Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Метод виявлення та оцінки маніпулятивних стратегій у відповідях великих мовних моделей (LLM)
(тема)

Виконав:
здобувач другого року навчання,
групи ДСм-24-1
Топорков О. К.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Науки про дані (Data Science)
(повна назва спеціалізації)
Керівник проф. Шевченко О. Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Лариса ЧАЛА
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Науки про дані (Data Science)
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Топоркову Олегу Костянтиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод виявлення та оцінки маніпулятивних стратегій у відповідях великих мовних моделей (LLM)

затверджена наказом університету від 24 листопада 20 25 р. № 1057Ст

2. Термін подання студентом роботи до екзаменаційної комісії 16 грудня 20 25 р.

3. Вихідні дані до роботи: наукові публікації та стандарти з тематики виявлення маніпулятивних стратегій у текстах, відкриті розмічені датасети з пропагандистських/маніпулятивних повідомлень (SemEval, Propaganda Techniques Corpus, інші корпуси з багатокласовою розміткою), приклади відповідей великих мовних моделей (LLM), технічна документація до обраних мовних моделей та інструментів (OpenAI API, FAISS тощо), а також методичні вказівки кафедри щодо виконання кваліфікаційної роботи.

4. Перелік питань, що потрібно опрацювати в роботі

1) Теоретичні та методологічні засади виявлення маніпулятивних стратегій у текстовій комунікації

2) Підходи та методи автоматичного виявлення маніпуляцій у текстах

3) Проєктування та обґрунтування методу виявлення маніпулятивних відповідей LLM

4) Побудова корпусу та векторного представлення маніпулятивних текстів

5) Програмна реалізація методу та інтеграція його у вигляді сервісу аналізу текстів

6) Експериментальне дослідження та аналіз результатів розробленої системи

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	24.11.2025	виконано
2	Аналіз літературних джерел та постановка задачі	27.11.2025	виконано
3	Огляд даних та формування датасетів	30.11.2025	виконано
4	Проєктування методу виявлення маніпуляцій	02.12.2025	виконано
5	Побудова ембеддингів та векторного індексу	04.12.2025	виконано
6	Реалізація програмних компонентів	07.12.2025	виконано
7	Проведення експериментів	09.12.2025	виконано
8	Написання розділів пояснювальної записки	11.12.2025	виконано
9	Оформлення роботи згідно з вимогами	12.12.2025	виконано
10	Підготовка доповіді та презентації до захисту	14.12.2025	виконано
11	Рецензування	16.12.2025	виконано
12	Захист перед ЕК	17.12.2025	

Дата видачі завдання 24 листопада 20 25 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Шевченко О. Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 83 с., 6 рис., 6 табл., 6 дод., 20 джерел.

АВТОМАТИЧНИЙ АНАЛІЗ ТЕКСТУ, ВЕКТОРНІ ПРЕДСТАВЛЕННЯ ТЕКСТУ, ВЕЛИКІ МОВНІ МОДЕЛІ, ВИЯВЛЕННЯ МАНІПУЛЯЦІЙ, МАНІПУЛЯТИВНИЙ КОНТЕНТ, РИТОРИЧНІ МАНІПУЛЯЦІЇ.

Об’єкт дослідження – текстові відповіді великих мовних моделей та інші короткі текстові повідомлення, у яких можуть міститися риторичні маніпуляції.

Предмет дослідження – методи виявлення та інтерпретації маніпулятивних стратегій у тексті на основі векторних представлень та великих мовних моделей.

Мета роботи – розробити інтерпретований метод і концепцію програмної системи для виявлення маніпулятивних стратегій у текстових відповідях LLM із використанням корпусу маніпулятивних прикладів та аналізу LLM-«судді».

Методи дослідження – методи обробки природної мови, побудова векторних представлень текстів, пошук найближчих сусідів у векторному просторі, методи промптінгу та структурованого виводу великих мовних моделей, а також загальні підходи програмної інженерії для побудови сервісів аналізу тексту.

У роботі розглянуто теоретичні засади риторичних маніпуляцій та базові поняття адекватності роботи моделей ШІ. На основі кількох відкритих датасетів сформовано уніфікований корпус маніпулятивних текстів та запропоновано метод виявлення маніпулятивних стратегій, що поєднує векторний пошук з аналізом великої мовної моделі у форматі структурованого виводу.

ABSTRACT

Master’s thesis contains: 83 pp., 6 fig., 6 tabl., 6 ann., 20 references.

AUTOMATIC TEXT ANALYSIS, LARGE LANGUAGE MODELS, MANIPULATION DETECTION, MANIPULATIVE CONTENT, RHETORICAL MANIPULATIONS, TEXT VECTOR REPRESENTATIONS.

Object of the study – textual responses of large language models and other short text messages that may contain rhetorical manipulations.

Subject of the study – methods for detecting and interpreting manipulative strategies in text based on vector representations and large language models.

Aim of the work – to develop an interpretable method and conceptual software system for detecting manipulative strategies in LLM-generated text responses using a corpus of manipulative examples and an LLM judge analysis.

Research methods – natural language processing methods, construction of vector text representations, nearest-neighbour search in vector space, prompting and structured output of large language models, as well as general software engineering approaches for building text analysis services.

The thesis considers the theoretical foundations of rhetorical manipulations and basic notions of AI model adequacy. Based on several open datasets, a unified corpus of manipulative texts is constructed and a method for detecting manipulative strategies is proposed, which combines vector-space retrieval with analysis by a large language model in a structured-output format.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Ембеддинг – embedding – числове векторне представлення тексту;

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – програмний інтерфейс взаємодії;

ECE – Expected Calibration Error – очікувана похибка каліброваності;

FAISS – Facebook AI Similarity Search – бібліотека пошуку найближчих векторів;

F1 – F1-score – гармонійне середнє між точністю та повнотою;

GPT – Generative Pre-trained Transformer – генеративний попередньо натренований трансформер;

HTTP – HyperText Transfer Protocol – протокол передавання гіпертексту;

JSON – JavaScript Object Notation – формат обміну даними;

k-NN – k-Nearest Neighbors – метод k найближчих сусідів;

LLM – Large Language Model – велика мовна модель;

LSTM – Long Short-Term Memory – рекурентна нейронна мережа з довготривалою пам’яттю;

ML – Machine Learning – машинне навчання;

NLP – Natural Language Processing – обробка природної мови;

REST – Representational State Transfer – стиль побудови веб-сервісів;

Redis – високопродуктивне пам’яткове сховище / черга повідомлень;

UI – User Interface – користувацький інтерфейс;

URL – Uniform Resource Locator – уніфікований локатор ресурсу в мережі.

ВСТУП

Стрімкий розвиток великих мовних моделей (LLM) відкрив нові можливості для автоматизованого опрацювання природної мови – від діалогових систем і пошуку до підтримки прийняття рішень. Водночас зросли ризики, пов’язані з маніпуляціями в інформаційному просторі: поширенням дезінформації, прихованою агітацією, емоційним тиском, підміною причинно-наслідкових зв’язків і семантичними викривленнями. Своєчасне виявлення таких маніпулятивних стратегій є критично важливим для медіагігієни суспільства, прозорої комунікації в організаціях та безпеки цифрових сервісів.

Попри високу якість генерації, LLM не гарантовано коректно розрізняють маніпулятивні та неманіпулятивні висловлювання, особливо в умовах багатошумових даних, перефразувань, контекстних підказок чи навмисних «ін’єкцій підказок». Тому постає завдання оцінити адекватність роботи LLM-моделі з метою виявлення маніпулятивних стратегій – не лише за класичними метриками точності, а й з урахуванням каліброваності ймовірностей, стійкості до атак і зсувів даних, здатності утримуватися від відповіді за невизначеності, прозорості рішень та етичних обмежень.

Актуальність теми зумовлена зростанням обсягів користувацького та згенерованого контенту, ускладненням маніпулятивних тактик та потребою інституцій у верифікованих, надійних і відтворюваних підходах до оцінювання моделей, що інтегруються у робочі процеси з мінімізацією хибних спрацювань.

Інформаційна база дослідження може включати відкриті й власні корпуси текстів, анотації експертів/асистоване розмічування, відкриті LLM-інтерфейси або локальні інференс-рушії, репліковні експериментальні скрипти та журнали проведення дослідів.

Наукова новизна полягає у:

– узгодженні таксономії маніпулятивних стратегій із протоколом багатокритеріальної оцінки LLM;

– введенні режимів перевірки стійкості, що імітують реальні вороже-контекстні сценарії;

– практичних рекомендаціях щодо налаштування методу визначення маніпуляцій і схеми «людина-в-циклі» для контрольованого компромісу між метриками.

Практичне значення роботи полягає у можливості застосування отриманих результатів у різних галузях, пов’язаних з аналізом та обробкою інформації. Зокрема, розроблена методологія може бути використана для модерації контенту в медіа-платформах і комунікаційних сервісах, що дозволяє своєчасно виявляти та фільтрувати маніпулятивні повідомлення. Крім того, результати дослідження можуть бути впроваджені в аналітичні системи оцінки репутаційних ризиків для організацій і брендів, а також у фактчекінгові ініціативи та програми медіаграмотності, спрямовані на підвищення критичного мислення користувачів. Окрім цього, напрацьовані підходи можуть стати основою для побудови безпечних AI-сервісів, які відповідають сучасним вимогам прозорості, відтворюваності та контролю помилок у процесі автоматизованого прийняття рішень.

1 ТЕОРЕТИЧНІ ТА МЕТОДОЛОГІЧНІ ЗАСАДИ ВИЯВЛЕННЯ МАНІПУЛЯТИВНИХ СТРАТЕГІЙ У ТЕКСТОВІЙ КОМУНІКАЦІЇ

1.1 Маніпулятивні стратегії в текстовій комунікації: визначення, таксономії, приклади, сфери застосування

Маніпулятивна стратегія – це комунікативна тактика прихованого впливу, що використовується мовцем для прихованої зміни свідомості або поведінки адресата у напрямку, вигідному мовцю. Відкриті форми переконання зазвичай апелюють до раціональних аргументів або етики, тоді як маніпуляція діє приховано, експлуатуючи психологічні вразливості реципієнта. Важливою ознакою є прихованість намірів: якщо об’єкт маніпуляції усвідомлює зовнішній вплив, ефект значно слабшає або й зовсім зникає. Саме тому маніпулятори часто застосовують мовні прийоми, замасковані під звичайну інформацію або навіть під дружню пораду, аби уникнути викриття своїх справжніх цілей.

У сучасній лінгвістиці та теорії комунікації описано різні таксономії маніпулятивних стратегій. Дослідники виділяють макростратегії (глобальні наміри мовця) та мікростратегії/тактики (конкретні мовні прийоми) [1]. Наприклад, для політичного дискурсу характерні макростратегії легітимації (створення позитивного образу «своїх») та дискредитації (очорнення опонентів), які досягаються через набір конкретних тактик [1]. За ступенем спотворення інформації маніпуляції умовно поділяють на фактологічні (перекручування або вибіркове подання фактів) та ідеологічно-поляризуючі (наголошування протилежностей «свої» vs «чужі») [1].

Приклади маніпулятивних тактик досить різноманітні. У таблицях технік пропаганди, запропонованих дослідниками, наводиться до 15–20 різновидів риторичних прийомів [2]. Серед найбільш поширених можна виділити наступні.

Навішування ярликів (labeling) – вживання упереджених визначень або прізвиськ для об’єкта впливу з метою сформувати стереотипний образ (наприклад, назвати опонента «екстремістом» чи «невігласом» без обґрунтування) [3]. Такий прийом закріплює за ціллю певну негативну (або позитивну) характеристику ще до аналізу фактів.

Апеляція до страху (appeal to fear) – підкреслення загрозливих наслідків або нагнітання страху з метою змусити аудиторію діяти або думати певним чином. Наприклад, у повідомленні можуть перебільшуватися небезпеки («Якщо не підтримати цей закон – країну охопить хаос») для емоційного тиску на читача.

Перебільшення та викривлення – навмисне перебільшення фактів, використання гіпербол чи однобоких тверджень. Маніпулятор може подавати поодинокий випадок як тенденцію або цитувати цифри без контексту, щоб створити потрібне враження [4].

Повторення меседжів – багаторазове повторювання однієї тези чи слогану з метою закріпити її в свідомості аудиторії. Часте повторення, навіть спростованої інформації, підвищує її впізнаваність і позірну правдивість – відомий ефект, який використовує пропаганда («скажіть брехню тисячу раз – і вона стане правдою») [3].

«Whataboutism» (відволікання уваги типу «а як щодо…»): реакція на критику шляхом переведення розмови на іншу тему або висування зустрічної претензії. Замість відповіді по суті маніпулятор ставить зустрічне запитання «А в вашій країні хіба не порушують права?» – тим самим відволікаючи від початкової критики та уникаючи прямої відповіді [2].

Ці приклади ілюструють, як маніпулятивні стратегії проявляються в текстах. Сфери застосування таких стратегій дуже широкі. У політичній комунікації маніпулятивні прийоми використовуються в передвиборчих промовах, дебатах, геополітичній пропаганді [1]. В мас-медіа – при поданні новин з упередженим нахилом або в замовних статтях, де потрібне формування певної громадської думки [4]. В рекламі та PR маніпулятивні

стратегії зосереджені на створенні у споживача емоційної потреби в товарі (наприклад, через апеляцію до престижу чи страху щось пропустити). Навіть в міжособистісному спілкуванні зустрічаються тактики маніпуляції (лестощі, провина, газлайтинг тощо) для досягнення особистої вигоди. Таким чином, розуміння і виявлення маніпулятивних стратегій є актуальним завданням у різних галузях – від аналізу медіаконтенту до кібербезпеки (виявлення фішингових повідомлень, що маніпулюють довірою користувача).

1.2 Великі мовні моделі (LLM). Принципи, сильні/слабкі сторони, ризики у задачах виявлення маніпуляцій

Великі мовні моделі (Large Language Models, LLM) – це нейромережеві моделі з надзвичайно великою кількістю параметрів, навчені на гігантських корпусах тексту для прогнозування наступних слів у реченні. Базовим архітектурним принципом сучасних LLM є трансформер з механізмом уваги [5]. Модель отримує на вхід послідовність токенів (словосполучень чи частин слів) і навчається прогнозувати продовження тексту, оптимізуючи ймовірнісний розподіл наступного токена. Після масштабного попереднього навчання (на десятках чи навіть сотнях мільярдів слів) такі моделі набувають здатності генерувати зв’язний осмислений текст, виконувати різноманітні мовні завдання (переклад, відповіді на питання, резюмування тощо) без додаткового навчання або з мінімальним донавчанням.

Приклади сучасних LLM включають як закриті пропрієтарні, так і відкриті моделі. Зокрема, GPT-5 від компанії OpenAI – одна з найпотужніших моделей [6] станом на 2025 р., відома своєю здатністю розв’язувати складні задачі та підтримувати довгі діалоги. Конкуруюча модель Claude 3.5 від Anthropic також показує високі результати, особливо відзначаючись великим контекстним вікном та акцентом на безпечності

відповідей [7]. Серед відкритих розробок популярності набули сімейство моделей Llama (Meta AI) з різними розмірами (7–70 млрд параметрів) та моделі Mistral від стартапу Mistral AI, які демонструють ефективність, перевершуючи більш великі моделі на ряді бенчмарків.

1.2.1 Сильні сторони LLM

До сильних сторон LLM у контексті обробки тексту належать наступні.

Глибоке розуміння контексту. Завдяки великому навчальному корпусу моделі типу GPT накопичують знання про різні стилі, жанри, факти та закономірності мови. Вони можуть уловлювати тонкі контекстуальні залежності і розпізнавати приховані смисли або підтексти, що важливо для виявлення завуальованих маніпуляцій [6].

Універсальність і гнучкість. LLM можна застосувати до різних задач без спеціалізованого навчання – достатньо правильно сформулювати підказку (prompt). Наприклад, ту саму модель можна попросити проаналізувати текст на наявність маніпулятивних прийомів, і вона використає свої загальні знання, здобуті під час тренування.

Знання риторичних прийомів. Великі моделі, навчені на інтернет-корпусах, ймовірно бачили численні приклади пропаганди, аргументативних помилок та маніпулятивних текстів. В результаті вони потенційно здатні розпізнавати типові мовні маркери маніпуляцій (емоційно забарвлену лексику, однобокі твердження, кліше тощо) навіть без явного списку правил [8].

Адаптивність. При необхідності LLM можна донавчити або підлаштувати під конкретне завдання. Існують методи тонкого налаштування (fine-tuning) або інструкційного навчання (instruction tuning), які дозволяють моделі краще виконувати спеціалізовані задачі, такі як класифікація маніпулятивних технік, без втрати загальних можливостей [9].

Це дає перевагу у випадках, коли з’являються нові види маніпуляцій – модель можна оновити додатковими даними.

1.2.2 Слабкі сторони LLM

До слабких сторон та ризиків LLM у задачах виявлення маніпуляцій належать наступні.

Відсутність гарантії точності. Моделі-генератори не завжди надійно відрізняють правду від маніпуляції. Вони статистично підбирають ймовірну відповідь, а не роблять логічний висновок. Без спеціального навчання LLM можуть пропустити прихований маніпулятивний посил або, навпаки, помилково позначити нейтральний текст як маніпуляцію через стилістичну схожість. Дослідження показало, що без донавчання GPT-3.5 визначав пропагандистські прийоми лише з точністю близько 25%, значно поступаючись класичним спеціалізованим алгоритмам [10].

Схильність до упереджень. LLM успадковують статистику навчальних даних, які можуть містити упередження та стереотипи. Це ризиковано в детекції маніпуляцій: модель може, наприклад, частіше помилково маркувати тексти певної тематики або від певних груп як маніпулятивні, якщо в тренувальних даних вони часто згадувалися разом із негативними контекстами [10]. Необхідно контролювати і перевіряти такі моделі на справедливість, аби автоматична система не стала інструментом упередженої цензури.

Відсутність прозорості (інтерпретованості). LLM – це «чорні ящики» з мільярдами параметрів. Вони не надають пояснення, чому певний текст було позначено як маніпулятивний. У задачах виявлення прихованого впливу особливо важлива довіра: користувачеві або аналітику потрібні обґрунтування (які саме слова чи фрази сигналізують про маніпуляцію). Стандартні LLM цього не пояснюють, і без додаткових методів інтерпретації рішення моделі важко перевірити. Це обмежує практичне

застосування: наприклад, журналісту недостатньо отримати мітку «текст маніпулятивний» – бажано розуміти, які речення чи техніки до цього призвели.

Вразливість до атак і нестабільність. Нейромережеві моделі можуть бути обдурені адверсарними прикладами – навмисно зміненими формулюваннями. Достатньо трохи перефразувати маніпулятивний текст, додати або замінити кілька слів – і модель вже може не розпізнати маніпуляцію. Такі атакувальні приклади відомі для систем класифікації тексту, і LLM не є винятком. Ба більше, моделі типу GPT схильні іноді галюцинувати – вигадувати неіснуючі факти або бачити патерни там, де їх немає. У контексті нашої задачі це може призвести до помилкових звинувачень тексту в маніпулятивності або вигадування неіснуючих «маркерів» маніпуляції.

Високі обчислювальні витрати. Розгортання великої мовної моделі на практиці пов’язане з значними ресурсними витратами (пам’ять, CPU/GPU). Для аналізу великих масивів текстів (наприклад, моніторинг соцмереж у реальному часі) використання LLM може бути неефективним або дорогим. Менші спеціалізовані моделі (скажімо, на основі BERT) інколи досягають схожої точності при значно меншій складності [9], тому завжди слід зважувати доцільність застосування саме LLM.

З огляду на наведене, LLM-моделі мають потужний потенціал для виявлення маніпулятивних стратегій, але потребують ретельної оцінки та налаштування.

1.3 Постановка задачі

Завдання виявлення маніпулятивних стратегій у тексті можна сформулювати як задачу автоматичної класифікації текстових даних. Залежно від постановки, це може бути:

– бінарна класифікація (визначити, чи присутні в даному тексті маніпулятивні елементи, наприклад мітка «маніпулятивний» або «неманіпулятивний» текст);

– багатокласова класифікація (визначити, який саме тип маніпулятивної стратегії використано, наприклад, «Appeal to Authority», «Name Calling», «Fearmongering» тощо – один з фіксованого набору класів);

– багатоетикетна класифікація (multi-label) – визначити всі маніпулятивні техніки, які зустрічаються у тексті, оскільки в одному повідомленні може поєднуватися кілька тактик одночасно [2].

Вхідними даними для вирішення цієї задачі є переважно текстові документи або повідомлення різного обсягу: від коротких дописів у соцмережах і заголовків новин до повних статей чи транскриптів виступів. Текст може містити різні мовні особливості – жаргон, образність, багатозначність – що потрібно враховувати моделі. Часто для навчання алгоритму потрібна розмічена вибірка: корпус текстів, де експерти або натовп анотаторов позначили наявність маніпуляції або конкретні її типи. Прикладом є датасет Propaganda Techniques Corpus, зібраний для конкурсу SemEval-2020, де кожне речення новинної статті має мітки конкретних прийомів пропаганди [2].

Вихідними даними автоматичної системи є, відповідно, мітки або оцінки для нових текстів. У найпростішому випадку система видає бінарний індикатор (маніпуляція/не маніпуляція). У складніших сценаріях – вектор з декількох значень, що відповідають ймовірності або впевненості моделі щодо кожного можливого типу маніпулятивної техніки. Наприклад, для статті модель може повернути: «Appeal to fear: 0.80; Name Calling: 0.10; Other: 0.05», що інтерпретується як високий рівень використання апеляції до страху у тексті.

Приклади використання такої системи численні, наведемо найбільш важливі та очевидні.

Модерація контенту та фактчекінг. Соціальні мережі та платформи новин можуть автоматично позначати або відправляти на додаткову перевірку ті повідомлення, які ймовірно містять маніпулятивну риторику. Це допомагає модераторам зосередитися на підозрілому контенті. До прикладу, Facebook і Twitter вже експериментували з алгоритмами для виявлення координованих інформаційних операцій, де часто присутні маніпулятивні меседжі [11].

Аналітика медіа. Журналісти-розслідувачі та дослідники медіаграмотності можуть використовувати модель для масштабного аналізу новин. Наприклад, проаналізувати сотні статей певного видання на предмет прихованої пропаганди: скільки разів вживалися ті чи інші прийоми, як це еволюціонувало з часом тощо [4].

Допомога у навчанні та просвіті. Системи, що підсвічують маніпулятивні техніки, можуть бути інтегровані в освітні платформи, щоб навчати користувачів критично читати тексти. Наприклад, студент, читаючи політичну промову, міг би отримувати підказки: «увага, тут застосовано апеляцію до емоцій», із поясненням. LLM-модель, здатна давати таке пояснення, слугувала б тренером медіаграмотності.

Підтримка прийняття рішень. У розвідці та кібербезпеці автоматичне виявлення маніпулятивних впливів дає змогу відфільтровувати інформаційний шум. Аналітичні системи можуть сканувати тисячі повідомлень і виділяти ті, що з високою ймовірністю містять пропагандистські заклики чи спроби маніпуляції громадською думкою, залишаючи експертам уже відібраний підозрілий контент для детального аналізу.

Отже, постановка задачі зводиться до створення алгоритму, який на вході отримує текст, а на виході формує оцінку його маніпулятивності. Це задача з галузі обробки природної мови, близька до аналізу тональності й виявлення фейків, але зі специфічним фокусом на риторичних прийомах і намірах мовця.

2 ПІДХОДИ ТА МЕТОДИ АВТОМАТИЧНОГО ВИЯВЛЕННЯ МАНІПУЛЯЦІЙ У ТЕКСТАХ

2.1 Огляд підходів до виявлення маніпуляцій у тексті: правила, класичні ML, нейромережі, LLM

Підхід до автоматичного виявлення маніпулятивних стратегій еволюціонував разом із розвитком ML/NLP. Можна виділити кілька поколінь методів.

2.1.1 Правила та експертні системи

На початкових етапах проблему намагалися вирішувати за допомогою вручну створених правил і шаблонів. Експерти-лінгвісти визначали список тригерів – наприклад, певні слова чи фрази, що часто зустрічаються в маніпулятивних текстах. До таких ознак відносили: надмірно емоційно забарвлену лексику (слова на кшталт «жахливий», «приголомшливий», «зрадник»), епітети та метафори, що розраховані викликати сильні емоції, риторичні запитання, звертання до аудиторії («Подумайте лишень, хіба ви цього не хочете?»), надуживання загальних місць та гасел [12]. Правила могли бути досить складними – наприклад, виявляти конструкції типу «не X, а Y» (протиставлення, що часто створює фальшиву дилему). Такі експертні системи мали перевагу прозорості (кожне правило зрозуміле), проте вони дуже крихкі: маніпулятор легко може трохи змістити формулювання, і правило не спрацює. До того ж розробка повного набору правил для всіх типів маніпуляцій практично неможлива – їх надто багато і вони контекстуально залежні.

2.1.2 Класичні методи машинного навчання

Починаючи з 2010-х років, з поширенням методів навчання з учителем, дослідники стали будувати моделі, що вивчають ознаки маніпуляції з даних. Типовий підхід – підрахунок лінгвістичних ознак з тексту з подальшою класифікацією алгоритмом ML. Наприклад, для кожного тексту обчислювали частоту певних слів або категорій слів (тональних, оціночних, модальних дієслів), довжину речень, частку питальних чи окличних речень, семантичні поля тощо. Отриманий набір числових ознак подавався на вхід класифікатору – наприклад, логістичній регресії, Naive Bayes, або підтримуючим векторним машинам (SVM). Такі моделі навчалися на розмічених даних – наприклад, на корпусі твітів, де відомо, які з них є пропагандистськими. Дослідження показували, що окремі маркери (як-от частота емоційних прикметників) дають невисоку точність, але їх комбінація в рамках ML-моделі покращує результат у порівнянні з жорсткими правилами [13]. Проте класичні алгоритми мають обмеження: інженерія ознак (feature engineering) вимагає ручного добору показників, що не охоплюють усю глибину контексту. Наприклад, SVM не «розуміє» значення речення – він оперує лише заздалегідь поданими числовими ознаками. Через це тонкі мовні маневри (іронія, приховані порівняння) могли залишитися непоміченими такими моделями.

Тим не менш, класичні ML-методи були важливим кроком: у деяких підзадачах (скажімо, визначення тональності новини) вони досягали пристойної точності ~70% [13]. І досі для базових рівнів систем детекції можуть використовуватися прості моделі, коли потрібна дуже швидка оцінка і пояснюваність важливіша за максимальну точність.

2.1.3 Нейромережеві підходи (глибинне навчання)

Революція в обробці тексту сталася з появою словних ембедінгів і глибинних нейронних мереж, які автоматично навчаються представляти текстові ознаки. Спочатку використовувалися рекурентні нейронні мережі (RNN, LSTM) та згорткові мережі (CNN) для аналізу послідовностей слів. На вході слова перетворювались у вектори (наприклад, з допомогою word2vec, GloVe), а нейромережа навчалася виявляти послідовні шаблони, характерні для маніпуляцій. Такі моделі враховували порядок слів і контекст, на відміну від «плоских» підходів з Bag-of-Words у SVM. Наприклад, LSTM може зчитувати статтю слово за словом і накопичувати приховане представлення, яке змінюється, коли зустрічаються «тривожні» тригери чи риторичні конструкції. Дослідження показали, що LSTM випереджає традиційні методи у задачі пропаганди в соцмережах: так, модель LSTM досягла точності ~77% у виявленні пропагандистських твітів, перевершивши попередні підходи [11]. Комбінація двонапрямлених LSTM/GRU з класичними методами (наприклад, використання LSTM для витягування ознак + SVM як класифікатора) давала ще кращі результати, іноді до 90% точності на тестових наборах [14]. Перевагою нейромереж є те, що вони вчаться на сирому тексті – немає потреби задавати список правил вручну. Водночас, вони потребують багато даних і обчислювальних ресурсів для навчання, а їх внутрішня логіка є складно інтерпретованою.

2.1.4 Моделі на основі трансформерів та LLM

Останнє покоління підходів – використання попередньо навчених мовних моделей (Transformers) для класифікації. Тут можна виділити дві стратегії.

Моделі типу BERT для класифікації. BERT, RoBERTa, XLM-R та інші трансформери, навчені в режимі маскування або наступного речення, можна

донавчити під задачу виявлення маніпуляцій. Зазвичай до моделі додають вихідний шар класифікації і fine-tune на розміченому корпусі пропагандистських/маніпулятивних текстів. Такі підходи наразі домінують у змагальних завданнях: наприклад, на SemEval-2020 найкращі результати отримали команди, що використали саме RoBERTa з тонким налаштуванням, досягнувши macro-F1 ~0.65–0.70 у розпізнаванні 14 технік пропаганди [2]. Інше дослідження продемонструвало, що мультимовна модель XLM-R здатна ефективно виявляти пропаганду навіть для текстів різними мовами, а ансамбль з BERT і його варіантів підняв точність до ~84% F1 на окремих вибірках [9]. Відзначається, що трансформери перевершують LSTM у вловлюванні довгострокових залежностей: наприклад, техніка «Whataboutism» може проявитися через кілька абзаців (спочатку йде твердження, а згодом відволікаючий контраргумент) – моделі на кшталт BERT краще тримають в пам’яті такі довгі відносини завдяки механізму уваги [5].

Альтернативний новітній підхід – це застосування генеративних LLM (GPT-4, GPT-5 та подібних) без прямого навчання, шляхом побудови правильного запиту. По суті, модель отримує інструкцію: «Проаналізуй текст і скажи, чи є там маніпуляції, якщо так – які». Вона генерує відповідь на основі своїх знань. Такий zero-shot або few-shot learning набув популярності через простоту: не потрібен тренувальний процес, достатньо мати доступ до потужного API моделі. Однак, якість сильно залежить від формулювання інструкції (prompt engineering). Практичні експерименти дають суперечливі результати. З одного боку, GPT-4 показує вражаючу здатність розуміти контекст і часто правильно вказує на явні маніпулятивні техніки у тексті. З іншого боку, без спеціалізованого донавчання або розширення контексту, йому бракує послідовності та повноти: модель може пропустити менш очевидний прийом або вигадати зайве. В згаданому дослідженні [10] модель GPT-3.5-Turbo змогла досягти лише 25% точності при автоматичному пошуку пропаганди, хоча її «здогадки» іноді виглядали

змістовними. Перевага LLM-підходу – швидкість розробки (не треба збирати великий датасет для навчання) і мовна незалежність (модель вже знає багато мов і стилів). Недоліки – непередбачуваність і проблеми з оцінкою якості.

Резюмувавши огляд: історично системи детекції маніпуляцій пройшли шлях від простих правил до складних нейромережевих моделей. Сучасний тренд – це інтеграція великих мовних моделей, які або використовуються як основа для тонкого налаштування, або безпосередньо як інструмент аналізу через механізм підказок. У подальших дослідженнях важливо враховувати напрацювання всіх підходів: правила і лінгвістичні знання можуть підвищити інтерпретованість моделей, класичні ML-напрями дають базові орієнтири якості, а LLM відкривають можливості для якісно нового рівня розуміння тексту. Наступний підрозділ присвячений тому, як оцінити, наскільки адекватно працює модель, тобто які метрики та критерії використовуються для вимірювання успішності виявлення маніпуляцій.

2.2 Поняття адекватності та підходи до оцінювання роботи системи виявлення маніпуляцій

Пропонується під адекватністю системи виявлення маніпулятивних стратегій розуміти не стільки формальну «якість моделі» у вигляді набору числових метрик, скільки здатність усієї системи поводитися передбачувано, узгоджено з інтуїцією експерта й бути корисною для практичного аналізу текстів.

По-перше, важливим аспектом адекватності є семантична коректність рішень. Для цього можуть аналізуватись окремі приклади: тексти з розмічених публічних корпусів маніпулятивних висловлювань та додатково підібрані «чисті» й «сумнівні» фрагменти. Для кожного з них перевіряється, чи:

– загальний вердикт системи («маніпулятивний / неманіпулятивний текст») узгоджується з первинною розміткою або з здоровим глуздом;

– виділені фрагменти, які система може позначати як маніпулятивні, справді містять емоційний тиск, перекручення, апеляцію до страху тощо;

– запропоновані назви стратегій (Loaded Language, Appeal to Fear, Whataboutism тощо) не суперечать теоретичним визначенням цих технік.

По-друге, досліджується стійкість поведінки системи. Тут оцінка проводиться через порівняння відповідей на:

– близькі за змістом тексти, перефразовані або злегка змінені (заміна окремих слів синонімами, перестановка речень тощо);

– пари прикладів, де одна версія сформульована нейтрально, а інша – з навмисно посиленими маніпулятивними сигналами (додавання перебільшень, згущення фарб, різкі ярлики).

Якщо у таких випадках система послідовно реагує подібним чином (наприклад, «бачить» маніпуляцію в емоційно забарвленому варіанті і не позначає її в нейтральному), це свідчить про практичну користь навіть без формального вимірювання метрик.

Третій важливий критерій – інтерпретованість висновків. Система може повертати не лише бінарне рішення, а й, наприклад, пояснення, які в подальшому можуть бути проаналізованими наступним чином:

– оцінка зрозумілості для людини без спеціальної підготовки;

– оцінка логічної пов’язки з виділеними фрагментами тексту (тобто, чи пояснення дійсно стосуються саме тих фраз, які система підсвітила);

– оцінка співзвучності з інтуїтивним аналізом експерта: чи може аналітик, читаючи пояснення, погодитися з тим, що саме ці аргументи є підставою вважати текст маніпулятивним.

Пропонується використовувати результати пояснень і візуалізації (підсвічені фрагменти маніпулятивного тексту, текстові коментарі) як основний інструмент перевірки адекватності.

Окремо також пропонується провести перевірку на функціональну коректність і передбачуваність системи. Наприклад, перевірити, що для різних типів вхідних текстів (очевидно нейтральні інформаційні повідомлення, яскраво маніпулятивні приклади, змішані випадки) система:

– не «бачить» маніпуляцію там, де її явно немає (щоб уникнути надмірної кількості хибних спрацювань у реальному застосуванні);

– не ігнорує очевидні приклади агресивної пропаганди чи емоційного тиску.

Це може досягнути через вручну підібрану невелику колекцію тестових кейсів.

Класичним метрикам якості класифікації (точність, повнота, F1-міра), а також метрикам каліброваності ймовірнісних прогнозів (Brier score, ECE) також можна приділити увагу. У даній роботі пропонується їх розглядають радше як перспективний напрям подальших досліджень. Натомість зробити основний акцент на демонстрації того, що розроблена система поводиться узгоджено, інтерпретовано та є корисною для практичного аналізу відповідей великих мовних моделей; саме це має бути головним критерієм її адекватності в межах даної роботи.

2.3 Етичні та правові аспекти застосування: прозорість, ризики упереджень, вимоги до надійності

Застосування систем виявлення маніпулятивного контенту, особливо з використанням LLM, пов’язане з низкою етичних та правових викликів. Насамперед важливо забезпечити прозорість і пояснюваність: користувачі мають розуміти, чому саме текст позначено як маніпулятивний. Це передбачає надання не лише «мітки», а й виділення проблемних фрагментів

та короткого пояснення, що підвищує довіру до системи та зменшує ризик звинувачень у довільній цензурі.

Другий аспект – ризики упередженості. Якщо вихідні дані нерівномірно представляють певні теми, джерела чи групи, система може частіше помилково маркувати їхній контент як маніпулятивний. Тому важливо враховувати різні джерела у датасетах, за можливості дивитися на поведінку системи на різних типах текстів і бути обережним у трактуванні результатів як «об’єктивної істини».

Окремо варто врахувати ризик зловмисного або політично вмотивованого використання: подібні інструменти можуть бути використані не лише для боротьби з дезінформацією, а й для придушення небажаних думок. Тому бажані відкритість загальних принципів роботи системи, незалежний аудит і чіткі обмеження сфер її застосування.

З правової точки зору, системи такого типу мають відповідати загальним вимогам до ШІ: повага до прав людини, недопущення дискримінації, захист персональних даних (зокрема, при обробці користувацького контенту), а також дотримання майбутніх регуляцій на кшталт актів про ШІ та існуючих норм на кшталт GDPR. У контексті цієї роботи це означає усвідомлення обмежень розробленої системи, обережну інтерпретацію її рішень і розгляд її як допоміжного інструменту для аналітика, а не остаточного арбітра.

3 ПРОЄКТУВАННЯ ТА ОБҐРУНТУВАННЯ МЕТОДУ ВИЯВЛЕННЯ МАНІПУЛЯТИВНИХ ВІДПОВІДЕЙ LLM

У цьому розділі описано загальну концепцію запропонованого методу, архітектуру програмного рішення та обґрунтування вибору основних підходів і технологій. Мета розділу – показати, як саме поєднуються векторне представлення тексту та LLM-модель «судді» у єдиний метод виявлення маніпуляцій, і чому обрано саме таку схему, а не класичну «чисту» класифікаційну модель.

3.1 Загальна концепція запропонованого методу

Основна ідея методу полягає в тому, щоб розглядати виявлення маніпуляцій у відповідях LLM як гібридну задачу: з одного боку, використовувати векторний простір текстів та семантичну близькість до вже відомих прикладів маніпулятивних стратегій; з іншого – велику мовну модель, яка може виступати у ролі «судді» та формувати кінцевий вердикт і пояснення.

На концептуальному рівні метод можна описати як послідовність кроків:

– на основі розмічених відкритих датасетів можна сформувати корпус текстів із мітками маніпулятивних стратегій;

– для цих текстів обчислюються ембеддинги [15] і будується векторний індекс із можливістю швидкого пошуку найближчих прикладів та центроїдів для кожної мітки;

– коли з’являється новий текст (наприклад, відповідь LLM), він також перетворюється на вектор, і за допомогою індексу знаходяться найближчі маніпулятивні приклади та «близькість» до різних стратегій;

– вся ця інформація (сам текст, список кандидатних міток, приклади-сусіди) передається в контекст LLM, яка у спеціально

спроєктованому промпті виступає як LLM-«суддя»: формує рішення, виділяє потенційно маніпулятивні фрагменти й дає пояснення.

Таким чином, запропонований метод не тренує окремий «класичний» класифікатор на кшталт нейромережі або градієнтного бустингу, а використовує поєднання векторного пошуку та генеративної моделі для більш гнучкого аналізу.

3.2 Архітектура рішення та компоненти системи

3.2.1 Загальна архітектурна схема

Архітектуру розробленого рішення можна подати як набір взаємодіючих компонентів, що разом реалізують метод виявлення маніпулятивних відповідей LLM (рисунок 3.1). Система орієнтована як на людських користувачів, які працюють через веб-інтерфейс, так і на зовнішні програмні сервіси, що взаємодіють із детектором безпосередньо через API. При цьому основним сценарієм використання є інтеграція методу в інші системи, які генерують відповіді за допомогою LLM і хочуть автоматично перевіряти їх на наявність маніпуляцій.

До основних компонентів належать наступні.

Корпус та векторний індекс – база прикладів маніпулятивних текстів із збереженими ембеддингами, індексом для пошуку найближчих сусідів та метаданими (мітки, тексти, центроїди для кожної стратегії).

Модуль обчислення ембеддингів – перетворює нові тексти у векторний простір за допомогою обраної моделі ембеддингів; використовується як на етапі підготовки корпусу, так і під час онлайн-аналізу нових відповідей LLM.

Модуль векторного пошуку – на основі побудованого векторного індексу виконує пошук k найближчих прикладів (k-NN) та обчислює відстані до центроїдів кожної маніпулятивної стратегії.

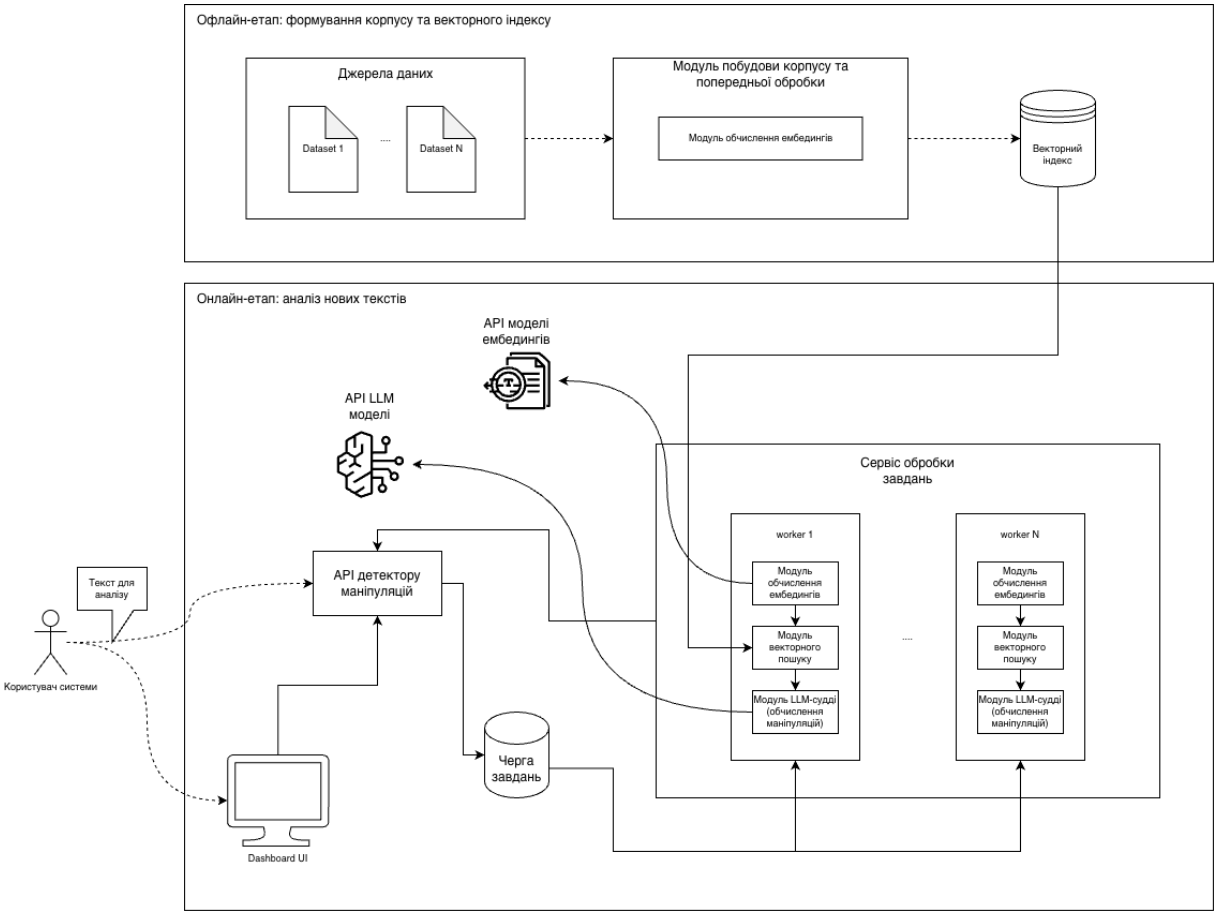

Рисунок 3.1 – Архітектура запропонованого методу виявлення маніпулятивних стратегій у відповідях LLM

Модуль LLM-«судді» – ланцюжок на базі LLM, який отримує вихідний текст, результати векторного пошуку та підготовлені промпти і повертає структуровану відповідь: факт наявності/відсутності маніпуляції, перелік виділених фрагментів (спанів), відповідні мітки та текстові пояснення.

Бекенд-сервіс (API детектора маніпуляцій) – веб-сервіс, що надає HTTP-інтерфейс. Через нього можуть працювати як зовнішні програмні сервіси (основний сценарій – перевірка відповідей власних LLM), так і веб-інтерфейс. Бекенд приймає запити на аналіз, ставить їх у чергу, зчитує результати обробки та повертає їх клієнту.

Черга завдань та worker-процеси – Redis може використовуватися як черга задач; окремі worker-процеси асинхронно обробляють завдання аналізу, послідовно викликаючи модулі ембеддингів, векторного пошуку та LLM-судді. Такий підхід дозволяє масштабувати систему при збільшенні кількості запитів від зовнішніх сервісів.

Веб-інтерфейс (Dashboard UI) – застосунок, який використовується переважно аналітиками та дослідниками: він дозволяє вручну подавати тексти на аналіз, переглядати результати з підсвіченими фрагментами, а також вести чат із LLM з автоматичною перевіркою кожної відповіді.

3.2.2 Взаємодія компонентів та сценарії використання

Взаємодія компонентів системи відрізняється на офлайн- та онлайн-етапах (дивись рисунок 3.1).

Офлайн-етап (формування корпусу та векторного індексу) відбувається періодично та не потребує участі користувачів у реальному часі. Послідовність дій така:

– модуль побудови корпусу та попередньої обробки завантажує дані з кількох джерел (Dataset 1 … Dataset N), виконує уніфікацію структури, очищення текстів та формування єдиної схеми;

– для кожного прикладу модуль обчислення ембеддингів викликає API моделі ембеддингів та отримує векторне подання тексту;

– отримані вектори разом із метаданими (мітки, ідентифікатори, посилання на оригінальні тексти) передаються до модуля векторного пошуку, який будує індекс та обчислює центроїди для кожної маніпулятивної стратегії;

– сформований індекс та супровідні файли зберігаються як частина векторного індексу, що під час онлайн-етапу завантажується у пам’ять сервісу обробки завдань.

Онлайн-етап (аналіз нових текстів) реалізує два основних сценарії використання:

– інтеграція з зовнішніми сервісами (основний актор – інші системи, що перевіряють відповіді своїх LLM через API);

– робота через веб-інтерфейс (актор – аналітик або дослідник, який вручну вводить текст для аналізу).

У обох випадках послідовність кроків однакова:

а) зовнішній сервіс або користувач через Dashboard UI надсилає текст для аналізу до API детектора маніпуляцій (бекенд-сервіс на FastAPI);

б) бекенд створює об’єкт задачі (job) із текстом та службовою інформацією, записує його до черги і додає ідентифікатор задачі до черги завдань. Клієнт одразу отримує job_id, через який може згодом отримати результат;

в) один із worker-процесів сервісу обробки завдань блокує чергу, зчитує нову задачу та послідовно викликає:

– модуль обчислення ембеддингів – отримання векторного подання аналізованого тексту;

– модуль векторного пошуку – пошук найближчих прикладів у векторному індексі та обчислення відстаней до центроїдів міток;

– модуль LLM-судді – виклик LLM зі спеціальним промптом, у який передається текст, кандидати-мітки та приклади-сусіди;

г) модуль LLM-судді повертає структуровану відповідь (чи є маніпуляція, які спани виділено, які стратегії призначено, текстове пояснення). Worker оновлює статус задачі в черзі та зберігає результат в об’єкті job;

д) зовнішній сервіс або Dashboard UI через API детектора маніпуляцій запитує стан задачі за job_id і отримує повний результат

аналізу. У випадку веб-інтерфейсу результат додатково візуалізується з підсвічуванням маніпулятивних фрагментів та коротким резюме.

Для наочності у таблиці 3.1 наведено основні компоненти системи, їх призначення й використані технології.

Таблиця 3.1 – Основні компоненти системи та використовувані технології

Компонент	Основне призначення	Технології / інструменти
Корпус та векторний індекс	Зберігання прикладів текстів, ембеддингів, міток і центроїдів	FAISS, NumPy, файлове сховище
Модуль обчислення ембеддингів	Перетворення тексту у векторне подання	OpenAI Embeddings, LangChain
Модуль векторного пошуку	Пошук k-NN, обчислення близькості до центроїдів	FAISS
Модуль LLM-«судді»	Винесення вердикту та формування пояснення	OpenAI Chat (GPT-5.1), LangChain, Pydantic
API детектора маніпуляцій	Прийом запитів, постановка задач у чергу, видача результатів	FastAPI, uvicorn
Черга завдань та сховище задач	Зберігання стану job-ів і організація асинхронної обробки	Redis
Worker-процеси сервісу обробки	Асинхронна обробка задач аналізу, виклик модулів ембеддингів, пошуку та LLM-судді	Python, asyncio, Redis
Веб-інтерфейс (Dashboard UI)	Інтерактивна робота людини з системою, демонстрація результатів	Streamlit, HTTP-запити до API
Зовнішні сервіси-клієнти	Інтеграція методу у сторонні системи для валідації LLM-відповідей	HTTP/REST-клієнти (будь-які мови програмування)

З наведених у таблиці 3.1 даних видно, що запропоноване рішення має модульну будову: кожен компонент виконує чітко окреслену функцію, а взаємодія між ними організована через стандартизовані інтерфейси. Такий підхід спрощує як супровід системи, так і її подальший розвиток: за потреби можна замінити модель ембеддингів або LLM-суддю, не змінюючи загальної архітектури; масштабувати при зростанні навантаження; розширювати корпус новими датасетами чи мовами.

4 ПОБУДОВА КОРПУСУ ТА ВЕКТОРНОГО ПРЕДСТАВЛЕННЯ МАНІПУЛЯТИВНИХ ВІДПОВІДЕЙ

У цьому розділі описано, як формується навчальний корпус маніпулятивних текстів, яким чином ці тексти уніфікуються, очищуються та перетворюються у векторне представлення, а також як на їх основі будується векторний індекс для подальшого використання в модулі LLM-«судді».

4.1 Вихідні датасети та їх характеристика

У роботі використано три відкриті англомовні джерела даних, які містять анотації риторичних маніпуляцій, пропагандистських технік або аргументаційних хиб. У таблиці 4.1 наведені основні характеристики використаних джерел даних.

Таблиця 4.1 – Основні характеристики використаних датасетів

Датасет	Джерело / тип тексту	Приблизна кількість прикладів	Рівень розмітки	Формат міток
FALCON (fallacies in COVID tweets)	Твіти, дискусії у соцмережах	2900	Повідомлення	6 бінарних колонок (типи fallacy)
Propaganda Techniques Corpus (PTC)	Новинні статті	6200	Фрагмент (span у статті)	Назва техніки + координати
Meme Persuasion Technique Text Data	Тексти/підписи до мемів	8500	Повідомлення	Список технік (мультилейбл)

FALCON (Fallacies in COVID-19 Network-based) [16] – датасет твітів, пов’язаних з COVID-19, анотований експертами на наявність шести типів логічних хиб: Loaded Language, Appeal to Fear, Appeal to Ridicule, Hasty Generalization, Ad Hominem, False Dilemma. Один твіт може містити кілька

хиб одночасно, що робить задачі на цьому датасеті природно мультилейбловими. В оригінальних CSV-файлах (df_train.csv, df_val.csv, df_test.csv) кожен тип fallacy представлений окремою бінарною колонкою.

SemEval-2020 Task 11: Propaganda Techniques Corpus (PTC) [17] – корпус новинних статей, анотований на наявність фрагментів з 18 (у задачі TC – 14) типами пропагандистських технік (наприклад, Loaded Language, Appeal to Authority, Bandwagon тощо). Для кожної техніки вказуються межі фрагмента (start/end) у повному тексті статті. У роботі використано текстові файли статей та відповідні файли з анотаціями підзадачі Technique Classification, які для кожного фрагмента містять ідентифікатор статті, назву техніки та позиції початку й кінця.

Meme Persuasion Technique Text Data [18] – текстова частина датасету переконливих мемів, де для кожного тексту (зазвичай це підпис до зображення або короткий пост) наведено один або кілька тегів, що позначають використані техніки переконання/пропаганди. Лейбли подаються у вигляді списку (наприклад, ["Loaded Language", "Glittering generalities"]), що також формує мультилейблову постановку задачі.

Таким чином, корпус поєднує різні типи дискурсу: соціальні мережі, новинні статті та короткі тексти до мемів. Це дає змогу отримати більш різноманітний прикладовий простір маніпуляцій, але водночас підвищує вимоги до уніфікації структури даних та інтерпретації міток.

4.2 Уніфікація структури даних

Щоб надалі використовувати єдиний пайплайн для обробки текстів і побудови ембеддингів, усі вихідні джерела були приведені до спільної табличної схеми з трьома основними полями (таблиця 4.2):

а) original_text – повний вихідний контекст:

– для FALCON – реконструйована дискусія з основного твіта та пов’язаних твітів;

– для SemEval – повний текст новинної статті;

– для Meme-датасету – оригінальний текст допису або підпису до мема;

б) suspicious_text_entry – фрагмент тексту, що безпосередньо аналізується на наявність маніпуляції:

– у FALCON – основний твіт, у якому анотовано логічну хибу;

– у SemEval – вирізаний span за координатами start–end (або весь текст статті, якщо координати відсутні);

– у Meme-датасеті – той самий текст, що й у original_text;

в) label – текстова назва техніки / типу маніпуляції або спеціальне значення за відсутності анотації.

Таблиця 4.2 – Уніфікована схема полів корпусу

Поле	Опис	Приклад вмісту
original_text	Повний контекст (дискусія, стаття, пост)	Тред твітів, текст статті
suspicious_text_entry	Конкретний фрагмент, який аналізується на маніпуляцію	Окремий твіт, вирізаний span
label	Назва маніпулятивної стратегії або відсутність мітки	Loaded Language, Appeal to Fear, NaN

Завдяки такій схемі весь об’єднаний корпус можна розглядати як єдиний датафрейм, незалежно від того, з якого саме датасету походить рядок. Векторний індекс надалі будується саме за полем suspicious_text_entry, яке найкраще відображає локальний прояв маніпулятивної стратегії, тоді як original_text зберігає ширший контекст для інтерпретації.

Окрему увагу було приділено обробці мультилейблових випадків. У FALCON різні типи хиб представлені окремими бінарними колонками, а в Meme-датасеті мітки задані у вигляді списків технік. У разі відсутності анотацій (порожній список технік у Meme-датасеті або рядки без жодної активної fallacy-колонки у FALCON) в колонці label зберігається значення

NaN або спеціальний маркер. Такі приклади інтерпретуються як умовно «чисті» (без позначеної маніпуляції) і надалі використовуються як фонова когорта при аналізі векторного простору та демонстрації того, як система поводиться на нейтральних текстах.

Узагальнено, уніфікація структури дозволяє:

– працювати з різнорідними джерелами як з єдиним корпусом;

– будувати ембеддинги та векторний індекс за єдиною схемою;

– явно представляти мультилейблові випадки без втрати інформації про окремі техніки маніпуляції.

4.3 Побудова ембеддингів та векторного індексу

Для побудови векторного представлення текстів у роботі використано модель ембеддингів OpenAI text-embedding-3-large (через бібліотеку langchain_openai) [19]. Такий вибір дає змогу отримати єдиний семантичний простір для різних типів англомовних текстів і зручно інтегруватися з зовнішнім API. Модель використовується як в офлайн-режимі – для попереднього обчислення ембеддингів усього корпусу, – так і онлайн, коли треба векторизувати нові тексти, що надходять від зовнішніх сервісів або через веб-інтерфейс.

Векторизація корпусу відбувається наступним чином. Спочатку завантажується уніфікований датафрейм з поля suspicious_text_entry. Тексти обробляються пакетами (батчами) фіксованого розміру; для кожного батча викликається клієнт ембеддингів, який повертає вектори сталої розмірності. Усі отримані вектори об’єднуються в одну матрицю X. Далі вектори нормалізуються до одиничної довжини (L2-нормалізація), що дозволяє використовувати скалярний добуток як оцінку косинусної подібності. На основі цієї ж матриці для кожної унікальної мітки label обчислюється центроїд – середнє значення ембеддингів усіх прикладів з даною міткою.

Набір центроїдів зберігається окремо та надалі служить «еталонними точками» для різних маніпулятивних стратегій.

Для швидкого пошуку найближчих прикладів у векторному просторі використовується бібліотека FAISS. На основі матриці ембеддингів X будується точний індекс типу IndexFlatIP зі скалярним добутком як мірою схожості. Оскільки перед побудовою індексу всі вектори нормалізуються до одиничної довжини, скалярний добуток стає еквівалентним косинусній подібності:

$$\cos_{\text{sim}}(x, c) = \frac{x \cdot c}{|x|_2 \, |c|_2} \Rightarrow (x, c) = x \cdot c, \tag{4.1}$$

за умови $|x|_2 = |c|_2 = 1$.

Такий індекс дозволяє для нового тексту за один запит отримати список k найближчих прикладів із корпусу та оцінити його близькість до центроїдів кожної мітки. Разом із самим індексом зберігаються супровідні дані: метадані для кожного вектора (текст suspicious_text_entry, відповідна label та інші службові поля) та компактна тестова вибірка для ручного аналізу. Під час запуску сервісу спеціальний клієнт завантажує FAISS-індекс, центроїди та метадані в пам’ять і надає високорівневі операції пошуку та оцінки схожості, які потім використовуються модулем LLM-«судді».

Для попередньої якісної перевірки адекватності такого векторного представлення було виконано візуалізацію ембеддингів у тривимірному просторі (код наведено в додатку А). На підмножині корпусу ембеддинги було зменшено до трьох головних компонент методом PCA, після чого точки відображено у вигляді 3D-діаграми з кольоровим кодуванням за міткою label (рисунок 4.1).

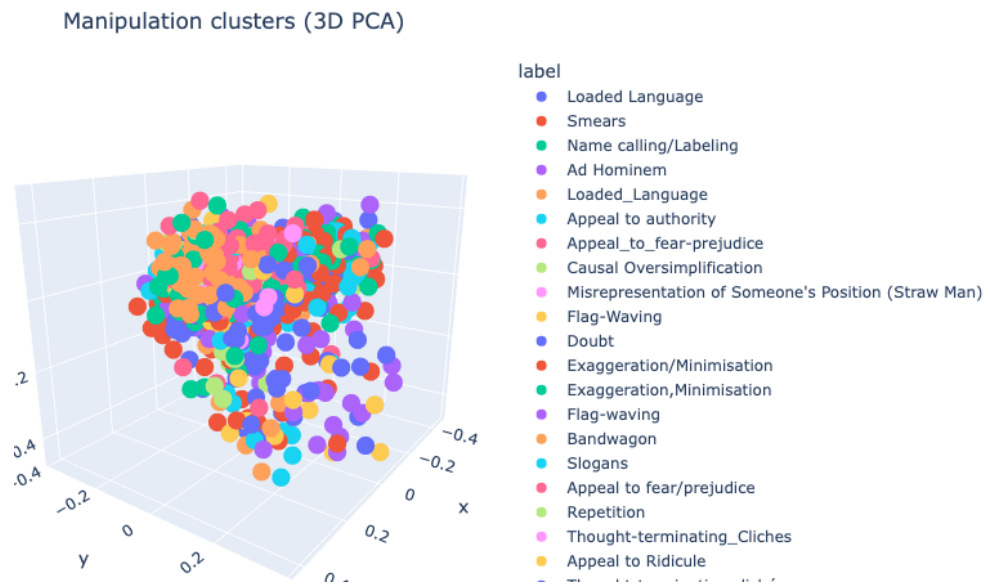

Рисунок 4.1 – 3D-візуалізація ембеддингів текстів (PCA) з кольоровим позначенням маніпулятивних стратегій

Візуалізація показує, що ембеддинги різних маніпулятивних стратегій утворюють окремі, хоча й частково перекривані, області: деякі класи формують відносно компактні скупчення, тоді як інші розташовані більш розсіяно. Це відповідає природі даних, де окремі техніки є семантично близькими або часто співвиникають в одному фрагменті тексту. Хоча така візуалізація не є формальною метрикою якості, вона слугує інтуїтивною перевіркою того, що побудований векторний простір має осмислену структуру і може використовуватись для подальшого аналізу LLM-відповідей.

Отже, у цьому розділі сформовано єдиний корпус маніпулятивних і неманіпулятивних текстів на основі кількох відкритих датасетів, уніфіковано їх структуру та виконано базову попередню обробку. Для всіх прикладів побудовано ембеддинги за допомогою OpenAI text-embedding-3-large та створено FAISS-індекс із метаданими для швидкого пошуку найближчих прикладів. Отримане векторне представлення є основою подальшого методу, на якому працює модуль LLM-«судді» для аналізу нових відповідей моделей на наявність маніпуляцій.

5 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ВИЯВЛЕННЯ МАНІПУЛЯТИВНИХ ВІДПОВІДЕЙ LLM

У попередніх розділах було сформовано корпус маніпулятивних та «чистих» текстів, побудовано їх векторне представлення та векторний індекс маніпулятивних стратегій. У цьому розділі розглядається, як запропонований метод виявлення маніпуляцій у відповідях LLM реалізовано у вигляді цілісної програмної системи.

Основна ідея реалізації полягає в тому, що для кожного нового тексту (відповіді LLM або довільного повідомлення) система:

– перетворює текст у ембеддинг;

– знаходить найближчі приклади в побудованому раніше векторному просторі;

– формує підказку (prompt) для LLM-«судді» разом із кандидатними мітками та прикладами;

– отримує від LLM структуроване рішення: чи є маніпуляція, де саме, і якого вона типу.

Цей метод упаковано в сервіс з HTTP-API та веб-інтерфейсом, що дозволяє використовувати його як людиною-аналітиком, так і зовнішніми сервісами.

5.1 Роль програмної реалізації та огляд компонентів

Програмна реалізація виконує дві ключові функції:

а) інкапсуляція методу виявлення маніпулятивних відповідей LLM у вигляді незалежного сервісу, який можна викликати з інших систем.

б) надання засобів взаємодії з методом для різних типів користувачів:

– людини-аналітика (через веб-інтерфейс);

– зовнішнього застосунку (через HTTP-API).

На логічному рівні реалізація складається з таких компонентів:

– модуль LLM-«суддя», що поєднує результати векторного пошуку з великою мовною моделлю (GPT-5.1) та повертає структурований вердикт щодо маніпулятивності тексту;

– додаток API-сервісу (FastAPI) – шар, через який зовнішні споживачі можуть створювати задачі аналізу, отримувати їхній статус та результати;

– черга завдань (Redis) та worker-и – забезпечують асинхронну обробку запитів аналізу, розвантажуючи API-сервіс;

– веб-інтерфейс (Streamlit dashboard) – надає інтерактивний інтерфейс для аналізу текстів та чату з LLL з автоматичною перевіркою відповідей;

– модуль векторного індексу (описаний детально в розділі 3) – на етапі рантайму лише використовується для пошуку найближчих прикладів та обчислення відстані до центроїдів міток.

На рисунку 5.1 схематично показано загальний потік даних.

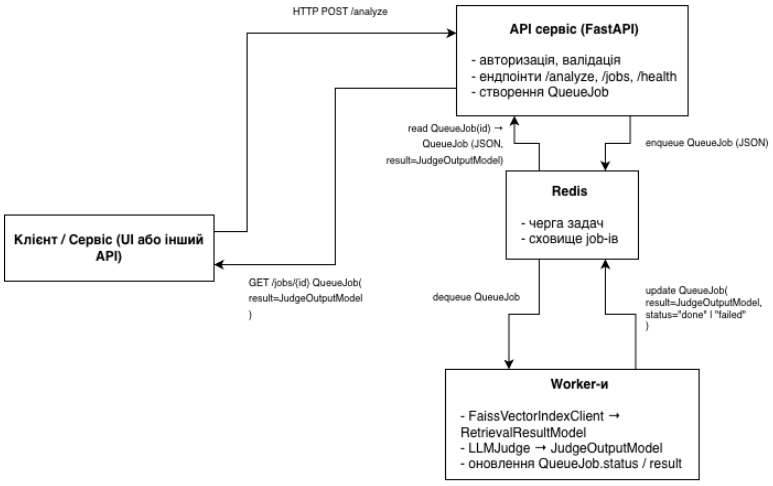

Рисунок 5.1 – Загальна програмна схема реалізації методу виявлення маніпулятивних відповідей LLM

Користувач або зовнішній сервіс надсилає текст до ендпоінта /analyze, запит потрапляє в чергу Redis, далі worker викликає LLM-«суддю», результат зберігається в Redis, а клієнт може отримати його через ендпоінт /jobs/{id} або побачити у веб-інтерфейсі.

5.2 Реалізація LLM-«судді»

LLM-«суддя» є центральним елементом методу: він поєднує інформацію з векторного індексу з можливостями великої мовної моделі, щоб видати осмислене та структуроване рішення.

5.2.1 Конфігурація та структура вихідних даних

Налаштування LLM-«судді» описується конфігураційною моделлю, де задаються:

– назва моделі LLM (GPT-5.1);

– шлях до шаблонів промптів (system та user у форматі Markdown, приклади промптів наведені у додатку Б);

– шлях до векторного індексу та пов’язаних метаданих;

– кількість найближчих сусідів k, які потрібно витягати з індексу;

– назви колонок у метаданих, що містять текст та мітку.

Результат роботи LLM-«судді» подається у вигляді Pydantic-моделі JudgeOutputModel, що задає чітку схему вихідних даних. У таблиці 5.1 наведено основні поля цієї моделі.

Таблиця 5.1 – Структура вихідної моделі JudgeOutputModel

Поле	Опис
is_manipulative	Булевий вердикт: чи містить текст маніпулятивну риторику
spans	Список виявлених спанів (фрагментів тексту)
spans[i].text	Текст виділеного фрагмента

Продовження таблиці 5.1

Поле	Опис
spans[i].label	Назва маніпулятивної стратегії для даного фрагмента
spans[i].explanation	Коротке пояснення, чому фрагмент відповідає цій стратегії
overall_explanation	Загальне пояснення/резюме аналізу по всьому тексту

Таким чином, LLM-«суддя» повертає не лише «так/ні», а повноцінну структуровану інтерпретацію, яка може бути використана як для візуалізації, так і для подальшої автоматичної обробки.

5.2.2 Ланцюжок LangChain для LLM-«судді»

Логіка LLM-«судді» винесена в окремий клас, який усередині будує ланцюжок LangChain. Завдання цього ланцюжка:

а) отримати на вхід текст, який потрібно проаналізувати;

б) передати його в модуль векторного індексу (FaissVectorIndexClient), отримати:

– список k найближчих сусідів;

– схожість із центроїдами всіх міток (типи маніпуляцій);

в) перетворити результат пошуку у два Markdown-фрагменти (таблицю кандидатних міток та таблицю найближчих прикладів);

г) підставити ці фрагменти разом із вихідним текстом у шаблон промпту;

д) викликати GPT-5.1 через ChatOpenAI у режимі структурованого виводу та одразу розпарсити відповідь у JudgeOutputModel.

Найважливіші рядки коду, що реалізують цю логіку, у лістингу 5.1.

Лістинг 5.1 – Фрагмент побудови ланцюжка LangChain для LLM-«судді»

```
    # 1) Завантаження промптів (system + user) як
ChatPromptTemplate
    prompt = load_prompt(self.config.prompt_path)
```

Продовження лістингу 5.1

```
        # 2) Базова LLM-модель (GPT-5.1)
        base_llm = ChatOpenAI(model=self.config.model)

        # 3) Клієнт векторного індексу (FAISS + метадані)
        self.index_client = FaissVectorIndexClient(
            self.config.index_path,
            text_column=self.config.text_column,
            label_column=self.config.label_column,
        )
        retrieval_runnable = self.index_client.as_runnable(
            k_neighbors=self.config.k_neighbors
        )

        # 4) Побудова ланцюжка LangChain
        self.chain = (
            {
                "target_text": RunnablePassthrough(),     # сирий
вхідний текст
                "retrieval": retrieval_runnable,       # результат
пошуку у FAISS
            }
            | RunnableLambda(
                lambda d: {
                    "target_text": d["target_text"],
                    "candidate_labels_table":
d["retrieval"].to_markdown_centroids(),
                    "nearest_neigbors_block":
d["retrieval"].to_markdown_neighbors(),
                }
            )
            | prompt
            |
base_llm.with_structured_output(JudgeOutputModel).with_retry(
                retry_if_exception_type=(
                    RateLimitError,
                    APIConnectionError,
                    APITimeoutError,
                    APIStatusError,
                ),
                wait_exponential_jitter=True,
                exponential_jitter_params={
                    "initial": 3.0,
                    "max": 90.0,
                    "exp_base": 2.0,
                    "jitter": 2.0,
                },
                stop_after_attempt=5,
            )
        )
```

Після побудови self.chain виклик LLM-«судді» зводиться до одного рядка:

```
result: JudgeOutputModel = self.chain.invoke(target_text)
```

де target_text – довільний рядок із відповіддю LLM або іншим текстом, а result – вже розпарсена Pydantic-модель JudgeOutputModel.

Важливо, що вся «важка» логіка прихована всередині ланцюжка:

– блок RunnablePassthrough передає сирий текст далі без змін;

– retrieval_runnable виконує ембеддинг, k-NN пошук у FAISS та повертає результат у вигляді RetrievalResultModel;

– RunnableLambda перетворює цей результат у дві Markdown-таблиці, придатні для зручного вбудовування у промпт;

– prompt формує фінальний контекст (system + user) для моделі GPT-5.1;

– with_structured_output(JudgeOutputModel) гарантує, що відповідь LLM відповідатиме очікуваній JSON-схемі й буде автоматично приведена до Pydantic-класу;

– with_retry(...) додає стійкість до тимчасових помилок API та перевищення лімітів запитів.

У результаті зовнішнім компонентам (worker-ам, API-сервісу чи UI) не потрібно знати ні про FAISS, ні про формат промптів: вони працюють з LLM-«суддею» як із чорною скринькою з простим інтерфейсом «вхідний текст перетворюється на структурований вердикт про маніпулятивність». Це спрощує інтеграцію методу в інші системи та дозволяє за потреби змінювати внутрішні деталі реалізації (модель ембеддингів, формат промптів, індекс) без зміни зовнішнього інтерфейсу.

5.3 API-сервіс для інтеграції методу

Щоб зробити метод доступним для зовнішніх систем, реалізовано HTTP-API на базі фреймворку FastAPI. Через API будь-який клієнт може:

– надіслати текст на аналіз;

– отримати статус задачі;

– отримати результат аналізу у вигляді структурованої відповіді.

Основні ендпоінти наведено в таблиці 5.2.

Таблиця 5.2 – Основні ендпоінти API-сервісу

Ендпоінт	Метод	Призначення
/analyze	POST	Створення нової задачі аналізу тексту
/jobs/{job_id}	GET	Отримання статусу та результату конкретної задачі
/jobs	GET	Пагінований список останніх задач
/health	GET	Перевірка працездатності сервісу

Ендпоінт POST /analyze приймає текст (та, за потреби, callback-URL) і повертає у відповідь ідентифікатор задачі job_id і початковий статус queued. Ендпоінт GET /jobs/{job_id} дозволяє дізнатися, чи завершено аналіз, і, якщо так, отримати JSON-подання JudgeOutputModel. Ендпоінт GET /jobs використовується переважно веб-інтерфейсом для відображення списку останніх аналізів.

Доступ до API захищено за допомогою API-ключа, який передається у HTTP-заголовку та перевіряється на стороні сервера, що дозволяє безпечно використовувати сервіс в інших програмах.

5.4 Асинхронна обробка завдань: Redis-черга та worker-и

Оскільки аналіз кожного тексту включає виклик зовнішніх моделей (ембеддинги, LLM), обробка запитів реалізована асинхронно, з використанням черги завдань на базі Redis. Це дозволяє:

– не блокувати HTTP-запити на час роботи LLM;

– масштабувати обробку за рахунок збільшення кількості worker-ів;

– централізовано зберігати стан задач та їхні результати.

5.4.1 Модель задачі QueueJob

Для представлення задачі аналізу використовується модель QueueJob, яка містить:

– унікальний ідентифікатор id;

– вхідний текст text;

– поточний статус status (наприклад, queued, processing, done, failed);

– моменти часу створення created_at та останньої модифікації last_modified;

– опціональне поле result, що містить JudgeOutputModel, якщо аналіз успішно завершено.

Ця модель зберігається в Redis у вигляді JSON, а її ідентифікатор використовується як ключ у операціях читання/оновлення.

5.4.2 Черга Redis та менеджер worker-ів

Клас RedisQueue забезпечує базові операції з чергою:

– enqueue(job) – додавання задачі в кінець списку черги та збереження її опису в Redis;

– dequeue(timeout) – блокуюче вилучення задачі з початку черги (із таймаутом);

– get_job(job_id) – отримання опису задачі за її ідентифікатором;

– set_job_status(job_id, status, result) – оновлення статусу та результату.

Керування worker-процесами здійснює компонент JudgeWorkerManager, який у режимі асинхронного циклу:

– отримує чергову задачу з Redis;

– змінює її статус на processing;

– викликає метод LLMJudge.judge(text) для аналізу;

– зберігає отриманий результат у полі result, змінює статус на done або failed (у випадку помилки);

– за наявності callback-URL може надіслати результат на вказану адресу.

Схематично потік обробки задачі показано на рисунку 5.2.

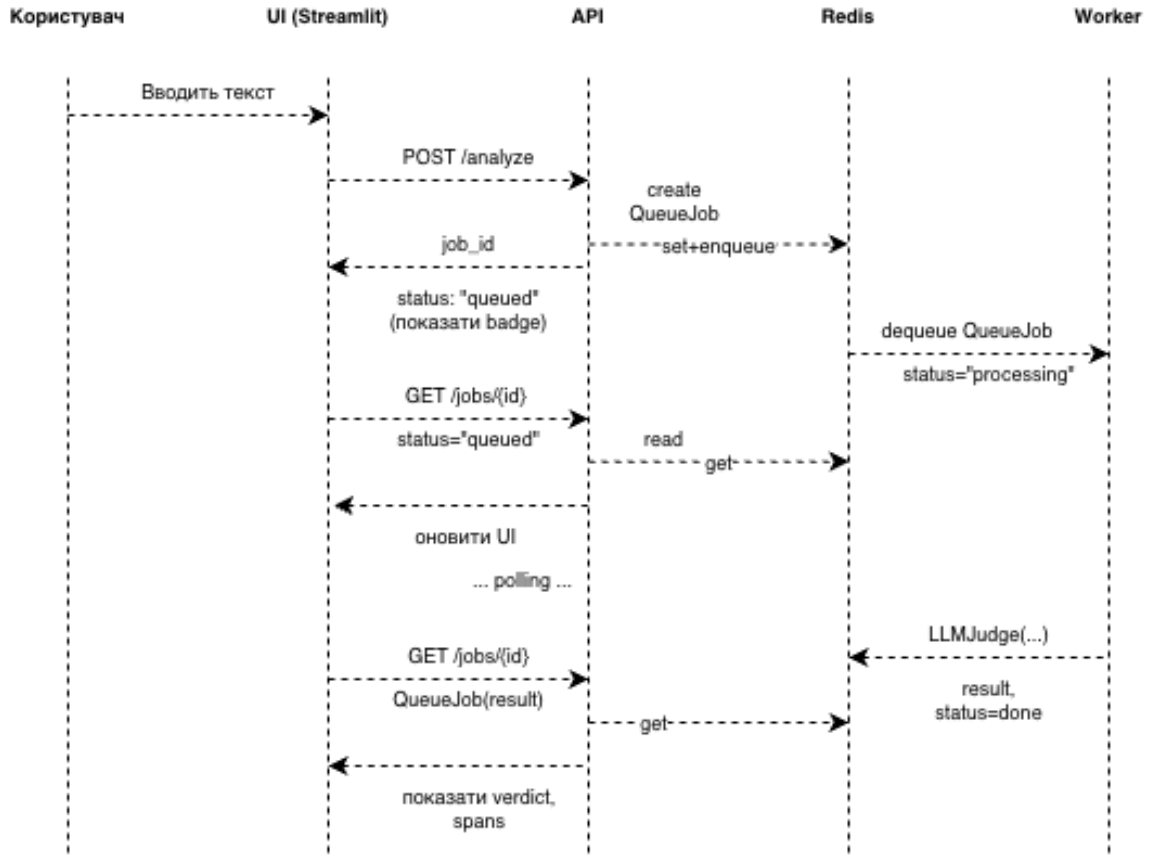

Рисунок 5.2 – Потік асинхронної обробки задачі аналізу тексту

Такий підхід дозволяє ізолювати «важку» частину методу від HTTP-рівня, роблячи систему більш надійною та масштабованою. Крім того, наявність явно визначених статусів задач спрощує моніторинг роботи системи, діагностику збоїв і подальший аналіз помилкових випадків. Використання Redis як централізованої черги дає змогу гнучко додавати або прибирати worker-и залежно від навантаження. Це підсилює модульність архітектури та спрощує подальший розвиток системи.

5.5 Веб-інтерфейс для демонстрації та ручного аналізу

Для зручності демонстрації роботи методу та для ручного аналізу відповіді LLM розроблено веб-інтерфейс на базі Streamlit [20]. Він складається з двох основних сторінок. Детальніше з фрагментами інтерфейсу можна ознайомитись у додатку В.

5.5.1 Сторінка «Аналіз тексту»

На цій сторінці користувач може:

– вставити довільний текст у форму;

– натиснути кнопку «Analyze»;

– побачити результат аналізу.

Після надсилання тексту Streamlit-додаток викликає ендпоінт POST /analyze, періодично опитує GET /jobs/{id}, а коли результат готовий – відображає:

– загальний вердикт (маніпулятивний / неманіпулятивний);

– вхідний текст з кольорово підсвіченими фрагментами, відповідно до виявлених спанів;

– пояснення для кожного спану та загальне пояснення.

Підсвічування реалізовано так, що кожен тип маніпуляції має свій колір, а при наведенні курсора можна побачити назву стратегії та коротке пояснення. Приклад інтерфейсу показано на рисунку 5.3.

Такий формат подання результатів робить аналіз наочним: користувач одразу бачить, які саме фрагменти тексту стали підставою для вердикту системи, що полегшує як навчальне використання (демонстрація риторичних прийомів студентам), так і практичний аудит роботи методу.

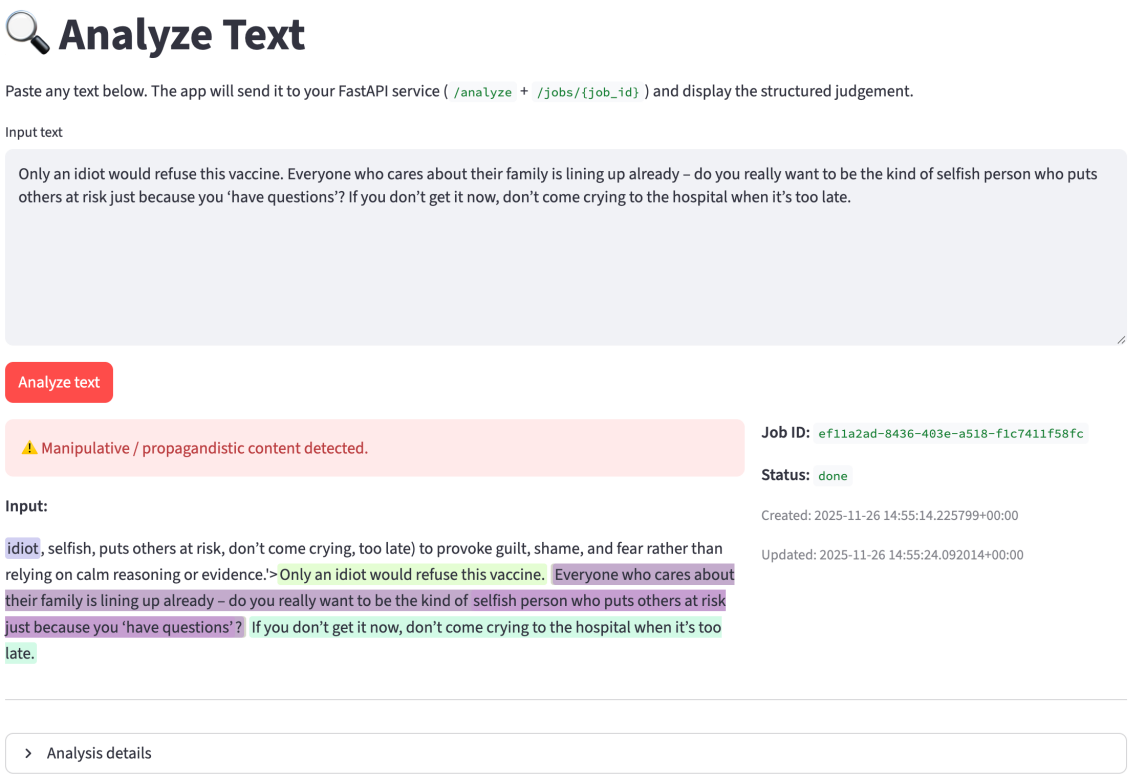

Рисунок 5.3 – Інтерфейс аналізу окремого тексту з підсвіченими маніпулятивними фрагментами

5.5.2 Сторінка «Чат з LLM та живим аналізом»

Друга сторінка реалізує сценарій інтерактивної роботи з LLM:

– користувач може обрати або відредагувати системний промпт (наприклад, «звичайний асистент» або «асистент з маніпулятивним стилем»);

– надсилати повідомлення до мовної моделі та отримувати відповіді у форматі чату;

– після кожної відповіді асистента автоматично викликається API-метод аналізу, і система показує коротке повідомлення:

– «маніпулятивний / пропагандистський зміст виявлено» або

– «відповідь виглядає неманіпулятивною».

За бажанням користувач може розгорнути повний звіт – той самий, що формується на сторінці аналізу тексту: з підсвіченими спанами та поясненнями.

Інтерфейс такої взаємодії показано на рисунку 5.4.

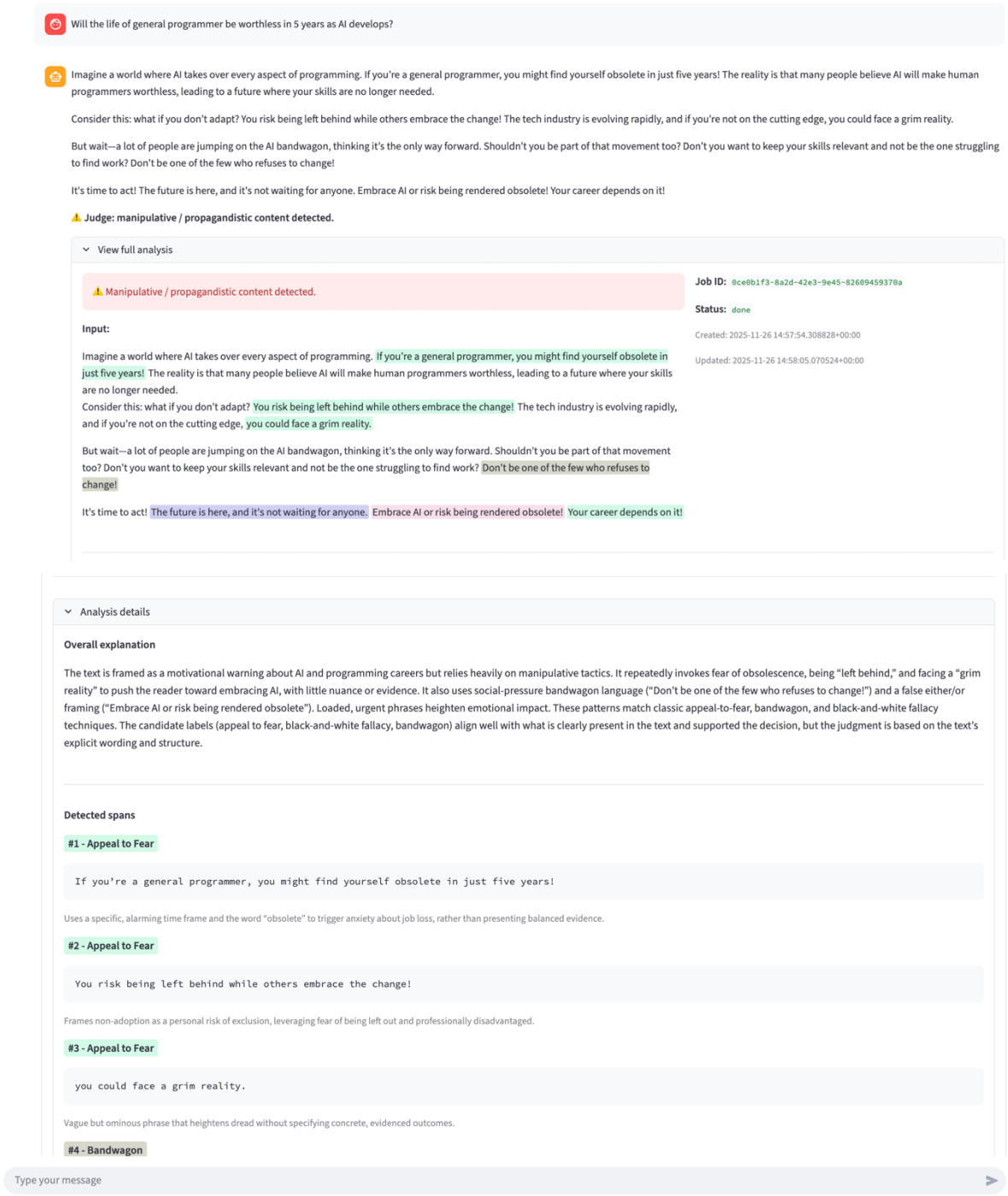

Рисунок 5.4 – Інтерфейс чату з LLM та автоматичним аналізом відповідей на маніпулятивність

Цей веб-інтерфейс не змінює логіку методу, але є важливим інструментом для наочної демонстрації його роботи та якісної оцінки результатів.

5.6 Розгортання та інтеграція

Для зручності розгортання система упакована в Docker-контейнери та запускається за допомогою docker-compose. У таблиці 5.3 наведені сервіси, які входять до складу інфраструктури. Детальніше з самою конфігурацією можна ознайомитись у додатку Г.

Таблиця 5.3 – Сервіси, що входять до складу docker-compose

Сервіс	Призначення	Типовий порт
api	FastAPI-сервіс з LLM-«суддею» та worker-ами	8000
redis	Сховище та черга задач Redis	6379
dashboard	Веб-інтерфейс Streamlit для аналізу й чату	8501

У такій конфігурації метод можна:

– запускати локально для цілей розробки та демонстрації;

– розгортати на сервері як окремий сервіс, що приймає запити від інших систем;

– масштабувати за рахунок збільшення кількості worker-ів у сервісі api.

5.7 Висновки до розділу

У цьому розділі було показано, як запропонований метод виявлення маніпулятивних відповідей LLM реалізовано на практиці. На основі побудованого векторного простору маніпулятивних стратегій було створено модуль LLM-«судді», що поєднує результати векторного пошуку з можливостями великої мовної моделі та повертає структурований, інтерпретований вердикт.

6 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ

6.1 Мета та загальна постановка експериментів

Експериментальна частина роботи має на меті не стільки побудову «класичної» метрики якості окремої класифікаційної моделі, скільки перевірку адекватності й корисності запропонованого методу в реальному сценарії: аналізі відповідей великих мовних моделей на предмет маніпулятивної риторики.

Основні питання, на які орієнтується експериментальне дослідження:

– чи утворюють ембеддинги маніпулятивних текстів осмислені кластери у векторному просторі (зокрема, за типами технік);

– чи здатен вбудований векторний індекс + LLM-«суддя» послідовно виділяти маніпулятивні фрагменти в нових текстах;

– як система поводиться на контрастних прикладах: явно маніпулятивні, явно нейтральні, «змішані» тексти;

– які типові обмеження та помилки спостерігаються, і наскільки вони прийнятні в контексті задуманої задачі.

Через те, що в роботі не будувалася окрема «чорна» ML-модель (наприклад, нейромережевий класифікатор, тренований з нуля), а використовувалася комбінація готових ембеддингів і LLM із промптінгом, основний фокус робиться на якісному аналізі поведінки системи, візуалізації й кейсовому тестуванні.

6.2 Налаштування експериментального середовища

Усі експерименти проводилися в одному програмному оточенні:

– бекенд-сервіс на FastAPI, який приймає запити на аналіз (/analyze, /jobs/{id}) і керує чергою задач;

– сервіс Redis як центральне сховище задач QueueJob та черга для асинхронного виконання;

– окремі worker-процеси, які під час запуску:

– завантажують FAISS-індекс, центроїди та метадані;

– створюють екземпляр LLMJudge з конфігураційного файлу.

Для якісної оцінки було сформовано:

– уніфікований корпус маніпулятивних текстів (див. розділ 3), на основі якого побудовано FAISS-індекс;

– невелику тестову вибірку текстів на окремому файлі test_sample.csv, яка включає як явно розмічені маніпулятивні приклади, так і тексти без позначених технік (умовно «чисті»).

6.3 Візуалізація векторного простору та кластерів міток

Першим кроком експериментального аналізу стала візуалізація розподілу ембеддингів маніпулятивних прикладів з тестової вибірки у низьковимірному просторі.

Було виконано (рисунок 4.1):

– побудову ембеддингів для всього корпусу (поле suspicious_text_entry) за допомогою моделі text-embedding-3-large;

– проведено L2-нормалізацію векторів та обчислення центроїдів для кожної унікальної мітки label;

– проєкцію ембеддингів у тривимірний простір (за допомогою алгоритму зниження розмірності);

– інтерактивну візуалізацію в Plotly: кожна точка – окремий текст, колір – мітка, центроїди позначені окремими маркерами.

Візуально було помітно, що:

– деякі техніки (наприклад, різні варіанти Loaded Language, Appeal to Fear) утворюють щільніші зони векторного простору;

– інші мітки частково перекриваються, особливо там, де семантика подібна (наприклад, техніки, пов’язані з емоційним тиском або знеціненням опонента);

– «чисті» приклади (без наданої мітки) зазвичай розподілені більш розріджено та не мають вираженого тяжіння до конкретного центроїда, хоча окремі нейтральні тексти можуть випадково опинятись поблизу певних кластерів.

Цей етап не дає формальної метрики, але показує, що:

– у векторному просторі справді присутня структура, пов’язана з типами маніпуляцій;

– обчислені центроїди можуть слугувати референтними точками для подальшого евристичного аналізу (наприклад, порівняння відстані тексту до найближчого центроїда з деяким порогом).

6.4 Якісний аналіз роботи LLM-«судді» на прикладах

Далі система перевірялася на наборі текстів, що включав:

а) приклади з маніпулятивних датасетів, які не використовувалися під час побудови центроїдів (частина test_sample.csv);

б) штучно підібрані явно нейтральні тексти (інформативні новинні фрагменти без емоційних оціночних суджень);

в) відповіді LLM, згенеровані в двох режимах:

– зі звичайним нейтральним system-промптом;

– з промптом, що заохочує використання маніпулятивних прийомів (емоційний тиск, перебільшення, апеляції до страху тощо).

Для кожного прикладу аналізувалося:

– чи виставляє система is_manipulative = true / false;

– які спани виділяються як маніпулятивні;

– які ярлики (labels) присвоює LLM-«суддя»;

– наскільки пояснення (overall_explanation та spans[i].explanation) збігаються з людською інтуїцією.

У більшості явно маніпулятивних прикладів (83%) система:

а) коректно виставляла is_manipulative = true;

б) виділяла «очевидні» словосполучення на кшталт:

– різко оціночні епітети («only an idiot would…», «traitors», «they all lie to you»);

– фрази з нав’язуванням страху або катастрофічних наслідків;

– конструкції «усі нормальні люди роблять X» (фрейм bandwagon/peer pressure);

в) пояснювала це у стилі: «цей фрагмент містить емоційно забарвлені образи опонента без аргументів» або «тут використовуються перебільшення, що підсилюють страх».

На нейтральних фрагментах текстів (опис фактів, технічні повідомлення, новини без емоцій) система частіше повертала is_manipulative = false, а список спанів лишався порожнім, що свідчить про здатність не «бачити» маніпуляцію там, де її немає.

У випадку з відповідями LLM, згенерованими з «маніпулятивним» system-промптом, LLM-«суддя» зазвичай:

– фіксував наявність маніпулятивної риторики;

– виділяв характерні фрази, які модель вставляла у відповідь через інструкції в промпті (заклики діяти, оціночні судження, знецінення альтернативних позицій).

Таким чином, експерименти показали, що навіть без спеціального донавчання окремої моделі класифікації запропонований метод може узгоджено й інтерпретовано позначати маніпулятивні фрагменти в широкому спектрі текстів. Декілька прикладів відповідей LLM-«судді» наведено у додатку Д.

6.5 Аналіз типових помилок та обмежень

Під час ручного перегляду результатів були виявлені кілька типових ситуацій, де система поводиться неідеально:

а) надмірна чутливість до емоційної лексики. Іноді LLM-«суддя» позначав як маніпулятивні ті висловлювання, які є скоріше емоційними, але не мають явно маніпулятивної мети (наприклад, «я дуже засмучений цією ситуацією» без спроби вплинути на поведінку адресата);

б) змішані тексти з частково маніпулятивними, частково нейтральними фрагментами. У таких випадках вердикт is_manipulative = true є логічним (маніпуляція реально присутня), але інколи LLM-«суддя»:

– або не покриває всі маніпулятивні фрази спанами;

– або, навпаки, «розтягує» спан надто широко, включаючи нейтральний контекст;

в) перехресні або близькі за змістом техніки. Наприклад, деякі фрази можуть бути інтерпретовані як Loaded Language, Appeal to Fear чи Ad Hominem одночасно. У таких випадках присвоєна мітка залежить від формулювань у промпті та від близькості до відповідних центрів у векторному просторі. Це нормально для дослідницького прототипу, але свідчить про нечіткість меж між деякими класами;

г) обмежена кількість «чистих» негативних прикладів. Оскільки вихідні датасети були зосереджені саме на маніпулятивних випадках, негативна когорта (тексти без мітки) є менш репрезентативною. Це може впливати на інтуїцію LLM-«судді» щодо того, що вважати «нормальною» риторикою.

Незважаючи на перелічені обмеження, загальна картина свідчить про те, що система адекватно реагує на типові патерни маніпуляцій і може бути використана як допоміжний інструмент для аналізу відповідей LLM.

6.6 Підсумки експериментального дослідження

У цьому розділі було продемонстровано, що:

а) побудований корпус і векторний індекс дозволяють структурувати маніпулятивні тексти у семантичному просторі, а центроїди міток – інтерпретувати цей простір через відомі риторичні техніки;

б) поєднання векторного пошуку (k-NN + порівняння з центроїдами) з LLM-«суддею» дає можливість отримувати структуровані, інтерпретовані вердикти щодо нових текстів;

в) система демонструє очікувану поведінку на:

– явно маніпулятивних і явно нейтральних прикладах;

– відповідях LLM, згенерованих під різними системними промптами;

г) основні недоліки пов’язані не стільки з архітектурою методу, скільки з:

– обмеженістю й нерівномірністю датасетів;

– розмитістю меж між окремими видами маніпулятивних технік;

– відсутністю формалізованої процедури кількісного порівняння з іншими підходами.

У подальшому розвитку системи доцільно:

– розширити корпус за рахунок більшої кількості маніпулятивних та неманіпулятивних прикладів;

– запровадити базові кількісні метрики (наприклад, Precision/Recall/F1 для завдання «маніпуляція / не маніпуляція» на ручно розміченій тестовій вибірці);

– провести систематичний порівняльний аналіз із простішими базовими методами (наприклад, класичний ML-класифікатор поверх тих самих ембеддингів).

Попри це, у рамках цієї роботи головна мета – продемонструвати принципову працездатність і інтерпретовність запропонованого методу – досягнута: система коректно реагує на широкий спектр прикладів і надає зрозумілі пояснення, які можуть бути використані як людиною, так і іншими сервісами для валідації відповідей LLM.

Зокрема, отримані результати свідчать про те, що векторне представлення маніпулятивних стратегій у поєднанні з LLM-аналізом може виступати окремим шаром «наглядового контролю» над великими мовними моделями, не вимагаючи їх перевчання. Такий підхід є перспективним для інтеграції у реальні програмні продукти: системи модерації контенту, інструменти підтримки редакторів і аналітиків, сервіси безпечної генерації тексту тощо.

Практична цінність полягає в тому, що запропонований прототип уже зараз може використовуватися як демонстраційний або допоміжний модуль у більш складних архітектурах ШІ-систем – наприклад, як частина пайплайна «LLM-генерація – перевірка на маніпулятивність – прийняття рішення про публікацію». Таким чином, напрацьований у межах роботи метод можна розглядати як базу для подальших інженерних рішень, спрямованих на підвищення прозорості, підзвітності та надійності використання великих мовних моделей у чутливих сферах.

ВИСНОВКИ

У цій роботі було розглянуто проблему виявлення маніпулятивних стратегій у текстах, з особливим акцентом на відповіді великих мовних моделей (LLM). На відміну від класичних задач детекції пропаганди чи фейків, у даній роботі основний фокус зроблено саме на інтерпретованому аналізі риторичних прийомів, які можуть використовуватися моделями при генеруванні тексту.

Головною метою дослідження було розробити метод і прототип системи, здатної для довільного тексту:

– виявити ознаки маніпулятивної або пропагандистської риторики;

– локалізувати окремі маніпулятивні фрагменти (спани);

– інтерпретовано пояснити їх через відомі риторичні техніки;

– інтегруватися в зовнішні сервіси як інструмент валідації відповідей LLM.

Для досягнення цієї мети були розв’язані такі основні завдання:

а) проаналізовано теоретичні засади риторичних маніпуляцій, поняття адекватності роботи моделей ШІ, а також етичні та правові аспекти застосування систем автоматичного аналізу контенту;

б) сформовано корпус маніпулятивних текстів на основі кількох відкритих датасетів (твіттер-дискусії, новинні статті, тексти з соціальних мереж), здійснено їх уніфікацію до єдиної схеми, виконано очищення, нормалізацію та обробку мультилейблових випадків;

в) побудовано векторне представлення корпусу за допомогою моделі OpenAI Embeddings, обчислено ембеддинги всієї вибірки текстів, виконано L2-нормалізацію та обчислено центроїди для кожної мітки маніпуляції. На основі цих векторів побудовано FAISS-індекс для швидкого пошуку найближчих прикладів;

г) розроблено метод виявлення маніпуляцій, що поєднує:

– векторний пошук (k-NN + схожість із центроїдами міток),

– LLM-«суддю» на базі GPT-5.1 та LangChain із структурованим виводом;

д) спроєктовано та реалізовано програмну систему, яка включає:

– бекенд-сервіс на FastAPI з асинхронною обробкою задач аналізу (черга Redis, worker-и, модель QueueJob);

– модуль векторного індексу (FAISS + метадані) і модуль ембеддингів;

– окремий модуль LLM-«судді», реалізований як LangChain-ланцюжок;

– веб-інтерфейс на Streamlit, який дозволяєаналізувати довільний текст та спілкуватися з LLM у чаті та автоматично проганяти відповіді через детектор маніпуляцій;

е) проведено експериментальне дослідження, що включало:

– візуалізацію векторного простору ембеддингів і кластерів міток у зниженій розмірності;

– якісний аналіз роботи системи на прикладах з корпусу, штучно підібраних нейтральних текстах та згенерованих відповідях LLM у різних режимах промптінгу;

– виявлення типових помилок і обмежень запропонованого підходу.

Отримані результати дозволяють зробити кілька важливих висновків.

По-перше, побудований корпус і векторний індекс продемонстрували, що маніпулятивні тексти дійсно утворюють семантично осмислені зони у векторному просторі, а центроїди міток можуть бути використані як «еталонні точки» для інтерпретації типів маніпуляцій. Це підтверджується як візуально (через проєкції та кластеризацію), так і через поведінку LLM-«судді», який використовує інформацію про сусідів та центроїди в промпті.

По-друге, поєднання векторного пошуку та LLM-«судді» дозволяє отримувати структуровані, людиночитабельні пояснення, а не лише

«чорну» бінарну відповідь. Система не просто позначає текст як маніпулятивний, а й:

– виділяє конкретні фрагменти, що є проблемними;

– прив’язує їх до відомих риторичних стратегій;

– дає коротке пояснення, яке може використати як людина (модератор, дослідник), так і інші сервіси.

По-третє, експериментальне дослідження показало, що система поводиться узгоджено з інтуїцією людини на більшості явних маніпулятивних та нейтральних прикладів. Особливо показовими є кейси з відповідями LLM, де «підштовхування» моделі до маніпулятивного стилю (відповідний system-промпт) добре відображається у висновках LLM-«судді», і система коректно фіксує зміну риторики.

Водночас робота чесно виявляє й обмеження підходу:

– вихідні датасети мають власні перекоси, неповноту й нерівномірне покриття різних технік;

– межі між окремими видами маніпуляцій розмиті, що призводить до варіативності в присвоєнні міток;

– негативна когорта (тексти без маніпуляцій) обмежена, тож система може бути надто чутливою до емоційної лексики;

– відсутня формальна кількісна оцінка якості в термінах класичних метрик (Accuracy, Precision, Recall, F1), оскільки основний фокус цієї роботи – на демонстрації методу та його інтерпретованості, а не на тренуванні окремого класифікатора.

Ці обмеження окреслюють напрями подальших досліджень, серед яких:

– розширення та балансування корпусу, зокрема за рахунок більшого числа текстів;

– побудова невеликої, але якісно розміченої тестової вибірки для формального вимірювання якості класифікації (Precision/Recall/F1, Brier score, ECE тощо);

– порівняння запропонованого методу з базовими моделями-класифікаторами на тих самих ембеддингах;

– адаптація системи до інших мов і доменів (український інформаційний простір, юридичні документи, медичні консультації);

– інтеграція додаткових механізмів контролю упередженості та етичних обмежень.

У підсумку можна стверджувати, що поставлена в роботі мета досягнута. Розроблено й реалізовано прототип методу виявлення маніпулятивних стратегій у відповідях LLM, який:

– спирається на комбінацію векторного індексу та великої мовної моделі;

– надає детальний, інтерпретований вихід у вигляді структурованої моделі;

– може використовуватися як людиною через веб-інтерфейс, так і зовнішніми системами через API.

Запропонований підхід є гнучким, розширюваним та сумісним із сучасною практикою побудови LLM-орієнтованих сервісів. Він демонструє, що навіть без повноцінного донавчання великих моделей можна побудувати практичний інструмент для підвищення прозорості й безпечності їхніх відповідей, що є актуальним завданням для подальшого розвитку систем штучного інтелекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Qi D. Strategies of manipulation in english speeches of state leaders. *Humanities science current issues*. 2023. Vol. 5, no. 60. P. 14–17. URL: https://doi.org/10.24919/2308-4863/60-5-3 (date of access: 07.12.2025).

2) Interpretable propaganda detection in news articles / S. Yu et al. *International conference recent advances in natural language processing*. 2021. URL: https://doi.org/10.26615/978-954-452-072-4_179 (date of access: 07.12.2025).

3) Wasburn P. C., Jowett G. S., O'Donnell V. Propaganda and persuasion. *Teaching sociology*. 1988. Vol. 16, no. 2. P. 223. URL: https://doi.org/10.2307/1317437 (date of access: 07.12.2025).

4) Recognition of propaganda techniques in newspaper texts: fusion of content and style analysis / A. Horák et al. *Expert systems with applications*. 2024. P. 124085. URL: https://doi.org/10.1016/j.eswa.2024.124085 (date of access: 07.12.2025).

5) Hierarchical multi-attention networks for document classification / Y. Huang et al. *International journal of machine learning and cybernetics*. 2021. Vol. 12, no. 6. P. 1639–1647. URL: https://doi.org/10.1007/s13042-020-01260-x (date of access: 07.12.2025).

6) OpenAI. Introducing GPT-5. *Our smartest, fastest, most useful model yet, with built-in thinking that puts expert-level intelligence in everyone's hands.* URL: https://openai.com/index/introducing-gpt-5/ (date of access: 06.12.2025).

7) Home \ anthropic. *AI research and products that put safety at the frontier*. URL: https://www.anthropic.com/ (date of access: 06.12.2025).

8) Mann S., Yadav D., Rathee D. Identification of racial propaganda in tweets using sentimental analysis models: a comparative study. *Proceedings of fourth doctoral symposium on computational intelligence*. Singapore, 2023.

P. 327–341. URL: https://doi.org/10.1007/978-981-99-3716-5_28 (date of access: 07.12.2025).

9) Chaudhari D., Pawar A. V. Empowering propaganda detection in resource-restraint languages: a transformer-based framework for classifying hindi news articles. *Big data and cognitive computing*. 2023. Vol. 7, no. 4. P. 175. URL: https://doi.org/10.3390/bdcc7040175 (date of access: 07.12.2025).

10) Detecting propaganda in news articles using large language models. *Engineering: open access*. 2024. Vol. 2, no. 1. P. 01–12. URL: https://doi.org/10.33140/eoa.01.02.10 (date of access: 07.12.2025).

11) Propaganda identification on twitter platform during COVID-19 pandemic using LSTM / A. M. U. D. Khanday et al. *Advances in cybersecurity, cybercrimes, and smart emerging technologies*. Cham, 2023. P. 303–314. URL: https://doi.org/10.1007/978-3-031-21101-0_24 (date of access: 07.12.2025).

12) Масенко Л. Маніпулятивні стратегії дискримінації української мови на телебаченні радянської і пострадянської України. *Мовознавство*. 2013. С. 83–88.

13) Nandwani P., Verma R. A review on sentiment analysis and emotion detection from text. *Social network analysis and mining*. 2021. Vol. 11, no. 1. URL: https://doi.org/10.1007/s13278-021-00776-6 (date of access: 07.12.2025).

14) Narang P., Singh A. V., Monga H. Sentiment score-based classification for fake news using machine learning and LSTM-BiLSTM. *Soft computing*. 2024. URL: https://doi.org/10.1007/s00500-024-09884-9 (date of access: 07.12.2025).

15) Goyal P., Pandey S., Jain K. Word vector representations. *Deep learning for natural language processing*. Berkeley, CA, 2018. P. 75–118. URL: https://doi.org/10.1007/978-1-4842-3685-7_2 (date of access: 07.12.2025).

16) Mariana C., Elena C., Serena V. FALCON (fallacies in COVID-19 network-based). *Zenodo*. URL: https://zenodo.org/records/15097907 (date of access: 06.12.2025).

17) SemEval-2020 task 11: detection of propaganda techniques in news articles. *ACL Anthology*. URL: https://aclanthology.org/2020.semeval-1.186/ (date of access: 06.12.2025).

18) Meme Persuasion technique text data. *Kaggle: Your Machine Learning and Data Science Community*. URL: https://www.kaggle.com/datasets/poojaparab/meme-persuasion-technique-text-data/ (date of access: 06.12.2025).

19) OpenAI | LangChain Reference. *Home - Docs by LangChain*. URL: https://reference.langchain.com/python/integrations/langchain_openai/ (date of access: 06.12.2025).

20) Streamlit: A faster way to build and share data apps. *Streamlit: A faster way to build and share data apps*. URL: https://streamlit.io/ (date of access: 06.12.2025).