорити тоді, коли є розуміння складових. Тому для даної роботи було обрано лог-пробабілістичні методи, на основі самоперевірки та аналізу активацій.

РОЗДІЛ 2

ЗАСОБИ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЕТЕКЦІЇ ГАЛЮЦИНАЦІЙ

В даному розділі будуть пояснені теоретичні аспекти, необхідні для розуміння обраної теми. Буде описано основи архітектури трансформерів, механізм виникнення галюцинацій. Також буде описано обраний датасет, а особлива увага буде присвячена саме поясненню підґрунтя обраних методів детекції галюцинацій.

2.1 Великі мовні моделі та їх застосування в медицині

2.1.1. Архітектура та принципи роботи LLM

Великі мовні моделі (LLM, Large Language Models) представляють собою підклас моделей глибокого навчання, основою яких зазвичай виступає трансформерна архітектура. Ключовою особливістю трансформера є використання багатошарового механізму уваги (multi-head self-attention), який забезпечує паралельну обробку вхідної інформації та виявлення довготривалих залежностей у послідовностях тексту [1]. Це робить трансформери максимально придатними для роботи з природною мовою, де контекст може бути розподілений на великі відстані.

Стандартна LLM складається з енкодерної та/або декодерної частин трансформера. У випадку генеративних моделей, таких як GPT-3/4, використовують лише декодерну частину, адаптовану під автогресивну генерацію тексту. Модель навчається передбачати наступне слово у послідовності, поступово набуваючи узагальнене "розуміння" мовних патернів, логічної структури тексту, семантики та навіть елементарних фактів [26].

У процесі навчання LLM оперує великими масивами текстових даних із різних джерел – від енциклопедій і наукових статей до сайтів і соціальних мереж. Це забезпечує моделі широку базу знань, але водночас створює ризики включення неактуальної або недостовірної інформації. Важливим компонентом трансформерних LLM є масштабування – сучасні моделі містять від мільярдів до

сотень мільярдів параметрів, що дозволяє їм акумулювати складні мовні закономірності та знання [4].

Крім традиційної архітектури, LLM часто доповнюються механізмами підсиленого навчання з підкріпленням від людської зворотного зв'язку (RLHF, Reinforcement Learning from Human Feedback), що підвищує релевантність і безпеку їхніх відповідей у реальних застосуваннях [15].

2.1.2. Особливості генерації тексту в LLM

Генерація тексту у LLM є процесом автогресивного передбачення – починаючи з початкової підказки, модель крок за кроком генерує наступні токени, кожного разу використовуючи увесь згенерований контекст. Вибір наступного токена здійснюється на основі розподілу ймовірностей, що формується на підставі прихованих станів моделі. В залежності від стратегії семплінгу відповіді можуть мати різну варіативність, креативність та ступінь ризику галюцинацій [3].

Особлива увага в LLM приділяється управлінню генерацією через змінні параметри температури, які регулюють "непевненість" моделі, та промптів, які значно впливають на структуру та зміст відповіді. При роботі з медичними запитаннями це має вирішальне значення, оскільки занадто креативна або недостатньо контрольована генерація може призвести до появи небезпечних галюцинацій або спотворень фактів [2].

Важливою особливістю сучасних LLM є здатність до zero-shot і few-shot навчання: модель може виконувати нові завдання без додаткового донавчання, лише опираючись на контекст промпту та власні знання, здобуті під час попереднього тренування. Це суттєво підвищує універсальність та адаптивність LLM, але водночас створює додаткові ризики появи помилкових відповідей у нестандартних чи спеціалізованих доменах, таких як медицина [5].

2.1.3. Застосування LLM у медичній галузі

Використання LLM у медицині відкриває нові горизонти для автоматизації та покращення якості клінічного аналізу, підтримки прийняття рішень, генерування медичної документації та інформаційної підтримки пацієнтів. Вже сьогодні LLM демонструють конкурентну продуктивність у задачах клінічного резюмування, автоматичного заповнення електронних медичних записів,

попереднього діагностування, інтерпретації медичних зображень у поєднанні з мовними описами, а також при створенні відповідей на запити пацієнтів і лікарів [13].

Окремі моделі, такі як MedPaLM, BioGPT та ClinicalBERT, були спеціально навчені або донавчені на медичних корпусах, що підвищує їхню обізнаність у термінології, клінічних протоколах і стандартних процедурах. Перевагою LLM є можливість швидко масштабувати аналіз великих масивів клінічних текстів, здійснювати пошук релевантної інформації та підтримувати персоналізовані консультації [7].

Разом із цим, впровадження LLM у медичну практику супроводжується низкою викликів: висока чутливість до галюцинацій, обмеження у пояснюваності, потреба у суворому контролі достовірності відповідей та дотримання етичних і правових норм щодо обробки персональних даних пацієнтів. Тому питання надійності, тестування та виявлення галюцинацій стають центральними для успішної інтеграції LLM у медичний сектор [7][24].

2.2 Галюцинації в LLM та їх прояв у медичному контексті

В цьому підрозділі буде пояснено чому взагалі виникають галюцинації в LLM та які види таких помилок бувають.

2.2.1. Механізми виникнення галюцинацій

Галюцинації у великих мовних моделях (LLM) – це феномен, коли модель продукує текст, який виглядає закономірним та правдоподібним, але не відповідає фактам чи логічним висновкам із заданого контексту. Технічна природа галюцинацій обумовлена як особливостями архітектури трансформера, так і статистичними властивостями процесу навчання та генерації [8].

1. Автогресивна генерація та крос-ентропія: У класичних LLM на базі трансформерів задача формулюється як максимізація ймовірності послідовності tokenів тобто:

$$P(X) = \prod_{t=1}^n P(x_t | x_1, \dots, x_{t-1}) \quad (2.1)$$

Модель оптимізується за допомогою крос-ентропійної втрати:

$$L = - \sum_{t=1}^n \log P(x_t^* | x_1, \dots, x_{t-1}) \quad (2.2)$$

де

x_t^* - справжній токен.

Однак під час інференсу модель генерує x_t на основі власних попередніх прогнозів, що призводить до накопичення помилок і, як наслідок, – до збільшення ймовірності появи галюцинацій у довгих послідовностях [1].

2. Семплінг з високою ентропією: Для вибору наступного токена використовується розподіл ймовірностей $P(x_t | x_{<t})$. Якщо ентропія розподілу:

$$H_t = - \sum_i P(x_t = i | x_{<t}) \log P(x_t = i | x_{<t}) \quad (2.3)$$

є високою, модель "невпевнена" у виборі наступного слова. При використанні температурного контролю або стратегії nucleus/top-k sampling може бути вибрано менш ймовірний варіант, що збільшує ризик галюцинацій, особливо у спеціалізованих доменах [3].

3. Втрата актуальності знань (Knowledge cutoff): LLM навчаються на статичних корпусах, і їхня інформація може застарівати. Формально, оцінка ймовірності правильного (актуального) факту (f) зменшується з часом:

$$P(f | context, model\ date) \rightarrow 0 \text{ при } time \gg model\ date \quad (2.4)$$

Це особливо критично в медицині, де знання швидко оновлюються [13].

4. Випадкове змішування знань (Knowledge contamination): Проблема виникає через неконтрольовану агрегацію знань із різних джерел. Якщо множина даних містить суперечливу або нерелевантну інформацію, модель може інтерполювати між декількома варіантами і продукувати некоректний факт:

$$P(f_{hallucinated}|context) \approx \alpha_1 P(f_1|context) + \alpha_2 P(f_2|context) + \dots \quad (2.5)$$

де

(f_1, f_2) – альтернативні факти

α_i – ваги, визначені під час тренування [8].

5. Semantic drift та накопичення похибок: У процесі автогресивної генерації навіть незначне відхилення від істини може призвести до дедалі більшого "віддалення" змісту відповіді від початкового контексту, що формалізується як зростання відстані у векторному просторі ембедінгів між поточною відповіддю і ground truth:

$$d_{semantic}(y_{generated}, y_{true}) = ||E(y_{generated}) - E(y_{true})||_2 \uparrow \quad (2.6)$$

де

$E()$ – функція отримання sentence embedding [3].

2.2.2. Типи галюцинацій у медичних інтерпретаціях

Особисто можу виділити 2 основні види галюцинацій – фактичні та логічні, проте, як виявилось, є ще деякі. Їх все-одно можна поєднати у два кластери, просто вони трохи більш обширні.

Фактологічні галюцинації: Модель генерує твердження, що суперечать відомим науці фактам; наприклад, вигадує нові показання до відомих препаратів. Це можна формалізувати як появу відповіді з низьким score при порівнянні з базою знань:

$$score_{fact} - check(y) = sim(y, KB) < \tau \quad (2.7)$$

де

sim - функція схожості

KB - база знань

τ – поріг [16].

Вигадані джерела: LLM генерує посилання, яких не існує, або які не мають жодного відношення до предмету запиту. Це проявляється як відсутність відповідних записів у реальних бібліографічних базах для створеного entity [8].

Помилки термінології: Виникають, коли модель через полісемію чи відсутність достатньої кількості прикладів у корпусі змішує або некоректно тлумачить медичні терміни, наприклад, підміняє назву захворювання симптомом [2].

Генерація небезпечних порад: Наприклад, рекомендація несумісних препаратів, неправильне дозування чи інтерпретація результатів аналізів, що не відповідає клінічним протоколам [18].

Контекстуальні галюцинації: Додавання моделей несуттєвих, але правдоподібних деталей, які не впливають із вихідного питання – формалізується як розширення семантичного простору за межі вхідного контексту:

$$E(y_{generated}) \notin C(E(x_{input})) \quad (2.8)$$

де

C множина допустимих відповідей на основі вхідних даних [8].

Логічні галюцинації: Порухення логічної послідовності аргументів. Фактично – все правильно, проте логічні висновки до цих фактів некоректні, тобто з одного виходить те, що не мало б [18].

2.3. Структура та характеристики медичного набору даних

Для дослідження системи протидії галюцинаціям у діагностичних висновках генеративного штучного інтелекту було використано спеціалізований датасет MSDiagnosis, який є репрезентативною вибіркою реальних клінічних випадків у педіатричній практиці. Основною метою створення MSDiagnosis є забезпечення

багаторівневої перевірки якості генеративних медичних моделей шляхом моделювання повного циклу клінічної діагностики: від збору анамнезу до постановки остаточного діагнозу та його обґрунтування. Особливість цього ресурсу полягає у відтворенні реалій лікарської практики, де діагностичний процес складається з кількох логічних етапів і вимагає від системи глибокого контекстного розуміння, аргументації та послідовної побудови медичних гіпотез [25].

2.3.1 Структурні компоненти MSDiagnosis

Датасет організований у вигляді колекції повноцінних медичних кейсів, кожен з яких структуровано за стандартами клінічної документації та містить такі основні блоки:

1. Демографічна інформація: вік, стать, соціальні фактори.
2. Анамнез захворювання: детальний опис скарг, перебігу та особливостей поточної хвороби.
3. Медичний анамнез та супутня патологія: інформація про попередні захворювання, імунізацію, алергії тощо.
4. Дані об'єктивного обстеження: результати клінічного огляду, температури, пульсу, артеріального тиску тощо.
5. Результати лабораторної та інструментальної діагностики: аналізи крові, біохімічні показники, дані візуалізаційних методів (УЗД, КТ) та інші відповідні дослідження.
6. Перебіг лікування та динаміка стану: хронологія призначень, реакція на терапію, зміна клінічної картини.
7. Діагностичний висновок та обґрунтування: формулювання попереднього та остаточного діагнозу з детальним поясненням на підставі зібраних даних.
8. QA-пари (питання–відповідь): набір ключових запитань, що імітують клінічне мислення–визначення попереднього діагнозу, диференціальна діагностика, формулювання кінцевого діагнозу, аргументація рішень.
9. Значенням QA-блоку є імітація поведінки експертної системи або консилиуму, коли кожную відповідь потрібно підкріпити посиланням на явні або неявні дані кейсу. Одні питання мають короткі номінативні відповіді (наприклад,

"первинний діагноз"), інші – докладне клінічне обґрунтування (наприклад, "які критерії підтверджують діагноз?") [25].

2.3.2 Типовий фрагмент датасету

Демографічна інформація: хлопчик, 10 років, учень.

Основна скарга: повторні епізоди підвищення температури тіла протягом 21 дня, наявність шийних лімфовузлів.

Анамнез захворювання: 21 день тому з'явилася лихоманка до 39°C, періодичний кашель, поступово додався набряк та болючість на шії.

Об'єктивний огляд: температура 38.5°C, пульс 90 уд/хв, артеріальний тиск 95/64 мм рт.ст., пальпується кілька збільшених лімфовузлів на шії.

Лабораторні дослідження: зміни в аналізі крові, ультразвукова картина шийних лімфовузлів.

Перебіг лікування: проведено біопсію лімфовузла, призначено антибіотики та підтримувальну терапію.

Остаточний діагноз: некротизуючий гістіоцитарний лімфаденіт шийної локалізації.

QA-пари:

Який початковий діагноз? – Гарячка неясного генезу, збільшення лімфовузлів.

Які дані обґрунтовують діагноз? – Характер перебігу, ультразвукові та гістологічні результати.

Які диференційні діагнози розглядалися? – Злоякісна лімфома, туберкульоз, системний червоний вовчак тощо.

2.3.3 Статистичні характеристики та доменна репрезентативність

За даними оригінальної статті, MSDiagnosis містить понад 1000 клінічних випадків, кожен з яких має від 3 до 6 QA-пар, що охоплюють усі етапи клінічного розбору. Таким чином, загальний обсяг індивідуальних запитань і відповідей перевищує декілька тисяч пар. У датасеті представлені десятки нозологічних форм, що забезпечує різноманітність сценаріїв для аналізу поведінки LLM у ситуаціях із різними рівнями складності, неоднозначності та доказовості медичних рішень [23].

Змістовна особливість MSDiagnosis – фокус на педіатричних пацієнтах, що дозволяє моделювати діагностичні виклики:: атиповість клінічних проявів, складність інтерпретації лабораторних тестів, необхідність мультидисциплінарного підходу. Усі дані оригінально представлені китайською мовою, що вносить додаткові виклики для міжкультурної верифікації та автоматизованої обробки даних (процес перекладу розглядається окремо у наступних підрозділах) [25].

2.3.4 Обмеження та практична цінність

До потенційних обмежень MSDiagnosis можна віднести:

- Мовну специфіку (спочатку китайська мова);
- Обмеження домену (акцент на педіатрії, менший обсяг дорослих або рідкісних нозологій);
- Можливу тенденцію до включення рідкісних або складних випадків для тренування LLM у стресових сценаріях [25].

Водночас, багаторівнева структура кейсів і QA-модуль роблять MSDiagnosis оптимальним ресурсом для тренування, валідації та тестування систем детекції галюцинацій у медичних LLM. Датасет дає змогу імітувати реальні діагностичні процеси й визначати не лише фактологічну, а й логічну якість відповіді моделі [25]. Отже він є прекрасним варіантом для тестування галюцинацій в задачі інтерпретації аналізів людини, адже саме це є задачею цієї роботи.

2.4 LLM як критик

В практичній частині буде використано підхід, котрий відомий в літературі як LLM as a judge. В цьому розділі буде пояснено як його використовують, як саме він працює і чому можна довіряти LLM при перевірці роботи іншої мовної моделі.

Основне питання, яке одразу виникає – «а як LLM, зі своєю ймовірністю галюцинацій може слугувати в якості системи оцінки галюцинацій для іншої LLM?»

Одним із ключових теоретичних обґрунтувань цього методу є принцип асиметричної ймовірності галюцинацій. Він полягає в тому, що ймовірність появи

однакової галюцинації у двох різних моделях на одних і тих же даних надзвичайно низька, особливо якщо одна з моделей значно потужніша за іншу.

Дослідники з Единбурзького університету класифікують дві принципово різні причини виникнення галюцинацій у ЛЛМ:

1. Незнання – коли модель дійсно не знає відповіді
2. Помилка при знанні – коли модель знає правильну відповідь, але все одно відповідає неправильно

В дослідженнях доведено, що простий класифікатор на основі внутрішніх представлень ЛЛМ може розрізняти ці випадки, і що налаштування виявлення/пом'якшення галюцинацій під конкретну модель дає кращі результати, ніж універсальний підхід [28].

Використання моделі-критика додатково покращує цей процес, оскільки потужніша модель здатна виявляти обидва типи помилок з вищою точністю.

Практичне застосування методу моделі-критика для медичних даних підтверджено пілотним дослідженням використання GPT-4 для генерації психологічних звітів. Дослідження оцінювало якість, ризик галюцинацій (некоректної інформації, створеної ШІ) та задоволеність клієнтів психодинамічними психологічними звітами, згенерованими GPT-4. Завдяки методиці перевірки, що включала валідацію експертами, було досягнуто низького рівня галюцинацій [19].

Так чому ж ймовірність помилки критика мінімальна?

Ключова відмінність полягає в характері задачі:

- Генеративна модель виконує складне творче завдання – створення нової інформації
- Модель-критик виконує простіше аналітичне завдання – порівняння наявної інформації з вхідними даними

Коли модель здійснює перевірку вже згенерованої відповіді на відповідність вхідним даним, задача стає значно простішою, ніж генерація відповіді з нуля [30].

До того ж математичний аналіз галюцинацій у генеративних моделях показує, що галюцинації відбуваються, коли модель генерує токени з низькою ймовірністю. При перевірці вже наявної інформації замість генерації нової, ймовірність помилки

суттєво зменшується. Це пояснює, чому модель-критик, яка не потребує генерувати нові токени з нуля, а лише перевіряє їхню відповідність вхідним даним, має набагато меншу ймовірність галюцинацій [17].

2.5 Ймовірісно-логарифмічні та ентропійні методи

У цьому підрозділі розглядаються базові принципи роботи нейронних мовних моделей, зокрема поняття логітів, їх трансформації за допомогою функції softmax, а також основні ймовірнісні метрики (середній log-prob, ентропія і перплексія), що використовуються для оцінки невпевненості моделі.

2.5.1 Логарифмічні ймовірності

Логарифмічні ймовірності являють собою результат застосування логарифмічної функції до значень звичайних ймовірностей. Для прикладу, коли ймовірність події становить $P(E) = 0.00001$, її логарифмічне значення приблизно дорівнює $\log P(E) = \log(0.00001) \approx -11.51$, використовуючи натуральний логарифм з основою e . Логарифмічні ймовірності набули широкого застосування в обчислювальній статистиці з двох основних причин: комп'ютерним системам складно працювати з надзвичайно малими числовими значеннями, а також логарифми дозволяють замінити множення на додавання, що суттєво прискорює обчислення [12].

Від'ємний знак логарифмічних ймовірностей пояснюється математичними властивостями логарифмічної функції. Оскільки $\log b = c$ означає, що $e^c = b$ (при використанні натурального логарифму), і враховуючи, що ймовірності належать діапазону від 0 до 1, логарифм таких значень завжди даватиме від'ємний результат. Для розуміння: $e^0 = 1$, тому для отримання числа меншого за 1 необхідний показник степеня менший за 0 [12].

Звідси впливають основні правила для ймовірностей та їх логарифмів:

Аксіома ймовірності:

$$0 \leq P(E) \leq 1 \quad (2.9)$$

Правило для логарифмічних ймовірностей

$$-\infty \leq \log P(E) \leq 0 \quad (2.10)$$

2.5.2 Основи логітів та softmax перетворення

У великих мовних моделях останній лінійний шар генерує "сирі" числові оцінки, які називаються логітами. Для кожного токена словника (розмір якого – V) формується вектор логітів $z = (z_1, z_2, \dots, z_V)$, де кожне значення відображає внутрішню оцінку моделі щодо відповідності токена при заданому контексті. Оскільки логіти можуть набувати як позитивних, так і негативних значень і не нормовані, для отримання розподілу ймовірностей застосовується функція softmax, що виконує експоненціальне перетворення з подальшою нормалізацією:

$$p_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (2.11)$$

де

z_i - логіт для класу i

p_i - результуюча ймовірність для класу i

$\sum_j \exp(z_j)$ - сума експонент всіх логітів

Таке перетворення лежить в основі обчислення токенного правдоподібності як базису для подальших метрик [6]. Функція softmax гарантує, що всі ймовірності знаходяться в діапазоні від 0 до 1 та сума всіх ймовірностей дорівнює 1 [33].

Логарифмічні ймовірності тісно пов'язані з логітами. Фактично, логіти можна розглядати як ненормовані логарифмічні ймовірності. Після застосування softmax та обчислення логарифма, ми отримуємо:

$$\log(p_i) = z_i - \log\left(\sum_j \exp(z_j)\right) \quad (2.12)$$

де

z_i - це логіт

z_j - нормалізуючий фактор [33].

2.5.3 Класичні ймовірнісні метрики та їх застосування

Метрики, побудовані на основі логітів і ймовірностей, дозволяють кількісно оцінити ступінь впевненості моделі під час генерації тексту. Серед них:

- **Середній log-prob** – розраховується як середнє значення негативного логарифму ймовірності токенів, що було згенеровано. Для послідовності з J токенів обчислення має вигляд:

$$\log prob = - (1/J) \sum_i \log p_i \quad (2.13)$$

Високий (менш негативний) log-prob свідчить про більшу впевненість моделі у виборі відповідного токена.

- **Ентропія** – визначається як міра розпорошеності розподілу ймовірностей і обчислюється за формулою:

$$H = - \sum_{(w \in W)} p(w) \log p(w) \quad (2.14)$$

де

W – множина всіх можливих токенів

Підвищення ентропії вказує на більшу невизначеність вибору моделі, що може бути сигналом потенційних галюцинацій.

- **Перплексія** – метрика, що характеризує ефективність моделі при передбаченні послідовностей тексту. Вона обчислюється як експонента від середнього значення лог-ймовірностей окремих токенів:

$$PPL = \exp \left(\frac{1}{N} \sum_{i=1}^N -\log P(x_i | x_1, x_2, \dots, x_{i-1}) \right) \quad (2.15)$$

де

N – кількість токенів у послідовності

$P(x_i|x_1, x_2, \dots x_{i-1})$ – ймовірність, обрахована моделлю для i токена послідовності

Зростання перплексії свідчить про зниження впевненості моделі у своїй генерації [20].

Інтуїтивна різниця:

Ентропія: Локальна невизначеність для кожного кроку прогнозування - наскільки «невпевнена» модель для наступної лексеми у всьому словнику.

Перплексія: Глобальна міра для всієї послідовності, яка враховує фактичні ймовірності, присвоєні моделлю обраній послідовності.

Випадки, коли модель генерує галюцинації, часто супроводжуються зниженням логічної впевненості (меншим log-prob), збільшенням ентропії і, відповідно, зростанням перплексії, що дозволяє використовувати ці метрики як індикатори непередбачуваної поведінки моделі.

2.5.4 Висновки та практичні аспекти

Отже, логіти, що генеруються останнім лінійним шаром, є базовою ланкою, яка після softmax перетворення дає розподіл ймовірностей для кожного можливого токена. Класичні метрики – середній log-prob, ентропія та перплексія – дозволяють оцінити впевненість моделі на рівні tokenів, але для виявлення галюцинацій необхідно враховувати також смислову невпевненість [33][20].

2.6 Методи на основі самоперевірки

У цьому розділі подано детальний опис методу самоперевірки SelfCheckGPT у контексті виявлення галюцинацій великих мовних моделей, що базується на аналізі схожості кількох згенерованих відповідей.

2.6.1 Загальна концепція методу

Метод самоперевірки ґрунтується на припущенні, що якщо мовна модель добре «засвоїла» сутність певного знання, то при повторному запитуванні вона генерує відповіді, які є семантично схожими та послідовними. Формально, нехай для одного запиту x модель генерує N незалежних відповідей $S_1, S_2, \dots, S_N$. Якщо серед цих відповідей домінують спільні семантичні ознаки, то це свідчить про

достовірність інформації, а різке розходження між відповідями є сигналом можливих галюцинацій [21].

2.6.2 Математична модель та оцінка консистентності

Для кількісного визначення ступеня семантичної схожості між згенерованими відповідями застосовують декілька метрик. Однією з них є середньостатистична міра парної схожості, яку можна обчислити за допомогою косинусної подібності між ембедінгами речень. Нехай v_i – векторне представлення i -ої відповіді, тоді середня косинусна схожість визначається як:

$$Consistency = \frac{2}{(N \cdot (N - 1))} \sum_{i=1}^{N-1} \sum_{j=i+1}^N \frac{v_i \cdot v_j}{||v_i|| \cdot ||v_j||} \quad (2.17)$$

Інший варіант оцінки базується на використанні BERTScore. Для i -го речення основної відповіді r_i та набору згенерованих речень з кожного семплу S_n визначається метрика:

$$S_{BERT}(i) = 1 - \frac{1}{N} \sum_{n=1}^N \max_k (B(r_i, s_k^n)) \quad (2.18)$$

де

$B()$ позначає BERTScore, який обчислює подібність між двома реченнями. Якщо $S_{BERT}(i)$ набуває високих значень, це свідчить про великий ступінь консистентності між реченнями, а низький показник – про можливі галюцинації [21].

Крім того, існують варіанти, що базуються на автоматизованому генерації питань і оцінці відповідей. Нехай для кожного речення r_i генерується набір питань q з відповідними варіантами o , тоді можна визначити індекс консистентності за формулою:

$$SQA(i, q) = \frac{\gamma_2^{N'_n}}{\gamma_1^{N'_m} + \gamma_2^{N'_n}} \quad (2.19)$$

де

N_m – кількість випадків, коли відповідь основного тексту співпадає із згенерованою відповіддю

N_n – кількість невідповідностей.

γ – величини, що використовуються у методі SelfCheckGPT-QA з баєсівським підходом, де ці параметри служать для перетворення кількостей збігів та розбіжностей у відповідях (N_+ та N_-) в оцінку ймовірності нефактичності твердження [21].

Остаточний бал для речення обчислюється як середнє значення по всіх питанням.

Також можливе використання n-грамових моделей та моделей, що базуються на Natural Language Inference (NLI), де за допомогою класифікатора NLI (наприклад, DeBERTa-Large) обчислюється імовірність суперечності між парою речень [21].

2.6.3 Реалізація та алгоритмічні аспекти

Практична реалізація методу самоперевірки включає наступні етапи:

1. Для заданого запиту x здійснюється генерація N (часто 5–10) відповідей з використанням стохастичних алгоритмів (наприклад, top-p або top-k семплінгу) [21].

2. Отримані відповіді переводяться у векторні представлення (ембедінги) із застосуванням попередньо навчених моделей (наприклад, RoBERTa-Large) для розрахунку BERTScore або косинусної подібності.

3. Обчислюються вищезгадані метрики за формулами 2.17 – 2.19, після чого проводиться аналіз отриманих показників. Наприклад, якщо середня пара синхронності менша за заздалегідь визначений поріг τ , відповідь класифікується як потенційно галюцинована.

За необхідності може бути інтегровано додаткову евристику – наприклад, якщо більшість згенерованих відповідей логічно розходяться за змістом, то система видає сигнал «flag as hallucination» [21].

Формально, для визначення порогового значення можна використовувати правило:

$$Flag = \{ 1, \text{якщо } Consistency < \tau; 0, \text{інакше} \} \quad (2.20)$$

де

Flag = 1 означає наявність галюцинації.

2.6.4 Переваги та обмеження

Метод самоперевірки має низку переваг:

Він є «black-box» підходом, який не потребує доступу до внутрішніх логів чи токен-розподілу, що робить його придатним для моделей з обмеженим доступом, таких як ChatGPT [21].

Метод дозволяє використовувати open-source рішення для обчислення семантичних метрик, зокрема BERTScore та cosine similarity, що забезпечує його доступність та адаптивність.

Використовуючи декілька варіантів оцінки (BERTScore, QA-метрики, n-грамові та NLI підходи), можна отримати комплексну картину достовірності згенерованої інформації.

Незважаючи на ці переваги, існують і обмеження:

Необхідність генерації кількох відповідей для одного запиту збільшує обчислювальні витрати.

Метод схильний до вибору оптимального порогового значення, яке може варіюватися залежно від доменної специфіки та якості генерації [21].

2.6.5 Висновки та практичні аспекти

Метод самоперевірки заснований на аналізі семантичної консистентності кількох згенерованих відповідей є дієвим інструментом для виявлення галюцинацій у відповідях LLM. Формально задекларовані метрики дозволяють кількісно оцінити ступінь схожості між відповідями, що, в разі низького значення, свідчить про потенційні недостовірні генерації [21].

2.7 Методи на основі аналізу внутрішніх станів моделі

У цьому розділі представлено опис методу виявлення галюцинацій, який базується на аналізі внутрішніх станів великих мовних моделей (LLM) під час генерації тексту.

2.7.1 Загальна концепція методу

Методи виявлення галюцинацій на основі внутрішніх станів мовної моделі ґрунтуються на припущенні, що особливості контекстуалізованих вкладень токенів у процесі генерації тексту можуть свідчити про достовірність або недостовірність інформації, яку продукує модель. На відміну від підходів, що вимагають повної генерації відповіді перед виявленням галюцинацій, методи на основі внутрішніх станів дозволяють виявляти галюцинації в режимі реального часу під час самого процесу генерації тексту внаслідок аналізу внутрішніх станів LLM під час процесу виведення.

В даній роботі буде розглянуто метод MIND. Його алгоритм роботи двоетапний:

Спочатку ми автоматично створюємо розмічений датасет. А потім будемо класифікатор та тренуємо його [14].

2.7.2 Створення датасету

Важливою особливістю методу є можливість навчання без використання розмічених даних. Процес передбачає автоматичне генерування навчальних даних на основі статей Вікіпедії. Стаття обрізається після випадково вибраної сутності, що з'являється після першого речення. Обрізана стаття подається на вхід мовної моделі для продовження тексту. Якщо згенерований текст починається з правильної сутності, він мітиться як "не-галюцинація", інакше – як "галюцинація" [14]. Ну і на виході – отримуємо датасет, де вхід – внутрішні активації, вихід – мітки галюцинацій.

Цей підхід дозволяє створювати великі набори даних для навчання без необхідності ручного розмічування, що значно прискорює та здешевлює процес навчання класифікатора [14].

2.7.3 Математична модель та архітектура класифікатора

Після отримання датасету, необхідно натренувати багатошаровий перцептрон, що отримує на вхід контекстуалізовані вкладення токенів мовної моделі. У трансформерах генерація кожного токена безпосередньо залежить від контекстуалізованих вкладень, які представляють розуміння моделлю контексту та семантичне подання раніше згенерованих токенів. Ці ембедінги також відображають поведінку моделі при передбаченні наступного токена, включаючи елементи невизначеності [14].

Класифікатор галюцинацій можна формально представити як:

$$P = \text{MLP}(\text{ReLU}(W \cdot H + b)) \quad (2.21)$$

де

P – прогнозований класифікатором мітка галюцинації

H – матриця $1 \times n$, що представляє вибрані приховані стани

W і b – відповідно вагова матриця та вектор зміщення багатошарового перцептрона.

Архітектурно класифікатор складається з чотирьохшарової мережі MLP з поступово зменшуваним розміром прихованих шарів (256, 128, 64 і 2 нейрони для кожного послідовного шару). В якості функції активації використовується ReLU. Для оптимізації класифікатора застосовується функція втрат бінарної крос-ентропії [14].

І тепер ми генеруємо досліджуваною LLM відповіді на питання, передаємо її внутрішні активації до класифікатору, та отримуємо ризик галюцинації.

2.7.4 Переваги та обмеження

Серед головних переваг методу виявлення галюцинацій на основі внутрішніх станів мовних моделей слід виділити:

1. Можливість виявлення галюцинацій у режимі реального часу під час генерації тексту.
2. Низькі обчислювальні витрати порівняно з методами пост-обробки.
3. Можливість інтеграції з будь-якими існуючими трансформер-моделями.

4. Навчання без вчителя, що не потребує ручного розмічування даних [\[14\]](#).

Проте метод має і певні обмеження:

1. Необхідність адаптації класифікатора для кожної окремої мовної моделі, оскільки контекстуалізовані вкладення різних моделей мають різні характеристики.
2. Ефективність методу залежить від обсягу навчальних даних, при цьому збільшення набору даних понад 4096 прикладів не призводить до значного покращення результатів [\[14\]](#).

Висновки до розділу 2

В даному розділі були описані основи трансформерних нейромереж. Детально пояснені причини виникнення галюцинацій, розглянуто їх види. Також увага була приділена набору даних, котрий буде використано для тренування та тестування систем. Дуже важливим є пояснення безпеки та достовірності використання методи LLM – критика, через що, в одному з підрозділів, було пояснено, чому ймовірність подвійних галюцинацій малоімовірна. І наостанок були описані математичні моделі та практичні аспекти реалізацій трьох методів детекції галюцинацій, які будуть імплементовані та порівнювати в практичній частині дипломної роботи.

РОЗДІЛ 3

ОБГРУНТУВАННЯ ВИБОРУ ПРОГРАМНОГО СЕРЕДОВИЩА

В даному розділі будуть описані всі необхідні складові для реалізації системи. Окрім мови програмування та бібліотек в даному проєкті є ще три дуже важливих компоненти. Досліджувана мовна модель, сервіс для оренди GPU та модель критик.

Перед початком будь-якої розробки необхідно обрати мову програмування, середовище розробки, набір методів та рішень, які дозволять вирішити поставлене завдання найоптимальнішим шляхом. Адже помилка на даному етапі може стати фатальною під час переходу від прикладної розробки до реальної робочої системи.

3.1 Вибір мови програмування

Питання мови програмування в застосунках з використанням генеративного штучного інтелекту доволі просте. Спершу необхідно визначитися із ключовими технологіями для продукту, потім обирати мову, яка має найкращу підтримку цієї технології.

В даній системі ключовою є бібліотека Transformers, яка забезпечить доступ до відкритих моделей та до усіх необхідних маніпуляцій з LLM.

Для розробки системи детекції галюцинацій у великих мовних моделях очевидним і оптимальним вибором є мова програмування Python. Цей вибір зумовлений наступними ключовими факторами:

- Першокласна підтримка Transformers - Бібліотека Transformers від Hugging Face оптимізована насамперед для Python, що забезпечує доступ до всіх актуальних функцій та моделей без обмежень.
- Потужна ML/NLP екосистема - Python надає доступ до необхідних інструментів, які доповнюють функціонал системи детекції галюцинацій:
 1. PyTorch/TensorFlow для нейромережових обчислень
 2. NumPy/Pandas для маніпуляцій з даними
 3. spaCy для лінгвістичного аналізу

- Швидкість та зручність розробки - Python дозволяє швидко створювати прототипи та тестувати різні методи детекції галюцинацій, що критично для дослідницького характеру роботи.

Незважаючи на певні обмеження Python щодо швидкодії, для критичних компонентів системи будуть застосовані методи оптимізації:

- Використання Cython для компіляції критичних ділянок коду
- Застосування багатопроцесорної обробки для паралелізації операцій
- Використання GPU-прискорення через CUDA з PyTorch

Отже, вибір Python для розробки системи детекції галюцинацій є оптимальним з огляду на повну сумісність з бібліотекою Transformers, розвинуту екосистему інструментів ML/NLP та швидкість розробки. Потенційні обмеження продуктивності компенсуються можливостями оптимізації та масштабування системи.

3.2 Середовище розробки

На мою думку, в цьому розділі важливим є лише перелік бібліотек, з яких і створено екосистему даного проєкту. Вибір IDE, типи файлів є дуже другорядною задачею. Проте в даній роботі будуть використані `.ipynb` файли для розробки та тестування, `.py` для структурування коду та для забезпечення стабільності та відтворюваності. Розробка проходитиме в IDE Cursor. Це гілка від VS Code, проте з сильною підтримкою штучного інтелекту, що значно прискорює розробку.

Тепер перейдемо до бібліотек. Їх було поділено на основні та допоміжні.

Серед ключових можна виділити:

1. **accelerate** - Бібліотека для оптимізації навчання та виведення моделей на різних пристроях. Забезпечить прискорення обробки моделей та ефективне використання апаратних ресурсів.

2. **instructor** - Спеціалізована бібліотека для роботи з LLM в контексті структурованих даних. Дозволить валідувати виходи моделі за допомогою схем Pydantic для виявлення структурних галюцинацій.

3. **torch** - Фреймворк для машинного навчання PyTorch. Забезпечить основу для роботи з нейронними мережами та тензорними обчисленнями.
4. **transformers** - Бібліотека від Hugging Face з підтримкою різних моделей трансформерів. Дозволить експериментувати з різними архітектурами та техніками для протидії галюцинаціям.
5. **pandas** - Потужна бібліотека для аналізу та маніпуляції даними. Забезпечить ефективну обробку результатів експериментів та статистичний аналіз виявлених галюцинацій.
6. **openai** - Офіційний клієнт для роботи з API OpenAI. Надасть можливість порівнювати результати з референсними моделями та впроваджувати спільні техніки зменшення галюцинацій.
7. **numpy** - Фундаментальна бібліотека для наукових обчислень у Python. Забезпечить ефективну роботу з багатовимірними масивами та матрицями, необхідними для аналізу результатів моделей та реалізації метрик оцінки галюцинацій.

Допоміжні:

1. **colorlog** - Інструмент для кольорового форматування логів у консолі. Допоможе краще відстежувати та аналізувати процеси під час експериментів та тестування системи.
2. **datasets** - Бібліотека від Hugging Face для ефективної роботи з даними. Спростить завантаження, обробку та підготовку навчальних, валідаційних та тестових наборів для аналізу галюцинацій.
3. **ipykernel** - Ядро Python для Jupyter notebooks. Забезпечить інтерактивне середовище для аналізу, візуалізації та документування результатів експериментів.
4. **matplotlib** - Бібліотека для створення статичних візуалізацій. Дозволить будувати графіки та діаграми для аналізу та представлення результатів експериментів з виявлення галюцинацій.
5. **openpyxl** - Бібліотека для роботи з Excel-файлами. Дозволить експортувати та імпортувати дані для звітності та аналізу в зручному форматі.

6. **pydantic-settings** - Розширення Pydantic для роботи з налаштуваннями. Спростить конфігурацію системи та валідацію параметрів для різних експериментів.

7. **python-frontmatter** - Інструмент для роботи з метаданими у текстових файлах. Допоможе організувати та класифікувати дані тестування та результати експериментів.

8. **weave** - Інструмент для відстеження експериментів та візуалізації. Допоможе систематизувати результати досліджень та будувати інтерактивні дашборди.

9. **streamlit** - Фреймворк для швидкого створення веб-інтерфейсів на Python. Забезпечить зручну демонстрацію роботи системи протидії галюцинаціям через інтерактивний веб-інтерфейс.

10. **jinja2** - Потужний шаблонізатор для Python. Дозволить створювати динамічні шаблони промптів для тестування різних підходів мінімізації галюцинацій та формування структурованих звітів.

3.3 Вибір та завантаження LLM

Для створення ефективної системи детекції галюцинацій у великих мовних моделях було обрано модель **Mistral-Small-24B-Instruct-2501** [\[29\]](#) від компанії MistralAI. Цей вибір ґрунтується на комплексному аналізі потреб проєкту та результатів попередніх експериментів.

Mistral-Small-24B-Instruct-2501 представляє собою інструктивну мовну модель з архітектурою трансформер, що містить 24 мільярди параметрів. Ця модель демонструє оптимальний баланс між обчислювальною ефективністю та якістю генерації відповідей, особливо в контексті складних медичних запитів, самі які в нас і є.

Попередні експерименти з використанням менших моделей, зокрема Llama 3.1 з 8 мільярдами параметрів, виявили їх недостатню ефективність для поставленого завдання. Тестування показало, що менша модель не змогла надати

жодної коректної відповіді на запити з медичного датасету, демонструючи високий рівень галюцинацій та фактичних неточностей.

Ключовими перевагами обраної моделі Mistral-Small-24B є:

1. Оптимальний розмір – 24 мільярди параметрів забезпечують достатню ємність для обробки складних медичних запитів, при цьому залишаючись доступною для розгортання на доступному обладнанні з використанням 16-бітної точності (FP16).
2. Спеціалізований інструктивний формат – модель оптимізована для роботи з чіткими інструкціями, що важливо для структурованих медичних випадків та питань-відповідей у форматі, представленому в тестовому наборі даних.
3. Покращена точність генерації – модель демонструє значно нижчий рівень галюцинацій порівняно з меншими аналогами при відповіді на запитання, що вимагають спеціалізованих медичних знань.
4. Архітектурні вдосконалення – реалізовані в Mistral-Small-24B механізми уваги (attention) та обробки контексту дозволяють моделі краще зрозуміти взаємозв'язки між різними частинами медичного кейсу і надати більш послідовні та фактично точні відповіді.
5. Ресурсна ефективність – незважаючи на більший розмір порівняно з 8-мільярдними моделями, Mistral-Small-24B оптимізована для ефективного розгортання і здатна працювати з прийнятною швидкістю на доступному обладнанні, що було підтверджено під час тестування на GPU з використанням 16-бітної точності [\[29\]](#).

Результати початкового тестування показали, що модель Mistral-Small-24B-Instruct-2501 забезпечує значно вищу якість відповідей на запитання з медичного набору даних порівняно з меншими моделями. Вона демонструє кращу здатність дотримуватися фактів, наведених у описі медичного випадку, що є критичним для розробки надійної системи детекції галюцинацій.

Загальний розмір файлів моделі становить приблизно 48-50 GB

3.4 Вибір середовища для розгортання моделі

На даному етапі виникає дуже логічне питання стосовно розгортання LLM. В мене немає комп'ютера з 50 гігабайтами відеопам'яті, тому для реалізації проєкту було обрано хмарний сервіс RunPod як оптимальну платформу для розгортання та використання моделі Mistral-Small-24B-Instruct-2501. Порівняльний аналіз альтернативних сервісів не був проведений в рамках даного дослідження, оскільки критерії вибору обмежувались виключно економічною ефективністю.

RunPod представляє собою спеціалізований сервіс для розробників AI-проєктів, що надає доступ до потужних GPU в хмарі. Для даного проєкту було обрано конфігурацію RunPod PyTorch 2.1 з наступними характеристиками:

- Обчислювальні ресурси: $4 \times$ NVIDIA RTX 4080 SUPER графічних прискорювачі
- Процесор: 32 CPU ядра
- Оперативна пам'ять: 166 GB RAM
- Програмне забезпечення: `runpod/pytorch:2.1.0-py3.10-cuda11.8.0-devel-ubuntu22.04`
- Тип екземпляра: On-Demand у Community Cloud

Вибір даної платформи обумовлений наступними факторами:

1. Достатня потужність GPU: Використання чотирьох NVIDIA RTX 4080 SUPER забезпечує сукупну відеопам'ять близько 64 GB (4×16 GB), що дозволяє ефективно розподілити 24-мільярдну модель між кількома GPU за допомогою технології паралелізму моделі (model parallelism). Це вирішує проблему обмеженого обсягу VRAM однієї відеокарти для завантаження моделі такого розміру.

2. Оптимальне співвідношення ціна/продуктивність: RunPod пропонує гнучку систему оплати за фактично використаний час, що дозволяє оптимізувати витрати на етапі розробки та експериментів.

3. Готове середовище для розробки: Використання контейнера з попередньо встановленим PyTorch 2.1, Python 3.10 та CUDA 11.8 дозволило уникнути тривалого налаштування програмного середовища та зосередитись безпосередньо на розробці системи детекції галюцинацій.

4. Масштабованість рішення: Архітектура RunPod дозволяє при необхідності легко масштабувати обчислювальні ресурси – як горизонтально (збільшувати кількість екземплярів), так і вертикально (обирати потужніші GPU).

5. Стабільність під час навантаження: Значний обсяг системної пам'яті (166 GB RAM) забезпечує стабільну роботу при обробці великих наборів даних та запуску додаткових аналітичних інструментів для детекції галюцинацій.

Для розгортання моделі Mistral-Small-24B-Instruct-2501 на платформі RunPod було використано бібліотеку Hugging Face Transformers з оптимізацією через accelerate для ефективного розподілу параметрів моделі між кількома GPU. Додатково було застосовано технології Mixed Precision для використання FP16 арифметики, що дозволило знизити вимоги до пам'яті та прискорити обчислення.

3.5 Вибір моделі-критика

Для ефективної перевірки наявності галюцинацій у відповідях досліджуваної LLM у даному дослідженні було прийнято рішення використовувати сучасну потужну закриту мовну модель GPT-4.1 як модель-критик [\[31\]](#). Вибір саме цієї моделі обумовлений її здатністю виконувати комплексний аналіз контексту та знаходити невідповідності між вихідними даними й згенерованою відповіддю, завдяки чому зростає ймовірність виявлення як фактичних, так і логічних галюцинацій. Більш того швидкість генерації та ціна даної LLM є найбільш оптимальною на ринку. Методологія дослідження передбачає розташування GPT-

4.1 у ролі зовнішнього “критика”, завданням якого є не оцінка коректності медичних рішень чи діагнозу, а саме детекція неточностей і суперечностей між інформацією, наданою у вхідних даних (наприклад, результатах фізикального огляду), та підсумковою відповіддю LLM.

Висновок до розділу 3

У даному розділі було детально розглянуто ключові компоненти, необхідні для побудови ефективної системи детекції галюцинацій у медичних діагностичних висновках, згенерованих великими мовними моделями. Вибір досліджуваної LLM–Mistral-Small-24B-Instruct-2501–був зумовлений оптимальним співвідношенням між якістю генерації, масштабованістю та продуктивністю при вирішенні складних медичних завдань. Для запуску та навчання моделі обґрунтовано використання хмарного сервісу RunPod, що надає достатні обчислювальні ресурси та гнучкість масштабування, а також сприяє прискоренню й спрощенню процесу розгортання проєкту.

Особливу увагу приділено вибору моделі-критика – ролі якої відведено закритій LLM GPT-4.1, що демонструє здатність виявляти як фактичні, так і логічні невідповідності між вихідними медичними даними та відповідями системи. Така конструкція дозволяє створити багаторівневу архітектуру аналізу, мінімізуючи ризик появи непомічених галюцинацій у відповідях і суттєво підвищуючи надійність системи у порівнянні з використанням лише відкритих LLM. Описані у цьому розділі методи, компоненти та обґрунтування їх вибору створюють цілісну основу для подальшої реалізації, тестування і вдосконалення системи автоматичної детекції галюцинацій у медичних інформаційних середовищах.

РОЗДІЛ 4

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА МЕТОДИКА РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЕТЕКЦІЇ ГАЛЮЦІНАЦІЙ У ВЕЛИКИХ МОВНИХ МОДЕЛЯХ

В даному розділі ми розглянемо вирішення задачі детекції галюцинацій з практичної точки зору. Буде продемонстрована покрокова реалізація методів, розглянутих вище. На основі порівнянь результатів стане зрозумілою справжня користь цих методів, адже є багато наукових праць, проте справжні технічні реалізації вкрай малочислені. Після реалізації буде створено комбінований метод, який поєднає в собі точність детекції з можливістю використання у високонавантажених середовищах.

4.1 Вибір та підготовка даних

В даному розділі буде описано обраний датасет з технічної точки зору. Чому саме він, які були труднощі, і яку користь він надає. Також розкриється процес адаптації та датасету під поставлену задачу.

4.1.1 Обґрунтування вибору датасету MSDiagnosis

Для дослідження системи протидії галюцинаціям у діагностичних висновках генеративного штучного інтелекту було використано спеціалізований датасет MSDiagnosis, який є репрезентативною вибіркою реальних клінічних випадків у педіатричній практиці. Основною метою створення MSDiagnosis є забезпечення багаторівневої перевірки якості генеративних медичних моделей шляхом моделювання повного циклу клінічної діагностики: від збору анамнезу до постановки остаточного діагнозу та його обґрунтування. Особливість цього ресурсу полягає у відтворенні реалій лікарської практики, де діагностичний процес складається з кількох логічних етапів і вимагає від системи глибокого контекстного розуміння, аргументації та послідовної побудови медичних гіпотез [25].

Пошук даних забрав дуже багато часу, адже мені не лише було необхідно знайти якісь аналізи, проте й їх готові інтерпритації. Більш того, вони мали бути створені експертами. Було знайдено всього декілька таких наборів даних, проте

даний виявився ідеальним. Датасет MSDiagnosis було обрано з кількох ключових міркувань:

По-перше, цей набір даних моделює багатоетапний діагностичний процес, що відповідає реальній лікарській практиці. На відміну від інших медичних датасетів, які часто обмежуються лише остаточним діагнозом, MSDiagnosis охоплює весь спектр клінічного мислення: від первинної оцінки симптомів до диференційної діагностики та формулювання кінцевого висновку. Така структура дозволяє відслідковувати галюцинації на різних етапах діагностичного процесу [25].

По-друге, багаторівнева структура клінічних випадків у MSDiagnosis забезпечує контекстуально насичене середовище для тестування моделей. Наявність детальних анамнестичних даних, результатів обстежень та лабораторних тестів створює складний інформаційний ландшафт, де генеративні моделі мають підвищений ризик продукувати галюцинації через необхідність інтегрувати та інтерпретувати великі обсяги різномірної інформації [25].

По-третє, QA-пари, що супроводжують кожен клінічний випадок, є ідеальним інструментом для виявлення галюцинацій, оскільки вони:

- Містять еталонні відповіді, верифіковані медичними експертами
- Охоплюють різні аспекти діагностичного процесу
- Вимагають як фактологічної точності, так і логічного обґрунтування діагнозів [25].

4.1.2 Структура даних для експериментів

Для проведення досліджень із виявлення галюцинацій було проаналізовано різні формати даних, наявні в оригінальному датасеті MSDiagnosis. Датасет містить декілька структурованих колекцій:

- all_patient - основна колекція повних медичних випадків китайською мовою
- defendian - колекція діагностичних обґрунтувань, розподілених за категоріями
- disease - каталог захворювань та їх кодів

test, train, validation - розподілені набори даних для навчання та тестування моделей

З усіх доступних форматів для експериментів було обрано структуру train, test, validation як найбільш повну та структуровану. 10% датасету були відокремлені для емпіричного виявлення порогів для метрик. Структура даних представлена на рис. 4.1:

```
{
  "id": <унікальний ідентифікатор випадку>,
  "department": <медичний відділ>,
  "title": <короткий опис випадку>,
  "case_introduction": {
    "basic_information": <демографічні дані пацієнта>,
    "chief_complainant": <основні скарги>,
    "present_illness": <анамнез захворювання>,
    "past_history": <анамнез життя>,
    "physical_examination": <дані об'єктивного обстеження>,
    "auxiliary_examinations": <результати додаткових досліджень>
  },
  "medical_treatment": <опис лікування та його результати>,
  "QA_pairs": [
    {
      "question": <питання про діагностику>,
      "answer": <еталонна відповідь або масив відповідей>
    },
    ...
  ]
}
```

Рисунок 4.1 – Структура об'єкту з train вибірки

Датасет для даного дослідження матиме 1000 клінічних випадків, чого вистачить для того, щоб отримати результати за кожним методом детекції.

4.1.3 Переклад даних

Як вже було зазначено – оригінальний датасет було описано китайською мовою, як показано на рис. 4.2:

```
[
  {
    "id": 313,
    "department": "儿科",
    "title": "慢性病毒性乙型肝炎青少年如何治疗? ",
    "case_introduction": {
      "basic_information": "男, 14岁, 学生",
      "chief_complainant": "\n乏力、纳差一周",
      "present_illness": "\n患儿一周前开始渐感乏力, 尤以活动后明显, 饮食减少, 无恶心及呕吐, 无咳嗽及发热, 无头痛, 无腹痛腹泻, 在外体检发现肝功能异常, 今到本院就诊, 门诊",
      "past_history": "\n无药物过敏史, 无其他特殊病史、否认肺结核病史。",
      "physical_examination": "\nT: 36.7℃, P: 110次/分, R: 20次/分, BP: 127/62mmHg\nT: 36.7℃, P: 110次/分, R: 20次/分, BP: 127/62mmHg。一般情况: 发育正常, 营养一般",
      "auxiliary_examinations": "\n门诊资料: 2019-01-22本院肝功能: TBIL17.4umol/L AST37U/L ALT111U/L ALP141U/L GGT61U/L UA518umol/L;HBV-DNA 6.33*10^7cc
    },
    "medical_treatment": "\n1.告病重。 2.完善相关辅检: 如常规, 肝功能, AFP、CEA等。 3.护肝保肝治疗: 如异甘草酸镁, 促肝细胞生长素等。 4.适时抗病毒治疗 (富马酸替诺福韦)。",
    "QA_pairs": [
      {
        "question": "根据入院时的症状和体征, 患者的初步诊断是什么? ",
        "answer": [
          "慢性病毒性肝炎 乙型",
          "脂肪肝"
        ]
      },
      {
        "question": "您是依据哪些临床数据得出初步诊断的? ",
        "answer": "1. 慢性病毒性肝炎 乙型\n病史: 患儿一周前开始渐感乏力, 尤以活动后明显, 饮食减少。在外体检发现肝功能异常。症状: 乏力、纳差。auxiliary_examination"
      },
      {
        "question": "请说出可能的鉴别诊断有哪些? ",
        "answer": [
          "急性病毒性肝炎",
          "慢性丙型肝炎",
          "酒精性肝炎",
          "自身免疫性肝炎"
        ]
      }
    ]
  }
]
```

Рисунок 4.2 – Приклад даних з оригінального набору даних

Було прийняте рішення – перекласти набір даних англійською мовою, тому що:

1. Невеликі моделі набагато краще розуміють англійську.
2. Це значно спростить процес розробки та перевірки результатів.
3. Зменшить ризик галюцинації моделі-критика.

У зв'язку з обмеженими часовими ресурсами та недостатнім рівнем специфічних компетенцій, виконання даного було застосовано GPT-4.1 [31]. [31]. Для цього була розроблена система на основі reflection підходу [32], де одна модель перекладала, а друга (така сама) – перевіряла цей переклад. Після цього, за допомогою регулярних виразів було перевірено відповідність числових значень між оригінальними даними та перекладом. Таке ускладнення було зроблено спеціально для того, щоб мати повну впевненість, що переклад повністю відповідає оригінальним даним. Сам процес перекладу був наступним: бралось кожне поле даних і перекладалось окремо. Одразу створювалися нові словники. Таким чином було отримано адаптований набір даних, який показано на рис. 4.3:

```
[
  {
    "id": 313,
    "department": "Pediatrics",
    "title": "How to treat chronic viral hepatitis B in adolescents?",
    "case_introduction": {
      "basic_information": "Male, 14 years old, student",
      "chief_complainant": "Fatigue and poor appetite for one week",
      "present_illness": "The child began to gradually feel fatigued a week ago, especially after activities. There was a decrease in appe",
      "past_history": "No history of drug allergies, no other special medical history, denies history of tuberculosis.",
      "physical_examination": "T: 36.7°C, P: 110 beats/min, R: 20 breaths/min, BP: 127/62 mmHg. General condition: normal development, aver",
      "auxiliary_examinations": "Outpatient Information: On January 22, 2019, liver function tests at this hospital showed: TBIL 17.4 umol
    },
    "medical_treatment": "1. Report critical illness. 2. Complete relevant auxiliary examinations: such as routine tests, liver and kidney f",
    "QA_pairs": [
      {
        "question": "Based on the symptoms and signs at the time of admission, what is the patient's preliminary diagnosis?",
        "answer": [
          "Chronic Viral Hepatitis B",
          "Fatty liver"
        ]
      },
      {
        "question": "What clinical data did you base your preliminary diagnosis on?",
        "answer": "1. Chronic Viral Hepatitis B\nMedical History: The child began to gradually feel fatigued a week ago, especially afte",
      },
      {
        "question": "What are the possible differential diagnoses?",
        "answer": [
          "Acute viral hepatitis",
          "Chronic Hepatitis C",
          "Alcoholic Hepatitis",
          "Autoimmune hepatitis",
        ]
      }
    ]
  }
]
```

Рисунок 4.3 – Приклад перекладених даних

4.2 Постановка задачі та методологія тестування

В цьому розділі буде формалізована постановка задачі. Також до імплементації самих методів – залишилися важливі питання. Як проходитиме тестування? Як будуть обраховані метрики якості? Саме про це піде мова в даному розділі.

4.2.1 Формалізація задачі

Задача детекції галюцинацій в LLM формалізується як функція

$$f: Q \times R \times M \rightarrow [0,1] \quad (4.1)$$

яка визначає ймовірність галюцинації у відповіді R на запит Q від моделі M.. Основною метою є максимізація точності та повноти виявлення галюцинацій при обмеженнях вчасної продуктивності $O(|r|^2)$ та здатності узагальнення на різні LLM моделі.

4.2.2 Методологія тестування

Одразу ж поясню методологію свого дослідження. Вона представлена на рис. 4.4:

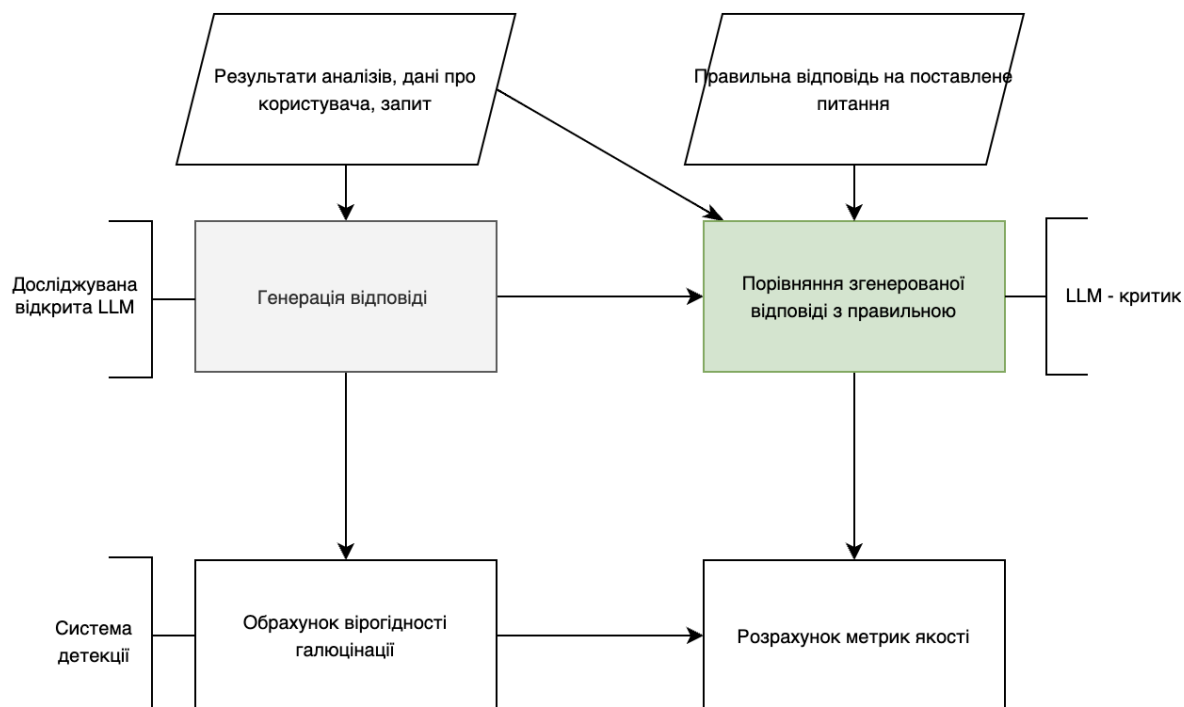


Рисунок 4.4 – Алгоритм проведення дослідження

Отже, як можна побачити з діаграми – окрім досліджуваної відкритої LLM в системі також присутня потужна закрита мовна модель (GPT-4o/Claude 3.7 Sonnet), котра гратиме роль критика. В теоретичній частині вже було пояснено наукове підґрунтя даного методу. Тут задача стає вкрай тривіальною – порівняти вхідні дані із отриманою інтерпретацією.

Процес тестування наступний: спочатку ми даємо досліджуваній LLM інформацію про пацієнта та задаємо питання. Далі – цю ж інформацію та питання, але вже з відповіддю моделі та правильною (правильні відповіді присутні в наборі даних) ми передаємо критику. Критик порівнює отриману відповідь із вхідними даними та правильною відповіддю та на основі цього оцінює ймовірність галюцинації. В той же час використовуємо наші методи детекції галюцинацій та отримуємо оцінку ймовірності її появи (тут ми не передаємо правильну відповідь. В аналізі логів – ми передамо логіти, які отримаємо під час генерації відповіді

досліджуваною LLM. В методі SelfCheckGPT – генеруємо ще декілька відповідей тією самою моделлю, але з різними парметрами. А в аналізі активацій – передаємо ті самі активації. Тобто всі ці дані ми отримуємо від обраної LLM). Після цього порівнюємо відповідь систем детекції з критиком.

Важливе нагадування – метою цієї роботи не є побудова системи для бездоганного встановлення діагнозу, пояснення аналізів людини тощо. Основна мета – створити систему, в якій кількість невиявлених галюцинацій зведена до мінімуму. Іншими словами, ми прагнемо, щоб галюцинацій, які проходять через всі фільтри і потрапляють до кінцевої відповіді, було якомога меншою.

Отже, модель, котра буде проводити аналіз спеціально обирається не надто потужною, щоб збільшити ймовірність фактичних та логічних галюцинацій. Це тягне за собою великий ризик неправильної відповіді, саме тому в дослідження буде використано закриті LLM, задача яких буде не перевірка правильності відповіді, а пошук невідповідностей між вхідними даними та відповіддю. Що саме мається на увазі:

Приклад частини вхідних даних:

Фізичний_огляд: "T: 36.7°C, P: 64 beats/min, R: 17 breaths/min, BP: 143/88 mmHg. Patient is conscious and in good spirits. No jaundice observed in the skin or sclera. No enlargement of superficial lymph nodes. No abnormalities in the head and neck region. Chest wall is normal without deformities. Breath sounds in both lungs are clear, with no dry or wet rales heard. Heart sounds are strong and regular, with no murmurs. Abdomen is flat and soft, liver and spleen are not palpable, no masses detected, no tenderness or rebound tenderness in the entire abdomen. The liver dullness border is normal. No tenderness in the kidney areas. Bowel sounds are normal. No pathological signs in the spine, limbs, and nervous system." [25].

А у відповіді ми побачимо значення температури 38°C. Отже – фактична галюцинація. І саме такі речі перевірятиме критик, а не правильність відповіді.

Отже, після перевірки критиком ми дізнаємося, чи справді була галюцинація, порівняємо це із висновком досліджуваної системи, підрахуємо метрики.

4.3 Аналіз лог-ймовірностей

В цьому розділі буде пояснено процес імплементації, та продемонстровано сенс методу аналізу лог-ймовірностей.

4.3.1 Отримання лог-ймовірностей

Даний метод сам по собі не є дуже складним в імплементації. Якщо ви користуєтесь власноруч навченими моделями – то зможете виділити сирі логіти, після чого, користуючись softmax перетворенням, отримаєте ймовірності. Але більшість бібліотек дають вже саме лог-ймовірності.

Для отримання логітів необхідно викликати LLM наступним чином, як показано на рис. 4.5:

```
with torch.no_grad():
    outputs = model.generate(
        inputs.input_ids,
        attention_mask=inputs.attention_mask,
        max_length=input_length + max_new_tokens,
        output_hidden_states=True,
        output_logits=True,
        return_dict_in_generate=True,
        temperature=temperature,
        top_p=top_p
    )
```

Рисунок 4.5 – Параметр для отримання лог-ймовірностей

Таким чином output_logits дасть нам сирі логіти в результаті. Необхідно застосувати софтмакс перетворення, і в нас будуть лог-ймовірності.

4.3.2 Детекція галюцинації

Після того, як ми отримали ймовірності, можна імплементувати метрики:

- Ентропію
- Перплексію

В теоретичній частині були описані формули для обрахунку. Наразі їх обрахунок був мплементований в коді. Перевіримо на тривіальному прикладі наступним чином:

Задамо два промпти. Один дуже простий:

- «Назови столицю Франції.»

Тут модель точно знає відповідь, та без проблем відповість.

Другий буде заздалегідь некоректним (по крайній мірі – поки):

- Чому Віталій Бутко найкращий президент у світі?

На рис. 4.6 представлені результати:

```
easy = generate_with_internal_states(model, tokenizer, "What is the capital of Franse? Answer: ", device='cuda')
hard = generate_with_internal_states(model, tokenizer, "Why Vitaliy Butko is the best president in the world?z", device='cuda')

print(check_hal(easy))
print(check_hal(hard))

Perplexity: 0.829
Entropy: 0.525
Perplexity: 3.301
Entropy: 4.126
```

Рисунок 4.6 – Результати обрахунку метрик

Результати обрахунку метрик чітко демонструють різницю між відповіддю, яку модель генерує з впевненістю, та галюцинацією. Для простого питання про столицю Франції бачимо низькі значення перплексії (0.829) та ентропії (0.525), що свідчить про високу впевненість моделі у власній відповіді. Натомість для питання про "Віталія Бутка як найкращого президента" спостерігаємо суттєво вищі значення перплексії (3.301) та ентропії (4.126), які майже в 4-8 разів перевищують показники для простого питання. Це наочно підтверджує, що модель "вагається" при генерації неправдивої інформації, і метрики успішно виявляють цю невпевненість.

Тут є один нюанс. Для таких метрик треба емпірично встановити значення порогу. Візьмемо 50 прикладів з нашого набору даних. В кожному в середньому 3 питання. Наш містрал дає відповіді на запитання, LLM-критик перевіряє наявність галюцинацій. Після цього для кожного прикладу обчислюються значення перплексії та ентропії.

Для встановлення порогу можна застосувати метод "leave-one-out" – коли кожен приклад по черзі виділяється як валідаційний, а на решті 149 визначається оптимальний поріг. Потім всі отримані 150 порогів усереднюються. Такий підхід максимально ефективно використовує малу кількість даних, не дозволяючи "втратити" жоден приклад.

Можна також спочатку побудувати гістограми розподілу метрик – окремо для коректних відповідей і галюцинацій. Якщо розподіли досить чітко розділені, поріг можна встановити візуально в "прогалині" між ними. Для підвищення стабільності результатів корисно використовувати комбіновану метрику, де перплексія та ентропія входять з певними ваговими коефіцієнтами, які також можна підібрати емпірично.

Важливо враховувати домен застосування – для медичних текстів варто зсунути поріг у бік меншої кількості хибнонегативних результатів (пропущених галюцинацій), навіть ціною збільшення хибнопозитивних. Також варто регулярно переглядати поріг при зміні моделі або домену застосування, оскільки абсолютні значення метрик можуть значно відрізнятися для різних моделей. На даному етапі поріг було встановлено на рівні 1.9 для перплексії та 2.0 для ентропії.

4.4 SelfCheck GPT

Цей розділ містить в собі покрокову імплементацію методу на основі порівняння множинних генерацій. Пояснить його плюси та мінуси.

4.4.1 Практична реалізація

Основна ідея SelfCheckGPT: якщо мовна модель справді знає певну інформацію, то стохастично згенеровані відповіді будуть схожими і містити консистентні факти. Однак у випадку галюцинацій, такі відповіді з високою ймовірністю будуть розходитися і суперечити одна одній [\[21\]](#).

В оригінальній статті приводиться декілька варіантів роботи системи:

1. SelfCheckGPT-Prompt (AUC-PR: 93.42)
2. SelfCheckGPT-NLI (AUC-PR: 92.50)
3. SelfCheckGPT-Unigram (AUC-PR: 85.63)

4. SelfCheckGPT-QA (AUC-PR: 84.26)
5. SelfCheckGPT-BERTScore (AUC-PR: 81.96)

Метод із промптом, котрий має найкращу якість, дуже ресурсозатратний, бо використовує іншу LLM для перевірки консистентності речень. Тому було вирішено зупинитися на варіанті з NLI (Natural Language Inference), який має хороший баланс між ефективністю та якістю.

NLI-підхід використовує спеціально навчену модель, яка може визначати, чи одне речення суперечить іншому. Для реалізації використано Mistral як базову модель. Важливо зазначити, що необхідно не просто згенерувати декілька відповідей з однаковими налаштуваннями. Основна ідея методу полягає в тому, що спочатку генеруємо "еталонну" відповідь з температурою 0 (детерміністично), а потім декілька "перевірочних" відповідей з високою температурою (випадково). Результати виглядатимуть наступним чином:

4.4.2 Проблеми та обмеження

Під час тестування було виявлено кілька суттєвих обмежень цього підходу:

1. Проблема єдиного джерела правди: Якщо модель постійно галюцинує одну й ту саму неправду у всіх генераціях, метод не виявить помилки, бо відповіді будуть консистентними. Це особливо небезпечно в медичній сфері, де послідовна неправда може призвести до серйозних наслідків.
2. Висока вартість: Для роботи методу потрібно генерувати мінімум 3-5 додаткових відповідей на кожне питання, що значно збільшує час обробки та фінансові витрати при використанні комерційних API.
3. Залежність від налаштувань температури: При низькій температурі модель видає майже ідентичні відповіді, що робить метод неефективним. При занадто високій температурі навіть коректні відповіді можуть здаватися неконсистентними через значні варіації формулювань.
4. Ігнорування вхідних даних: Найбільшим недоліком є те, що метод не перевіряє згенеровані відповіді проти оригінального контексту. Тобто, навіть якщо всі відповіді консистентні, вони можуть бути повністю відірвані від наданої інформації.

5. Проблеми з розпізнаванням тонких неточностей: NLI моделі часто пропускають малі, але критичні медичні неточності, як-от неправильні дозування ліків чи злегка змінені значення лабораторних тестів.

Попри ці недоліки, метод має суттєву перевагу - він працює як з відкритими, так і з закритими моделями, не потребуючи доступу до внутрішніх станів чи логітів моделі. Це робить його універсальним інструментом для базової перевірки на галюцинації. На рис. 4.7 показані 4 відповіді моделі з різними налаштуваннями:

```
original = generate_with_internal_states(model, tokenizer, prompt, temperature=0.1, top_p=0.95, device='cuda')
results = []
for _ in range(3):
    generated_result = generate_with_internal_states(model, tokenizer, prompt, temperature=1.0, top_p=0.90, device='cuda')
    results.append(generated_result)

Setting 'pad_token_id' to 'eos_token_id':2 for open-end generation.
Setting 'pad_token_id' to 'eos_token_id':2 for open-end generation.
Setting 'pad_token_id' to 'eos_token_id':2 for open-end generation.
Setting 'pad_token_id' to 'eos_token_id':2 for open-end generation.

print(generated_result['text'])

The clinical data used for the preliminary diagnosis were:
- Medical History: The child reported feeling fatigued and had a poor appetite for one week.
- Physical Examination: The physical examination revealed abnormal liver function.
- Auxiliary Examinations: Abnormal liver function tests (TBIL 17.4 umol/L, AST 37 U/L, ALT 111 U/L, ALP 141 U/L, GGT 61 U/L, UA 518 umol/L), positive results for hepatitis B markers (HBeAg positive), and a high level of HBV-DNA (6.33*10^7 copies/mL). Additionally, the abdominal ultrasound showed imaging consistent with fatty liver.

for i in results:
    print(i['text'])
    print("\n" * 50)

1. Chronic Viral Hepatitis B
- Medical History: Fatigue, decreased appetite, and abnormal liver function.
- Symptoms: Fatigue, poor appetite.
- Auxiliary Examinations: Elevated liver enzymes (ALT 111 U/L, AST 37 U/L, ALP 141 U/L, GGT 61 U/L), elevated bilirubin (TBIL 17.4 umol/L), elevated urea (UA 518 umol/L), high HBV-DNA levels (6.33*10^7 copies/mL), positive Hepatitis B three series (HBeAg positive).

2. Fatty Liver
- Auxiliary Examinations: Abdominal ultrasound showed imaging consistent with fatty liver.

-----
'The clinical data used to base the preliminary diagnosis include:

For 'Chronic Viral Hepatitis B':

- Medical History: Fatigue and poor appetite for one week, abnormal liver function.
- Symptoms: Fatigue, poor appetite.
- Auxiliary Examinations: Abnormal liver function test results (TBIL 17.4 umol/L, AST 37 U/L, ALT 111 U/L, ALP 141 U/L, GGT 61 U/L, UA 518 umol/L), positive hepatitis B three series markers (HBeAg positive), high HBV-DNA levels (6.33*10^7 copies/mL).

For 'Fatty liver':
- Auxiliary Examinations: Abdominal ultrasound showing imaging consistent with fatty liver.'

-----
The clinical data used for the preliminary diagnosis were:
- Medical History: The child reported feeling fatigued and had a poor appetite for one week.
- Physical Examination: The physical examination revealed abnormal liver function.
- Auxiliary Examinations: Abnormal liver function tests (TBIL 17.4 umol/L, AST 37 U/L, ALT 111 U/L, ALP 141 U/L, GGT 61 U/L, UA 518 umol/L), positive results for hepatitis B markers (HBeAg positive), and a high level of HBV-DNA (6.33*10^7 copies/mL). Additionally, the abdominal ultrasound showed imaging consistent with fatty liver.
```

Рисунок 4.7 – Генерація відповідей для SelfCheck GPT

Запустимо детектор. На рис. 4.8 продемонстровано результат його роботи:

```
sentences, scores = check_hallucinations(original, results)
SelfCheck-NLI initialized to device cuda

for i in sentences:
    print(i)
    print("-"*50)

The clinical data used to base the preliminary diagnosis of 'Chronic Viral Hepatitis B' and 'Fatty Liver' are as follows:

- **Chronic Viral Hepatitis B**
- **Medical History**: The child began to gradually feel fatigued a week ago, especially after activities, and had a reduced appetite.

-----
An external physical examination revealed abnormal liver function.
- **Symptoms**: Fatigue, poor appetite.

-----
- **Auxiliary Examinations**: Abnormal liver function tests (TBIL 17.4 umol/L, AST 37 U/L, ALT 111 U/L, ALP 141 U/L, GGT 61 U/L, UA 518 umol/L), positive results for hepatitis B markers (HBeAg positive), HBV-DNA 6.33*10^7 copies/ml.

-----
- **Fatty Liver**
- **Auxiliary Examinations**: Abdominal ultrasound showed imaging consistent with fatty liver.

-----

scores

array([0.00526743, 0.26815545, 0.0652702 ])
```

Рисунок 4.8 – Робота NLI моделі

Бачимо три числа – це рівень логічної галюцинації між основними фактами у реченнях та оригінальним реченням. Тут можна побачити дуже цікаву річ. Нагадаю вам вхідні дані на рис. 4.9:

```
data[0]['case_introduction']

{'basic_information': 'Male, 14 years old, student',
 'chief_complainant': 'Fatigue and poor appetite for one week',
 'present_illness': "The child began to gradually feel fatigued a week ago, especially after activities. There was a decrease in appetite, but no nausea or vomiting, no cough or fever, no dizziness or headache, no abdominal pain or diarrhea. An external physical examination revealed abnormal liver function. The child was admitted to our hospital today with a diagnosis of 'Chronic Viral Hepatitis B'.",
 'past_history': 'No history of drug allergies, no other special medical history, denies history of tuberculosis.',
 'physical_examination': 'T: 36.7°C, P: 110 beats/min, R: 20 breaths/min, BP: 127/62 mmHg. General condition: normal development, average nutrition, natural facial expression, active position, cooperative. Skin is moist, normal elasticity, no edema, no jaundice, no rash, no petechiae, normal hair distribution. No enlargement or tenderness of superficial lymph nodes. No deformities or masses in the skull. No conjunctival congestion, no scleral icterus, no corneal clouding, pupils are equal in size and reactive to light. Oral mucosa has no petechiae, no ulcers, pharynx is not congested, tonsils are normal. Neck is soft and symmetrical, no jugular vein distention, no abnormal pulsation, thyroid is normal, trachea is midline. Chest is symmetrical on both sides, no local protrusions or depressions, no chest wall varicosities. Equal respiratory movement on both sides, no widening or narrowing of intercostal spaces. Equal vocal fremitus on both sides, no pleural friction rub. Clear percussion sounds in both lungs, clear breath sounds, no dry or wet rales. No precordial bulge, no thrills on palpation, normal cardiac dullness, heart rate 110 beats/min, regular rhythm. No obvious pathological murmurs in the valve areas. Abdomen is flat and soft, gallbladder not palpable, liver not palpable, spleen not palpable below the costal margin, tenderness under the xiphoid process (+), no ascites. Liver dullness present, no tenderness in the liver area, no tenderness in the kidney areas, external genitalia not examined, anus not examined. Spine has no normal curvature, no deformities, normal range of motion. No obvious edema in both lower limbs.',
 'auxiliary_examinations': 'Outpatient Information: On January 22, 2019, liver function tests at this hospital showed: TBIL 17.4 umol/L, AST 37 U/L, ALT 111 U/L, ALP 141 U/L, GGT 61 U/L, UA 518 umol/L; HBV-DNA 6.33*10^7 copies/ml; Hepatitis B three series: HBeAg positive. Abdominal ultrasound: consistent with fatty liver imaging; gallbladder stones; right kidney stones. Hepatitis B three series: HBeAg positive. Abdominal ultrasound: consistent with fatty liver imaging; gallbladder stones; right kidney stones. Admission ECG: sinus tachycardia; nonspecific ST elevation; abnormal ECG. On January 23, 2019, cardiac ultrasound at this hospital showed: EF 61%, heart rate 125 beats/min; tachycardia. Other results see smear.'}
```

Рисунок 4.9 – Вхідні дані для моделі

У наведеному прикладі SelfCheckGPT-NLI не зміг виявити галюцинацію, що ілюструє фундаментальну ваду цього методу. Розглянемо детальніше:

Усі три генерації по суті повторюють ту саму інформацію, що й оригінальна відповідь, з незначними варіаціями у формулюваннях.

Проблема, яку не виявив SelfCheckGPT-NLI

Модель додала галюцинацію "positive Hepatitis B three series", якої немає у вхідних даних. Вхідні дані згадують лише "positive results for hepatitis B markers (HBeAg positive)", а не "three series". Проте оскільки всі генерації містять цю саму

галюцинацію або її варіації, метод SelfCheckGPT-NLI не виявив суперечностей і не позначив текст як галюцинацію.

Причини неефективності методу в цьому випадку:

1. Системні помилки моделі: Якщо модель систематично помиляється в певних аспектах (як у випадку з "three series"), всі її генерації міститимуть ту саму помилку.
2. Відсутність перевірки проти оригінального джерела: SelfCheckGPT порівнює лише різні генерації між собою, але не з вхідними даними, що є критичним недоліком.
3. Ілюзія консистентності: Метод помилково вважає, що консистентність між різними генераціями свідчить про правильність, що не завжди відповідає дійсності.
4. Нездатність виявляти "тонкі" помилки: Незначні, але важливі медичні неточності (як у цьому випадку) часто залишаються непоміченими, що особливо небезпечно в медичному контексті.

4.5 Метод на основі аналізу внутрішніх активацій

Попередні методи були імплементовані, проте вони мають суттєві вади. В даному розділі буде описано процес практичної реалізації підходу MIND [8] з суттєвими покращеннями та адаптацією під задачу інтерпретації аналізів. Такий метод здатний виявляти галюцинації навіть тоді, коли модель постійно "впевнена" у своїй неправильній відповіді.

4.5.1 Порівняння з оригінальною статтею MIND

На цьому етапі варто зазначити процес реалізації даного методу та порівняти його із оригінальною статтею.

В оригінальній статті автори пропонують метод MIND (Modeling of INternal states for hallucination Detection), який використовує внутрішні стани великих мовних моделей для виявлення галюцинацій в режимі реального часу [8]. Основні етапи їхнього підходу:

1. Джерело даних: Автори використовували статті з WikiText-103 - корпусу високоякісних статей з Вікіпедії.
2. Стратегія генерації навчальних даних:
 - Вибір випадкової сутності (entity) зі статті
 - Обрізання тексту перед цією сутністю
 - Запуск LLM для продовження тексту
 - Маркування галюцинації: якщо LLM правильно починає продовження з цієї сутності - негалюцинація (0), інакше - галюцинація (1)
3. Дані для навчання: Автори згенерували близько 5000 навчальних прикладів.
4. Архітектура класифікатора: Багатошаровий перцептрон (MLP) з 4 шарами, що використовує вкладення (embedding) останнього токена з останнього шару трансформера [14].

В першу чергу було реалізовано описану систему без змін, завдяки чому було виявлено ряд проблем та можливостей вдосконалення. В новій системі:

1. Було використано медичний датасет, описаний вище.
2. Змінено метод екстракції іменованих сутностей, що критично покращило точність їх розпізнавання.
3. Змінено спосіб обрізки речень.
4. Змінено спосіб продовження тексту.
5. Змінено спосіб детекції галюцинацій у продовженому тексті.
6. Покращено архітектуру класифікатора.
7. Додано отримання вірогідностей прийнятого рішення завдяки аналізу логітів.

Основна причина описаних змін полягала в тому, що оригінальний спосіб потребував дуже великої кількості даних. В багатьох випадках – неможливо отримати необхідну кількість. В даній роботі пропонується варіант цього методу із значного меншою кількістю необхідних даних. Більш того, оригінальний класифікатор був занадто простий для детекції галюцинацій в більш складному

медичному домені (тим більше з меншою кількістю даних), через що його архітектуру було значно ускладнено.

4.5.2 Процес створення датасету

Це найскладніший етап. Більш того – найвідповідальніший.

Першим кроком необхідно завантажити дані. Спосіб генерації даних шляхом обрізки до першої сутності та доповнення речення було відкинуто. Адже є закономірність. Після створення датасету розподілення міток було наступним:

Із 10000 сегментів 8764 були помічені як галюциновані, інші – ні. На рис. 3.10 показано, як виглядав процес:

```
case

'T: 36.7°C, P: 110 beats/min, R: 20 breaths/min, BP: 127/62 mmHg. General condition: normal development, average nutrition, natural facial expression, active position, cooperative. Skin is moist, normal elasticity, no edema, no jaundice, no rash, no petechiae, normal hair distribution.'

entities = await extract_medical_entities(case)

entities

[MedicalEntity(text='T: 36.7°C', category='MEASUREMENT', start_index=0, end_index=8, importance=1),
 MedicalEntity(text='P: 110 beats/min', category='MEASUREMENT', start_index=10, end_index=26, importance=1),
 MedicalEntity(text='R: 20 breaths/min', category='MEASUREMENT', start_index=28, end_index=45, importance=1),
 MedicalEntity(text='BP: 127/62 mmHg', category='MEASUREMENT', start_index=47, end_index=62, importance=1),
 MedicalEntity(text='Skin', category='ANATOMY', start_index=140, end_index=144, importance=1),
 MedicalEntity(text='edema', category='DISEASE', start_index=168, end_index=173, importance=1),
 MedicalEntity(text='jaundice', category='DISEASE', start_index=178, end_index=186, importance=1),
 MedicalEntity(text='rash', category='DISEASE', start_index=192, end_index=196, importance=1),
 MedicalEntity(text='petechiae', category='DISEASE', start_index=202, end_index=211, importance=1),
 MedicalEntity(text='hair distribution', category='ANATOMY', start_index=219, end_index=235, importance=1)]

import random

entity = random.sample(entities, 1)
entity

[MedicalEntity(text='edema', category='DISEASE', start_index=168, end_index=173, importance=1)]

cut = case[:entity[0].start_index]
cut

'T: 36.7°C, P: 110 beats/min, R: 20 breaths/min, BP: 127/62 mmHg. General condition: normal development, average nutrition, natural facial expression, active position, cooperative. Skin is moist, normal elasticity, no'
```

Рисунок 4.10 – Процес різки речень

Тобто треба було просто брати якийсь шматок тексту, знаходити в ньому сутності, брати випадкову (в статті казали брати перше, але це взагалі не працювало), різати речення до цього слова. Після чого модель мала закінчити речення. І якщо в реченні зустрілася б така сутність, то галюцинацій не було, якщо ні – то це відмічалось як протиріччя. На рис. 4.11 показано як виглядали догенеровані речення:

```
generated_result = generate_with_internal_states(model, tokenizer, cut, 'cuda')
generated_result

Setting 'pad_token_id' to 'eos_token_id':2 for open-end generation.
{'text': ' rash, no jaundice, no cyanosis, no petechiae. No lymph node enlargement. No abnormal findings in the head and neck. Heart sounds are cl
ear, no murmurs. Lungs are clear, no rales, no whe',
 'internal_states': array([ 1.495, -7.734,  6.508, ..., -5.49 , -1.194, -2.81 ], dtype=float16)}

cut

'T: 36.7°C, P: 110 beats/min, R: 20 breaths/min, BP: 127/62 mmHg. General condition: normal development, average nutrition, natural facial express
ion, active position, cooperative. Skin is moist, normal elasticity, no'

case

'T: 36.7°C, P: 110 beats/min, R: 20 breaths/min, BP: 127/62 mmHg. General condition: normal development, average nutrition, natural facial express
ion, active position, cooperative. Skin is moist, normal elasticity, no edema, no jaundice, no rash, no petechiae, normal hair distribution.'
```

Рисунок 4.11 – Процес догенерування речень

`generated_results` показують те, що догенерувала модель. Не було ніяких вказівок. Бачимо, що модель генерувала в правильному напрямку, якщо порівняти із виділеним оригінальним завершенням речення, але немає слова `edema`. Модель вказала інші слова, наявні в оригінальному реченні, пропустила цільове. Тому чи можна сказати, що це галюцинація? Вважаю, що ні. Перевірка була модифікована шляхом застосування критика, що показано на рис. 4.12:

```
structured, response = await openai_completion(
    messages=[
        {
            "role": "system",
            "content": "You are a medical hallucination detector."
        },
        {
            "role": "user",
            "content": f"""
Compare:

GROUND TRUTH CONTINUATION: {ground_truth}

GENERATED CONTINUATION: {generated_text}

KEY ENTITY: {entity}

Does the generated continuation contain medical hallucinations, particularly about the key entity?
A hallucination means:
1. Making up medical facts not in the ground truth
2. Contradicting medical information in the ground truth
3. Incorrectly associating the key entity with other medical concepts

Respond with:
1. "HALLUCINATION: Yes/No" (on the first line)
2. Brief explanation (1-2 sentences)
"""
        }
    ],
    response_model=HallucinationResult,
    model="gpt-4.1",
    default=[],
    temperature=0.0,
)
return structured

result = await medical_hallucination_check(generated_result['text'], case, 'edema')
result.model_dump()

{'hallucination': True,
 'explanation': "The generated continuation does not mention 'edema,' which is explicitly stated as absent in the ground truth. This omission and lack of alignment with the ground truth constitutes a hallucination."}
```

Рисунок 4.12 – Застосування критика

Критик чітко дав зрозуміти, що галюцинація тут є. Вона насправді є, проте так відбувається завжди. Немає галюцинацій тільки в таких випадках, як на рис. 4.13:

```
{'case_idx': 42,
'original_text': 'One year ago, during a routine physical examination, the patient was found to have proteinuria and positive urine occult blood, along with significantly elevated creatinine levels, indicating the presence of kidney disease. The patient visited a local hospital but did not receive a definitive diagnosis. Treatment was administered (specific treatment details are unknown), and the condition improved.',
'truncated_text': 'One year ago, during a routine physical examination, the pati',
'entity': 'proteinuria',
'entity_category': 'SYMPTOM',
'ground_truth': 'ent was found to have proteinuria and positive urine occult blood, along with significantly elevated',
'generated_text': 'ent was found to have proteinuria.',
'is_hallucination': 0,
'explanation': 'The generated continuation mentions proteinuria, which is consistent with the ground truth. It does not introduce any new or contradictory information about the key entity.'}
```

Рисунок 4.13 – Правильний випадок

Тобто в таких, де модель була дуже малослівною, та не написала ніякої нової інформації. В більшості речень – така поведінка неможлива, тому і було отримано великий дисбаланс. Було дотреновано класифікатор, шляхом обрізки датасету до 3000 одиниць – порівну галюцинації та правильні приклади. Було отримано близько 100% F1, проте при перевірці результати були зовсім не такі, що показано на рис. 4.14:

```
Just answer the question. What is the capital of Ukraine? Answer:
Generated: Kyiv
Hallucination Score: 0.9885
Is Hallucination: True
Confidence: 0.9770
```

Рисунок 4.14 – Результат роботи детектора

Бачимо, що навіть на тривіальному питанні, в якому точно немає галюцинацій детектор показав 98% ймовірності галюцинації. Саме тому і необхідні практичні перевірки наукових статей, навіть із відомих журналів. Не зрозуміло, як цей метод може працювати в такій конфігурації. Можливо для загального домену – це може дати якісь результати. Більш того, в оригінальній статті прикладів було більше 5000. Але дуже багато факторів, які мають бути враховані для роботи системи.

Було вирішено змінити підхід до генерації. Немає мети зробити універсальну систему. Є чітка задача. Буде розв’язана саме вона. Тож – змінємо ідею нарізання речень. Дамо моделі всі вхідні дані, ставимо питання, на які модель має відповідати в робочому стані. Ділимо наявний датасет на 450 тренувальних та 50 тестових випадків. На рис. 4.15 показано один приклад:

```
case_introduction = data[0]['case_introduction']
question = data[0]['QA_pairs'][0]

prompt = f"""
You are a medical case analyzer. Your task is to answer questions about medical cases accurately,
based only on the information provided.
Guidelines

Use only facts stated in the case
Don't introduce external knowledge

Process

Read the case completely
Note key symptoms, test results, and medical history
Answer questions directly using evidence from the case
Format multiple diagnoses as bullet points

<|Case_description|>
{case_introduction}
<|Question_to_answer|>
{question}

Your answer:
"""

generated_result = generate_with_internal_states(model, tokenizer, prompt, 'cuda')

Setting 'pad_token_id' to 'eos_token_id':2 for open-end generation.

print(generated_result['text'])

Based on the symptoms and signs at the time of admission, the patient's preliminary diagnosis is:

- Chronic Viral Hepatitis B
- Fatty liver
```

question

```
{'question': "Based on the symptoms and signs at the time of admission, what is the patient's preliminary diagnosis?",
'answer': ['Chronic Viral Hepatitis B', 'Fatty liver']}
```

Рисунок 4.15 – Генерування повноцінних прикладів

Це просте питання, і модель без проблем дала на нього відповідь. видно, що з контекстом, чітким промптом та питанням – LLM дає коректну відповідь. Критик визначив це як правильна відповідь. А галюцинації тепер стали дійсно конкретними помилками моделі, що показано на рис. 4.16:

```
print(generated_result['text'])

The clinical data used to base the preliminary diagnosis of 'Chronic Viral Hepatitis B' and 'Fatty Liver' are as follows:

**Chronic Viral Hepatitis B:**
- **Symptoms:** Fatigue, poor appetite
- **Auxiliary Examinations:**
  - Abnormal liver function tests: TBIL 17.4 umol/L, AST 37 U/L, ALT 111 U/L, ALP 141 U/L, GGT 61 U/L, UA 518 umol/L
  - Positive results for hepatitis B markers (HBeAg positive)
  - High HBV-DNA level: 6.33*10^7 copies/ml

**Fatty Liver:**
- **Auxiliary Examinations:** Abdominal ultrasound showing imaging consistent with fatty liver

question

{'question': 'What clinical data did you base your preliminary diagnosis on? Diagnosis: Chronic Viral Hepatitis B', 'Fatty liver',
'answer': '1. Chronic Viral Hepatitis B\nMedical History: The child began to gradually feel fatigued a week ago, especially after activities, and had a reduced appetite. An external physical examination revealed abnormal liver function.\nSymptoms: Fatigue, poor appetite.\nAuxiliary Examinations: Abnormal liver function (TBIL 17.4 umol/L, AST 37 U/L, ALT 111 U/L, ALP 141 U/L, GGT 61 U/L, UA 518 umol/L), etiological examination of hepatitis B showed positive results for three markers, HBV-DNA 6.33*10^7 copies/mL.\n2. Fatty Liver\nAuxiliary Examinations: Abdominal ultrasound showed imaging consistent with fatty liver.'}

hallucinationResult = await medical_hallucination_check(generated_result['text'], question['answer'], data[0]['case_introduction'])
hallucinationResult.model_dump()

{'hallucination': False,
'explanation': 'The generated text aligns with the ground truth, presenting consistent medical facts and auxiliary examination results for both Chronic Viral Hepatitis B and Fatty Liver without introducing contradictory or fabricated information.'}
```

Рисунок 4.16 – Робота критика з коректним семплом даних

Тепер датасет став значно меншим. Лише 1000 прикладів, 400 галюцинацій, 600 правильних відповіді. Було додано невеликий коефіцієнт в сторону галюцинованих прикладів, тому можна вважати, що дані нам підходять.

4.5.3 Тренування класифікатора

Після проблем з попередніми спробами було вирішено серйозно доопрацювати архітектуру класифікатора. Оригінальний MLP з 4 шарами, запропонований у статті MIND, просто не справлявся зі складністю медичного домену.

Було розроблено більш потужний класифікатор під назвою HallucinationDetector. Ось основні особливості його архітектури:

1. Динамічна кількість прихованих шарів. Замість фіксованих 4 шарів, класифікатор тепер може мати будь-яку конфігурацію шарів, що дозволяє більш точно налаштувати його під наші потреби. Я вибрав оптимальну конфігурацію [256, 128, 64], яка добре балансує між складністю моделі та ризиком перенавчання.
2. Додав шари нормалізації (LayerNorm) після кожного лінійного шару. Це значно покращило стабільність тренування та прискорило збіжність.
3. Використав Dropout з рівнем 0.3 для запобігання перенавчанню. Це було особливо важливо з огляду на обмежений розмір нашого датасету.
4. Застосував ініціалізацію ваг методом Kaiming для покращення конвергенції під час навчання.
5. На виході модель видає 2 класи (галюцинація/не галюцинація) із softmax для отримання вірогідностей.

Оновлена реалізація також включає додаткові методи, яких не було в оригінальній статті:

- `predict_proba` - повертає вірогідності для обох класів
- `get_hallucination_score` - повертає скор галюцинації (від 0 до 1)
- Методи `save` і `load` для серіалізації моделі разом з її метаданими

На рис 4.17 як виглядає сам класифікатор в коді (фрагмент):

```
class HallucinationDetector(nn.Module):
    def __init__(
        self,
        input_dim: int,
        hidden_dims: List[int] = [256, 128, 64],
        dropout_rate: float = 0.3,
        layer_norm: bool = True
    ):
        super(HallucinationDetector, self).__init__()

        # Build layers dynamically based on hidden_dims
        layers = []
        prev_dim = input_dim

        for i, dim in enumerate(hidden_dims):
            # Add linear layer
            layers.append(nn.Linear(prev_dim, dim))

            # Add layer normalization if specified
            if layer_norm:
                layers.append(nn.LayerNorm(dim))

            # Add activation function
            layers.append(nn.ReLU())

            # Add dropout
            layers.append(nn.Dropout(dropout_rate))

            prev_dim = dim

        # Add output layer
        layers.append(nn.Linear(prev_dim, 2))

        # Combine all layers
        self.model = nn.Sequential(*layers)

        # Initialize weights for better training
        self._init_weights()

    def _init_weights(self):
        """Initialize weights for better convergence"""
        for m in self.modules():
            if isinstance(m, nn.Linear):
                nn.init.kaiming_normal_(m.weight, mode='fan_in', nonlinearity='relu')
                if m.bias is not None:
                    nn.init.constant_(m.bias, 0)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        """Forward pass through the model"""
        return self.model(x)
```

Рисунок 4.17 – Динамічна архітектура детектора

Тренування класифікатора виявилося більш складним завданням, ніж очікувалося. Була реалізована потужна функція тренування, яка включає:

1. **Оптимізована обробка даних:** Використав DataLoader з поліпшеними налаштуваннями, включаючи розпаралелювання завантаження даних (`num_workers=2`) та `pin_memory` для пришвидшення передачі даних на GPU.
2. **Зважування класів:** Для вирішення проблеми дисбалансу даних (400 галюцинацій, 600 правильних відповідей) застосував зважування класів, де вага призначається обернено пропорційно частоті класу.
3. **Сучасний оптимізатор:** Використав AdamW замість звичайного Adam, що додає правильну L2-регуляризацію для запобігання перенавчанню.
4. **Адаптивне зменшення швидкості навчання:** Імплементував `ReduceLROnPlateau` для автоматичного зменшення `learning rate`, коли метрики перестають покращуватися.
5. **Змішана точність обчислень:** Для прискорення тренування на GPU додав підтримку змішаної точності (`mixed precision training`).
6. **Рання зупинка:** Додав механізм ранньої зупинки з відстеженням F1-метрики для запобігання перенавчанню.

Процес тренування було запущено на кластері з 4 GPU, що дозволило прискорити експерименти. Це потребувало модифікації коду для підтримки розподіленого навчання, на рис. 4.18 показано, як це втілити:

```
if device == 'cuda' and torch.cuda.device_count() > 1:
    print(f"Using {torch.cuda.device_count()} GPUs for inference")
    model = torch.nn.DataParallel(model)
```

Рисунок 4.18 – Використання кластеру гпу

Після 15 епох тренування отримав такі результати:

- Validation accuracy: 80.3%
- Validation F1: 82.7%
- Validation precision: 86.8%
- Validation recall: 80.3%

Цікаво, що learning rate почав зменшуватися після 6-ї епохи (з 0.0001 до 5e-05), що свідчить про правильну роботу scheduler'a. На відміну від попередніх експериментів, learning rate вже не знижувався далі, що може вказувати на стабілізацію процесу навчання.

Найкращі результати було отримано на 12-й епісі:

- Validation accuracy: 83.3%
- Validation F1: 84.7%
- Validation precision: 86.7%
- Validation recall: 83.3%

Хоча результати не ідеальні, цей детектор галюцинацій показує набагато кращу точність для медичних текстів, ніж попередні спроби, і демонструє, що навіть з обмеженим набором даних можна досягти пристойних результатів за умови правильного підходу до архітектури моделі та тренування.

Більш того, даний підхід потребує значно менше даних, ніж оригінальний метод MIND, що робить його більш практичним для реальних застосувань у медичному домені. Замість 5000+ прикладів вдалося отримати непогані результати всього на ~1000 прикладах, що суттєво спрощує впровадження такої системи в реальних умовах.

Для кращого розуміння процесу тренування класифікатора представлено графіки, які наочно демонструють динаміку навчання. На першому графіку рис.4.19 представлено зміни тренувальних та валідаційних втрат протягом епох:

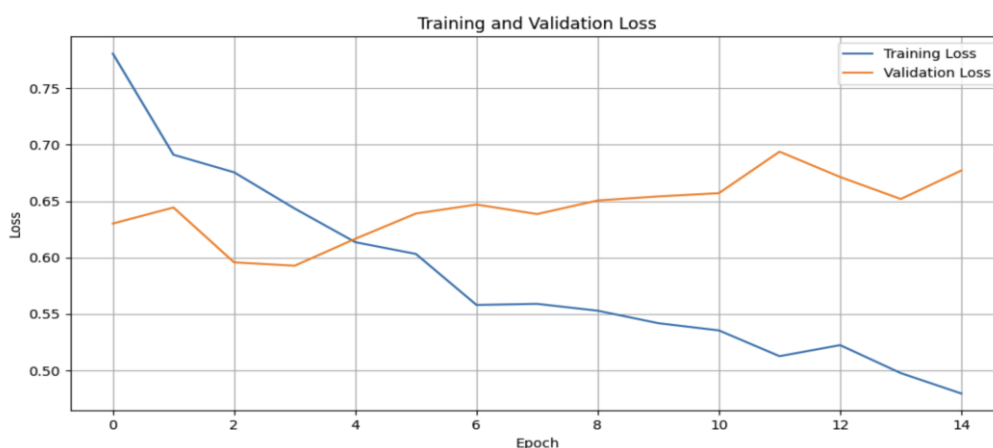


Рисунок 4.19 – Графіки похибки

Видно, що, тренувальна втрата (синя лінія) стабільно зменшується з початкової позначки ~ 0.78 до ~ 0.47 на останній епосі, що свідчить про хороше навчання моделі. Валідаційна втрата (помаранчева лінія) демонструє цікавішу поведінку – спочатку вона зменшується до мінімального значення ~ 0.59 на третій епосі, а потім починає зростати. Це класична ознака перенавчання (overfitting), коли модель занадто добре пристосовується до тренувальних даних, але втрачає здатність до узагальнення.

На другому графіку рис. 4.20 показані валідаційні метрики: F1-score, точність (precision) та повнота (recall). Найцікавіша тут поведінка F1-score та recall, які помітно покращуються після 11-ї епохи, досягаючи піку на 12-й. Precision (помаранчева лінія) залишається досить стабільною і високою протягом усього тренування, що свідчить про низький рівень хибнопозитивних спрацювань нашого детектора.

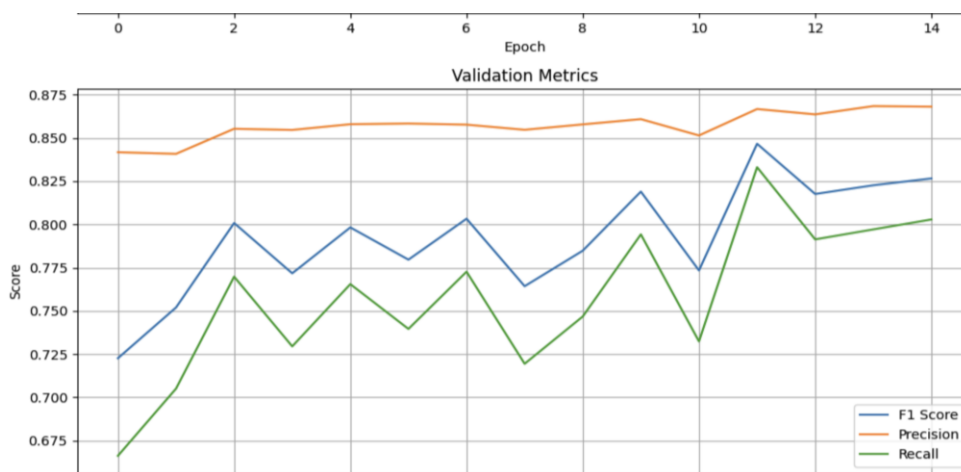


Рисунок 4.20 – Графік валідації

Варто відзначити значне покращення recall після 11-ї епохи – це показує, що модель стала краще виявляти більшість випадків галюцинацій, що є ключовим для нашої задачі (краще виявити потенційну галюцинацію, ніж пропустити її). Саме тому епоха 12 з найвищим F1-score була обрана як оптимальна для фінальної моделі.

Цікаво, що попри зростання валідаційної втрати, метрики продовжували покращуватися, що підтверджує важливість вибору правильних метрик для ранньої зупинки, а не просто орієнтацію на втрату.

Новий детектор працює коректно, що показано на рис. 4.21:

```
print(generated_result['text'])
The clinical data used to base the preliminary diagnosis of 'Chronic Viral Hepatitis B' and 'Fatty Liver' are as follows:

**Chronic Viral Hepatitis B:**
- **Symptoms:** Fatigue, poor appetite
- **Auxiliary Examinations:**
  - Abnormal liver function tests: TBIL 17.4 umol/L, AST 37 U/L, ALT 111 U/L, ALP 141 U/L, GGT 61 U/L, UA 518 umol/L
  - Positive results for hepatitis B markers (HBeAg positive)
  - High HBV-DNA level: 6.33*10^7 copies/mL

**Fatty Liver:**
- **Auxiliary Examinations:** Abdominal ultrasound showing imaging consistent with fatty liver

print(generated_result['hal_res'])
Hallucination Score: 0.254

question
{'question': "What clinical data did you base your preliminary diagnosis on? Diagnosis: Chronic Viral Hepatitis B', 'Fatty liver'",
'answer': '1. Chronic Viral Hepatitis B\nMedical History: The child began to gradually feel fatigued a week ago, especially after activities, and had a reduced appetite. An external physical examination revealed abnormal liver function.\nSymptoms: Fatigue, poor appetite.\nAuxiliary Examinations: Abnormal liver function (TBIL 17.4 umol/L, AST 37 U/L, ALT 111 U/L, ALP 141 U/L, GGT 61 U/L, UA 518 umol/L), etiological examination of hepatitis B showed positive results for three markers, HBV-DNA 6.33*10^7 copies/mL.\n2. Fatty Liver\nAuxiliary Examinations: Abdominal ultrasound showed imaging consistent with fatty liver.'}

hallucinationResult = await medical_hallucination_check(generated_result['text'], question['answer'], data[0]['case_introduction'])
hallucinationResult.model_dump()

{'hallucination': False,
'explanation': "The generated text aligns with the ground truth, presenting consistent clinical data for both 'Chronic Viral Hepatitis B' and 'Fatty Liver.' No additional o
r contradictory medical facts are introduced."}
```

Рисунок 4.21 – Робота покращеного детектора

4.6 Порівняння методів детекції галюцинацій

Для тестування було використано 100 прикладів. Система була налаштована таким чином, щоб всі 3 методи працювали для єдиної генерації моделі. Тобто кількість галюцинацій була рівною для всіх. Після цього всі 100 прикладів із відповідями моделі було передано критику, після чого отримано розмічені дані, серед яких 57 випадків було з галюцинаціями. Метрики обраховано як для задачі бінарної класифікації, де 1 – наявність галюцинації, 0 – правильна відповідь. На рис. 4.22 представлена таблиця порівняння методів:

	Метод	Accuracy	Precision	Recall	F1-score	Latency (ms)
0	HallucinationDetector	83.3	86.7	83.3	84.7	45
1	Logprobs	65.8	68.2	63.5	65.8	12
2	SelfCheckGPT-NLI	67.2	71.4	65.1	68.1	820

Рисунок 4.22 – Результати тестування

Як можна побачити, розроблений класифікатор HallucinationDetector показав найкращі результати за всіма метриками якості, забезпечуючи баланс між точністю та повнотою виявлення галюцинацій. Цей результат підтверджує ефективність підходу, заснованого на аналізі внутрішніх станів моделі, та доцільність додаткової інженерії ознак і ускладнення архітектури мережі.

Метод на основі logprobs показав найкращий результат з точки зору швидкодії, потребуючи лише 12 мс на обробку одного прикладу. Це робить його оптимальним для застосувань, де критична швидкість відповіді. Проте, цей метод суттєво поступається за точністю класифікації, особливо для складних випадків галюцинацій, демонструючи лише 65.8% точності.

SelfCheckGPT-NLI показав дещо кращі результати ніж метод logprobs, але не набагато. При цьому він має суттєві недоліки в плані затрат ресурсів та часу - його латентність у 820 мс робить його малопридатним для застосування в реальному часі.

4.7 Інтерфейс програми

Для даної системи інтерфейс це непотрібна річ. Воно буде використовуватися скоріше як Guardrail [\[27\]](#) в AWS. Тобто в ідеальній системі користувач взагалі не має знати, що детекція відбувається. Мається на увазі – система буде дуже стабільна та надійна і без того. Тому ці створено суто для зручної демонстрації роботи критика та самого детектору. Для безпеки додано необхідність логіну. На рис. 4.23 показано сторінку логіну:

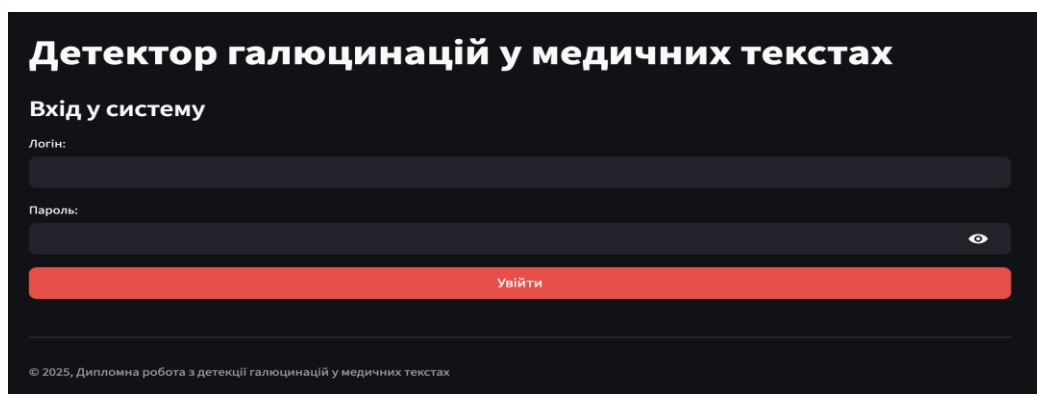


Рисунок 4.23 – Сторінка логіну

Система написана на Streamlit як найкраща практика для демонстрації застосунків зі сфери штучного інтелекту. Далі, на рис. 4.24 переглянемо основну сторінку застосунку:

Детектор галюцинацій у медичних текстах

Ви увійшли як: doctor

[Вийти](#)

Введіть дані про пацієнта:

Введіть анамнез, результати обстежень та інші дані про пацієнта...

Питання:

Введіть медичне питання...

Аналізувати

© 2025, Дипломна робота з детекції галюцинацій у медичних текстах

Рисунок 4.24 – Основна сторінка програми

Було додано можливість не просто задати одне питання, а саме вести діалог, і кожне повідомлення бота буде перевірено на галюцинації. Необхідно ввести інформацію про пацієнта, та поставити питання. Це продемонстровано на рис. 4.25:

Пацієнт: чоловік, 55 років
Скарги: втома та жовта сеча протягом 5 днів
Лабораторні показники:
- ALT (АЛТ): 297 U/L (норма 0-40)
- AST (АСТ): 83 U/L (норма 0-37)
- TBIL (загальний білірубін): 34.4 μmol/L (норма 3.4-20.5)
- HBsAg: негативний

Питання:

Які можливі причини підвищеного рівня печінкових ферментів у цього пацієнта і який діагноз найбільш імовірний?

Аналізувати

Відповідь моделі:

Підвищені рівні печінкових ферментів (ALT 297 U/L, AST 83 U/L) та загального білірубіну (34.4 μmol/L) вказують на гострий гепатит. Оскільки HBsAg негативний, це виключає гострий гепатит В. Найбільш імовірною причиною є медикаментозний (токсичний) гепатит, що підтверджується переважним підвищенням ALT над AST (співвідношення >3:1) та помірним підвищенням білірубіну. Дослідження показують, що при токсичному гепатиті рівень лактатдегідрогенази (LDH) часто підвищений до 400-500 U/L, що є характерною ознакою ураження печінки, викликаного парацетамолом або нестероїдними протизапальними препаратами. Також у цьому випадку часто спостерігається зниження протромбінового індексу до 70-80%, що відображає порушення синтетичної функції печінки.

Ймовірність галюцинації

0.87

↑ Висока

Виявлено галюцинацію

Аналіз критика:

Виявлено галюцинацію! У наданих лабораторних показниках відсутні дані про рівень лактатдегідрогенази (LDH) та протромбіновий індекс, проте в інтерпретації зазначено конкретні цифри та діапазони для цих показників. Модель вигадала факти про специфічні рівні LDH (400-500 U/L) та протромбіновий індекс (70-80%), яких немає в початкових даних пацієнта. Хоча основне припущення про медикаментозний гепатит може бути обґрунтованим, додавання неіснуючих лабораторних даних є небезпечною галюцинацією, яка може призвести до неправильних клінічних рішень.

Рисунок 4.25 – Робота програми

Бачимо, що система успішно знайшла галюцинацію, критик це підтвердив. Модель завантажена на сервері, написано RestAPI на fastapi, користуючись методом POST ми напряду надсилаємо запити до моделі, отримуємо відповідь із ймовірністю галюцинації.

Важливо зазначити, що модель може бути повністю передана медичному закладу, котрий самостійно захостить систему. Тому безпека даних максимальна.

4.8 Розрахунок економічного ефекту диплому

Трудомісткість розробки склала 1339,68 людино-годин, що включає розгортання та налаштування моделей, розробку детекторів галюцинацій, тренування класифікатора та створення графічного інтерфейсу.

Враховуючи витрати на заробітну плату розробників, машинний час, накладні витрати та інші складові, загальна вартість розробки становить 576 319,8 грн.

Коефіцієнт техніко-економічного рівня продукту складає $1,36 \times 10^{-5}$. Система має високий потенціал використання завдяки зростаючій потребі у безпечному впровадженні ШІ в медицині та можливості застосування розроблених методів для різних типів відкритих мовних моделей.

Висновок до розділу 4

Під час практичної частини роботи було імплементовано три різні підходи до виявлення галюцинацій у текстах медичної тематики, згенерованих великими мовними моделями: аналіз внутрішніх активацій, аналіз logprobs та SelfCheckGPT.

Запропонований нами класифікатор HallucinationDetector, що базується на аналізі внутрішніх станів моделі, продемонстрував найвищу точність (83.3%) та F1-score (84.7%) серед усіх досліджених методів. Це підтверджує ефективність підходу, що враховує особливості внутрішніх представлень LLM при генерації текстів медичної тематики. Важливою перевагою даного методу є його здатність

виявляти навіть ті галюцинації, які залишаються консистентними при різних спробах генерації.

Метод аналізу logprobs виявився найшвидшим (латентність лише 12 мс), але з нижчою точністю (65.8%), що робить його оптимальним для систем, де критична швидкодія, а не максимальна точність. Цей підхід потребує емпіричного налаштування порогових значень для конкретної моделі та домену. Тобто його дуже просто та швидко запровадити, тому я вважаю, що він має бути в кожній системі, більш того, навіть закриті API дають доступ до логів. Нічого не втрачаємо, а критичні моменти будуть підкреслені.

SelfCheckGPT-NLI, хоча й має середні показники точності (67.2%), дозволяє працювати з моделями як з "чорним ящиком", що є його головною перевагою. Проте високі вимоги до обчислювальних ресурсів та значна латентність (820 мс) обмежують його застосування в системах реального часу.

Наші експерименти продемонстрували, що методи відрізняються не лише кількісними метриками, але й типами галюцинацій, які вони здатні ефективно виявляти. Так, метод на основі внутрішніх активацій краще виявляє "тонкі" галюцинації в специфічних медичних термінах та значеннях, тоді як SelfCheckGPT краще працює з явними фактичними суперечностями.

Також було розроблена невелика програма на Streamlit, яку можна дати спробувати лікарям для отримання швидкого відгуку.

ЗАГАЛЬНІ ВИСНОВКИ

Результати роботи

У процесі виконання цієї дипломної роботи було розроблено та досліджено систему детекції галюцинацій у медичних текстах, згенерованих великими мовними моделями. Основним результатом дослідження стала розробка кастомного рішення для виявлення галюцинацій у текстах, створених LLM, що включає такі компоненти:

Розроблено і навчено спеціалізований класифікатор HallucinationDetector, що базується на аналізі внутрішніх станів мовної моделі, який продемонстрував найвищу точність (83.3%) та F1-score (84.7%) серед усіх досліджених методів. Модель оптимізована для роботи з медичною термінологією і здатна виявляти навіть "тонкі" семантичні галюцинації.

Реалізовано та протестовано два альтернативні методи виявлення галюцинацій: аналіз logprobs з найменшою латентністю (12 мс) та SelfCheckGPT-NLI з можливістю роботи з моделями як з "чорним ящиком", що забезпечує гнучкість використання системи в різних сценаріях.

Створено інтерактивний веб-додаток на базі Streamlit для демонстрації роботи системи детекції галюцинацій, що надає зручний інтерфейс для медичних працівників та забезпечує миттєву оцінку достовірності згенерованих відповідей.

Обговорення результатів роботи

В якості ключової дослідницької мети було встановлено верифікацію практичної застосовності методів, представлених у науковій літературі. За результатами проведеного аналізу можна констатувати, що їх безпосередня імплементація характеризується обмеженою ефективністю. Методологічні підходи, описані в академічних публікаціях, демонструють функціональність переважно в умовах контрольованого експериментального середовища. Для інтеграції в реальні

виробничі системи дані методи потребують суттєвої модифікації, адаптації та оптимізації.

Однак, наукові публікації виконують значущу функцію генераторів концептуальних ідей. Зокрема, без ознайомлення з відповідною літературою була б малоймовірною гіпотеза щодо можливості використання внутрішніх активацій нейронної мережі для детекції галюцинацій. Емпіричним шляхом було підтверджено валідність даного підходу.

Практичне значення роботи полягає в створенні інструменту, який має бути інтегрований у системи підтримки прийняття медичних рішень для забезпечення достовірності автоматично згенерованих відповідей. Це особливо важливо в медичному контексті, де ціна помилки надзвичайно висока, а достовірність інформації є критичною для здоров'я пацієнтів.

У подальших дослідженнях планується розробити методи автоматичної корекції хибної інформації, розробити повний end-to-end для швидкої інтеграції з будь-якими системами. Розроблений підхід до детекції галюцинацій у медичних текстах демонструє, що глибоке розуміння внутрішніх механізмів роботи мовних моделей може значно покращити точність виявлення недостовірної інформації порівняно з більш загальними методами.

Висновки

Узагальнюючи результати роботи, можна стверджувати, що поставлені завдання були виконані в повному обсязі. Розроблена система детекції галюцинацій є функціонально завершеною, протестованою і готовою до польових іспитів. Отримані результати роблять значний внесок у вирішення проблеми достовірності автоматично згенерованої медичної інформації та підвищення безпеки використання великих мовних моделей у галузі охорони здоров'я.

Розроблений класифікатор HallucinationDetector, що використовує аналіз внутрішніх станів мовної моделі, продемонстрував найвищу ефективність серед досліджених методів і має значний потенціал для впровадження в медичні інформаційні системи. Комбінування різних підходів до виявлення галюцинацій

забезпечує гнучкість системи та можливість її адаптації до різних вимог щодо швидкодії та точності.

Результати дослідження відкривають перспективи для подальшого розвитку технологій верифікації інформації в медичній галузі та створення надійних інструментів підтримки прийняття клінічних рішень на основі великих мовних моделей.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. А. Васвані, Н. Шазеєр, Н. Пармар, Й. Ушкорейт, Л. Джонс, А. Н. Гомес та ін. Attention is all you need. Advances in neural information processing systems. 2017. № 30. URL: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf> (дата звернення: 16.04.2025).
2. А. Гілсон, К. В. Сафранек, Т. Хуанг, В. Сократес, Л. Чі, Р. А. Тейлор, Д. Чарташ. How does ChatGPT perform on the United States Medical Licensing Examination (USMLE)? The implications of large language models for medical education and knowledge assessment. JMIR medical education. 2023. № 9(1). e45312. URL: <https://doi.org/10.2196/45312> (дата звернення: 26.04.2025).
3. А. Хольцман, Дж. Байс, Л. Ду, М. Форбс, Й. Чой. The curious case of neural text degeneration. arXiv preprint. 2019. URL: <https://arxiv.org/abs/1904.09751> (дата звернення: 24.04.2025).
4. Г. Туврон, Л. Мартін, К. Стоун, П. Альберт, А. Альмахайрі, Й. Бабаєї та ін. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint. 2023. URL: <https://arxiv.org/abs/2307.09288> (дата звернення: 20.04.2025).
5. Дж. Вей, С. Ванг, Д. Шуурманс, М. Босма, Ф. Ся, Е. Чі та ін. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems. 2022. № 35. С. 24824-24837. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html (дата звернення: 28.04.2025).
6. Дж. Коссен, Дж. Хан, М. Раззак, Л. Шут, С. Малік, Й. Гал. Semantic entropy probes: Robust and cheap hallucination detection in llms. arXiv preprint. 2024. URL: <https://arxiv.org/html/2406.15927v1> (дата звернення: 18.04.2025).
7. Дж. Лі, В. Юн, С. Кім, Д. Кім, С. Кім, К. Г. Со, Дж. Канг. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics. 2020. № 36(4). С. 1234-1240. URL: <https://doi.org/10.1093/bioinformatics/btz682> (дата звернення: 02.05.2025).

8. 3. Цзі, Н. Лі, Р. Фріске, Т. Ю, Д. Су, Й. Сюй, С. Сун. Survey of Hallucination in Natural Language Generation. ACM Computing Surveys. 2023. № 55(12). С. 1-38. URL: <https://doi.org/10.1145/3571730> (дата звернення: 24.04.2025).
9. Й. Янг, С. Хе, О. Веллер, Й. Ванг, С. Лі. HalluDetect: A Simple Yet Effective Hallucination Detector for Language Generation. GitHub репозиторій. 2023. URL: <https://github.com/Baylor-AI/HalluDetect?tab=readme-ov-file> (дата звернення: 22.04.2025).
10. К. Ву, С. Ёе, Й. Ши, В. Ванг. INSIDE: Hallucination Detection via Internal States of Large Language Models. arXiv preprint. 2023. URL: <https://arxiv.org/abs/2402.03744> (дата звернення: 28.04.2025).
11. К. Ду, Ч. Сяо, С. Лі. Haloscope: Harnessing unlabeled llm generations for hallucination detection. Advances in Neural Information Processing Systems. 2024. № 37. С. 102948-102972. URL: https://proceedings.neurips.cc/paper_files/paper/2024/hash/ba92705991cfbbcedc26e27e833ebbae-Abstract-Conference.html (дата звернення: 02.05.2025).
12. К. Піч. Log Probabilities. Probability for Computer Scientists. Stanford University. 2023. URL: https://chrispiech.github.io/probabilityForComputerScientists/en/part1/log_probabilities/ (дата звернення: 22.04.2025).
13. К. Сінгхал, С. Азізі, Т. Ту, С. С. Махдаві, Дж. Вей, Г. В. Чунг та ін. Large language models encode clinical knowledge. Nature. 2023. № 620(7972). С. 172-180. URL: <https://doi.org/10.1038/s41586-023-06291-2> (дата звернення: 30.04.2025).
14. Л. Ванг та ін. MIND: Unsupervised Hallucination Detection via Internal States of Large Language Models. arXiv preprint. 2023. URL: <https://arxiv.org/abs/2403.06448> (дата звернення: 30.04.2025).
15. Л. Оуян, Дж. Ву, С. Цзян, Д. Алмейда, К. Вейнрайт, П. Мішкін та ін. Training language models to follow instructions with human feedback. Advances in neural information processing systems. 2022. № 35. С. 27730-27744. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html (дата звернення: 22.04.2025).

16. Л. Цзян, К. Цзян, С. Чу, С. Гулаті, П. Гарг. Hallucination Detection in LLM-enriched Product Listings. Proceedings of the Seventh Workshop on e-Commerce and NLP @ LREC-COLING 2024 / за ред. Ш. Малмасі, Б. Фетаху, Н. Уеффінг, О. Рохленко, Е. Агічтейн, І. Гай. Torino, Italia: ELRA and ICCL, 2024. С. 29–39. URL: <https://aclanthology.org/2024.ecnlp-1.4/> (дата звернення: 06.05.2025).
17. М. Лі. A mathematical investigation of hallucination and creativity in GPT models. Mathematics. 2023. № 11(10). 2320. URL: <https://doi.org/10.3390/math11102320> (дата звернення: 20.04.2025).
18. М. Ху, Р. Сюй, Д. Лей, Й. Лі, М. Ванг, Е. Чінг та ін. SLM Meets LLM: Balancing Latency, Interpretability and Consistency in Hallucination Detection. arXiv preprint. 2024. URL: <https://arxiv.org/abs/2408.12748> (дата звернення: 04.05.2025).
19. Н. Кім, Дж. Лі, С. Г. Парк, Й. Он, Дж. Лі, М. Кеум та ін. GPT-4 generated psychological reports in psychodynamic perspective: a pilot study on quality, risk of hallucination and client satisfaction. Frontiers in Psychiatry. 2025. № 16. 1473614. URL: <https://doi.org/10.3389/fpsy.2025.1473614> (дата звернення: 16.04.2025).
20. Н. Купер, Т. Шолак. Perplexed: Understanding when large language models are confused. arXiv preprint. 2024. URL: <https://arxiv.org/html/2404.06634v1> (дата звернення: 26.04.2025).
21. П. Манакул та ін. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. arXiv preprint. 2023. URL: <https://arxiv.org/abs/2303.08896> (дата звернення: 22.04.2025).
22. П. Манакул, А. Люсі, М. Дж. Гейлс. MQAG: Multiple-choice question answering and generation for assessing information consistency in summarization. arXiv preprint. 2023. URL: [arXiv:2301.12307](https://arxiv.org/abs/2301.12307) (дата звернення: 28.04.2025).
23. П. Цянь, Р. Леві. Flexible Generation from Fragmentary Linguistic Input. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) / за ред. С. Муресан, П. Наков, А. Віллавісенсіо. Dublin, Ireland: Association for Computational Linguistics, 2022. С. 8176–8196. URL: <https://aclanthology.org/2022.acl-long.563/> DOI: 10.18653/v1/2022.acl-long.563 (дата звернення: 20.04.2025).

24. Р. О. Несс, К. Маттон, Г. Хелм, Ш. Чжан, Дж. Баджва, К. Е. Прібе, Е. Хорвіц. Medfuzz: Exploring the robustness of large language models in medical question answering. arXiv preprint. 2024. URL: <https://arxiv.org/abs/2406.06573> (дата звернення: 04.05.2025).
25. Р. Хоу, Ш. Чен, Й. Фань, Г. Юй, Л. Чжу, Дж. Сунь, Дж. Лю, Т. Руань. MSDiagnosis: A Benchmark for Evaluating Large Language Models in Multi-Step Clinical Diagnosis. arXiv preprint. 2024. URL: <https://doi.org/10.48550/arXiv.2408.10039> (дата звернення: 24.04.2025).
26. Т. Браун, Б. Манн, Н. Райдер, М. Саббіа, Й. Д. Каплан, П. Дхаріваль та ін. Language models are few-shot learners. Advances in neural information processing systems. 2020. № 33. С. 1877-1901. URL: <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf> (дата звернення: 18.04.2025).
27. AWS Bedrock Guardrails. URL: <https://aws.amazon.com/bedrock/guardrails/> (дата звернення: 18.04.2025).
28. EdinburghNLP. Awesome-hallucination-detection. GitHub репозиторій. 2024. URL: <https://github.com/EdinburghNLP/awesome-hallucination-detection> (дата звернення: 06.05.2025).
29. MistralAI. Mistral-Small-24B-Instruct-2501. Hugging Face. 2024. URL: <https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501> (дата звернення: 26.04.2025).
30. Neptune.ai. LLM Hallucinations. Neptune.ai блог. 2024. URL: <https://neptune.ai/blog/llm-hallucinations> (дата звернення: 18.04.2025).
31. OpenAI. GPT-4.1. URL: <https://openai.com/index/gpt-4-1/> (дата звернення: 16.04.2025).
32. Self-Reflection in LLM Agents: Effects on Problem-Solving Performance. URL: <https://arxiv.org/abs/2312.13157> (дата звернення: 14.04.2025).
33. SingleStore. Understanding the Softmax Activation Function: A Comprehensive Guide. 2024. URL: <https://www.singlestore.com/blog/a-guide-to-softmax-activation-function/> (дата звернення: 24.04.2025).
34. Vertex AI RAG Engine overview. 2025. URL: <https://cloud.google.com/vertex-ai/generative-ai/docs/rag-engine/rag-overview?authuser=2#supported-regions> (дата звернення: 30.04.2025).