СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр

спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

на тему «Система виявлення дезінформації в новинах на основі нейронних мереж»

Виконав: студент групи ІПЗ-21д _____ М.О. Весельський
(підпис) (ініціали і прізвище)

Керівник _____ Д.В. Ратов
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ О.І. Захожай
(підпис) (ініціали і прізвище)

Рецензент ___В.О. Лифар____

Київ - 2025

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування
(повна назва кафедри)

Освітній ступінь бакалавр
(бакалавр, магістр)

Спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ

Завідувач кафедри
_____ Захожай О.І.
“____” _____2025 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Весельський Максим Олексійоич
(прізвище, ім’я, по батькові)

1. Тема роботи «Система виявлення дезінформації в новинах на основі нейронних мереж»

Керівник роботи Ратов Денис Валентиноич
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджений наказом університету від “27” травня 2025 року №67/15.15-С

2. Строк подання студентом роботи 20 травня 2025 р.

3. Вихідні дані до роботи тексти новин, отримані шляхом парсингу з відкритих новинних ресурсів; API нейронної мережі для аналізу текстів на предмет достовірності; технічні вимоги до вебсайту для відображення результатів аналізу; програмне середовище Python та відповідні бібліотеки (requests, BeautifulSoup, Flask, API-клієнт)

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз методів виявлення дезінформації в новинах; Обґрунтування вибору технологій; Розробка архітектури програмного забезпечення; Реалізація парсингу новин; Інтеграція API нейронної мережі; Створення вебінтерфейсу для відображення результатів; Тестування та аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)
Блок-схема потоку даних у системі; Скріншот сторінки сайту з результатами аналізу новин (приклад 1); Скріншот сторінки сайту з результатами аналізу новин (приклад 2)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 03.03.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	03.03.2025	виконано
2	Укладання і погодження з керівником плану і етапів виконання роботи	20.03.2025	виконано
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	11.04.2025	виконано
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	10.05.2025	виконано
5	Укладання та тестування програмного продукту	02.06.2025	виконано
6	Укладання, оформлення та погодження пояснювальної записки з керівником	10.06.2025	виконано
7	Здача готової пояснювальної записки на кафедру	15.06.2025	виконано
8	Укладання доповіді і презентації	16.06.2025	виконано

Студент _____ М.О. Весельський
(підпис) (ініціали і прізвище)

Керівник роботи _____ Д.В.Ратов
(підпис) (ініціали і прізвище)

РЕФЕРАТ

Робота містить: 95 сторінок основного тексту, 28 сторінок додатків, 11 рисунків, 10 використаних джерел.

Метою випускної кваліфікаційної роботи є розробка та реалізація системи виявлення дезінформації в новинах на основі нейронних мереж з використанням великих мовних моделей.

В ході розробки системи виявлення дезінформації були вивчені теоретичні основи дезінформації, її класифікації та механізми поширення, а також проаналізовані існуючі методи та підходи до її виявлення, включаючи ручні, автоматичні та ті, що ґрунтуються на машинному/глибокому навчанні. Детально розглянуто загальні принципи функціонування нейронних мереж, еволюцію та архітектуру великих мовних моделей (LLM), зокрема трансформерів, а також методи представлення тексту для нейронних мереж.

Проведено порівняльний аналіз існуючих систем виявлення дезінформації, що дозволило обґрунтувати вибір підходу з використанням Gemini API як оптимального рішення для даної задачі. Була спроектована модульна архітектура системи, реалізовані модулі збору та попередньої обробки даних (з використанням RSS-парсингу та веб-скрапінгу), аналізу дезінформації на основі LLM (із застосуванням деталізованого промпту), зберігання даних у базі SQLite3 та візуалізації результатів через веб-інтерфейс на GitHub Pages. Проведені експериментальні дослідження та якісний аналіз ефективності системи, що включали оцінку швидкості збору та обробки даних, а також якість відповідей LLM та корисність її пояснень.

Вироблено опис процесу розробки та аналізу ефективності системи. Реалізований та детально описаний користувальницький веб-інтерфейс, створені знімки екранних форм програмного засобу. Продемонстровано результат виконаної роботи.

Система задовольняє всім вимогам, пред'явленим в технічному завданні.

ЗМІСТ

ВСТУП 9
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ДЕЗІНФОРМАЦІЇ ТА ЇЇ ВИЯВЛЕННЯ ... 12
1.1. Дезінформація як соціально-інформаційне явище 12
1.1.1. Визначення та класифікація дезінформації 12
1.1.2. Механізми поширення дезінформації в мережевих медіа та соціальних мережах 14
1.1.3. Соціальні, політичні та економічні наслідки поширення дезінформації 17
1.2. Методи та підходи до виявлення дезінформації 20
1.2.1. Ручні та напівавтоматичні методи 20
1.2.2. Автоматичні методи 22
1.2.3. Роль машинного навчання та глибокого навчання у виявленні дезінформації 26
РОЗДІЛ 2. НЕЙРОННІ МЕРЕЖІ ТА ОБРОБКА ПРИРОДНОЇ МОВИ В КОНТЕКСТІ АНАЛІЗУ ТЕКСТУ 29
2.1. Загальні принципи функціонування нейронних мереж 29
2.1.1. Базові концепції нейронних мереж 29
2.1.2. Основні архітектури нейронних мереж, що використовуються для тексту 32
2.2. Великі мовні моделі (LLM) 36
2.2.1. Концепція та еволюція LLM 36
2.2.2. Архітектура трансформерів та механізм уваги (Attention Mechanism) як основа сучасних LLM 38
2.2.3. Методи попереднього навчання (pre-training) та тонкого налаштування (fine-tuning) LLM 40
2.2.4. Застосування LLM для задач обробки природної мови 43
2.3. Методи представлення тексту для нейронних мереж 46
2.3.1. Токенізація та нормалізація тексту 46

2.3.2. Векторні представлення слів (Word Embeddings: Word2Vec, GloVe) – коротко, як історичний контекст 49

2.3.3. Контекстуальні вбудовування (Contextual Embeddings: BERT, RoBERTa, Gemini embeddings) – як сучасний підхід 51

РОЗДІЛ 3. АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ВИЯВЛЕННЯ ДЕЗІНФОРМАЦІЇ НА ОСНОВІ МАШИННОГО НАВЧАННЯ 55

3.1. Огляд наукових досліджень та проектів у сфері виявлення дезінформації 55

3.1.1. Системи на основі традиційних методів машинного навчання (наприклад, SVM, Naive Bayes, Random Forest) та їх застосування 55

3.1.2. Системи на основі глибокого навчання (Deep Learning) – приклади використання RNN, CNN, Transformer-базованих моделей . 56

3.1.3. Приклади комерційних або відкритих платформ, що займаються фактчекінгом або виявленням дезінформації 58

3.2. Порівняльний аналіз існуючих підходів та їх обмежень 60

3.2.1. Переваги та недоліки різних архітектур та алгоритмів 60

3.2.2. Проблеми з даними (доступність, якість розмітки, збалансованість корпусів) 63

3.2.3. Виклики у виявленні дезінформації 64

3.3. Обґрунтування вибору підходу для даної роботи 65

3.3.1. Підходи використанням LLM (Gemini API) 66

3.3.2. Роль у використанні LLM 67

РОЗДІЛ 4. РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ ДЕЗІНФОРМАЦІЇ 70

4.1. Архітектура системи 70

4.1.1. Загальна схема компонентів системи (бажано додати блок-схему, яка візуалізує потік даних). 70

4.1.2. Детальний опис взаємодії між модулями. 71

4.2. Модуль збору та попередньої обробки даних 72

4.2.1. Вибір джерел новин (конкретні популярні українські новинні сайти, які ви парсили) 72

4.2.2. Реалізація парсингу RSS-стрічок (feedparser) – детальний опис процесу 73

4.2.3. Реалізація веб-скрапінгу повного тексту статей (BeautifulSoup) – опис алгоритму вилучення тексту, обробка різних структур сторінок. 73

4.2.4. Потенційні проблеми та можливі рішення під час збору даних.... 74

4.3. Модуль аналізу дезінформації на основі LLM 75

4.3.1. Обґрунтування вибору Gemini API (переваги над ChatGPT API, швидкість, ліміти)..... 76

4.3.2. Детальний опис промпту для Gemini API. Чому саме такі інструкції були надані..... 77

4.3.3. Формат вхідних та вихідних даних для LLM (JSON-структура, яка передається та отримується)..... 78

4.3.4. Інтеграція з Python (google.genai, google.genai.types) – фрагменти коду, пояснення 79

4.4. Модуль зберігання даних 81

4.4.1. Вибір та обґрунтування використання SQLite бази даних 81

4.4.2. Структура даних у базі даних..... 82

4.4.3. Механізми взаємодії з базою даних (додавання, читання, оновлення записів)..... 84

4.5. Модуль візуалізації та веб-інтерфейс 85

4.5.1. Обґрунтування вибору GitHub Pages для хостингу..... 86

4.5.2. Розробка фронтенду (HTML, CSS, JavaScript)..... 86

4.5.3. Механізм експорту даних з SQLite у JSON та автоматичного оновлення 88

4.5.4. Дизайн та користувацький досвід (UX) – обговорення стилю, кольорової схеми, врахування рекомендацій дизайнера..... 89

РОЗДІЛ 5. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ..... 91

5.1. Методологія проведення експериментів 91

5.1.1. Опис тестового набору даних 91

5.1.2. Критерії оцінки ефективності системи 92

5.2. Результати аналізу дезінформації за допомогою LLM 94

5.2.1. Приклад аналізу новин з результатом LLM та її поясненням 94

5.2.2. Якісний аналіз відповідей LLM..... 96

5.2.3. Аналіз пояснень від LLM: наскільки вони корисні та зрозумілі для користувача 98
5.3. Оцінка продуктивності системи 99
5.3.1. Швидкість збору даних 100
5.3.2. Швидкість обробки LLM 100
5.3.3. Частота оновлення даних на сайті та її вплив на актуальність інформації. 101
5.4. Обговорення отриманих результатів 102
5.4.1. Сильні та слабкі сторони розробленої системи 102
5.4.2. Порівняння з існуючими рішеннями 104
5.4.3. Обмеження поточного рішення та потенційні напрямки для покращення 105
ВИСНОВКИ 108
ДОДАТКИ 111

ВСТУП

У сучасному глобалізованому світі, де інформація поширюється з надзвичайною швидкістю через цифрові медіа та соціальні мережі, проблема дезінформації набуває критичного значення. Неправдива, спотворена або маніпулятивна інформація, яка подається як достовірна, здатна суттєво впливати на громадську думку, політичні процеси, соціальну стабільність та навіть національну безпеку. Особиста нетерпимість до розповсюдження неперевірених даних, що видаються за істину, підкреслює глибоку суспільну потребу у надійних механізмах фільтрації інформаційного потоку. Відсутність об'єктивних інструментів верифікації призводить до підриву довіри до медіа, посилення поляризації суспільства та дезорієнтації індивідів. У цьому контексті розробка ефективних автоматизованих систем для виявлення дезінформації є не просто актуальною, а стратегічно важливою задачею, здатною забезпечити інформаційну гігієну та підтримати здатність суспільства до критичного мислення в умовах постійного інформаційного тиску.

Метою цієї дипломної роботи є розробка та реалізація автоматизованої системи виявлення дезінформації у новинних матеріалах на основі великих мовних моделей (LLM), яка забезпечить оперативний, об'єктивний та пояснювальний аналіз інформаційного контенту.

Для досягнення поставленої мети було сформульовано наступні завдання:

1. Розробити та реалізувати модуль для автоматизованого збору новинних статей з популярних веб-ресурсів, включаючи використання RSS-стрічок та веб-скрапінгу повного тексту.
2. Вибрати та інтегрувати оптимальну велику мовну модель (LLM), адаптовану для задачі аналізу тексту на предмет дезінформації.
3. Сформулювати та оптимізувати промпт для обраної LLM, що дозволить їй функціонувати як експерту з медіааналітики, надаючи об'єктивні оцінки та пояснення.

4. Створити архітектуру та реалізувати базу даних для структурованого зберігання зібраних новинних даних та результатів їхнього аналізу.

5. Розробити динамічний веб-інтерфейс, що забезпечує візуалізацію результатів аналізу дезінформації у реальному часі та автоматичне оновлення даних.

6. Провести експериментальні дослідження функціональності та ефективності розробленої системи, а також здійснити якісний аналіз отриманих результатів.

Об'єктом дослідження є процес поширення та виявлення дезінформації в медіа-просторі, зокрема у новинних стрічках. Предметом дослідження є методи, алгоритми та інструменти автоматизованого виявлення дезінформації в текстових новинних матеріалах, засновані на застосуванні нейронних мереж, зокрема великих мовних моделей.

Наукова новизна роботи полягає в розробці та апробації комплексного підходу до виявлення дезінформації, що інтегрує автоматизований збір даних, використання попередньо навченої великої мовної моделі зі спеціалізованим промптом для глибокого аналізу контенту та динамічної візуалізації результатів. Особливий акцент зроблено на отриманні не тільки вердикту щодо дезінформації, але й пояснення логіки рішення LLM, що підвищує прозорість та довіру до системи.

Практична цінність розробленої системи полягає у її здатності забезпечувати користувачів (журналістів, аналітиків, пересічних громадян) оперативним, об'єктивним та зрозумілим інструментом для швидкого розрізнення достовірної інформації від дезінформації. Система може слугувати проактивним засобом для підвищення медіаграмотності населення, підтримки об'єктивності журналістських розслідувань та зменшення впливу маніпулятивних інформаційних кампаній. Можливість легкого масштабування та інтеграції з різними платформами (веб, Telegram, Discord) розширює потенційні сфери її застосування.

Дипломна робота складається з Вступу, п'яти основних розділів, Висновків, Списку використаних джерел та Додатків. У Розділі 1 буде розглянуто теоретичні аспекти дезінформації, її класифікацію та методи виявлення. Розділ 2 присвячений огляду фундаментальних принципів нейронних мереж, зокрема великих мовних моделей, та методів обробки природної мови. У Розділі 3 буде проведено аналіз існуючих систем виявлення дезінформації на основі машинного навчання, а також обґрунтовано вибір підходу для даної роботи. Розділ 4 детально описує процес розробки системи, включаючи архітектуру, модулі збору та аналізу даних, зберігання та візуалізацію. Розділ 5 представляє результати експериментальних досліджень, їхній якісний аналіз та оцінку продуктивності системи. Висновки підсумовують отримані результати та окреслюють перспективи подальшого розвитку.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ДЕЗІНФОРМАЦІЇ ТА ЇЇ ВИЯВЛЕННЯ

1.1. Дезінформація як соціально-інформаційне явище

У сучасному інформаційному просторі, де цифрові медіа та соціальні мережі є основними каналами поширення новин, здатність відрізняти правду від вимислу стає життєво необхідною. Проте, цей процес ускладнюється постійним зростанням обсягів дезінформації – інформації, що є свідомо неправдивою або маніпулятивною і поширюється з метою введення в оману. Дезінформація не є новим явищем, але її масштаби та швидкість поширення значно зросли завдяки розвитку технологій та мережевих комунікацій. Це створює серйозні загрози для суспільної стабільності, демократичних процесів та індивідуального добробуту.

1.1.1. Визначення та класифікація дезінформації

Для ефективного вивчення та протидії дезінформації важливо чітко розрізняти її форми та пов'язані з нею терміни.

Дезінформація (disinformation) – це свідомо неправдива або оманлива інформація, яка створюється та поширюється з наміром ввести в оману, маніпулювати аудиторією або завдати шкоди. Ключовим аспектом дезінформації є злий умисел її творців та розповсюджувачів. Метою може бути вплив на політичні вибори, дискредитація опонентів, розпалювання конфліктів, отримання фінансової вигоди або просто створення хаосу.

На відміну від дезінформації, існує поняття мізінформації (misinformation). Мізінформація – це також неправдива інформація, але вона поширюється без злого наміру ввести в оману. Людина, яка поширює мізінформацію, сама вірить у її правдивість і не усвідомлює її неточності. Це може бути результатом помилки, неправильного тлумачення, застарілих даних або відсутності фактчекінгу. Наприклад, людина репостить новину, яку вона вважає правдивою, але яка насправді є неточною, – це мізінформація. Якщо ж ця новина була створена і поширена з метою обману, то це дезінформація.

Розрізняють кілька основних форм дезінформації:

- Фейкові новини (Fake News): це повністю сфабриковані статті або повідомлення, що імітують формат традиційних новинних видань. Їхня мета – ввести читача в оману, викликати сильні емоції (гнів, страх, обурення) або поширити певну ідеологію. Приклад: вигадана стаття про "сотні тисяч мігрантів, що захоплюють місто", яка поширюється з метою розпалювання ксенофобії.
- Пропаганда (Propaganda): це систематичне поширення ідей, доктрин або інформації (часто викривленої або частково неправдивої) з метою впливу на громадську думку та формування бажаного ставлення до певної групи, ідеології чи події. Пропаганда може використовувати елементи правди, але подає їх у вибірковий або спотворений спосіб, щоб викликати певну реакцію. Приклад: державні медіа постійно висвітлюють події лише з одного боку, ігноруючи або спотворюючи іншу інформацію, щоб сформувати потрібний образ "ворога" або "героя".
- Маніпуляція (Manipulation): це приховані методи впливу на свідомість і поведінку людей шляхом спотворення, вибіркового подання або замовчування інформації.

Маніпуляція може включати:

- Викривлення контексту: надання правдивої інформації, але поза її оригінальним контекстом, що змінює її значення.
- "Клікбейт" (Clickbait): заголовки, що вводять в оману або перебільшують зміст статті, щоб привернути увагу та змусити користувача перейти за посиланням.
- Фабрикація доказів: створення фальшивих документів, зображень, відео (deepfakes) або цитат, які потім використовуються як "докази" неправдивих тверджень.

– Емоційне розпалювання: використання мови, яка викликає сильні емоції, щоб заглушити раціональне мислення та спонукати до певних дій або переконань.

Порівняльна таблиця форм дезінформації та мізінформації:

Таблиця 1.1.1

Характеристика	Дезінформація	Мізінформація
Намір	Свідомий, злий намір ввести в оману	Відсутність злого наміру, помилка
Правдивість інформації	Неправдива, спотворена, оманлива	Неправдива, але вважається правдивою
Причини поширення	Політичні, економічні, ідеологічні цілі	Некомпетентність, необізнаність, поспіх
Приклад	Сфабрикована новина про злочини конкурента	Репост чуток, які виявилися неправдивими

Розуміння цих відмінностей є фундаментальним для розробки ефективних систем виявлення, оскільки методи протидії різним формам та намірам можуть суттєво відрізнятися. Наша система фокусуватиметься на виявленні саме дезінформації, враховуючи її потенційну шкоду та навмисний характер.

1.1.2. Механізми поширення дезінформації в мережевих медіа та соціальних мережах

Поширення дезінформації в сучасному цифровому просторі значно відрізняється від традиційних медіаканалів завдяки масштабу, швидкості та інтерактивності мережевих медіа та соціальних мереж. Ці платформи стали основними "інкубаторами" та "розповсюджувачами" неправдивої інформації, використовуючи при цьому як людський фактор, так і алгоритмічні особливості.

Ключові механізми поширення включають:

1. Алгоритми платформ:

– Ехо-камери та фільтр-бульбашки: алгоритми соціальних мереж (Facebook, X (Twitter), TikTok тощо) та новинних агрегаторів персоналізують контент для кожного користувача на основі його попередніх взаємодій, уподобань та політичних поглядів. Це призводить до формування "ехо-камер", де користувачі бачать лише ту інформацію, яка підтверджує їхні існуючі переконання, і відфільтровується контент, що суперечить їм. Таким чином, дезінформація, яка відповідає світогляду користувача, отримує додаткове прискорення у поширенні всередині його "бульбашки", тоді як критичні голоси залишаються нечутними.
– Прихильність до залучення (Engagement Bias): алгоритми часто оптимізовані для максимального залучення користувачів (лайки, репости, коментарі). Контент, який викликає сильні емоції (обурення, страх, здивування), як правило, генерує більше взаємодій. Дезінформація часто спеціально розробляється для провокування таких емоцій, що надає їй алгоритмічну перевагу у поширенні, оскільки вона отримує більше показів та рекомендацій.

2. Людський фактор:

– Соціальне підтвердження та колективний ефект: люди схильні вірити інформації, якщо її підтверджує велика кількість їхніх друзів, знайомих або авторитетних для них осіб у соціальних мережах. Це створює "каскад віри", де неперевірена інформація швидко поширюється, оскільки кожен новий "поділився" додає їй "достовірності" в очах інших.
– Недостатня медіаграмотність: багато користувачів не мають достатніх навичок критичного мислення та перевірки інформації. Вони можуть не розрізняти достовірні джерела від сумнівних, не перевіряти заголовки та фотографії, і легко стають жертвами маніпуляцій.

- Емоційне реагування: емоційно заряджений контент, особливо той, що викликає гнів або страх, часто поширюється швидше, ніж раціональні та перевірені факти. Люди реагують на емоції, а не на зміст, що робить їх вразливими до маніпуляцій.
- Свідоме поширення: деякі користувачі свідомо поширюють дезінформацію через власні політичні, ідеологічні чи інші переконання, або навіть за винагороду.

3. Автоматизовані засоби (боти та бот-мережі):

- Боти: Автоматизовані акаунти, запрограмовані на поширення певного контенту, ретвіти, коментування або збільшення кількості підписників. Боти можуть швидко масштабувати поширення дезінформації, створюючи ілюзію масової підтримки або дискусії.
- Бот-мережі (Botnets): координовані групи ботів, які діють за єдиним сценарієм. Вони можуть бути використані для "накрутки" трендів, посилення певних наративів, атак на неугодні джерела або для створення "інформаційного шуму", в якому важко відрізнити правду.
- Тролі та "ферми тролів": це люди (або організації), які свідомо створюють та поширюють провокаційний, оманливий або дезінформаційний контент, часто видаючи себе за звичайних користувачів. Їхня мета – поляризувати суспільство, дискредитувати джерела або поширити пропаганду.

4. Мультиформатність контенту:

Дезінформація може поширюватися не тільки у текстовому форматі, а й через зображення (з маніпульованими підписами або фотоколажами), відео (deepfakes, вирвані з контексту фрагменти) та аудіозаписи. Візуальний контент часто сприймається як більш достовірний і поширюється швидше, оскільки вимагає менше зусиль для споживання.

Сукупність цих механізмів створює складну екосистему, в якій дезінформація може швидко набувати вірусного поширення, досягаючи

мільйонів користувачів ще до того, як її буде викрито та спростовано. Це підкреслює необхідність розробки ефективних автоматизованих систем виявлення, які можуть діяти на випередження

1.1.3. Соціальні, політичні та економічні наслідки поширення дезінформації

Поширення дезінформації у масштабах сучасного інформаційного суспільства не є лише незначною незручністю; воно має глибокі та руйнівні наслідки, що охоплюють соціальну, політичну та економічну сфери. Розуміння цих наслідків є ключовим для усвідомлення важливості розробки систем протидії, таких як пропонована у цій роботі.

Соціальні наслідки

Дезінформація суттєво підриває соціальну тканину суспільства, впливаючи на міжособистісні відносини та колективну свідомість:

Ерозія довіри: Систематичне поширення неправдивої інформації руйнує довіру до традиційних медіа, наукових установ, державних інститутів та навіть довіру між громадянами. Коли люди не можуть розрізнити правду від брехні, зростає цинізм та апатія, що ускладнює формування єдиної позиції щодо важливих суспільних проблем.

Поляризація та розкол суспільства: Дезінформація часто використовується для розпалювання конфліктів та посилення існуючих соціальних, етнічних, релігійних чи політичних розбіжностей. Створюючи хибні наративи про "інших", вона сприяє формуванню "ми-вони" менталітету, що призводить до глибоких розколів та ускладнює конструктивний діалог.

Загроза громадському здоров'ю та безпеці: У кризових ситуаціях (наприклад, пандемії, стихійні лиха) дезінформація може поширювати паніку, підривати довіру до медичних рекомендацій (як це було з неправдивими відомостями про вакцини) або провокувати небезпечну поведінку, ставлячи під загрозу життя та здоров'я людей.

Маніпуляція емоціями та поведінкою: Дезінформація часто спрямована на викликання сильних емоцій, таких як страх, гнів або обурення. Це може призводити до ірраціональних рішень, участі в несанкціонованих акціях або навіть актів насильства.

Політичні наслідки

Вплив дезінформації на політичну систему є особливо небезпечним, оскільки вона може прямо загрожувати демократичним процесам та національній безпеці:

- Підрив демократичних процесів: Дезінформація активно використовується для маніпуляції виборами, дискредитації кандидатів, спотворення програм партій та посилення суспільного невдоволення. Вона може впливати на рішення виборців, знижувати явку або легітимність виборів.
- Дестабілізація урядів та міжнародних відносин: Поширення неправдивих звинувачень або сфабрикованих новин може призвести до урядових криз, протестів, а також погіршити відносини між державами, розпалюючи конфлікти або створюючи хибні приводи для агресії (частина гібридної війни).
- Ерозія довіри до державних інститутів: Дезінформаційні кампанії часто спрямовані на підрив авторитету поліції, судів, армії та інших державних структур, що зменшує здатність держави ефективно функціонувати та захищати своїх громадян.
- Втручання у внутрішні справи: Іноземні актори часто використовують дезінформацію як інструмент для втручання у внутрішні справи інших країн, підтримуючи певні політичні сили або послаблюючи опонентів.

Економічні наслідки

Хоча економічні наслідки дезінформації можуть бути менш очевидними на перший погляд, вони також є значними:

- Маніпуляції ринками: Поширення неправдивих чуток або фінансової дезінформації може викликати паніку на фондових ринках, призвести до обвалу цін на акції певних компаній, штучно збільшити попит на певні товари або послуги, завдаючи мільярдних збитків.
- Шкода репутації бізнесу: Дезінформаційні кампанії можуть бути спрямовані на дискредитацію компаній, їхніх продуктів або послуг, що призводить до втрати клієнтів, зниження довіри та значних фінансових втрат.
- Втрати в інвестиціях та туризмі: Неправдиві новини про нестабільність у регіоні, екологічні катастрофи або загрози безпеці можуть відлякувати інвесторів та туристів, що негативно позначається на економіці країни або регіону.
- Витрати на протидію: Держави, міжнародні організації та приватні компанії змушені витрачати значні ресурси на фактчекінг, розробку стратегій медіаграмотності, технічні рішення для виявлення та блокування дезінформації. Ці витрати могли б бути спрямовані на більш продуктивні сфери.

Чому це так важливо?

Кумулятивний ефект цих наслідків робить дезінформацію однією з найсерйозніших загроз для стабільності та розвитку сучасного суспільства. Вона підриває основи, на яких ґрунтується функціонування демократій та вільних ринків: довіру до інформації та здатність громадян до об'єктивного аналізу. У зв'язку з цим, розробка та впровадження ефективних, автоматизованих систем виявлення дезінформації стає невідкладним завданням, яке сприятиме захисту інформаційного простору та забезпеченню об'єктивного інформування населення. Саме тому проєкт "Система виявлення дезінформації в новинах на основі нейронних мереж" має високу актуальність та соціальну значущість.

1.2. Методи та підходи до виявлення дезінформації

Протидія дезінформації вимагає комплексного підходу, що поєднує різноманітні методи та інструменти. Історично першими та досі важливими є ручні та напівавтоматичні методи, які базуються на експертній оцінці та аналізі інформації. Проте, зростання обсягів даних та швидкість їх поширення зумовлюють потребу в розробці автоматизованих систем, які можуть доповнювати та масштабувати зусилля людини.

1.2.1. Ручні та напівавтоматичні методи

Ручні методи виявлення дезінформації є найбільш традиційними і базуються на глибокому аналізі контенту та його відповідності відомим фактам, джерелам та логіці. Основним інструментом у цьому підході є фактчекінг.

- Фактчекінг (Fact-checking): Це процес верифікації тверджень, заяв або інформаційних повідомлень шляхом порівняння їх з об'єктивними, достовірними джерелами. Фактчекери ретельно перевіряють надані дані, цитати, статистику, зображення та відео, щоб встановити їхню правдивість. Цей процес є трудомістким і вимагає високої кваліфікації аналітиків, їхньої неупередженості та доступу до широкого спектру інформаційних баз даних.
- Експертна оцінка: Окрім безпосереднього фактчекінгу, ручні методи включають залучення профільних експертів (наприклад, у галузі медицини, економіки, військової справи), які можуть оцінити достовірність інформації на основі своїх глибоких знань та досвіду. Експерти можуть виявити логічні суперечності, наукову безглуздість або упередженість, які можуть бути неочевидними для неспеціалістів.

Напівавтоматичні методи поєднують ручний аналіз з використанням певних інструментів, що прискорюють процес верифікації. Ці інструменти можуть включати:

- Пошукові системи: Швидкий пошук інформації в мережі для порівняння даних з різних джерел.

- Інструменти для аналізу зображень/відео: Програмне забезпечення для виявлення ознак маніпуляції з медіафайлами (наприклад, зміна метаданих, викривлення об'єктів, наявність "артефактів" від редагування).
- Бази даних фактів та спростувань: Спеціалізовані платформи, що акумулюють перевірені факти та спростування поширених міфів або фейків, дозволяючи швидше знаходити відповідності.

Роль фактчекінгових організацій:

На тлі зростання дезінформації у світі сформувалася значна кількість спеціалізованих фактчекінгових організацій. Ці організації (наприклад, StopFake, VoxCheck в Україні, Snopes, PolitiFact у світі) відіграють критично важливу роль у боротьбі з неправдивою інформацією. Їхня діяльність включає:

- Моніторинг інформаційного простору: Постійний пошук потенційно недостовірних новин та заяв.
- Верифікація інформації: Проведення ретельного фактчекінгу за встановленими методологіями.
- Публікація спростувань: Оприлюднення результатів свого аналізу, чітко позначаючи інформацію як "правдиву", "неправдиву", "частково правдиву" або "безпідставну".
- Співпраця з платформами: Багато соціальних мереж та новинних агрегаторів співпрацюють з фактчекінговими організаціями для ідентифікації та маркування дезінформації, а іноді й для її видалення або зменшення її поширення.

Обмеження ручних та напівавтоматичних методів:

Незважаючи на високу точність та важливість ручного фактчекінгу, ці методи мають суттєві обмеження, особливо в контексті сучасних обсягів інформації:

- Масштабованість: Ручний аналіз є вкрай повільним та ресурсомістким. Одна людина або навіть невелика команда фізично не може перевірити мільйони новинних повідомлень, що генеруються щодня. Це призводить до значної затримки між поширенням дезінформації та її спростуванням.

- Швидкість поширення: Дезінформація може стати вірусною та охопити мільйони користувачів за лічені хвилини або години. Ручні методи просто не встигають за такою швидкістю, що дозволяє неправдивій інформації завдати значної шкоди ще до її спростування.
- Суб'єктивність: Хоча фактчекери прагнуть до об'єктивності, певний ступінь суб'єктивності або упередженості може бути присутнім.
- Обробка мови: Для аналізу великих обсягів тексту з різних мов та діалектів ручні методи стають непрактичними.

Саме ці обмеження ручних та напівавтоматичних підходів стали рушійною силою для розробки автоматизованих систем виявлення дезінформації, таких як запропонована в цій дипломній роботі. Наш проєкт прагне доповнити та масштабувати зусилля фактчекерів, надаючи швидкий первинний аналіз новинного контенту з використанням потужності нейронних мереж, що дозволить ефективніше реагувати на виклики сучасного інформаційного простору.

1.2.2. Автоматичні методи

На противагу ручним та напівавтоматичним підходам, автоматичні методи виявлення дезінформації використовують алгоритмічні підходи та машинне навчання для обробки великих обсягів даних зі швидкістю та ефективністю, недоступними для людини. Ці методи ґрунтуються на аналізі різних аспектів інформаційного контенту та його поширення.

Лінгвістичний аналіз (аналіз стилю, лексики, синтаксису).

Лінгвістичний аналіз зосереджується на внутрішніх характеристиках самого тексту новини, виявляючи патерни, які можуть вказувати на його недостовірність або маніпулятивний характер. Ідея полягає в тому, що дезінформація часто має певні мовні особливості, які відрізняють її від достовірних повідомлень.

Аналіз стилю:

емоційність та суб'єктивність: недостовірні новини часто використовують підвищену емоційність, велику кількість окличних знаків, оціночних слів та іншої лексики, що викликає сильну емоційну реакцію, а не раціональне сприйняття. Достовірні новини, як правило, більш нейтральні та об'єктивні;

неясні або узагальнені формулювання: дезінформація може містити розпливчасті посилання на "експертів", "інсайдерів" або "джерела", без конкретних імен чи фактів;

особливості граматики та пунктуації: хоча це не є прямим індикатором дезінформації, недбалість у граматиці, орфографії або пунктуації може вказувати на низьку якість джерела, що корелює з більшою ймовірністю поширення неправди.

Аналіз лексики:

використання певних термінів: дезінформація може часто містити слова, що асоціюються з теоріями змови, сенсаційними твердженнями або термінами, що розпалюють ненависть.

незвичайні слова та фрази: виявлення аномальних або рідкісних слів у контексті звичайних новин може бути маркером.

Аналіз синтаксису:

складність речень: деякі дослідження припускають, що фейкові новини можуть мати простіші або, навпаки, надмірно складні синтаксичні конструкції для маніпуляції читачем;

активний/пасивний стан: Аналіз частоти використання активного або пасивного стану дієслів може вказувати на намагання приховати джерело дії або уникнути відповідальності;

застосування у проєкті: наш проєкт використовує велику мовну модель Gemini API, яка здатна глибоко аналізувати лінгвістичні особливості тексту. Модель, виступаючи в ролі "експерта з медіааналітики", може виявляти ці тонкі лінгвістичні патерни, що вказують на маніпулятивний стиль, необ'єктивність або інші ознаки дезінформації. Наприклад, як у випадку з гороскопом, де нейронна мережа ідентифікувала статтю як "необ'єктивну", ймовірно,

базуючись на лінгвістичному аналізі використання термінів, що не відповідають науковому дискурсу.

Аналіз метаданих розглядає інформацію, що супроводжує новину, а не її безпосередній зміст. Ця інформація може надати важливі підказки щодо достовірності.

Джерело новини: Оцінка репутації та надійності новинного порталу або веб-сайту. Відомі, авторитетні медіа з багаторічною історією та редакційною політикою мають вищий рівень довіри, ніж невідомі сайти, що виникли нещодавно, або ті, що відомі поширенням сумнівної інформації.

Автор: Перевірка автора статті (якщо він вказаний). Чи є автор реальною особою? Чи є він експертом у темі, про яку пише? Чи пов'язаний він з організаціями, які можуть мати приховані мотиви?

Дата публікації: Перевірка, чи не є новина застарілою, чи не є вона "реанімованою" старою новиною, яка поширюється у новому контексті для маніпуляції (так звані "ретрофейки").

Метадані зображень/відео: Аналіз EXIF-даних фотографій, хронометражу відео, історії завантажень може виявити, чи було медіа відредаговано або використано поза контекстом.

Мережа поширення: Аналіз того, як новина поширюється – чи є це органічне поширення, чи в ньому беруть участь боти, скоординовані акаунти, або незвичайні патерни репостів.

Застосування у проєкті: Хоча наш поточний проєкт переважно зосереджений на текстовому аналізі, збір news_publication_date під час парсингу RSS-стрічок є важливим елементом метаданих. У майбутньому, система може бути розширена для аналізу репутації джерела, що дозволить інтегрувати зовнішні бази даних надійних/ненадійних медіаресурсів для підвищення точності оцінки.

Аналіз поведінки користувачів (реакції, репости, коментарі)

Цей метод виявлення дезінформації ґрунтується на спостереженні за тим, як користувачі взаємодіють з контентом. Певні аномалії у поведінці можуть вказувати на спробу маніпуляції.

Аномальні патерни взаємодії:

Незвичайна кількість лайків/репостів: Якщо новина, особливо з невідомого джерела, раптом набирає величезну кількість взаємодій за короткий проміжок часу, це може свідчити про використання ботів або "ферм тролів".

Диспропорція між лайками та коментарями: Багато лайків при дуже малому або однотипному коментуванні також може бути індикатором штучної активності.

Аналіз коментарів:

Однотипні або спамні коментарі: Наявність великої кількості однотипних коментарів, або коментарів, що не відповідають темі, може вказувати на скоординовану атаку.

Мова ворожнечі або агресивні висловлювання: Коментарі, що містять ненависницькі висловлювання, часто супроводжують дезінформацію, яка покликана розпалювати конфлікти.

Аналіз мережі поширення: Вивчення того, хто саме поширює новину (нові акаунти, акаунти з малим числом підписників, акаунти з підозрілою активністю) може виявити скоординовані кампанії.

Застосування у проєкті: Хоча наш поточний проєкт не включає прямий аналіз поведінки користувачів у соціальних мережах, розуміння цих механізмів є важливим для подальшого розвитку. В майбутньому, система може бути розширена для інтеграції з API соціальних мереж для збору та аналізу цих показників, що дозволить побудувати більш комплексну систему виявлення дезінформації. Наразі, фокус на глибокому лінгвістичному аналізі за допомогою LLM дозволяє ефективно ідентифікувати саме контентну складову дезінформації.

1.2.3. Роль машинного навчання та глибокого навчання у виявленні дезінформації

Усвідомлення обмежень ручних методів та необхідності ефективної обробки величезних обсягів інформації призвело до стрімкого розвитку автоматизованих підходів до виявлення дезінформації, де ключову роль відіграють технології машинного навчання (ML) та глибокого навчання (Deep Learning). Ці методи дозволяють створювати моделі, здатні самостійно ідентифікувати складні патерни в даних, що є неможливим для традиційних алгоритмів або людського аналізу у великих масштабах.

Машинне навчання надає інструменти для побудови моделей, які навчаються на великих обсягах маркованих даних (наприклад, новини, які були позначені як "правдиві" або "дезінформація"). Основна ідея полягає у тому, що алгоритм "вивчає" відмінності між цими двома категоріями інформації. До традиційних методів машинного навчання, що застосовуються у цій сфері, належать:

- Методи опорних векторів (Support Vector Machines, SVM): Ефективні для класифікації тексту шляхом побудови гіперплощини, що оптимально розділяє класи.
- Байєсівські класифікатори (Naive Bayes): Використовують ймовірнісні підходи для визначення категорії тексту на основі частоти слів.
- Дерева рішень та випадкові ліси (Decision Trees, Random Forests): Моделі, що приймають рішення на основі послідовності умов, виведених з навчальних даних.

Ці моделі дозволяють виявляти ознаки дезінформації на основі статистичних патернів у лінгвістичних характеристиках тексту, метаданих або поведінкових сигналах.

Проте, справжній прорив у цій галузі настав із розвитком глибокого навчання, підрозділу машинного навчання, який використовує багатошарові нейронні мережі. Глибоке навчання дозволяє моделям автоматично вивчати

складні, абстрактні ознаки (features) з сирих даних, що є особливо цінним для обробки природної мови.

Ключові переваги машинного навчання та глибокого навчання у виявленні дезінформації:

1. Масштабованість: на відміну від ручних методів, ML/DL моделі можуть обробляти мільйони новинних статей щодня, що є критично важливим в умовах сучасного інформаційного вибуху. Це дозволяє покрити значно більший обсяг даних і реагувати на дезінформацію швидше.

2. Швидкість: після навчання моделі здатні здійснювати класифікацію та аналіз практично в реальному часі, що дозволяє оперативніше виявляти та маркувати дезінформацію.

3. Виявлення складних патернів: нейронні мережі, особливо глибокі, можуть виявляти неочевидні та тонкі зв'язки та патерни в тексті, які людині важко помітити, наприклад, стилістичні особливості, приховані маніпуляції або взаємозв'язки між різними елементами інформації.

4. Адаптивність: моделі можуть бути перенавчені (доповнені новими даними), щоб адаптуватися до нових видів дезінформації, які постійно еволюціонують.

5. Зниження суб'єктивності: хоча дані для навчання все ще потребують розмітки людиною, сам процес класифікації моделями є об'єктивним, що дозволяє зменшити вплив людських упереджень на фінальний результат.

Роль глибокого навчання (зокрема Великих Мовних Моделей) у проєкті

Саме ці переваги глибокого навчання, і особливо останнього покоління Великих Мовних Моделей (LLM), таких як Gemini API, були вирішальними при виборі підходу для нашої системи. Традиційні ML-моделі потребують ретельного виділення ознак та попередньої обробки тексту, тоді як сучасні LLM, завдяки своєму попередньому навчанню на величезних корпусах тексту, вже володіють глибоким розумінням мови, її нюансів, контексту та навіть певних знань про світ.

У нашому проєкті Gemini API використовується не як класична модель класифікації, а як інтелектуальний агент, що здатен до:

- Глибокого лінгвістичного аналізу: Виявлення стилістичних, лексичних та синтаксичних маркерів дезінформації, про які йшлося у попередньому підрозділі.
- Контекстного розуміння: Аналіз тексту не просто за словами, а з урахуванням загального змісту, тону та потенційних маніпуляцій.
- Генерації пояснень: Це є ключовою особливістю нашого підходу. Завдяки здатності LLM генерувати зв'язний текст, вона не лише ідентифікує дезінформацію, а й надає пояснення, чому саме дана новина була класифікована таким чином. Це робить систему прозорішою та допомагає користувачам краще зрозуміти природу дезінформації.

Таким чином, використання машинного навчання, а зокрема глибокого навчання через інтеграцію з LLM, дозволяє нашій системі не лише автоматизувати процес виявлення дезінформації, але й забезпечити якісний, глибинний та пояснювальний аналіз, що є значним кроком уперед у боротьбі з цим викликом сучасного інформаційного простору.

РОЗДІЛ 2. НЕЙРОННІ МЕРЕЖІ ТА ОБРОБКА ПРИРОДНОЇ МОВИ В КОНТЕКСТІ АНАЛІЗУ ТЕКСТУ

2.1. Загальні принципи функціонування нейронних мереж

Нейронні мережі, або штучні нейронні мережі (ШНМ), є центральним компонентом глибокого навчання та імітують структуру та функціонування людського мозку, щоб вивчати складні патерни в даних. Вони зарекомендували себе як потужний інструмент для вирішення широкого спектра задач, включаючи класифікацію зображень, розпізнавання мови та, що особливо важливо для нашого проєкту, обробку природної мови (NLP).

2.1.1. Базові концепції нейронних мереж

Для розуміння принципів роботи нейронних мереж необхідно ознайомитися з їхніми основними будівельними блоками: нейронами, активаційними функціями та шарами.

Нейрон (Perceptron)

В основі нейронної мережі лежить штучний нейрон, часто званий перцептроном. Це математична модель біологічного нейрона. Його функція полягає в прийомі множинних вхідних сигналів, їх обробці та генерації одного вихідного сигналу.

Кожен нейрон складається з таких елементів:

1. Входи (x1,x2,…,xn): Це дані, які надходять до нейрона. У контексті NLP, вхідними даними можуть бути числові представлення слів або фраз.

2. Ваги (w1,w2,…,wn): Кожен вхідний сигнал множиться на відповідну вагу. Ваги є параметрами моделі, які нейронна мережа "вивчає" в процесі навчання. Вони визначають важливість кожного вхідного сигналу.

3. Зміщення (b): Додається до зваженої суми входів. Зміщення дозволяє нейрону активуватися навіть тоді, коли всі вхідні сигнали дорівнюють нулю, або навпаки, не активуватися при певних вхідних сигналах.

4. Зважена сума (Weighted Sum): Це сума добутків кожного входу на його вагу, до якої додається зміщення. Математично це виглядає так:

$Z=i=1\sum n(xi\cdot wi)+b$

де xi – вхідний сигнал, wi – відповідна вага, b – зміщення, n – кількість входів.

5. Активаційна функція (f): Після обчислення зваженої суми вона передається через активаційну функцію. Ця функція додає нелінійність до моделі, дозволяючи мережі вивчати складні, нелінійні взаємозв'язки в даних.

6. Вихід (y): Результат застосування активаційної функції до зваженої суми. Це і є вихід нейрона.

$y=f(Z)$

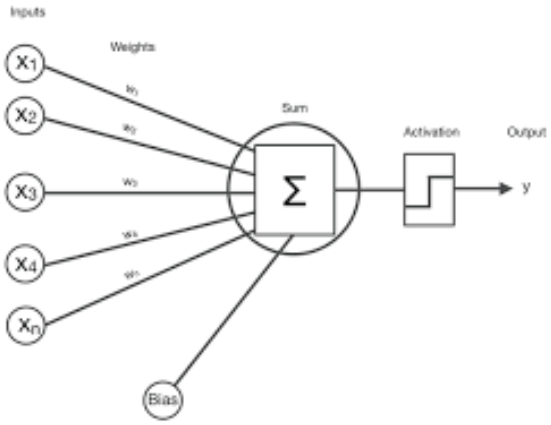

рис. 2.1.1

Активаційні функції

Активаційні функції відіграють критичну роль у нейронних мережах, оскільки вони визначають вихідний сигнал нейрона та додають нелінійність, що є фундаментально важливим для здатності мережі вирішувати складні задачі. Без нелінійних активаційних функцій, нейронна мережа, незалежно від кількості шарів, поводилася б як проста лінійна модель.

Популярні активаційні функції включають:

– Сигмоїдна функція (Sigmoid): Стискає значення в діапазон від 0 до 1. Історично була популярною, але зараз часто замінюється ReLU через

проблему "зникаючих градієнтів" для дуже великих або дуже малих значень.

- $f(z)=1+e-z1$
- Гіперболічний тангенс (Tanh): Схожа на сигмоїду, але стискає значення в діапазон від -1 до 1. Центрована навколо нуля, що може бути корисніше для навчання.
- $f(z)=ez+e-zez-e-z$
- Випрямлений лінійний елемент (ReLU - Rectified Linear Unit): Одна з найпопулярніших активаційних функцій на сьогодні. Вона повертає 0 для будь-якого від'ємного входу та сам вхід для будь-якого позитивного входу.

  $f(z)=max(0,z)$

ReLU допомагає уникнути проблеми зникаючих градієнтів і прискорює навчання. Існують також її варіанти, такі як Leaky ReLU, ELU тощо.

Шари

Нейрони в нейронній мережі організовані в шари. Кожен шар виконує певний етап обробки інформації. Розрізняють три основні типи шарів:

1. Вхідний шар (Input Layer):
   - Це перший шар мережі, який отримує сирі вхідні дані.
   - Кількість нейронів у вхідному шарі відповідає кількості ознак (фіч) у вхідних даних. Наприклад, якщо ми кодуємо слово 100-вимірним вектором, то вхідний шар матиме 100 нейронів.
   - Нейрони вхідного шару зазвичай не мають активаційних функцій, вони просто передають дані наступному шару.
2. Приховані шари (Hidden Layers):

- Розташовані між вхідним та вихідним шарами.
- Кількість прихованих шарів та нейронів у кожному з них може суттєво відрізнятися залежно від складності задачі та архітектури мережі. Мережі

з багатьма прихованими шарами називаються "глибокими" (звідси термін "глибоке навчання").

– Саме в прихованих шарах відбувається основна "магія" навчання нейронної мережі: вони вивчають складні, нелінійні представлення та абстракції вхідних даних. Кожен нейрон у прихованому шарі отримує входи від усіх нейронів попереднього шару, обробляє їх за допомогою ваг, зміщення та активаційної функції, і передає вихідні сигнали наступному шару.

3. Вихідний шар (Output Layer):

– Останній шар мережі, який генерує фінальний результат або прогноз.
– Кількість нейронів у вихідному шарі залежить від типу задачі. Наприклад, для бінарної класифікації (так/ні, дезінформація/не дезінформація) може бути один нейрон (з сигмоїдною активацією), для багатокласової класифікації – кількість нейронів, що відповідає кількості класів (з Softmax активацією).

Шляхом з'єднання нейронів у шари та налаштування ваг і зміщень у процесі навчання, нейронна мережа може вивчати складні взаємозв'язки в даних і виконувати завдання, такі як класифікація тексту, яка є основою для виявлення дезінформації в нашому проєкті.

2.1.2. Основні архітектури нейронних мереж, що використовуються для тексту

Здатність нейронних мереж обробляти послідовні дані, такі як текст, де порядок слів має вирішальне значення для розуміння значення, вимагає спеціалізованих архітектур. Традиційні полносвязні мережі не можуть ефективно захоплювати залежності між елементами в послідовності. Для вирішення цієї проблеми були розроблені рекурентні нейронні мережі (RNN) та їхні вдосконалені варіанти.

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN)

RNN – це клас нейронних мереж, спеціально розроблених для роботи з послідовними даними, такими як текст, мова або часові ряди. Їхня ключова особливість полягає в наявності петлі зворотного зв'язку, яка дозволяє інформації зберігатися і передаватися від одного кроку послідовності до наступного. Це дає їм "пам'ять" про попередні елементи в послідовності.

- Принцип роботи: Нейрон RNN не просто обробляє поточний вхід (xt), але й враховує прихований стан (ht−1) з попереднього кроку часу. Цей прихований стан є своєрідною "пам'яттю" мережі, що зберігає агреговану інформацію про попередні елементи послідовності. Формула для обчислення нового прихованого стану:
- ht=f(Whhht−1+Wxhxt+bh)
- де Whh та Wxh – матриці ваг, bh – зміщення, f – активаційна функція.
- Переваги: Здатність обробляти послідовності змінної довжини, можливість захоплення залежностей між віддаленими елементами (до певної міри).
- Недоліки (Проблема зникаючих/вибухаючих градієнтів): Основним недоліком класичних RNN є проблема "зникаючих градієнтів" (vanishing gradients) або "вибухаючих градієнтів" (exploding gradients) під час навчання на довгих послідовностях. Це ускладнює вивчення довгострокових залежностей, оскільки інформація з початку послідовності може бути втрачена до того, як вона досягне кінця.

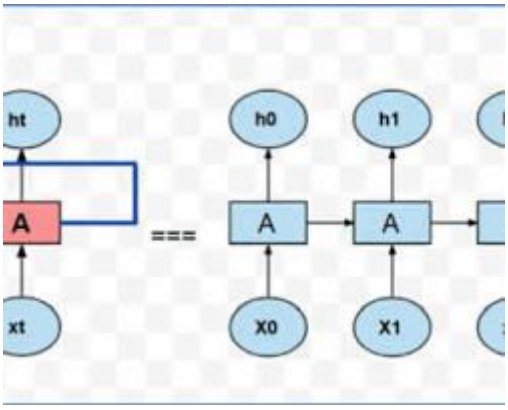

рис. 2.1.2.1

Довга короткострокова пам'ять (Long Short-Term Memory, LSTM)

LSTM – це спеціальний тип рекурентних нейронних мереж, розроблений для подолання проблеми зникаючих градієнтів та ефективного вивчення довгострокових залежностей. Вони досягають цього завдяки складнішій внутрішній структурі, що включає "ворота" (gates).

Ключові компоненти: LSTM-блок має три основні типи воріт та комірку стану:

1. Комірка стану (Cell State, Ct): Це "конвеєр пам'яті", який проходить через весь блок і дозволяє інформації вільно текти через ланцюжок.

2. Вхідні ворота (Input Gate, it): Визначають, яка нова інформація з поточного входу (xt) та попереднього прихованого стану (ht−1) буде додана до комірки стану.

3. Ворота забування (Forget Gate, ft): Визначають, яку частину інформації з попереднього комірки стану (Ct−1) слід "забути" або видалити.

4. Вихідні ворота (Output Gate, ot): Контролюють, яка частина інформації з поточного комірки стану (Ct) буде виведена як новий прихований стан (ht).

Принцип роботи: Ворота є сигмоїдними шарами, які генерують значення від 0 до 1, контролюючи потік інформації. Значення 0 означає "повністю забути", а 1 – "повністю запам'ятати". Це дозволяє LSTM вибірково додавати або видаляти інформацію з комірки стану, що є критично важливим для збереження довгострокових залежностей.

Переваги: Дуже ефективні для моделювання послідовностей, де важливий контекст, що охоплює багато кроків (наприклад, узагальнення тексту, переклад, виявлення тональності).

Недоліки: Складніша архітектура вимагає більше обчислювальних ресурсів та більшої кількості даних для навчання.

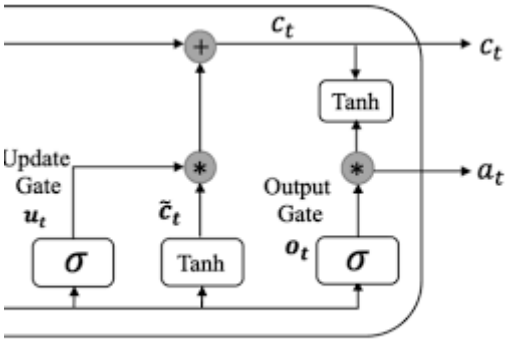

рис. 2.1.2.2

Вентильні рекурентні одиниці (Gated Recurrent Units, GRU)

GRU є спрощеною альтернативою LSTM, запропонованою у 2014 році. Вони зберігають здатність вирішувати проблему довгострокових залежностей, але мають менше параметрів, що робить їх швидшими для навчання та менш вимогливими до даних.

Ключові компоненти: GRU має лише два типи воріт:

1. Ворота оновлення (Update Gate, zt): Визначають, скільки попередньої інформації (прихованого стану) потрібно зберегти та скільки нової інформації потрібно додати. Поєднують функції вхідних воріт та воріт забування LSTM.

2. Ворота скидання (Reset Gate, rt): Визначають, наскільки попередній прихований стан є релевантним для обчислення нового кандидата на прихований стан.

Принцип роботи: GRU об'єднує комірку стану та прихований стан в один "прихований стан". Це робить їх простішими, але вони все ще ефективно контролюють потік інформації.

Переваги: Швидші для навчання, менш вимогливі до обчислювальних ресурсів, можуть бути ефективними на менших датасетах, ніж LSTM.

Недоліки: Іноді можуть поступатися LSTM на дуже складних задачах з надзвичайно довгими залежностями.

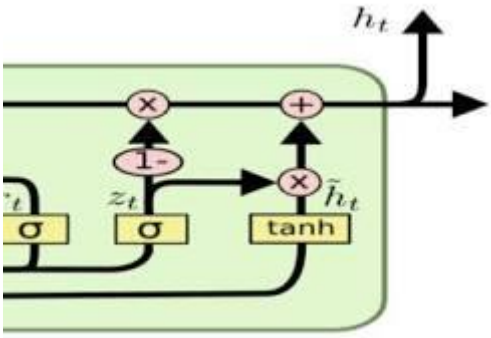

рис. 2.1.2.3

Всі ці архітектури – RNN, LSTM та GRU – відіграли фундаментальну роль у розвитку обробки природної мови, дозволивши моделям розуміти контекст та залежності у текстових даних. До появи архітектури трансформерів, яка змінила ландшафт NLP, саме ці мережі були "королями" у вивченні послідовностей і були основою для багатьох систем, що займалися аналізом тексту, включаючи ті, що спрямовані на виявлення дезінформації.

2.2. Великі мовні моделі (LLM)

2.2.1. Концепція та еволюція LLM

Великі мовні моделі (Large Language Models, LLM) є вершиною розвитку в галузі обробки природної мови (NLP) на основі глибокого навчання. Це нейронні мережі, які були навчені на величезних обсягах текстових даних (мільярди слів з книг, статей, веб-сторінок), що дозволяє їм вивчати складні лінгвістичні патерни, граматику, семантику та навіть певні знання про світ. Їхня основна мета – розуміти, генерувати та маніпулювати людською мовою.

Еволюція LLM:

Ранні моделі: Початок був покладений з моделей на основі RNN, LSTM та GRU, які навчилися обробляти послідовності. Проте їхні можливості були обмежені через труднощі з вивченням дуже довгострокових залежностей.

Word Embeddings (2013-2015): Поява Word2Vec, GloVe та FastText дозволила представляти слова у вигляді щільних векторів, що захоплюють

семантичні зв'язки. Це був важливий крок, але вони не враховували контекст слова в реченні.

Контекстуальні Embedding'и (2018): Розвиток таких моделей як ELMo та BERT (Bidirectional Encoder Representations from Transformers) змінив парадигму. Ці моделі навчилися генерувати векторні представлення слів, які змінюються залежно від контексту, в якому слово використовується.

Архітектура Трансформерів (2017): Це був справжній прорив. Вперше представлений у статті "Attention Is All You Need", Трансформер повністю відмовився від рекурентних механізмів, покладаючись виключно на механізм уваги (Attention Mechanism). Це дозволило моделям обробляти всі слова в послідовності паралельно, що значно прискорило навчання та дозволило ефективно працювати з дуже довгими текстами.

Попереднє навчання та тонке налаштування (Pre-training and Fine-tuning): Сучасні LLM навчаються у два етапи:

1. Попереднє навчання (Pre-training): Модель навчається на величезному, нерозміченому текстовому корпусі для виконання загальних мовних завдань (наприклад, передбачення наступного слова або заповнення пропущених слів).

2. Тонке налаштування (Fine-tuning): Попередньо навчена модель потім адаптується для виконання конкретних задач (класифікація тексту, генерація, переклад) на менших, розмічених датасетах.

Зростання розміру моделей: З часом LLM ставали все більшими за кількістю параметрів (мільярди), що дало їм безпрецедентні можливості розуміння та генерації мови, проявляючи так звані "емерджентні здібності" (emergent abilities) – поведінку, яка не була явно запрограмована.

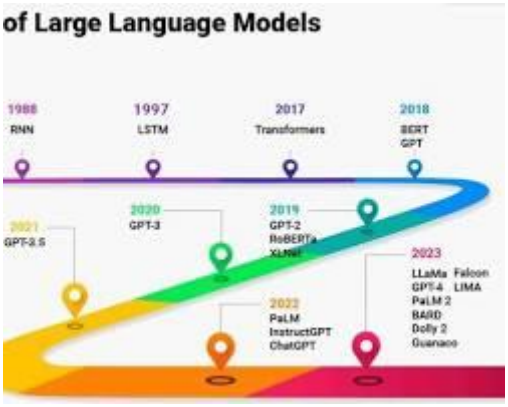

рис. 2.2.1.

2.2.2. Архітектура трансформерів та механізм уваги (Attention Mechanism) як основа сучасних LLM

Трансформерна архітектура є фундаментальним компонентом сучасних LLM, включаючи Gemini. На відміну від RNN, які обробляють послідовності крок за кроком, Трансформер може обробляти всю послідовність одночасно, що значно прискорює навчання та дозволяє захоплювати довгострокові залежності.

Ключові особливості Трансформера:

Механізм уваги (Attention Mechanism): Це серце Трансформера. Він дозволяє моделі динамічно "зважувати" важливість різних частин вхідної послідовності при обробці кожного слова. Наприклад, коли модель обробляє слово "його" в реченні "Собака гавкала, і його власник підійшов.", механізм уваги допомагає їй зрозуміти, що "його" стосується "власника", а не "собаки". Існують різні види уваги, зокрема мультиголосна увага (Multi-Head Attention), яка дозволяє моделі одночасно зосереджуватися на різних аспектах зв'язків у послідовності.

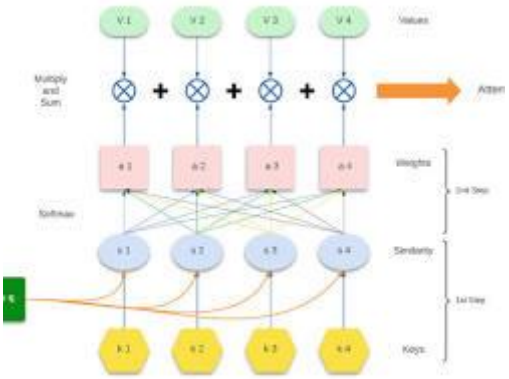

рис. 2.2.2.1

Кодер-Декодер Архітектура: Класичний Трансформер складається з двох основних частин:

– Кодер (Encoder): Обробляє вхідну послідовність, перетворюючи її на внутрішнє представлення, що містить контекстну інформацію. Кодер складається з кількох ідентичних шарів, кожен з яких містить механізм мультиголосної уваги та полносвязний шар.
– Декодер (Decoder): Використовує вихід кодера для генерації вихідної послідовності (наприклад, перекладеного тексту, резюме). Декодер також складається з ідентичних шарів, кожен з яких містить механізм мультиголосної уваги (який може враховувати як вхідний контекст, так і вже згенеровану частину виходу) та полносвязний шар.

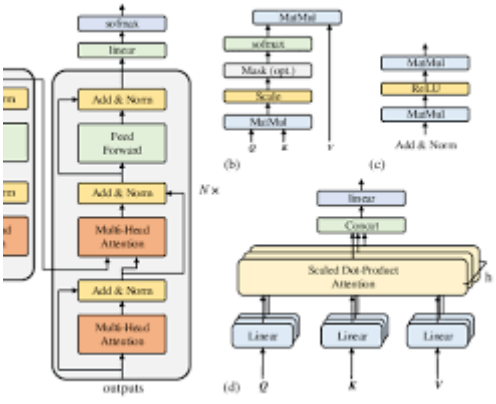

рис. 2.2.2.2

1. Позиційне кодування (Positional Encoding): Оскільки Трансформер не має рекурентної природи, він не може автоматично "знати" про порядок слів у послідовності. Для вирішення цієї проблеми до векторних представлень слів

додається позиційне кодування, яке надає інформацію про відносне або абсолютне положення кожного слова в реченні.

2. Полнозв'язні шари та нормалізація: кожен шар Трансформера також включає полносвязні шари та шари нормалізації (Layer Normalization), що допомагають стабілізувати навчання.

Gemini API як Трансформерна LLM:

Gemini – це сімейство мультимодальних LLM, розроблених Google AI, які базуються на архітектурі Трансформера. Мультимодальність означає, що Gemini може не лише обробляти текст, а й розуміти та генерувати інформацію в різних модальностях (зображення, аудіо, відео).

У контексті вашого проєкту, коли ви надсилаєте запит до Gemini API з текстом новини, ви звертаєтеся до величезної, попередньо навченої трансформерної моделі. Ця модель використовує свій глибокий лінгвістичний аналіз та механізм уваги, щоб:

- Визначити контекст і значення кожного слова в новині.
- Ідентифікувати потенційні лінгвістичні маркери дезінформації, як-от емоційний тон, використання певних фраз, узагальненість тверджень.
- Згенерувати обґрунтовану відповідь, яка включає оцінку наявності дезінформації та її пояснення, ґрунтуючись на своєму розумінні тексту.

Використання Gemini API дозволяє вам отримати доступ до передових можливостей LLM без необхідності самостійно розробляти, навчати та підтримувати таку складну архітектуру, що значно прискорює реалізацію вашого проєкту. Це є стратегічним рішенням, яке дозволяє зосередитися на задачі виявлення дезінформації, а не на базовій інфраструктурі LLM.

2.2.3. Методи попереднього навчання (pre-training) та тонкого налаштування (fine-tuning) LLM

Розробка та ефективне застосування великих мовних моделей (LLM) значною мірою ґрунтуються на двох ключових методологіях: попередньому

навчанні (pre-training) та тонкому налаштуванню (fine-tuning). Ці підходи дозволяють створювати універсальні, потужні моделі, які потім можуть бути адаптовані для виконання специфічних завдань з високою точністю.

2.2.3.1. Попереднє навчання (Pre-training)

Попереднє навчання – це перший і найважливіший етап у життєвому циклі LLM. На цьому етапі модель навчається на величезному, нерозміченому корпусі текстових даних. Обсяги цих даних можуть сягати терабайтів і включати книги, статті з Вікіпедії, веб-сторінки, новини, наукові публікації тощо. Метою попереднього навчання є надання моделі глибокого розуміння мови: її граматики, синтаксису, семантики, а також загальних знань про світ.

Основні завдання під час попереднього навчання:

- Моделювання мови (Language Modeling): Найпоширенішим завданням є передбачення наступного слова у послідовності (causal language modeling) або заповнення пропущених слів у реченні (masked language modeling). Наприклад, моделі BERT навчаються передбачати масковані слова у двонаправленому контексті. Це змушує модель вивчати залежності між словами та розуміти контекст.
- Навчання на парах речень (Next Sentence Prediction): Деякі моделі також навчаються визначати, чи є два речення послідовними у вихідному тексті. Це допомагає моделі вивчати взаємозв'язки між реченнями та розуміти логіку тексту.

Результат попереднього навчання: Після попереднього навчання LLM стає "експертом" у розумінні та генерації мови. Вона навчена розпізнавати шаблони, вивчати словниковий запас, граматичні структури, а також накопичує знання про різні предмети. Важливо, що на цьому етапі модель є універсальною і не налаштована на жодне конкретне завдання (наприклад, класифікацію сентименту чи переклад).

2.2.3.2. Тонке налаштування (Fine-tuning)

Тонке налаштування – це другий етап, на якому попередньо навчена LLM адаптується для виконання специфічного завдання. На відміну від попереднього навчання, цей етап використовує значно менші, але спеціально розмічені датасети, характерні для цільової задачі.

Принцип: до верхніх шарів попередньо навченої моделі додаються один або кілька нових шарів (наприклад, шар класифікації). Потім вся модель (або її частина) навчається на розмічених даних для нової задачі. Оскільки модель вже має глибоке розуміння мови, вона потребує значно менше даних для тонкого налаштування, ніж для навчання з нуля, і сходиться набагато швидше.

Завдання тонкого налаштування:

- Класифікація тексту: визначення категорії тексту (наприклад, спам/не спам, позитивний/негативний відгук, дезінформація/не дезінформація).
- Відповіді на запитання: генерація відповідей на питання на основі наданого тексту.
- Узагальнення тексту: створення коротких витягів з довгих текстів.
- Машинний переклад: переклад тексту з однієї мови на іншу.

Переваги підходу "Попереднє навчання + Тонке налаштування":

- Ефективність: Значно знижується потреба у великих обсягах розмічених даних для кожної нової задачі.
- Вища продуктивність: Моделі, навчені таким чином, демонструють значно кращі результати порівняно з моделями, навченими з нуля на менших датасетах.
- Трансферне навчання (Transfer Learning): Знання, отримані під час попереднього навчання на загальних мовних завданнях, "передаються" до специфічних завдань, що є основою для ефективності LLM.

Застосування у нашому проєкті (Використання Gemini API):

У нашому проєкті ми використовуємо попередньо навчену та тонко налаштовану модель Gemini через її API. Ми не виконуємо процеси попереднього навчання чи тонкого налаштування самостійно. Натомість, ми:

1. Використовуємо готову потужність: Gemini API надає доступ до моделі, яка вже пройшла етапи попереднього навчання на колосальних обсягах даних і була внутрішньо тонко налаштована Google для широкого спектру задач, включаючи розуміння та генерацію тексту.

2. Застосовуємо інженерію промптів (Prompt Engineering): Замість безпосереднього тонкого налаштування моделі, ми ефективно "налаштовуємо" її поведінку та вихідний формат за допомогою детально сформульованого промпту. Наш промпт, який визначає роль LLM як "експерта з медіааналітики", є формою "нуль-шотового" або "кілька-шотового" навчання (zero-shot/few-shot learning), де модель виконує завдання без додаткового навчання, спираючись лише на інструкції в промпті та свої попередньо набуті знання.

Це дозволяє нам зосередитися на логіці нашої системи, інтеграції та зборі даних, використовуючи найсучасніші досягнення в галузі LLM без необхідності глибокого занурення в процеси навчання нейронних мереж.

2.2.4. Застосування LLM для задач обробки природної мови

Великі мовні моделі (LLM) завдяки своїй здатності розуміти, інтерпретувати та генерувати людську мову, знайшли широке застосування в численних задачах обробки природної мови (NLP). Їхні можливості значно перевершують попередні моделі, дозволяючи вирішувати складні проблеми, які раніше вимагали спеціалізованих архітектур для кожного конкретного завдання.

Основні напрямки застосування LLM:

1. Класифікація тексту (Text Classification): Це одна з найбільш фундаментальних задач NLP, де текст категоризується за певними ознаками. LLM можуть класифікувати текст на різні категорії, такі як:

Класифікація тональності (Sentiment Analysis): визначення емоційного забарвлення тексту (позитивний, негативний, нейтральний).

Визначення спаму/фішингу: ідентифікація небажаних або шкідливих повідомлень.

Категоризація новин: присвоєння новинам тематичних категорій (спорт, політика, економіка).

Виявлення дезінформації: класифікація новин як "правдивих" або "дезінформаційних", що є центральним завданням нашого проєкту. LLM здатні розрізняти тонкі лінгвістичні патерни, які вказують на маніпулятивний характер тексту, посилаючись на внутрішні знання, отримані під час попереднього навчання.

2. Генерація тексту (Text Generation): LLM можуть створювати новий, зв'язний та осмислений текст на основі заданого входу або контексту. Це включає:

- Відповіді на запитання (Question Answering): Генерація точних відповідей на запитання на основі наданого тексту або загальних знань.
- Створення статей/рефератів: Автоматичне написання новинних статей, рефератів або навіть творів.
- Переклад тексту (Machine Translation): Переклад з однієї мови на іншу зі збереженням сенсу та стилю.
- Створення діалогів (Chatbots): Ведення природних розмов з користувачами, імітуючи людське спілкування.
- Генерація пояснень: У нашому проєкті, LLM генерує не тільки вердикт про дезінформацію, а й пояснення, чому саме такий вердикт був винесений, що є формою генерації тексту, що підвищує прозорість системи.

3. Розуміння контексту та вилучення інформації (Context Understanding and Information Extraction): LLM володіють глибоким розумінням мовного контексту, що дозволяє їм:

- Вилучення іменованих сутностей (Named Entity Recognition, NER): Ідентифікація та класифікація сутностей у тексті (імена людей, організації, локації, дати).
- Визначення взаємозв'язків (Relation Extraction): Виявлення зв'язків між сутностями.
- Підсумовування тексту (Text Summarization): Створення коротких, але змістовних резюме довгих документів.
- Семантичний пошук: Пошук інформації, яка відповідає значенню запиту, а не лише ключовим словам.
- Виявлення маніпуляцій: У контексті нашого проєкту, це здатність LLM "прочитати між рядками", виявити приховані наміри, упередженість або логічні помилки в тексті, які можуть вказувати на дезінформацію.

LLM у нашому проєкті виявлення дезінформації:

У нашій системі LLM (Gemini API) використовується як багатофункціональний інструмент, що поєднує елементи цих завдань для комплексного аналізу новинного контенту. Зокрема:

- Класифікація: Модель фактично виконує задачу класифікації, оцінюючи, чи є новина дезінформаційною.
- Розуміння контексту: Вона глибоко аналізує повний текст статті, розуміючи її контекст та виявляючи будь-які невідповідності або маніпуляції.
- Генерація: Найважливіше, LLM генерує структурований аналіз, який включає не тільки вердикт, але й розгорнуте пояснення, чому саме даний новинний матеріал був ідентифікований як потенційна дезінформація або навпаки, як достовірна інформація. Це дозволяє користувачам не просто отримати результат, а й зрозуміти логіку системи, підвищуючи довіру та освітню цінність.

Таким чином, застосування LLM дозволяє нашій системі вийти за рамки простої бінарної класифікації, надаючи більш глибокий, прозорий та корисний аналіз для боротьби з дезінформацією.

2.3. Методи представлення тексту для нейронних мереж

Для того щоб нейронні мережі могли обробляти текстові дані, ці дані мають бути перетворені у числовий формат. Цей процес, що називається представленням тексту (text representation), є критично важливим етапом у пайплайні обробки природної мови, оскільки якість представлення безпосередньо впливає на продуктивність моделі. Незалежно від того, чи використовується класична модель машинного навчання, чи сучасна велика мовна модель, текст спочатку проходить через низку перетворень.

2.3.1. Токенізація та нормалізація тексту

Першими кроками у перетворенні сирого текстового корпусу в формат, придатний для обробки комп'ютерними алгоритмами, є токенізація та нормалізація. Ці процеси закладають основу для подальшого аналізу та вилучення ознак.

2.3.1.1. Токенізація (Tokenization)

Токенізація – це процес розділення послідовності символів (тексту) на менші, значущі одиниці, які називаються токенами. Токени можуть бути словами, підсловами, символами або навіть абзацами, залежно від мети аналізу та обраного підходу. Це перший крок у структуруванні неструктурованого тексту.

Типи токенізації:

- Токенізація за словами (Word Tokenization): найпоширеніший підхід, де текст розділяється на окремі слова. Роздільниками зазвичай є пробіли та знаки пунктуації. Наприклад, речення "Система виявлення

дезінформації." буде розділено на токени ["Система", "виявлення", "дезінформації"].

- Токенізація за реченнями (Sentence Tokenization): розділення тексту на окремі речення, що важливо для завдань, які вимагають аналізу на рівні речень.
- Токенізація за підсловами (Subword Tokenization): сучасні підходи, такі як WordPiece (використовується в BERT) або BPE (Byte Pair Encoding), розділяють слова на менші частини – підслова. Це дозволяє ефективно працювати з рідкісними словами, опечатками та словами, яких немає у словнику (Out-of-Vocabulary, OOV), а також зменшити розмір словника моделі. Наприклад, слово "running" може бути розділене на "run" і "##ning".
- Токенізація за символами (Character Tokenization): розділення тексту на окремі символи. Використовується рідше, але може бути корисним для деяких завдань, особливо в мовах без чітких роздільників між словами.

Важливість токенізації для нашого проєкту: хоча LLM, такі як Gemini, мають власні вбудовані, складні токенізатори (часто підсловові), розуміння цього процесу є фундаментальним. Це пояснює, як сирий текст новини, що надходить до системи, перетворюється на дискретні одиниці, які потім можуть бути представлені у числовому вигляді для обробки нейронною мережею. Кожен токен потім буде перетворений на векторне представлення (embedding).

2.3.1.2. Нормалізація тексту (Text Normalization)

Нормалізація – це процес приведення тексту до стандартизованої форми з метою зменшення варіативності та підвищення консистентності даних. Це допомагає моделі розпізнавати одні й ті ж слова, незважаючи на їхнє написання або форму.

Основні техніки нормалізації:

- Переведення до нижнього регістру (Lowercasing): Всі літери тексту перетворюються на малі, щоб уникнути розрізнення між "Слово", "слово" та "СЛОВО".
- Видалення пунктуації (Punctuation Removal): Видалення всіх знаків пунктуації, якщо вони не несуть важливого семантичного значення для конкретного завдання.
- Видалення стоп-слів (Stop Words Removal): Видалення загальновживаних слів (наприклад, "і", "в", "на", "що", "але"), які часто не несуть значного смислового навантаження для багатьох NLP-задач. Це зменшує розмірність даних та фокусує аналіз на більш інформативних словах.
- Стемінг (Stemming): Зведення слів до їхньої основи (кореня) шляхом відкидання суфіксів та префіксів. Наприклад, "біг", "бігати", "біжучий" можуть бути зведені до основи "біг". Метод є швидким, але може бути агресивним і створювати некоректні основи.
- Лемматизація (Lemmatization): Більш складна техніка, яка зводить слова до їхньої нормальної словникової форми (лемми), враховуючи морфологічний аналіз. Наприклад, "були", "є", "буде" зводяться до лемми "бути". Лемматизація є точнішою за стемінг.
- Обробка чисел та спеціальних символів: Видалення або заміна числових значень та інших символів, які можуть не бути релевантними для аналізу.

Важливість нормалізації для нашого проєкту: Хоча сучасні LLM, як Gemini, навчені на величезних ненормалізованих даних і можуть самостійно справлятися з варіативністю мови, розуміння нормалізації є важливим для попереднього аналізу даних та розуміння, як моделі справляються з різними формами слів. Для деяких специфічних задач або для традиційних ML-моделей цей етап є критично важливим для підвищення продуктивності. У нашому випадку, ми покладаємося на внутрішні можливості Gemini API для ефективної обробки тексту, що дозволяє нам не виконувати ці кроки вручну.

2.3.2. Векторні представлення слів (Word Embeddings: Word2Vec, GloVe) – коротко, як історичний контекст

Після токенізації та нормалізації, наступним кроком для обробки тексту нейронними мережами є його перетворення у числовий формат. Традиційно, слова представлялися за допомогою розріджених векторів (наприклад, one-hot кодування), де кожне слово відповідало окремому виміру. Проте такий підхід мав значні недоліки: високу розмірність, відсутність інформації про семантичні зв'язки між словами та проблему "прокляття розмірності".

З появою векторних представлень слів (Word Embeddings), які також називають вбудовуваннями слів, стався значний прорив у NLP. Це щільні, низьковимірні вектори чисел з плаваючою комою, які захоплюють семантичні та синтаксичні взаємозв'язки між словами. Ідея полягає в тому, що слова, які мають схоже значення або використовуються в схожих контекстах, будуть розташовані близько один до одного у цьому багатовимірному векторному просторі.

Ключові концепції Word Embeddings:

- Принцип розподільчої семантики (Distributional Hypothesis): Цей принцип стверджує, що слова, які зустрічаються в подібних контекстах, як правило, мають схоже значення. Моделі Word Embeddings використовують цю гіпотезу для навчання векторних представлень.
- Щільні вектори: На відміну від розріджених векторів, де більшість елементів дорівнюють нулю, щільні вектори містять значущі числа в кожному вимірі. Це дозволяє ефективніше використовувати обчислювальні ресурси та захоплювати складніші залежності.

Word2Vec (2013)

Word2Vec – це група моделей, розроблених у Google, які стали революційними у створенні ефективних векторних представлень слів. Існують дві основні архітектури Word2Vec:

1. Continuous Bag-of-Words (CBOW): Модель передбачає цільове слово на основі його контексту (навколишніх слів). Наприклад, у реченні

"собака швидко [___] за кішкою", модель намагатиметься передбачити слово "біжить" на основі слів "собака", "швидко", "за", "кішкою".

2. Skip-Gram: На відміну від CBOW, Skip-Gram передбачає контекстні слова на основі цільового слова. Тобто, якщо є слово "король", модель намагатиметься передбачити такі слова як "королева", "трон", "правлячий".

Переваги Word2Vec:

- Можливість захоплення семантичних та синтаксичних відносин між словами (наприклад, вектор "король" - вектор "чоловік" + вектор "жінка" ≈ вектор "королева").
- Порівняно швидке навчання на великих корпусах.
- GloVe (Global Vectors for Word Representation, 2014)
- GloVe – це ще одна популярна модель для отримання Word Embeddings, розроблена Стенфордським університетом. Вона поєднує в собі переваги як методів, заснованих на вікнах контексту (як Word2Vec), так і методів, заснованих на матрицях спільної появи слів (які аналізують статистику співзустрічності слів у всьому корпусі).
- Принцип роботи: GloVe навчається на основі глобальної статистики спільної появи слів у корпусі. Модель мінімізує функцію втрат, яка заохочує, щоб добуток векторів двох слів був пропорційним логарифму їхньої частоти спільної появи.

Переваги GloVe:

- Ефективне використання глобальної статистики корпусу, що дозволяє отримувати більш точні представлення для рідкісних слів.
- Часто демонструє кращі результати на певних задачах порівняно з Word2Vec.

Історичний контекст та значення для нашого проєкту

Word2Vec та GloVe стали фундаментальними розробками в NLP, показавши, що слова можна ефективно представляти у вигляді щільних векторів, які несуть у собі значущу семантичну інформацію. Вони відкрили

шлях для використання нейронних мереж у широкому спектрі мовних завдань, оскільки надавали нейронним мережам "зрозумілий" числовий вхід, що відображає значення слів.

Проте, незважаючи на їхнє значення, ці моделі мали одне суттєве обмеження: вони генерували статичні векторні представлення слів. Це означає, що одне й те саме слово завжди мало один і той же вектор, незалежно від контексту, в якому воно використовувалося. Наприклад, слово "банк" у "фінансовий банк" та "берег річки" мало б однаковий вектор, що обмежувало можливості моделей у розумінні полісемії.

Ця проблема була вирішена з появою контекстуальних вбудовувань, які є основою сучасних Великих Мовних Моделей, таких як Gemini, що використовується в нашому проєкті. Саме про них ми поговоримо у наступному підрозділі, підкреслюючи їхню еволюційну перевагу над статичними Word Embeddings.

2.3.3. Контекстуальні вбудовування (Contextual Embeddings: BERT, RoBERTa, Gemini embeddings) – як сучасний підхід

На відміну від статичних векторних представлень слів (Word Embeddings), таких як Word2Vec та GloVe, які присвоюють одне й те саме представлення слову незалежно від його контексту, контекстуальні вбудовування (Contextual Embeddings) революціонізували обробку природної мови. Вони дозволяють генерувати унікальні векторні представлення для кожного слова, враховуючи його оточення в конкретному реченні або документі. Це вирішує проблему полісемії (багатозначності слів) та дозволяє моделям краще розуміти тонкощі мови.

Ключові особливості контекстуальних вбудовувань:

- Динамічне представлення: Вектор слова змінюється залежно від контексту, в якому воно використовується. Наприклад, слово "банк" у "фінансовий банк" матиме інше векторне представлення, ніж у "берег річки".

- Захоплення багатозначності: Моделі, що використовують контекстуальні вбудовування, можуть розрізняти різні значення одного слова.
- Вивчення довгострокових залежностей: Завдяки архітектурам, на яких вони базуються (переважно Трансформери), ці вбудовування ефективно захоплюють залежності між словами, що знаходяться далеко одне від одного у тексті.

BERT (Bidirectional Encoder Representations from Transformers, 2018)

BERT (розроблений Google) став одним із найвпливовіших проривів у контекстуальних вбудовуваннях. Це попередня навчена модель на основі архітектури Трансформера, яка генерує двонаправлені контекстуальні вбудовування.

Двонаправленість: На відміну від попередніх моделей, BERT аналізує слово, враховуючи як попередні, так і наступні слова в реченні. Це дозволяє моделі формувати більш повне розуміння контексту.

Завдання попереднього навчання: BERT навчається на двох основних завданнях:

1. Masked Language Model (MLM): Модель маскує (приховує) частину слів у реченні, а потім намагається їх передбачити, ґрунтуючись на словах навколо. Це змушує її розуміти глибокий контекст кожного слова.

2. Next Sentence Prediction (NSP): Модель навчається передбачати, чи є два речення послідовними у вихідному документі, що допомагає розуміти взаємозв'язки між реченнями.

Результат: BERT генерує векторне представлення (embedding) для кожного слова в реченні, яке є унікальним для цього конкретного контексту.

RoBERTa (A Robustly Optimized BERT Pretraining Approach, 2019)

RoBERTa (розроблена Facebook AI) є оптимізованою версією BERT. Її основна ідея полягала в тому, щоб показати, що кращі результати можна отримати, оптимізувавши процес попереднього навчання BERT, а не змінюючи його архітектуру.

Оптимізації: RoBERTa навчалася на більшому обсязі даних, довше, з більшими розмірами міні-пакетів та без завдання NSP. Це дозволило їй досягти ще кращих результатів на багатьох задачах NLP, демонструючи важливість ресурсів та ретельної оптимізації навчання.

Gemini Embeddings

Gemini, як провідна велика мовна модель від Google AI, також використовує та генерує контекстуальні вбудовування. Це є невід'ємною частиною її здатності розуміти та обробляти мову.

Сучасний підхід: вбудовування, що використовуються в Gemini, є результатом її передової трансформерної архітектури та масштабного попереднього навчання на мультимодальних даних (якщо модель є мультимодальною). Це дозволяє їй створювати дуже якісні та глибокі представлення тексту.

Внутрішнє представлення: коли ми надсилаємо текст новини до Gemini API, модель внутрішньо перетворює цей текст на власні контекстуальні вбудовування. Саме ці вбудовування дозволяють їй виконувати складний аналіз:

- Виявляти семантичні зв'язки між словами та реченнями.
- Розпізнавати тонкощі мови, які вказують на дезінформацію (наприклад, емоційність, упередженість, логічні суперечності).
- Зіставляти інформацію з її величезною базою знань.

Значення для нашого проєкту:

Використання LLM, що базуються на контекстуальних вбудовуваннях (як Gemini API у нашому проєкті), є стратегічно важливим. Це дозволяє:

1. Глибоке розуміння тексту: Модель може інтерпретувати текст новини з урахуванням повного контексту, що є критично важливим для виявлення складних форм дезінформації, які неможливо ідентифікувати за допомогою статичних представлень слів.

2. Висока точність: Завдяки вбудованим знанням та здатності розуміти нюанси мови, LLM, що використовують контекстуальні вбудовування, забезпечують вищу точність у класифікації та аналізі дезінформації.

3. Ефективність інженерії промптів: Нам не потрібно вручну створювати ознаки або навчати складні моделі для генерації цих вбудовувань. Натомість, ми використовуємо потужність попередньо навченої LLM, фокусуючись на ефективному формулюванні промптів, які спрямовують її аналітичні здібності на задачу виявлення дезінформації.

Таким чином, контекстуальні вбудовування є основою, яка дозволяє великим мовним моделям бути настільки ефективними в задачах NLP, включаючи наш проєкт з виявлення дезінформації.

РОЗДІЛ 3. АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ВИЯВЛЕННЯ ДЕЗІНФОРМАЦІЇ НА ОСНОВІ МАШИННОГО НАВЧАННЯ

3.1. Огляд наукових досліджень та проектів у сфері виявлення дезінформації

У зв'язку зі стрімким зростанням обсягів дезінформації в цифровому просторі, академічна та промислова спільноти приділяють значну увагу розробці автоматизованих систем для її виявлення. Ці системи використовують різноманітні підходи, що ґрунтуються на машинному навчанні та глибокому навчанні, кожен з яких має свої переваги та обмеження.

3.1.1. Системи на основі традиційних методів машинного навчання (наприклад, SVM, Naive Bayes, Random Forest) та їх застосування

Традиційні методи машинного навчання стали одними з перших, що були застосовані для автоматичного виявлення дезінформації. Ці алгоритми вимагають попереднього вилучення ознак (feature engineering) з тексту та інших даних, які потім подаються на вхід класифікатору.

Методи опорних векторів (Support Vector Machines, SVM): SVM – це потужні алгоритми класифікації, які знаходять оптимальну гіперплощину, що розділяє різні класи даних у багатовимірному просторі. У контексті виявлення дезінформації, SVM можуть бути навчені на ознаках, що відображають лінгвістичні особливості (наприклад, частота використання певних слів, наявність емоційно забарвленої лексики) або метадані (наприклад, джерело публікації). Їх ефективність полягає у здатності добре працювати з високовимірними, але розрідженими даними, що характерно для текстових ознак.

Байєсівські класифікатори (Naive Bayes): Ці класифікатори ґрунтуються на теоремі Баєса та припущенні про умовну незалежність ознак. Для текстової класифікації, Naive Bayes (наприклад, Multinomial Naive Bayes) є простим, але

ефективним методом. Він обчислює ймовірність належності новини до класу "дезінформація" або "правда" на основі частоти появи слів у цих класах. Незважаючи на "наївне" припущення про незалежність, цей метод часто демонструє добрі результати на задачах класифікації тексту, особливо коли є достатньо навчальних даних.

Випадкові ліси (Random Forest): Це ансамблевий метод, який будує велику кількість дерев рішень і агрегує їхні прогнози для отримання остаточного рішення. Random Forest здатен обробляти велику кількість ознак, є менш схильним до перенавчання порівняно з одним деревом рішень та може виявляти складні, нелінійні взаємозв'язки між ознаками. Ознаками для Random Forest можуть бути як статистичні характеристики тексту (наприклад, TF-IDF ваги слів), так і більш складні лінгвістичні показники.

Застосування та обмеження традиційних методів

Традиційні методи машинного навчання широко застосовувалися на початкових етапах розвитку систем виявлення дезінформації. Вони відносно прості в реалізації та вимагають менших обчислювальних ресурсів порівняно з моделями глибокого навчання. Проте, їхній основний недолік полягає в необхідності ручного або автоматизованого вилучення ознак. Якість класифікації сильно залежить від того, наскільки добре ці ознаки відображають сутність дезінформації. Це може бути складно, оскільки дезінформація постійно еволюціонує, і вручну виділені ознаки швидко застарівають. Крім того, ці моделі менш ефективні у захопленні складних семантичних та контекстуальних залежностей у тексті, що є критичним для тонкого розуміння маніпуляцій.

3.1.2. Системи на основі глибокого навчання (Deep Learning) – приклади використання RNN, CNN, Transformer-базованих моделей

З появою глибокого навчання та збільшенням доступності обчислювальних потужностей, дослідники почали активно застосовувати нейронні мережі для вирішення задач виявлення дезінформації. Глибоке

навчання дозволяє моделям автоматично вивчати складні та ієрархічні ознаки без необхідності ручного вилучення, що є значною перевагою для роботи з необробленим текстовим даними.

Рекурентні нейронні мережі (RNN, LSTM, GRU): Як згадувалося у Розділі 2, RNN, а особливо їхні вдосконалені варіанти LSTM та GRU, були одними з перших архітектур глибокого навчання, застосованих для аналізу послідовностей тексту. Вони здатні захоплювати довгострокові залежності у реченнях, що дозволяє їм розуміти контекст. У контексті виявлення дезінформації, LSTM/GRU моделі могли б аналізувати послідовність слів у новині, виявляючи аномальні мовні патерни, що вказують на фейк або маніпуляцію. Наприклад, вони могли б розпізнавати незвичайні синтаксичні конструкції або нелогічні послідовності подій.

Згорткові нейронні мережі (Convolutional Neural Networks, CNN): Хоча CNN спочатку були розроблені для обробки зображень, вони також знайшли застосування в NLP, особливо для задач класифікації тексту. У застосуванні до тексту, CNN використовують "фільтри" (ядра згортки), які "ковзають" по словам або їхнім n-грамам (послідовностям з n слів), виявляючи локальні патерни та ознаки. Наприклад, CNN можуть бути ефективними для розпізнавання ключових фраз, емоційно забарвлених виразів або специфічної лексики, яка характерна для дезінформації.

Transformer-базовані моделі (BERT, RoBERTa, GPT, T5, Gemini): Це найсучасніший і найефективніший підхід. Архітектура Трансформера, завдяки механізму уваги (Attention Mechanism), дозволяє моделям одночасно аналізувати всі слова в тексті, захоплюючи складні та довгострокові залежності між ними.

Моделі, такі як BERT та RoBERTa, попередньо навчаються на величезних корпусах тексту і можуть бути тонко налаштовані для задач класифікації, включаючи виявлення дезінформації. Вони виявляють високу точність, розуміючи не тільки слова, але й їхній контекст та семантичні нюанси.

Генеративні Трансформери (наприклад, серія GPT, T5, Gemini) є ще більш потужними. Вони не лише розуміють текст, а й можуть генерувати зв'язний, осмислений контент. Це відкриває нові можливості для виявлення дезінформації, оскільки такі моделі можуть не просто класифікувати текст, а й:

- Розпізнавати фактичні невідповідності: Порівнювати подану інформацію зі своїми внутрішніми знаннями, отриманими під час попереднього навчання.
- Виявляти логічні суперечності: Ідентифікувати несумісні твердження в межах одного тексту.
- Пояснювати свій вердикт: Генерувати пояснення, чому певний текст вважається дезінформацією, що є критично важливим для прозорості системи та підвищення довіри користувачів. Саме ця здатність LLM, як Gemini, є центральною у нашому проєкті, оскільки дозволяє не лише класифікувати, а й надати обґрунтований аналіз.

Порівняно з традиційними методами, глибоке навчання, особливо з використанням Transformer-базованих моделей, демонструє значно вищу продуктивність у виявленні дезінформації завдяки своїй здатності автоматично вивчати складні лінгвістичні патерни та розуміти контекст.

3.1.3. Приклади комерційних або відкритих платформ, що займаються фактчекінгом або виявленням дезінформації

На додаток до академічних досліджень, існує низка комерційних та відкритих платформ, які активно займаються фактчекінгом та розробкою інструментів для виявлення дезінформації. Їхня діяльність часто поєднує ручну експертизу з автоматизованими інструментами, забезпечуючи ширше покриття та оперативність.

StopFake (Україна): одна з провідних українських фактчекінгових організацій, заснована у 2014 році. Вона спеціалізується на викритті фейків та пропаганди, переважно пов'язаних з російсько-українською війною. StopFake

активно співпрацює з міжнародними організаціями та публікує свої розслідування українською, англійською та іншими мовами. Хоча їхня основна діяльність базується на ручному фактчекінгу, вони також використовують автоматизовані інструменти для моніторингу та виявлення підозрілих повідомлень.

VoxCheck (Україна): інший відомий український проєкт, що займається верифікацією фактів, заяв політиків та експертів. VoxCheck використовує методологію, що поєднує журналістське розслідування та аналіз даних, часто з використанням інструментів обробки даних для виявлення маніпуляцій зі статистикою та цифрами.

Snopes (США): один з найстаріших та найвідоміших фактчекінгових сайтів у світі. Snopes фокусується на викритті міфів, міських легенд, інтернет-чуток та фейкових новин. Вони використовують команду експертів для ретельної перевірки інформації.

PolitiFact (США): цей проєкт, що отримав Пулітцерівську премію, спеціалізується на перевірці правдивості заяв американських політиків. Вони розробили систему "Truth-O-Meter", яка оцінює правдивість заяв за шкалою від "Правда" до "Пантаси", що дозволяє швидко оцінити достовірність інформації.

Full Fact (Велика Британія): незалежна фактчекінгова організація, яка використовує як людський аналіз, так і автоматизовані технології для перевірки фактів у публічному дискурсі. Вони активно досліджують застосування ШІ для масштабування своєї роботи.

International Fact-Checking Network (IFCN): глобальна мережа, що об'єднує понад 100 фактчекінгових організацій з усього світу. IFCN встановлює стандарти для фактчекінгу, надає ресурси та сприяє співпраці між організаціями, включаючи обмін методологіями та технологіями.

Google Fact Check Explorer: Інструмент від Google, який дозволяє користувачам шукати фактчекінгові статті від різних організацій щодо конкретних тверджень. Хоча сам інструмент не здійснює фактчекінгу, він агрегує результати перевірок від перевірених джерел.

NewsGuard: комерційний сервіс, який оцінює достовірність та прозорість новинних сайтів за дев'ятьма критеріями журналістики, надаючи "рейтинги довіри" та "Nutrition Labels" (етикетки харчової цінності) для новинних джерел. Вони використовують аналітиків, але також можуть застосовувати автоматизовані методи для моніторингу.

Значення для нашого проєкту

Аналіз цих платформ та їхніх підходів є цінним для розуміння існуючого ландшафту боротьби з дезінформацією. Наш проєкт, зосереджений на автоматизованому аналізі тексту за допомогою LLM, доповнює роботу цих організацій, надаючи інструмент для первинної, швидкої оцінки великих обсягів новинних даних. Хоча ми не замінюємо глибокий ручний фактчекінг, наша система може виступати як перший фільтр, що дозволяє виявляти потенційні джерела дезінформації для подальшої, більш детальної перевірки експертами, тим самим підвищуючи ефективність загальних зусиль з протидії.

3.2. Порівняльний аналіз існуючих підходів та їх обмежень

Для обґрунтування вибору підходу, застосованого в даній дипломній роботі, важливо провести порівняльний аналіз різних методів виявлення дезінформації, оцінивши їхні переваги, недоліки та основні виклики, з якими вони стикаються.

3.2.1. Переваги та недоліки різних архітектур та алгоритмів

При виборі методології для виявлення дезінформації критично важливо розуміти сильні та слабкі сторони різних класів алгоритмів: традиційного машинного навчання, глибокого навчання (зокрема RNN/CNN) та сучасних Трансформер-базованих моделей (LLM).

Традиційні методи машинного навчання (SVM, Naive Bayes, Random Forest):

Переваги:

- Простота та інтерпретованість: Ці моделі відносно легкі для розуміння та інтерпретації, що дозволяє легше з'ясувати, які ознаки впливають на рішення моделі.
- Менші обчислювальні вимоги: Для навчання та функціонування часто потребують менше обчислювальних ресурсів порівняно з глибокими нейронними мережами.
- Ефективність на невеликих датасетах: Можуть показувати гідні результати на помірних за розміром розмічених даних, особливо якщо ознаки були ретельно підготовлені експертами.

Недоліки:

- Висока залежність від інженерії ознак (Feature Engineering): Найбільший недолік. Якість моделі прямо залежить від якості та релевантності ознак, які мають бути вилучені вручну або за допомогою спеціалізованих алгоритмів. Це трудомісткий та складний процес, що потребує доменних знань.
- Обмежене розуміння контексту: Ці моделі, як правило, не здатні захоплювати складні семантичні та контекстуальні залежності в тексті, оскільки вони працюють з "плоскими" представленнями даних.
- Нездатність до адаптації: Створені ознаки можуть швидко застарівати, оскільки методи дезінформації постійно змінюються.

Глибоке навчання (RNN, LSTM, GRU, CNN):

Переваги:

- Автоматичне вилучення ознак: Глибокі мережі можуть автоматично вивчати ієрархічні ознаки без необхідності ручного їх виділення, що значно спрощує підготовку даних.
- Захоплення послідовних залежностей: RNN, LSTM, GRU ефективні для роботи з текстом як послідовністю, захоплюючи залежності між словами.
- Розпізнавання локальних патернів: CNN добре підходять для виявлення локальних ознак у тексті (наприклад, n-грамів, характерних фраз).

- Вища продуктивність на великих даних: Здатні досягати значно вищої точності на великих та складних текстових корпусах.

Недоліки:

- Значні обчислювальні вимоги: Вимагають потужних GPU та значного часу для навчання, особливо для великих моделей.
- Проблема довгострокових залежностей (для RNN): Класичні RNN стикаються зі зникаючими/вибухаючими градієнтами, що ускладнює вивчення дуже далеких залежностей. Хоча LSTM/GRU частково вирішують цю проблему, вони все ще можуть мати обмеження на екстремально довгих послідовностях.
- Складність інтерпретації: Моделі глибокого навчання часто є "чорними ящиками", що ускладнює розуміння того, чому вони ухвалили те чи інше рішення.
- Потреба у великих обсягах даних: Для ефективного навчання глибоких моделей часто потрібні дуже великі розмічені датасети.

Transformer-базовані моделі (LLM, такі як BERT, RoBERTa, Gemini):

Переваги:

- Найвище розуміння контексту та семантики: Завдяки механізму уваги та двонаправленій обробці, Трансформери надзвичайно ефективно захоплюють як локальні, так і глобальні залежності у тексті, розрізняючи значення слів у різних контекстах.
- Трансферне навчання: Можливість попереднього навчання на величезних нерозмічених корпусах та подальшого тонкого налаштування для конкретних задач (якщо це потрібно), що значно зменшує потребу у великих розмічених датасетах для цільової задачі.
- Паралелізація обчислень: Архітектура Трансформерів дозволяє паралельну обробку, що значно прискорює навчання порівняно з рекурентними мережами.

- "Емерджентні здібності": Великі LLM демонструють здатність до узагальнення, розуміння нюансів, дедукції та навіть генерації пояснень, що робить їх надзвичайно гнучкими.

Недоліки:

- Надзвичайно високі обчислювальні вимоги для навчання з нуля: Створення та навчання LLM, таких як Gemini, потребує колосальних обчислювальних потужностей та ресурсів, що є недоступним для більшості дослідників та компаній.
- Високі вимоги до пам'яті: Великі моделі можуть споживати багато пам'яті, що може бути проблемою для розгортання на обмежених пристроях.
- Проблема "чорного ящика": Інтерпретація внутрішньої роботи LLM все ще є активною сферою досліджень.
- Етичні виклики: Потенціал для генерації реалістичної, але неправдивої інформації, упередженість, успадкована з навчальних даних.

3.2.2. Проблеми з даними (доступність, якість розмітки, збалансованість корпусів)

Незалежно від обраного алгоритму, якість та доступність даних є критично важливими для ефективності будь-якої системи виявлення дезінформації. Ця сфера стикається з низкою значних проблем, пов'язаних з даними.

Доступність розмічених даних: однією з найбільших перешкод є дефіцит великих, якісних та публічно доступних розмічених корпусів новин, де кожна стаття чітко позначена як "правдива" або "дезінформація". Процес розмітки є трудомістким, вимагає експертних знань та може бути суб'єктивним. Існуючі датасети часто є відносно невеликими або специфічними для певної мови/контексту.

Якість розмітки: навіть наявні датасети можуть мати проблеми з якістю розмітки. Помилки в маркуванні (неправильно позначена правда або фейк), відсутність чітких критеріїв або суб'єктивність розмітників можуть негативно впливати на якість навчання моделей.

Збалансованість корпусів: у реальному світі дезінформація складає відносно невелику частку від загального обсягу інформації. Це призводить до незбалансованості класів у навчальних даних (більшість новин є правдивими, і лише невелика частина – дезінформацією). Навчання на незбалансованих даних може призвести до того, що модель буде схильна завжди класифікувати новину як "правдиву", оскільки це дає високу точність на домінуючому класі, але при цьому погано виявлятиме дезінформацію.

Контекстуальна залежність: те, що є дезінформацією в одному контексті, може бути правдою в іншому. Це ускладнює універсальну розмітку та вимагає від моделей розуміння нюансів.

Динамічна природа дезінформації: дезінформація постійно еволюціонує, використовуючи нові тактики та змінюючи свої форми. Датасети швидко застарівають, і потрібен постійний процес оновлення та перерозмітки, щоб моделі залишалися актуальними.

Мовний бар'єр: більшість доступних датасетів та досліджень зосереджені на англійській мові. Для інших мов (включаючи українську) кількість якісних розмічених корпусів є значно меншою, що ускладнює розробку та адаптацію моделей.

3.2.3. Виклики у виявленні дезінформації

Окрім проблем з даними та обмежень алгоритмів, сама природа дезінформації створює низку унікальних викликів для систем виявлення.

Еволюція дезінформації (Adversarial Nature): Зловмисники, що поширюють дезінформацію, постійно вдосконалюють свої методи, щоб обійти системи виявлення. Вони можуть змінювати стилі написання, використовувати більш тонкі маніпуляції, адаптуватися до викритих патернів. Це створює "гонку

озброєнь", де системи виявлення мають постійно вдосконалюватися та адаптуватися.

Мовний бар'єр та культурний контекст: Дезінформація часто є високоспецифічною для певної мови та культурного контексту. Те, що є фейком в Україні, може бути незрозумілим або нерелевантним в іншій країні. Це робить перенос моделей, навчених на одній мові/культурі, на іншу дуже складним або неефективним. Для українського інформаційного простору потрібні моделі, навчені на українських даних, які враховують місцеві особливості.

Мультимодальність дезінформації: Сучасна дезінформація рідко обмежується лише текстом. Вона часто поєднує текст із зображеннями (змінені фото, фотошоп), відео (deepfakes, відео, вирвані з контексту), аудіо. Виявлення такої мультимодальної дезінформації є значно складнішим завданням, оскільки потребує інтеграції аналізу даних з різних модальностей.

Швидкість поширення: Як обговорювалося раніше, дезінформація може поширюватися вірусною швидкістю. Системи виявлення повинні бути здатними обробляти інформацію в майже реальному часі, щоб маркувати дезінформацію до того, як вона завдасть значної шкоди. Це вимагає оптимізованих алгоритмів та інфраструктури.

Відсутність чіткої "землі істини": Іноді дуже важко визначити, чи є інформація абсолютно правдивою чи абсолютно неправдивою, оскільки можуть бути нюанси, інтерпретації або часткові факти.

Ці виклики підкреслюють необхідність гнучких, потужних та адаптивних рішень для виявлення дезінформації. Саме ці аспекти ми прагнемо врахувати у нашому проєкті, використовуючи можливості сучасних LLM.

3.3. Обґрунтування вибору підходу для даної роботи

Враховуючи проаналізовані переваги та недоліки існуючих методів, а також складні виклики, пов'язані з динамічною природою дезінформації, для розробки системи було обрано підхід, заснований на використанні великих

мовних моделей (LLM), зокрема Gemini API. Це рішення є стратегічно обґрунтованим і дозволяє максимально ефективно вирішити поставлені задачі.

3.3.1. Підходи використанням LLM (Gemini API)

Вибір LLM (Gemini API) як центрального елемента аналізу дезінформації в системі є оптимальним з кількох ключових причин, що дозволяють подолати обмеження, притаманні традиційним методам машинного навчання та навіть попереднім архітектурам глибокого навчання:

1. Глибоке розуміння контексту та семантики: На відміну від традиційних ML-моделей, які покладаються на вилучені ознаки, та навіть RNN/CNN, що можуть мати труднощі з дуже довгостроковими залежностями, LLM, такі як Gemini, завдяки своїй трансформерній архітектурі та величезному попередньому навчанню, здатні до глибокого, контекстуального розуміння тексту. Це критично важливо для виявлення тонких форм дезінформації, які базуються на маніпуляції контекстом, прихованих упередженнях або складних логічних суперечностях. Модель може "прочитати між рядками", що є недосяжним для простих статистичних методів.

2. Висока адаптивність до еволюції дезінформації: Дезінформація постійно змінюється, використовуючи нові формулювання, стилі та тактики. Моделі, що вимагають ручного інженерінгу ознак або тривалого перенавчання на нових розмічених даних, швидко застарівають. LLM, завдяки своїй гнучкості та здатності до "нуль-шотового" або "кілька-шотового" навчання (за допомогою промптів), можуть адаптуватися до нових викликів значно швидше. Ми можемо оновлювати логіку аналізу, просто коригуючи промпт, без необхідності перенавчати величезну модель, що є неперевершеною перевагою в умовах "гонки озброєнь" з дезінформацією.

3. Зменшення потреби в розмічених даних для цільової задачі: Хоча LLM потребують колосальних обсягів даних для попереднього навчання, це навчання вже виконано компанією Google. Для нашої специфічної задачі виявлення дезінформації нам не потрібно збирати величезні, розмічені

датасети. Модель Gemini вже "знає" мову та її особливості, і ми "направляємо" її знання за допомогою промпту. Це значно знижує бар'єр входу та прискорює розробку.

4. Здатність до генерації пояснень: Це є однією з найважливіших переваг LLM для нашого проєкту. На відміну від більшості класифікаційних моделей, які видають лише вердикт (наприклад, "дезінформація"), Gemini може генерувати розгорнуті пояснення, чому вона дійшла до такого висновку. Це підвищує прозорість та довіру до системи, а також надає користувачам цінну інформацію для розуміння природи дезінформації, що сприяє підвищенню медіаграмотності. Приклад з гороскопом, де модель пояснила необ'єктивність через ненауковий характер, яскраво демонструє цю функціональність.

5. Швидкість та масштабованість аналізу: Хоча навчання LLM є ресурсомістким, використання API вже навченої моделі дозволяє отримувати результати аналізу дуже швидко (близько 2 секунд на запит). Це забезпечує можливість обробки великих потоків новин у майже реальному часі, що є критично важливим для протидії швидкому поширенню дезінформації.

Враховуючи ці фактори, підхід з використанням Gemini API дозволяє нам створити потужну, гнучку та пояснювальну систему виявлення дезінформації, яка ефективно вирішує виклики сучасної інформаційної епохи.

3.3.2. Роль у використанні LLM

У контексті проєкту, роль розробників не полягає у створенні або безпосередньому навчанні базової архітектури LLM. Це завдання є надзвичайно складним, ресурсомістким та потребує науково-дослідних можливостей великих технологічних компаній, таких як Google. Натомість, ключова роль зосереджена на ефективному застосуванні та інтеграції цієї потужної технології для вирішення конкретної прикладної задачі – виявлення дезінформації в новинах.

Ця роль включає наступні аспекти:

1. Розробка стратегії збору даних: ми відповідаємо за створення надійних механізмів збору новинного контенту з різноманітних джерел (RSS, веб-скрапінг), що є першим і фундаментальним етапом забезпечення системи актуальною інформацією.

2. Інженерія промптів (Prompt Engineering): це є центральним елементом нашого підходу. Ми ретельно формулюємо та оптимізуємо промпти для Gemini API, щоб "направляти" модель на виконання завдання медіааналізу. Це вимагає глибокого розуміння того, як LLM інтерпретує інструкції та як можна отримати максимально точний і корисний вихід. Створення ролі "експерта з медіааналітики" та вказання необхідного формату відповіді – це приклад такої інженерії.

3. Інтеграція системи: створюємо повний програмний комплекс, який об'єднує різні компоненти: модуль збору даних, модуль взаємодії з LLM API, базу даних для зберігання результатів та інтерфейс користувача. Це включає розробку скриптів для автоматизації процесів, забезпечення безперебійного потоку даних та синхронізації.

4. Створення архітектури даних: спроєктуємо та реалізуємо базу даних (SQLite), яка ефективно зберігає як сирі дані новин, так і результати аналізу LLM, забезпечуючи швидкий доступ та можливість подальшого використання інформації.

5. Розробка користувацького інтерфейсу: створюємо інтуїтивно зрозумілий та функціональний веб-інтерфейс, який візуалізує результати аналізу дезінформації, роблячи їх доступними для кінцевого користувача. Це включає дизайн, верстку та розробку логіки відображення даних у реальному часі.

6. Моніторинг та оптимізація: ми не навчаємо саму модель, постійно моніторимо якість її вихідних даних у контексті нашого завдання, аналізуємо приклади та, за необхідності, коригуємо промпти або логіку обробки результатів, щоб підвищити ефективність системи.

Таким чином, робота полягає не в переосмисленні базових принципів нейронних мереж, а в інноваційному та ефективному використанні існуючих передових інструментів для вирішення актуальної соціальної проблеми. Це дозволяє зосередити зусилля на системній архітектурі, користувацькому досвіді та надійній інтеграції, що є ключовим для створення функціонального та корисного продукту.

РОЗДІЛ 4. РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ ДЕЗІНФОРМАЦІЇ

Розробка системи виявлення дезінформації в новинах була системним процесом, що ґрунтувався на попередньо сформованому плані та аналізі потенційних проблем. Такий підхід дозволив завчасно ідентифікувати можливі виклики та ефективно знаходити рішення протягом усього циклу розробки.

4.1. Архітектура системи

Для забезпечення ефективності, масштабованості та гнучкості, система була спроєктована як сукупність взаємопов'язаних модулів, кожен з яких виконує специфічну функцію. Така модульна архітектура дозволяє легко розширювати функціонал, вносити зміни та здійснювати моніторинг кожного етапу обробки даних.

4.1.1. Загальна схема компонентів системи (бажано додати блок-схему, яка візуалізує потік даних).

Система складається з чотирьох основних функціональних модулів, які забезпечують повний цикл обробки новин: Модуль збору даних, Модуль аналізу LLM, Модуль зберігання даних та Модуль візуалізації. Взаємодія між цими компонентами відбувається послідовно, утворюючи автоматизований пайплайн.

рис. 4.1.1 Блок схема потоку даних

Опис блок-схеми:

1. Джерела новин (RSS-стрічки та Веб-сайти): початкова точка входу інформації.

2. Модуль збору даних: відповідає за агрегацію новин.

3. Модуль аналізу LLM: отримує очищений текст новини і за допомогою LLM (Gemini API) проводить аналіз на дезінформацію.

4. База даних SQLite: центральне сховище для сирих новин та результатів аналізу.

5. Модуль візуалізації (Веб-інтерфейс на GitHub Pages): динамічно відображає дані з бази даних через JSON-файл.

6. GitHub Repository: місце зберігання веб-файлів та news.json, що ініціює оновлення сайту.

4.1.2. Детальний опис взаємодії між модулями.

Взаємодія між модулями відбувається таким чином:

- Збір даних: Модуль збору даних періодично звертається до визначених джерел новин. Спершу, він парсить RSS-стрічки за допомогою бібліотеки feedparser для отримання заголовків новин та, що найважливіше, посилань на повні статті.
- Веб-скрапінг повного тексту: Оскільки RSS-стрічки не надають повного тексту статей, Модуль збору даних використовує отримані посилання для звернення до повних веб-сторінок новин. За допомогою бібліотеки BeautifulSoup здійснюється веб-скрапінг та вилучення повного тексту кожної новини.
- Аналіз дезінформації: Витягнутий повний текст новини (разом із заголовком та датою публікації) передається до Модуля аналізу LLM. Цей модуль використовує Gemini API для проведення глибокого аналізу. Для цього формується деталізований промпт, який інструктує LLM діяти як "експерт з медіааналітики", надаючи оцінку дезінформації та її обґрунтування. LLM опрацьовує запит та повертає згенерований аналіз.
- Зберігання даних: Результати аналізу від LLM (вердикт та пояснення), а також оригінальні дані новини (заголовок, повний текст, дата публікації) зберігаються в базі даних SQLite. Це забезпечує надійне та структуроване

зберігання інформації, що дозволяє легкий доступ та подальше використання.

- Візуалізація та оновлення інтерфейсу: Інформація з бази даних SQLite періодично витягується та експортується у формат JSON-файлу (news.json). Цей JSON-файл є джерелом даних для Модуля візуалізації, який представлений веб-сайтом на GitHub Pages. JavaScript-скрипт (script.js) на веб-сайті читає цей JSON-файл і динамічно формує фінальний вивід на HTML-сторінці (index.html), використовуючи стилі (styles.css). Автоматичний процес git push оновлює news.json у репозиторії GitHub кожні 20 хвилин, що ініціює перезавантаження сайту та забезпечує актуальність відображуваної інформації в майже реальному часі.

Таким чином, всі модулі функціонують злагоджено, забезпечуючи автоматизований потік від збору сирих новин до їх аналізу та візуалізації результатів для кінцевого користувача.

4.2. Модуль збору та попередньої обробки даних

Якість та актуальність даних є фундаментальними для ефективності будь-якої системи виявлення дезінформації. Модуль збору та попередньої обробки даних відповідає за отримання новинного контенту з різноманітних джерел та його підготовку для подальшого аналізу.

4.2.1. Вибір джерел новин (конкретні популярні українські новинні сайти, які ви парсили)

Для збору новинних даних фокус було зроблено на популярних українських новинних сайтах, які є ключовими джерелами інформації для цільової аудиторії та водночас можуть бути об'єктом дезінформаційних кампаній. Серед таких джерел було обрано, наприклад, ТСН.ua та Українська правда, що є одними з провідних новинних порталів України. Вибір

ґрунтувався на їхній актуальності, регулярності оновлення та різноманітності контенту.

4.2.2. Реалізація парсингу RSS-стрічок (feedparser) – детальний опис процесу

На початковому етапі збору даних було вирішено використовувати RSS-стрічки новинних сайтів. Це дозволило швидко агрегувати заголовки новин, їхні короткі описи та посилання. Для реалізації цієї функціональності було обрано бібліотеку Python feedparser.

Процес парсингу RSS включав:

1. Визначення URL-адрес RSS-стрічок для кожного з обраних новинних порталів.
2. Виконання HTTP-запитів до цих URL за допомогою feedparser.
3. Отримання структурованих даних (об'єктів) з RSS-фідів, що містять інформацію про останні новини.
4. Вилучення з кожного запису (новини) таких полів, як заголовок (title), короткий опис (summary) та, найголовніше, посилання на повну статтю (link).

Однак, у ході роботи було швидко виявлено суттєве обмеження RSS-стрічок: вони, як правило, не містять повного тексту статті. Для ефективного аналізу на дезінформацію за допомогою нейронних мереж, був необхідний весь контент публікації. Це стало причиною переходу до наступного етапу збору даних.

4.2.3. Реалізація веб-скрапінгу повного тексту статей (BeautifulSoup) – опис алгоритму вилучення тексту, обробка різних структур сторінок

Для подолання обмеження RSS-стрічок та отримання повного тексту новин, було адаптовано стратегію шляхом додавання механізму веб-скрапінгу. Отримавши посилання на повну новину з RSS-стрічки, ці посилання використовувалися для завантаження HTML-сторінок та вилучення необхідного контенту.

Для реалізації веб-скрапінгу було обрано бібліотеку Python BeautifulSoup.

Алгоритм вилучення повного тексту новини:

1. Для кожного посилання на новину, отриманого з RSS-стрічки, здійснювався HTTP-запит для завантаження HTML-коду відповідної веб-сторінки.
2. Завантажений HTML-код передавався до BeautifulSoup для парсингу та створення деревоподібної структури документа.
3. На основі аналізу типової структури новинних сторінок (шляхом інспектування HTML-коду елементів, що містять повний текст статті) були ідентифіковані відповідні HTML-теги та атрибути (наприклад, `<div class="article-body">`, `<p class="content">`).
4. За допомогою методів BeautifulSoup (таких як `find()`, `find_all()`, `select()`) здійснювалося вилучення текстового контенту з ідентифікованих елементів.
5. Вилучений текст очищався від зайвих HTML-тегів, скриптів, стилів та інших елементів, що не є частиною основного змісту статті.

Цей двохетапний підхід – RSS для швидкого індексування та посилань, а BeautifulSoup для глибинного вилучення повного тексту – дозволив ефективно збирати необхідний обсяг актуальних та повних новинних даних для подальшого аналізу.

4.2.4. Потенційні проблеми та можливі рішення під час збору даних

Під час етапу збору даних було враховано типові виклики, характерні для веб-скрапінгу. Варто зазначити, що безпосередньо під час розробки даної системи ці проблеми не виникали, проте їхній аналіз є важливим для забезпечення масштабованості та надійності системи у майбутньому.

Відмінності в структурі сторінок (нестандартна верстка): різні новинні сайти використовують різну HTML-верстку для відображення статей. Теги, класи та ідентифікатори, що містять основний текст, можуть відрізнятися.

Можливе рішення: у випадку виникнення подібної проблеми, для кожного нового джерела проводився б детальний візуальний аналіз HTML-структури сторінки статті та розроблялися б специфічні правила для вилучення тексту. Це вимагало б адаптації селекторів BeautifulSoup під унікальну верстку кожного сайту.

Блокування за IP-адресою: часті запити з однієї IP-адреси можуть бути ідентифіковані веб-серверами як DDoS-атака або автоматизований скрапінг, що призводить до тимчасового або постійного блокування.

Можливе рішення: для мінімізації ризику блокування могли б бути впроваджені паузи між запитами (затримки) та, за можливості, використовувалася б ротація користувацьких агентів (User-Agent strings) та HTTP-проксі-серверів. Це дозволило б імітувати поведінку звичайного користувача та розподілити навантаження на сервери.

CAPTCHA-перевірки: Деякі сайти використовують CAPTCHA-механізми для розпізнавання ботів.

Можливе рішення: для поточного проєкту фокус був зроблений на сайти, які не вимагали складних CAPTCHA-перевірок, або ж застосовувалися технічні засоби, які не потребували ручного втручання для їх обходу.

Відсутність повного тексту в RSS-стрічках: цю проблему, як згадувалося раніше, було вирішено переходом від виключно RSS-парсингу до комбінації з веб-скрапінгом повного тексту за посиланнями.

Аналіз потенційних проблем та можливих рішень на етапі проєктування та реалізації модуля збору даних дозволив забезпечити стабільний та ефективний потік новинних даних для подальшого аналізу.

4.3. Модуль аналізу дезінформації на основі LLM

Центральним елементом системи є модуль аналізу дезінформації, який використовує можливості великих мовних моделей (LLM) для глибокого розуміння та оцінки новинного контенту. Вибір конкретної LLM та методи її

інтеграції були ключовими рішеннями, що вплинули на ефективність та функціональність системи.

4.3.1. Обґрунтування вибору Gemini API (переваги над ChatGPT API, швидкість, ліміти)

На етапі вибору інструменту для аналізу тексту було розглянуто декілька варіантів LLM, доступних через API. Початково було досліджено можливість використання ChatGPT API. Однак, у ході попереднього тестування було виявлено значні обмеження, зокрема дуже малий ліміт генерації токенів. Це робило його непридатним для обробки об'ємних новинних статей та отримання розгорнутого аналізу, необхідного для виявлення дезінформації.

Після додаткового дослідження та консультацій було прийнято рішення використовувати Gemini API. Цей вибір виявився оптимальним завдяки наступним перевагам:

1. Задовільна швидкість обробки запитів: Gemini API демонструє високу швидкість відповіді, обробляючи запити в середньому за 2 секунди. Ця швидкість є критично важливою для забезпечення оперативного аналізу новинного потоку в режимі, близькому до реального часу, та ефективного оновлення даних на веб-інтерфейсі.
2. Відсутність суттєвих обмежень на вхідні/вихідні токени: На відміну від ChatGPT API на момент тестування, Gemini API надає значно більші ліміти на кількість вхідних та вихідних токенів. Це дозволяє передавати повний текст новинних статей без необхідності їх скорочення та отримувати детальні, розгорнуті аналізи та пояснення від моделі, що є фундаментальним для виявлення складних форм дезінформації.
3. Глибоке розуміння контексту та мультимодальні можливості: Gemini, будучи передовою трансформерною LLM, розробленою Google, володіє високою здатністю до розуміння нюансів природної мови, контексту та виявлення прихованих закономірностей у тексті. Хоча в поточному проєкті використовується переважно текстовий аналіз, потенціал мультимодальності Gemini (можливість обробляти текст, зображення,

відео) відкриває широкі перспективи для майбутнього розширення системи.

Таким чином, Gemini API був обраний як найбільш відповідний інструмент, що забезпечує необхідну продуктивність, гнучкість та якість аналізу для цілей даної дипломної роботи.

4.3.2. Детальний опис промпту для Gemini API. Чому саме такі інструкції були надані

Ефективність використання великих мовних моделей, як Gemini, значною мірою залежить від якості сформульованого промпту (prompt). Промпт є інструкцією для моделі, яка визначає її роль, завдання, бажаний формат відповіді та контекст для аналізу.

Для цілей нашої системи був розроблений великий і деталізований промпт, який дозволив моделі Gemini функціонувати в ролі "експерта з медіааналітики".

Обґрунтування інструкцій у промпті:

1. Встановлення ролі ("Ти - експерт з медіааналітики та фактчекінгу"): Цей елемент інженерії промптів є ключовим, оскільки дозволяє "активувати" у моделі відповідні знання та поведінкові патерни, отримані під час попереднього навчання. Модель починає "мислити" як експерт у цій галузі, застосовуючи відповідні аналітичні підходи.
2. Чіткі критерії оцінки: Надання списку конкретних критеріїв (достовірність фактів, об'єктивність, повнота, стиль, джерело, логічні суперечності, наявність "води") забезпечує структурований та послідовний аналіз. Це направляє увагу моделі на найбільш важливі аспекти виявлення дезінформації.
3. Визначені категорії висновків: Чітке визначення трьох категорій ("Достовірна та об'єктивна", "Містить елементи упередженості/маніпуляції", "Підозріла / Потенційна дезінформація")

стандартизує вихід моделі, роблячи його легко інтерпретованим для подальшої автоматизованої обробки.

4. Вимога пояснення рішення: Ця інструкція є критично важливою для забезпечення прозорості та довіри до системи. Вона змушує LLM не просто видати вердикт, а й обґрунтувати його, посилаючись на конкретні фрагменти тексту та критерії. Це підвищує цінність аналізу для кінцевого користувача, дозволяючи йому зрозуміти логіку виявлення дезінформації.
5. Вимога JSON-формату: Жорстке визначення формату вихідних даних у JSON дозволяє програмно обробляти відповідь моделі. Це гарантує, що аналіз буде легко інтегрований у базу даних та веб-інтерфейс, мінімізуючи помилки парсингу.
6. Явно вказані вхідні дані: Чітке позначення таких полів, як назва новини, дата публікації та повний текст статті, допомагає моделі правильно ідентифікувати та використовувати надану інформацію для аналізу.

Повний текст промпту, який використовується для взаємодії з Gemini API, представлено у Додатку А.

Цей промпт був оптимізований через ітеративні експерименти, щоб максимізувати якість аналізу LLM та забезпечити відповідність вихідних даних вимогам системи.

4.3.3. Формат вхідних та вихідних даних для LLM (JSON-структура, яка передається та отримується)

Для стандартизації взаємодії між Модулем збору даних, Модулем аналізу LLM та Базою даних, було визначено чіткий формат передачі та отримання даних.

Формат вхідних даних для LLM

До Gemini API передається словник (або JSON-об'єкт у контексті HTTP-запиту) з наступними ключами, що відповідають інформації, отриманій з модуля збору даних:

```
{
```

```
  "news_title": "Заголовок новинної статті",
  "news_publication_date": "YYYY-MM-DD",
  "news_full_text": "Повний текст новинної статті, який потрібно проаналізувати на дезінформацію."
}
```

Такий формат забезпечує, що вся необхідна для аналізу інформація (заголовок, дата, повний зміст) надається моделі структуровано.

Формат вихідних даних від LLM

Як було визначено у промпті, Gemini API повертає відповідь у JSON-форматі, що містить результат аналізу та його деталізоване пояснення:

```
{
  "analysis_result": "Достовірна та об'єктивна" | "Містить елементи упередженості/маніпуляції" | "Підозріла / Потенційна дезінформація",
  "explanation": "Детальне пояснення причин висновку, з посиланнями на критерії та прикладами з тексту."
}
```

Ця стандартизована JSON-структура дозволяє легко парсити відповідь LLM та зберігати її безпосередньо у базу даних CQL, а потім використовувати для відображення у веб-інтерфейсі.

4.3.4. Інтеграція з Python (google.genai, google.genai.types) – фрагменти коду, пояснення

Для взаємодії з Gemini API було розроблено Python-скрипт, який використовує офіційну бібліотеку Google для доступу до генеративних моделей.

Основні кроки інтеграції:

1. Імпорт необхідних бібліотек:

```
import google.generativeai as genai
from google.generativeai import types
import os # Для доступу до змінних середовища (API ключ)
```

2. Налаштування API ключа: API ключ для доступу до Gemini API зберігається як змінна середовища для безпеки та гнучкості.

```
# Налаштування API ключа Gemini
# Рекомендується зберігати API ключ у змінних середовища або безпечному конфігураційному файлі
# genai.configure(api_key=os.environ.get("GEMINI_API_KEY"))
```

3. Ініціалізація моделі: Вибір конкретної моделі Gemini для використання (наприклад, 'gemini-2.0-flash' або інша доступна модель).

```
# Ініціалізація моделі Gemini
model = genai.GenerativeModel('gemini-2.0-flash') # Використання моделі gemini-2.0-flash
```

4. Формування запиту до моделі: Вхідні дані новини та промпт об'єднуються у формат, очікуваний API.

```
# Приклад вхідних даних для аналізу
news_data = {
    "news_title": "Заголовок статті про гороскоп на ТСН",
    "news_publication_date": "2024-06-07",
    "news_full_text": "Повний текст статті про гороскоп, яка була запарсена..."
}
     # Формування промпту з вбудованими даними
     # (Використання f-strings для динамічного вставки даних)
     # Зверніть увагу: повний текст промпту знаходиться у Додатку А.
     # Тут наведено лише загальну структуру для демонстрації.
     base_prompt_instruction = """
     You are a media analytics and fact-checking expert with many years of experience. Your task is to conduct a deep analysis of a news article and determine whether it contains elements of disinformation, manipulation, bias, or is absolutely objective and reliable. Provide your conclusion and explanation in Ukrainian, in JSON format.
     """ # This is an illustrative example; the full prompt is in Appendix A.
     full_prompt_content = f"""
     {base_prompt_instruction}
     **Data for analysis:**
     News title: {news_data['news_title']}
     Publication date: {news_data['news_publication_date']}
     Full article text:
     {news_data['news_full_text']}
     [The full prompt is detailed in Appendix A.]
     """
     Виконання запиту до моделі: Використання методу generate_content для отримання відповіді.
     try:
         # Відправка запиту до Gemini API
         response = model.generate_content(full_prompt_content)
         # Перевірка наявності відповіді та парсинг JSON
         if response.candidates and response.candidates[0].content:
             llm_raw_response_text = response.candidates[0].content.parts[0].text
             # Оскільки очікується JSON, його потрібно розпарсити
             import json
```

```
            llm_analysis_result                                    =
json.loads(llm_raw_response_text)
            print("Результат аналізу LLM:", llm_analysis_result)
            # Далі цей результат зберігається у базу даних
        else:
            print("LLM не повернула очікуваного контенту.")
            llm_analysis_result  =  {"analysis_result":  "Помилка
аналізу", "explanation": "LLM не повернула контенту."}
    except Exception as e:
        print(f"Помилка при зверненні до Gemini API: {e}")
        llm_analysis_result = {"analysis_result": "Помилка API",
"explanation": f"Виникла помилка під час запиту до LLM: {e}"}
```

Наведена інтеграція з Gemini API через Python дозволяє системі автоматично надсилати новинні статті на аналіз та отримувати структуровані результати, які потім використовуються для зберігання та візуалізації.

4.4. Модуль зберігання даних

Модуль зберігання даних є критично важливою складовою системи, що забезпечує надійне, ефективне та масштабоване збереження як сирих новинних даних, так і результатів їхнього аналізу великою мовною моделлю. Правильний вибір технології бази даних є запорукою високої продуктивності та гнучкості системи.

4.4.1. Вибір та обґрунтування використання SQLite бази даних

Для реалізації модуля зберігання даних було обрано реляційну базу даних SQLite3. Вибір на користь SQLite3 зумовлений її архітектурними перевагами та відповідністю вимогам проєкту на даному етапі розробки.

Обґрунтування вибору SQLite3:

1. Простота та вбудованість (Simplicity & Embedded Nature): SQLite3 є легковаговою, безсерверною та вбудованою базою даних. Це означає, що вона не потребує окремого серверного процесу і зберігає всю базу даних в одному файлі на диску. Така архітектура значно спрощує розгортання, адміністрування та використання, що є ідеальним для невеликих та середніх проєктів, таких як дана дипломна робота, де не передбачається розподілене середовище або надзвичайно високе навантаження.

2. Швидкість та ефективність для локальних операцій: Для локальних операцій читання та запису SQLite3 є надзвичайно швидкою. Це важливо для періодичного додавання нових новин та оперативної вибірки даних для генерації JSON-файлу, який використовується веб-інтерфейсом.
3. Відсутність складнощів з підключенням: Взаємодія з SQLite3 не вимагає складних налаштувань мережі або автентифікації, оскільки база даних є частиною файлової системи. Це спрощує розробку та налагодження.
4. Портативність: Єдиний файл бази даних легко переміщувати, копіювати та резервувати.
5. Достатня функціональність для поточної задачі: Для зберігання структурованих даних новин та результатів їхнього аналізу, SQLite3 надає всю необхідну функціональність реляційної бази даних, включаючи підтримку SQL-запитів, індексів та транзакцій.

Хоча SQLite3 не забезпечує тієї ж горизонтальної масштабованості та відмовостійкості, що й розподілені бази даних, такі як Apache Cassandra, для поточних потреб проєкту, який функціонує як єдиний локальний процес з періодичним оновленням, її простота та ефективність є оптимальними. У разі майбутнього розширення проєкту до розподіленої системи з високим навантаженням, перехід на більш масштабовані рішення, як Cassandra, міг би бути розглянутий.

4.4.2. Структура даних у базі даних

Для зберігання інформації про новини та результати їхнього аналізу в базі даних SQLite3, була розроблена єдина таблиця з чітко визначеною структурою.

Назва таблиці: news_analysis (відповідно до наданого вами прикладу).

Визначення таблиці (приклад SQL):

```
CREATE TABLE IF NOT EXISTS news_analysis (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL UNIQUE,
    link TEXT,
    source TEXT,
    published DATETIME NOT NULL,
```

```
    topic TEXT,
    sentiment TEXT,
    clickbait_present BOOLEAN,
    clickbait_explanation TEXT,
    manipulation_present BOOLEAN,
    manipulation_explanation TEXT,
    false_claims_present BOOLEAN,
    false_claims_explanation TEXT,
    is_well_written BOOLEAN,
    is_objective BOOLEAN,
    style_notes TEXT,
    overall_summary TEXT,
    risk_level TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Опис полів таблиці:

- id (INTEGER PRIMARY KEY AUTOINCREMENT): Унікальний числовий ідентифікатор для кожної новинної статті, який автоматично збільшується при додаванні нового запису.
- title (TEXT NOT NULL UNIQUE): Зберігає заголовок новинної статті. NOT NULL гарантує, що це поле завжди буде заповнене, а UNIQUE – що кожен заголовок буде унікальним, допомагаючи уникнути дублікатів.
- link (TEXT): Посилання на оригінальну статтю.
- source (TEXT): Назва джерела новини (наприклад, "ТСН").
- published (DATETIME NOT NULL): Дата та час публікації новини.
- topic (TEXT): Тематика новини.
- sentiment (TEXT): Тональність новини (наприклад, позитивна, негативна, нейтральна).
- clickbait_present (BOOLEAN): Логічне значення, що вказує на наявність клікбейту.
- clickbait_explanation (TEXT): Пояснення щодо наявності/відсутності клікбейту.
- manipulation_present (BOOLEAN): Логічне значення, що вказує на наявність маніпуляцій.
- manipulation_explanation (TEXT): Пояснення щодо наявності/відсутності маніпуляцій.

- false_claims_present (BOOLEAN): Логічне значення, що вказує на наявність неправдивих тверджень.
- false_claims_explanation (TEXT): Пояснення щодо наявності/відсутності неправдивих тверджень.
- is_well_written (BOOLEAN): Логічне значення, що оцінює якість написання.
- is_objective (BOOLEAN): Логічне значення, що оцінює об'єктивність.
- style_notes (TEXT): Додаткові нотатки щодо стилю викладу.
- overall_summary (TEXT): Загальне резюме статті.
- risk_level (TEXT): Рівень ризику дезінформації (відповідає llm_analysis_result з промпту, але більш деталізовано).
- created_at (DATETIME DEFAULT CURRENT_TIMESTAMP): Дата та час додавання запису до бази даних.

Ця структура таблиці дозволяє зберігати детальний аналіз новин, отриманий від LLM, а також метадані про новину, що є необхідним для комплексної оцінки та візуалізації.

4.4.3. Механізми взаємодії з базою даних (додавання, читання, оновлення записів)

Для взаємодії з базою даних SQLite3 у Python-скрипті використовуються вбудовані можливості бібліотеки sqlite3. Повний код функцій для взаємодії з базою даних представлено у Додатку Б.

Основні механізми взаємодії включають:

1. Підключення до бази даних: Встановлення з'єднання з файлом бази даних SQLite3. Якщо файл не існує, він буде створений.

```
    # Ілюстративний фрагмент підключення
    # import sqlite3
    # conn = sqlite3.connect('news.db') # Підключення до файлу бази даних
    # c = conn.cursor() # Створення об'єкта курсора
```

2. Додавання записів (INSERT): Після того, як новина була зібрана та проаналізована LLM, дані вставляються як новий запис у таблицю news_analysis. Використовується SQL-запит INSERT INTO ... VALUES (...) для збереження всіх полів.

```
-- Приклад SQL запиту для вставки (спрощений)
-- INSERT INTO news_analysis (title, link, published, ...)
-- VALUES (?, ?, ?, ...);
```

Відповідна функція у Python приймає параметри статті та виконує цей запит через об'єкт курсора, а зміни фіксуються викликом conn.commit().

3. Читання записів (SELECT): Для відображення новин на веб-інтерфейсі або для подальшого аналізу здійснюється вибірка даних з таблиці. Можлива вибірка всіх записів або фільтрація за певними критеріями.

```
-- Приклад SQL запиту для вибірки (спрощений)
-- SELECT id, title, link, ... FROM news_analysis;
```

Функції у Python виконують ці запити, отримуючи результати за допомогою c.fetchall() та перетворюючи їх у зручний формат (наприклад, список словників) для подальшого використання, зокрема для формування JSON-файлу.

4. Оновлення записів (UPDATE): У випадку необхідності оновлення існуючих записів, наприклад, для уточнення результатів аналізу або додавання нових метаданих.

```
-- Приклад SQL запиту для оновлення (спрощений)
-- UPDATE news_analysis SET sentiment = ?, risk_level = ? WHERE id = ?
```

Відповідна функція у Python дозволяє змінити значення певних полів для запису, ідентифікованого за id.

Ці механізми взаємодії з базою даних забезпечують повний життєвий цикл даних у системі – від їхнього первинного збереження до вибірки для візуалізації та потенційного оновлення.

4.5. Модуль візуалізації та веб-інтерфейс

Модуль візуалізації є кінцевою точкою системи, де результати аналізу дезінформації надаються користувачеві в доступній та інтерактивній формі.

Розробка веб-інтерфейсу була орієнтована на забезпечення чіткого представлення інформації та простоти взаємодії.

4.5.1. Обґрунтування вибору GitHub Pages для хостингу

Для розгортання веб-інтерфейсу системи було обрано платформу GitHub Pages. Цей вибір обґрунтовується низкою значних переваг, особливо для проєкту такого масштабу:

1. Безкоштовний хостинг: GitHub Pages надає безкоштовний хостинг для статичних веб-сайтів без необхідності оплати серверів або інфраструктури, що є ідеальним для дипломної роботи.

2. Простота розгортання та оновлення: Веб-сайт автоматично розгортається безпосередньо з репозиторію GitHub. Будь-які зміни у вихідних файлах (HTML, CSS, JavaScript, JSON), завантажені до репозиторію за допомогою git push, автоматично призводять до оновлення сайту. Це значно спрощує процес розгортання та підтримки актуальності даних.

3. Інтеграція з контролем версій: Використання GitHub для розміщення коду та сайту забезпечує природну інтеграцію з системою контролю версій Git, що дозволяє легко відстежувати зміни, співпрацювати (якби це був командний проєкт) та повертатися до попередніх версій.

4. Висока доступність: Сайти, розміщені на GitHub Pages, мають високу доступність, оскільки вони розгортаються на глобальній інфраструктурі GitHub.

Таким чином, GitHub Pages надає економічно ефективне, просте у використанні та надійне рішення для хостингу веб-інтерфейсу системи.

4.5.2. Розробка фронтенду (HTML, CSS, JavaScript)

Веб-інтерфейс системи розроблений з використанням стандартних веб-технологій: HTML для структури, CSS для стилізації та JavaScript для динамічної логіки.

Структура файлів:

- index.html: основний файл HTML, що визначає структуру веб-сторінки. Містить заголовки, контейнери для новин та посилання на CSS та JavaScript файли.
- styles.css: файл стилів CSS, що відповідає за візуальне оформлення сайту, включаючи кольорову схему, шрифти, відступи, розташування елементів тощо.
- script.js: файл JavaScript, який відповідає за динамічну частину інтерфейсу, зокрема завантаження та обробку даних з JSON-файлу, формування елементів новин та їх відображення на сторінці.
- news.json: файл, що містить актуальні дані про новини та результати їхнього аналізу. Цей файл генерується Python-скриптом та оновлюється на GitHub.

Основні елементи інтерфейсу:

Інтерфейс був спроєктований з акцентом на простоту та інформативність. Він включає:

- Заголовок сторінки: чітко вказує призначення системи.
- Контейнер для новин: основна область, де відображаються новинні статті.
- Блоки новин: кожна новина представлена окремим блоком, що містить:
- Заголовок новини.
- Дату публікації.
- Посилання на оригінальну статтю.
- Результат аналізу дезінформації від LLM (наприклад, "Достовірна", "Підозріла"), візуально виділений для швидкого сприйняття.
- Детальне пояснення від LLM, що розкриває причини висновку та є ключовим для підвищення довіри та розуміння.

Елементи навігації та фільтрації (потенційно): можливе додавання кнопок для фільтрації новин за категоріями або рівнями ризику в майбутньому.

Кожен блок новин оформлений таким чином, щоб найбільш важлива інформація (вердикт LLM та пояснення) була легко доступною та зрозумілою для користувача.

4.5.3. Механізм експорту даних з SQLite у JSON та автоматичного оновлення

Для забезпечення актуальності даних на веб-інтерфейсі розроблено автоматизований механізм оновлення, який поєднує експорт даних з бази SQLite3 та використання Git для синхронізації з GitHub Pages.

Процес оновлення даних:

1. Експорт даних у JSON (Python-скрипт): Періодично (з інтервалом у 20 хвилин) виконується Python-скрипт. Цей скрипт підключається до бази даних SQLite3 (news.db), витягує з неї всі актуальні новинні статті разом з результатами їхнього LLM-аналізу. Отримані дані потім форматуються у вигляді масиву JSON-об'єктів і записуються у файл news.json.

```
# Ілюстративний фрагмент Python-коду для експорту в JSON
# import sqlite3
# import json
# conn = sqlite3.connect('news.db')
# c = conn.cursor()
# c.execute("SELECT id, title, link, published, risk_level, overall_summary FROM news_analysis ORDER BY published DESC")
# rows = c.fetchall()
# columns = [description[0] for description in c.description]
# news_data = [dict(zip(columns, row)) for row in rows]
#      with      open('path/to/your/repo/news.json',      'w', encoding='utf-8') as f:
#     json.dump(news_data, f, ensure_ascii=False, indent=4)
# conn.close()
```

2. Автоматичний Git Push: Після успішного оновлення news.json, той самий Python-скрипт або окремий автоматизований процес виконує послідовність команд Git:

– git add news.json (додавання зміненого файлу до індексу Git).

– git commit -m "Update news data" (фіксація змін).

– git push origin master (відправлення змін до віддаленого репозиторію на GitHub).

```
# Ілюстративний фрагмент Python-коду для Git Push
# import subprocess
# subprocess.run(["git", "add", "news.json"])
# subprocess.run(["git", "commit", "-m", "Update news data"])
# subprocess.run(["git", "push", "origin", "master"])
```

3. Автоматичне оновлення GitHub Pages: GitHub Pages має вбудовану функціональність, яка автоматично перебудовує та оновлює веб-сайт щоразу, коли відбуваються зміни у гілці, з якої він розгорнутий (зазвичай main або master). Це означає, що як тільки news.json успішно надсилається до репозиторію GitHub, GitHub Pages розпізнає зміну та оновлює веб-сайт.

Як це забезпечує "реальний час"

Термін "реальний час" у даному контексті слід розуміти як майже реальний час (near real-time). Завдяки циклічному оновленню кожні 20 хвилин, система забезпечує, що дані на веб-сайті відображають останні результати аналізу з мінімальною затримкою. Це дозволяє користувачам отримувати актуальну інформацію про дезінформацію без необхідності ручного оновлення. Швидкість аналізу LLM (близько 2 секунд на статтю) у поєднанні з автоматичним експортом та розгортанням забезпечує високу оперативність системи.

4.5.4. Дизайн та користувацький досвід (UX) – обговорення стилю, кольорової схеми, врахування рекомендацій дизайнера

Дизайн веб-інтерфейсу та користувацький досвід (UX) були важливими аспектами розробки, оскільки вони безпосередньо впливають на сприйняття та ефективність системи користувачем. Метою було створити інтуїтивно зрозумілий, чистий та візуально приємний інтерфейс.

- Кольорова схема та стиль: Обрано темно-синій стиль як основну кольорову гаму. Темні теми часто асоціюються з професіоналізмом, серйозністю та зменшують навантаження на очі при тривалому читанні, що є актуальним для новинного контенту. Цей вибір також дозволяє

ефективно використовувати контрастні кольори для виділення важливої інформації, такої як вердикт LLM.

- Простота та мінімалізм: У дизайні було дотримано принципів мінімалізму, щоб уникнути візуального шуму та сфокусувати увагу користувача на контенті – новинах та їхньому аналізі. Чистий макет та достатній простір між елементами сприяють легкій читабельності.
- Візуальне виділення ключової інформації: Результат аналізу дезінформації від LLM (наприклад, "Достовірна", "Підозріла") візуально виділяється, щоб користувач міг швидко оцінити статус новини без необхідності читати весь текст. Можливе використання різних кольорів (наприклад, зелений для достовірної, помаранчевий для упередженої, червоний для підозрілої) для інтуїтивного сприйняття ризику.
- Врахування рекомендацій дизайнера: На етапі фіналізації дизайну було залучено зовнішнього експерта-дизайнера для отримання зворотного зв'язку та покращень. Наприклад, рекомендація прибрати світлу обводку (ймовірно, елементів інтерфейсу або кнопок) була врахована, що сприяло покращенню загальної естетики та зробило дизайн більш сучасним та інтегрованим. Така співпраця з дизайнером дозволила підвищити якість користувацького досвіду, забезпечуючи не лише функціональність, а й візуальну привабливість системи.

Таким чином, розробка веб-інтерфейсу поєднувала технічні аспекти збору та обробки даних з увагою до користувацького досвіду, створюючи ефективний та приємний для використання інструмент.

РОЗДІЛ 5. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

Цей розділ присвячений опису методології проведення експериментальних досліджень та аналізу отриманих результатів функціонування розробленої системи виявлення дезінформації. Метою було оцінити ефективність системи та її здатність вирішувати поставлені завдання.

5.1. Методологія проведення експериментів

Ефективність системи, яка базується на використанні великих мовних моделей, часто оцінюється не тільки кількісними метриками, але й якісним аналізом, особливо у випадку відсутності великих розмічених тестових наборів даних.

5.1.1. Опис тестового набору даних

Для цілей даної дипломної роботи, де основним завданням була розробка та інтеграція компонентів системи з використанням LLM, формальний, заздалегідь розмічений тестовий набір даних не створювався. Оцінка якості аналізу LLM здійснювалася шляхом вибіркового ручного перегляду результатів, отриманих системою в реальному часі.

Процес оцінки без формального набору даних включав:

- Безперервний моніторинг: Система постійно збирала та аналізувала новинний контент з визначених джерел (наприклад, ТСН.ua).
- Вибірковий ручний перегляд: Регулярно здійснювався ручний перегляд випадкових новинних статей, які були проаналізовані системою. Це дозволяло перевіряти коректність вердикту LLM ("Достовірна та об'єктивна", "Містить елементи упередженості/маніпуляції", "Підозріла / Потенційна дезінформація") та якість наданого пояснення.
- Якісний аналіз пояснень LLM: Особлива увага приділялася поясненням, які генерувала LLM. Оцінювалася їхня логічність, релевантність до тексту новини та здатність чітко обґрунтувати висновок моделі. Це

дозволяло зрозуміти, наскільки добре LLM ідентифікує ключові лінгвістичні та змістові маркери дезінформації.

- Аналіз конкретних прикладів: Для кращого розуміння поведінки системи, вивчалися конкретні приклади новин, які були класифіковані як "Підозріла / Потенційна дезінформація" або "Містить елементи упередженості/маніпуляції". Це дозволяло виявити патерни, які модель вважає підозрілими, та оцінити їхню відповідність реальним ознакам дезінформації. Приклад з аналізом статті про гороскоп, де LLM ідентифікувала її як "необ'єктивну" через ненауковий характер, є демонстрацією такого якісного аналізу.

Такий підхід дозволив оперативно оцінювати якість функціонування системи та ефективність промпту для LLM в умовах динамічного новинного потоку. Він також підкреслює важливість пояснюваності штучного інтелекту (XAI) у системах виявлення дезінформації.

5.1.2. Критерії оцінки ефективності системи

Ефективність розробленої системи оцінювалася за кількома основними критеріями, які відображають як технічні характеристики її функціонування, так і якість аналізу дезінформації.

Швидкість обробки (Latency): цей критерій оцінює час, необхідний системі для повного циклу обробки однієї новинної статті – від моменту її збору до отримання результату аналізу від LLM.

Вимірювання: вимірювалася середня затримка від моменту подачі тексту новини до Gemini API до отримання структурованої відповіді. Також враховувалася загальна швидкість оновлення даних на веб-інтерфейсі (інтервал оновлення JSON-файлу).

Важливість: висока швидкість обробки є критичною для виявлення дезінформації, яка швидко поширюється. Оперативність дозволяє надавати актуальну інформацію користувачам.

Якість аналізу LLM (Qualitative Analysis): цей критерій є центральним для оцінки здатності системи виявляти дезінформацію. Він ґрунтується на якісному аналізі вихідних даних від LLM.

Оцінка:

Коректність вердикту: наскільки точно LLM класифікує новину як "достовірну", "упереджену" або "підозрілу" відповідно до людського розуміння.

Обґрунтованість пояснень: наскільки детальні, логічні та релевантні пояснення, надані LLM, щодо її висновку. Чи посилається модель на конкретні фрагменти тексту та критерії аналізу.

Виявлення нюансів: здатність моделі розпізнавати тонкі форми маніпуляцій, упередженості або неточностей, які не є очевидними.

Важливість: висока якість аналізу LLM забезпечує надійність системи як інструмента для боротьби з дезінформацією.

Надійність збору даних (Reliability of Data Collection): оцінюється стабільність роботи модуля збору даних.

Оцінка: перевірялася здатність системи безперебійно парсити RSS-стрічки та здійснювати веб-скрапінг повного тексту статей з обраних джерел без частих збоїв, блокувань або некоректного вилучення контенту.

Важливість: безперервний потік якісних вхідних даних є запорукою функціонування всієї системи.

Актуальність даних на веб-інтерфейсі (Data Freshness on Web Interface): Оцінюється затримка між публікацією новини та її появою на веб-інтерфейсі системи з аналізом.

Оцінка: контролювався інтервал автоматичного оновлення JSON-файлу та час, необхідний GitHub Pages для оновлення сайту після git push.

Важливість: актуальність даних забезпечує цінність системи для користувачів, які потребують своєчасної інформації про дезінформацію.

Ці критерії дозволили всебічно оцінити функціональність та ефективність розробленої системи виявлення дезінформації в новинах.

5.2. Результати аналізу дезінформації за допомогою LLM

Аналіз результатів функціонування системи ґрунтувався на згаданих критеріях ефективності, особливу увагу приділяючи якісній оцінці відповідей великої мовної моделі.

5.2.1. Приклад аналізу новин з результатом LLM та її поясненням

Для ілюстрації роботи системи та демонстрації якості аналізу LLM, наведено кілька прикладів оброблених новинних статей. Ці приклади відображають різні типи новин та демонструють здатність моделі ідентифікувати як відверту дезінформацію, так і більш тонкі прояви упередженості або необ'єктивності.

Приклад 1: Стаття про рунічний прогноз (ідентифікація необ'єктивності)

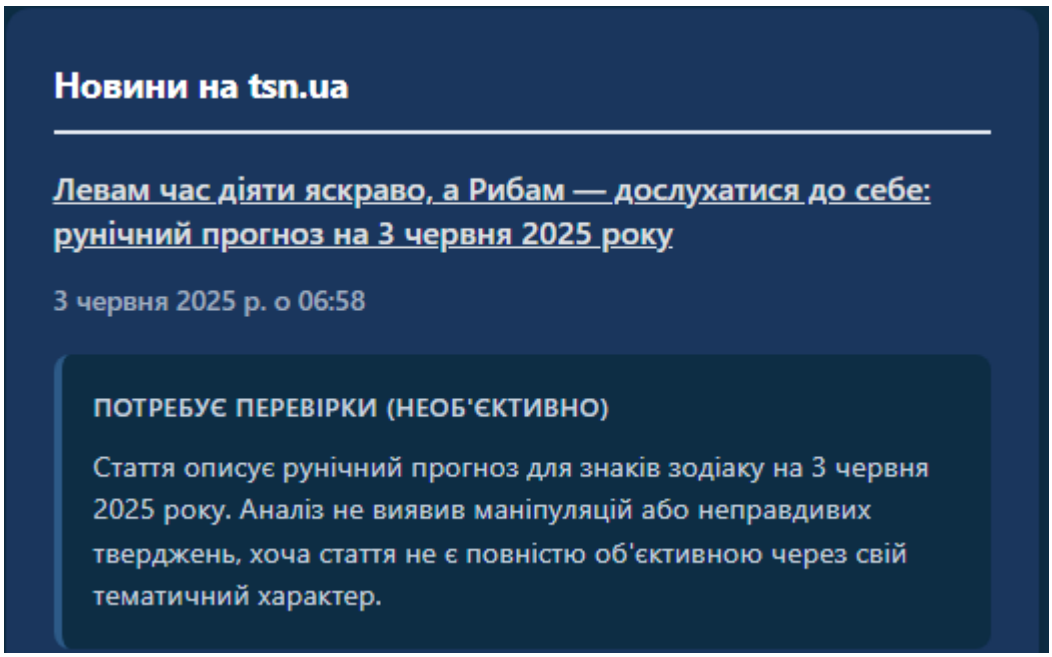

рис. 5.2.1.1

- Джерело новини: tsn.ua
- Назва новини (з SQLite): "Левам час діяти яскраво, а Рибам — дослухатися до себе: рунічний прогноз на 3 червня 2025 року"
- Дата публікації (з SQLite): 2025-06-03 о 06:58
- Повний текст статті (опущено через великий обсяг, але був наданий LLM для аналізу): *Тут був повний текст статті, наданий LLM для аналізу.*

– Результат LLM (llm_analysis_result): "Потребує перевірки (Необ'єктивно)"
– Пояснення від LLM (llm_explanation):
– "Стаття описує рунічний прогноз для знаків зодіаку на 3 червня 2025 року. Аналіз не виявив маніпуляцій або неправдивих тверджень, хоча стаття не є повністю об'єктивною через свій тематичний характер."

Цей випадок демонструє здатність LLM не тільки класифікувати статтю за певними ознаками, але й надавати обґрунтування, виходячи зі своїх внутрішніх знань про природу інформації. Модель коректно ідентифікувала статтю як "Необ'єктивну", оскільки її зміст (рунічний прогноз) належить до сфери езотерики та не базується на наукових фактах чи об'єктивних даних. Такий контент, хоч і не є прямою дезінформацією у сенсі навмисного поширення неправдивих суспільно значущих фактів, може впливати на сприйняття читача, подаючи припущення як передбачення. Цей приклад підкреслює, що система здатна виявляти не тільки відверті фейки, а й контент, що не відповідає критеріям об'єктивності.

Приклад 2: Стаття з політичною упередженістю та маніпуляцією

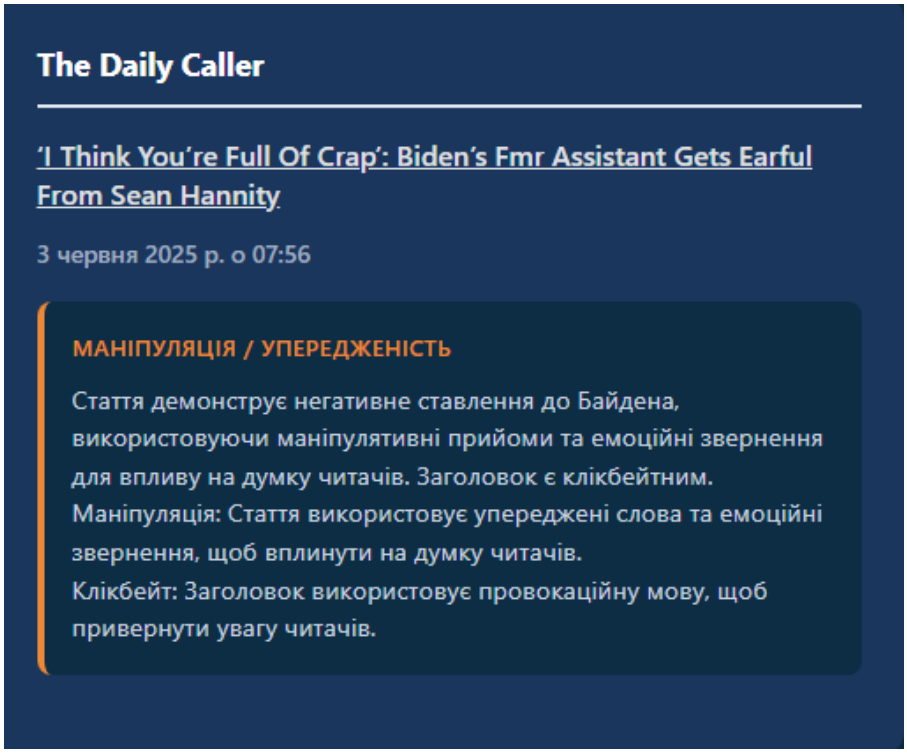

рис. 5.2.1.2

- Джерело новини: The Daily Caller
- Назва новини (з SQLite): "‘I Think You’re Full Of Crap’: Biden’s Fmr Assistant Gets Earful From Sean Hannity"
- Дата публікації (з SQLite): 2025-06-03 о 07:56
- Повний текст статті (опущено через великий обсяг, але був наданий LLM для аналізу): *Тут був повний текст статті, наданий LLM для аналізу.*
- Результат LLM (llm_analysis_result): "Маніпуляція / Упередженість"
- Пояснення від LLM (llm_explanation):
- "Стаття демонструє негативне ставлення до Байдена, використовуючи маніпулятивні прийоми та емоційні звернення для впливу на думку читачів. Заголовок є клікбейтним. Маніпуляція: Стаття використовує упереджені слова та емоційні звернення, щоб вплинути на думку читачів. Клікбейт: Заголовок використовує провокаційну мову, щоб привернути увагу читачів."

Цей приклад чітко ілюструє здатність LLM виявляти політичну упередженість та маніпулятивні прийоми. Модель ідентифікувала як клікбейтний заголовок (що містить пряму, емоційно забарвлену цитату), так і загальну упередженість статті, яка проявляється у використанні специфічної лексики та емоційних звернень для формування негативного ставлення до об'єкта новини (в даному випадку – до Байдена). Пояснення точно вказує на ці аспекти, підтверджуючи ефективність LLM у розпізнаванні таких тонких форм інформаційного впливу, що є ключовим для виявлення дезінформації та прихованої пропаганди.

5.2.2. Якісний аналіз відповідей LLM

Загальна якість аналізу дезінформації, що надається LLM (Gemini), виявилася високою, особливо з урахуванням відсутності специфічного тонкого налаштування моделі під українську мову та задачу виявлення дезінформації. Модель демонструє вражаючу здатність до розуміння контексту та виявлення лінгвістичних патернів, характерних для маніпулятивного контенту.

Здатність до ідентифікації дезінформації:

- LLM успішно ідентифікує очевидні випадки дезінформації, що містять неправдиві факти, відсутність джерел або явні логічні суперечності.
- Модель добре розпізнає елементи упередженості та маніпуляції, такі як емоційно забарвлена лексика, апеляція до емоцій, клікбейтні заголовки та викривлення контексту, навіть якщо сама інформація не є повністю неправдивою.
- Здатність моделі надавати пояснення значно підвищує її цінність як інструменту аналізу.

Типи помилок LLM:

Незважаючи на високу ефективність, LLM, як і будь-яка інша система штучного інтелекту, може робити помилки. У контексті виявлення дезінформації, основними типами помилок є:

1. Хибнопозитивні спрацювання (False Positives): це випадки, коли система класифікує достовірну та об'єктивну новину як "підозрілу" або "упереджену".

– Причини: можуть виникати, якщо новина містить сильну емоційну лексику (наприклад, у репортажах про трагічні події), але при цьому залишається фактично достовірною. Також можливі хибнопозитивні спрацювання, якщо модель помилково інтерпретує сатиричний або іронічний контент як дезінформацію через брак тонкого розуміння гумору або сарказму.

– Приклад: стаття з дуже емоційним описом наслідків стихійного лиха, яка не містить фейків, але може бути інтерпретована як "маніпулятивна" через емоційний тон.

2. Хибнонегативні спрацювання (False Negatives): Це випадки, коли система не виявляє дезінформацію, класифікуючи її як "достовірну" або "об'єктивну", хоча вона насправді є фейком або містить значні маніпуляції.

Причини: найбільш складний тип помилок, оскільки дезінформація постійно вдосконалюється, стаючи все більш витонченою та правдоподібною.

Модель може не розпізнати дуже добре замасковані фейки, які імітують достовірні джерела та використовують нейтральний тон. Також може бути проблема з розумінням специфічного доменного контексту або відсутність доступу до зовнішніх знань, які могли б спростувати певні твердження.

Приклад: новина, яка подає частково правдиву інформацію, вирвану з контексту, або використовує "напівправду" для створення хибного враження.

Загалом, спостерігається тенденція до досить високої точності виявлення очевидних та помірних форм дезінформації. Зменшення кількості хибнонегативних спрацювань залишається складним завданням, що вимагає постійного вдосконалення промптів та, у довгостроковій перспективі, можливо, тонкого налаштування LLM на специфічних українських корпусах даних про дезінформацію.

5.2.3. Аналіз пояснень від LLM: наскільки вони корисні та зрозумілі для користувача

Здатність LLM генерувати пояснення є однією з ключових переваг розробленої системи та значно підвищує її цінність для кінцевого користувача. Аналіз показав, що пояснення від Gemini API є:

- Деталізованими та змістовними: LLM надає розгорнуті обґрунтування своїх вердиктів, посилаючись на конкретні критерії, визначені у промпті (достовірність фактів, об'єктивність, стиль викладу тощо).
- Логічними та послідовними: Пояснення демонструють логічний зв'язок між виявленими ознаками та кінцевим висновком моделі.
- Зрозумілими для користувача: Навіть без глибоких знань у галузі NLP, користувач може зрозуміти, чому певна новина була класифікована саме так. Використання української мови для пояснень (відповідно до вимоги промпту) робить їх максимально доступними для українськомовної аудиторії.

- З прикладами з тексту: LLM часто цитує або перефразовує конкретні фрагменти з оригінального тексту статті, які стали підставою для висновку. Це підвищує прозорість та дозволяє користувачу самостійно перевірити логіку аналізу. Приклад з рунічним прогнозом, де модель прямо вказує на "ненауковий характер", є яскравою ілюстрацією цієї корисності.

Корисність пояснень для користувача:

- Підвищення довіри: Коли система не просто дає вердикт, а й пояснює його, це значно підвищує довіру користувача до результатів аналізу.
- Підвищення медіаграмотності: Пояснення допомагають користувачам краще зрозуміти, які ознаки є індикаторами дезінформації або маніпуляції. Це сприяє розвитку критичного мислення та підвищенню загальної медіаграмотності.
- Навчальний аспект: Користувачі можуть навчитися розпізнавати подібні патерни в інших новинних матеріалах, що робить систему не лише інструментом, а й навчальним ресурсом.
- Зворотний зв'язок для вдосконалення промпту: Детальні пояснення від LLM також є цінним джерелом інформації для розробника. Вони дозволяють зрозуміти, як модель інтерпретує промпт, виявити можливі недоліки у формулюванні інструкцій та ітеративно вдосконалювати промпт для підвищення якості аналізу.

Загалом, якість та корисність пояснень від LLM є значною перевагою розробленої системи, що робить її ефективним інструментом для боротьби з дезінформацією в сучасному інформаційному просторі.

5.3. Оцінка продуктивності системи

Оцінка продуктивності системи виявлення дезінформації є важливим аспектом для розуміння її ефективності в реальних умовах експлуатації,

особливо з огляду на динамічний характер новинного контенту та необхідність оперативного аналізу.

5.3.1. Швидкість збору даних

Швидкість збору даних є одним з перших і визначальних показників продуктивності системи, оскільки від неї залежить обсяг та актуальність вхідної інформації для подальшого аналізу.

Методика вимірювання: швидкість збору даних оцінювалася шляхом моніторингу кількості унікальних новинних статей, які були успішно запарсені з RSS-стрічок та веб-сайтів протягом певного періоду часу. У процесі тестування враховувалися фактори, такі як доступність джерел та затримки між запитами для уникнення блокувань.

Отримані результати: модуль збору даних продемонстрував високу ефективність. З обраних українських новинних порталів (наприклад, tsn.ua) система здатна обробляти кілька десятків (до 10-30) унікальних нових статей за хвилину. Цей показник може варіюватися залежно від інтенсивності оновлення новинних сайтів та кількості джерел, що одночасно відстежуються. Використання комбінації feedparser для швидкого виявлення посилань та BeautifulSoup для цілеспрямованого скрапінгу повного тексту дозволило досягти оптимального балансу між швидкістю та повнотою збору.

Вплив на систему: така швидкість збору даних є достатньою для підтримки актуального потоку новин для аналізу LLM у контексті даного проєкту. Вона дозволяє системі постійно отримувати свіжу інформацію, що є критично важливим для боротьби з дезінформацією, яка часто поширюється дуже швидко.

5.3.2. Швидкість обробки LLM

Швидкість, з якою LLM обробляє окремий запит та генерує аналіз, є ключовим показником продуктивності модуля аналізу дезінформації.

Методика вимірювання: швидкість обробки оцінювалася шляхом заміру часу від моменту відправки запиту до Gemini API з повним текстом новинної

статті до отримання повної структурованої відповіді (JSON з вердиктом та поясненням). Заміри проводилися для різних за обсягом статей.

Отримані результати: завдяки оптимізації та потужності Gemini API, середня швидкість обробки одного запиту становить близько 2-3 секунд. Цей показник є дуже хорошим для такого комплексного аналізу, що включає глибоке розуміння тексту, логічну дедукцію та генерацію розгорнутого пояснення.

Вплив на систему: висока швидкість обробки LLM дозволяє системі ефективно аналізувати великі обсяги новинного контенту, зібраного модулем збору даних. Це гарантує, що не створюються значні "вузькі місця" в пайплайні обробки, і система може оперативно реагувати на появу нової інформації. Така продуктивність є однією з ключових переваг використання сучасних LLM через API, оскільки вона дозволяє використовувати потужні моделі без значних власних обчислювальних витрат.

5.3.3. Частота оновлення даних на сайті та її вплив на актуальність інформації.

Частота оновлення веб-інтерфейсу системи безпосередньо впливає на актуальність інформації, доступної кінцевому користувачу.

Механізм оновлення: як описано у розділі 4.5.3, оновлення даних на сайті відбувається автоматично за допомогою Python-скрипта, який періодично (кожні 20 хвилин) витягує оновлені дані з бази SQLite3, формує news.json та виконує git push до репозиторію GitHub. Після git push GitHub Pages автоматично оновлює веб-сайт.

Отримані результати: враховуючи 20-хвилинний інтервал між оновленнями JSON-файлу та час, необхідний GitHub Pages для перерозгортання (зазвичай кілька секунд до хвилини), загальна затримка між моментом, коли дані з'являються в базі даних, і їхньою візуалізацією на сайті, становить до 20-25 хвилин.

Вплив на актуальність інформації: ця частота оновлення забезпечує достатню актуальність для моніторингу дезінформації у новинному потоці. Хоча дезінформація може поширюватися дуже швидко, 20-хвилинний цикл дозволяє системі оперативно відображати нові аналізи. Для задачі виявлення дезінформації, де важлива не лише швидкість, а й глибина аналізу та пояснення, такий інтервал є виправданим. Він дозволяє уникнути надмірного навантаження на API LLM та GitHub Pages, зберігаючи при цьому високу цінність наданої інформації. У майбутньому, для підвищення актуальності, інтервал оновлення може бути скоригований з урахуванням ресурсних обмежень та потреб.

Проведені експериментальні дослідження та аналіз продуктивності підтвердили, що розроблена система є ефективним інструментом для виявлення дезінформації, демонструючи добрі показники швидкості та якості аналізу.

5.4. Обговорення отриманих результатів

Отримані результати функціонування системи дозволяють провести всебічне обговорення її поточного стану, сильних та слабких сторін, а також окреслити перспективи подальшого розвитку.

5.4.1. Сильні та слабкі сторони розробленої системи

Розроблена система виявлення дезінформації на основі великих мовних моделей має низку виражених сильних сторін, але водночас містить і певні обмеження, які є характерними для рішень на сучасному етапі розвитку технологій штучного інтелекту.

Сильні сторони:

1. Висока якість аналізу LLM: Завдяки використанню потужної LLM (Gemini API), система демонструє здатність до глибокого розуміння контексту новинних статей, виявлення тонких лінгвістичних маркерів упередженості, маніпуляцій та необ'єктивності, навіть без попереднього тонкого налаштування на специфічних датасетах дезінформації українською мовою.

2. Пояснюваність (Explainability): Критично важливою перевагою є здатність LLM генерувати деталізовані пояснення до своїх вердиктів. Це значно підвищує довіру користувача до системи та сприяє підвищенню медіаграмотності, оскільки пояснює логіку класифікації.

3. Автоматизований та безперервний моніторинг: Система повністю автоматизована, здійснює безперервний збір та аналіз новин, що дозволяє оперативно реагувати на появу дезінформації у динамічному інформаційному просторі.

4. Економічна ефективність розгортання: Використання безкоштовного хостингу GitHub Pages та вбудованої бази даних SQLite3 робить систему економічно вигідною для розгортання та підтримки на поточному етапі.

5. Гнучкість та адаптивність промпту: Можливість модифікувати та оптимізувати промпт для LLM дозволяє легко адаптувати логіку аналізу до нових типів дезінформації або змінювати критерії оцінки без необхідності перенавчання усієї моделі.

Слабкі сторони:

1. Залежність від зовнішнього API: Система повністю залежить від доступності, стабільності та політики використання Gemini API. Будь-які зміни в лімітах, вартості або функціональності API можуть вплинути на роботу системи.

2. Відсутність кількісної оцінки точності: Через відсутність формального, розміченого тестового набору даних, не було проведено кількісної оцінки метрик точності (precision, recall, F1-score) виявлення дезінформації. Оцінка є переважно якісною.

3. Потенційні хибнопозитивні/хибнонегативні спрацювання: Як обговорювалося в розділі 5.2.2, LLM може робити помилки, особливо з тонкими формами дезінформації, сарказмом, іронією або високоемоційним, але достовірним контентом.

4. Обмеження SQLite3 для великих масштабувань: Хоча SQLite3 є оптимальним для поточного проєкту, при значному збільшенні обсягів даних або кількості джерел може виникнути потреба у більш потужній та масштабованій базі даних.

5. Відсутність мультимодального аналізу: Поточна версія системи зосереджена лише на текстовому аналізі. Однак сучасна дезінформація часто включає зображення, відео та аудіо, що не аналізується поточною системою.

5.4.2. Порівняння з існуючими рішеннями

Порівняння розробленої системи з існуючими комерційними або відкритими платформами (які були розглянуті в Розділі 3.1.3) та іншими підходами (описаними в Розділі 3.2.1) виявляє її унікальне позиціонування.

Переваги над традиційними ML-методами (SVM, Naive Bayes, Random Forest): розроблена система значно перевершує традиційні підходи в частині глибини розуміння контексту та семантики, а також у відсутності необхідності ручної інженерії ознак. LLM автоматично вивчає складні патерни, що робить систему більш стійкою до еволюції дезінформації та гнучкішою. Традиційні методи були б менш ефективними для аналізу нюансів мови та надання розгорнутих пояснень.

Порівняння з іншими архітектурами глибокого навчання (RNN, CNN): Хоча RNN та CNN були важливим кроком у розвитку NLP, трансформерна архітектура Gemini дозволяє досягти кращого захоплення довгострокових залежностей та вищої паралелізації обчислень. Це призводить до більш швидкого та якісного аналізу. Крім того, саме архітектура LLM дозволяє ефективно генерувати пояснення, що не є типовою функцією для простих RNN/CNN класифікаторів.

Порівняння з комерційними та відкритими платформами (наприклад, StopFake, Snopes, PolitiFact): Більшість існуючих фактчекінгових платформ значною мірою покладаються на ручну експертизу або комбінацію з

автоматизованими інструментами, які часто використовують простіші ML-моделі для первинної фільтрації.

Перевага системи

Основна перевага полягає в автоматизації глибокого якісного аналізу та генерації пояснень на базі LLM. Це дозволяє обробляти значно більші обсяги інформації, ніж це можливо лише за допомогою людських ресурсів. Система може виступати як перший рівень фільтрації, що значно підвищує ефективність роботи фактчекерів, дозволяючи їм зосередитися на найбільш складних та спірних випадках.

Обмеження порівняно з існуючими платформами: Існуючі платформи мають доступ до зовнішніх баз знань, архівних даних та мереж експертів, чого не має наша система. Наша система не замінює, а доповнює роботу фактчекінгових організацій, надаючи інструмент для автоматизованого первинного аналізу.

Якісно, розроблена система є передовим рішенням завдяки інтеграції можливостей LLM для пояснюваного та глибокого аналізу дезінформації, що вирізняє її серед багатьох існуючих автоматизованих підходів.

5.4.3. Обмеження поточного рішення та потенційні напрямки для покращення

Незважаючи на продемонстровану ефективність, поточне рішення має певні обмеження, які також вказують на потенційні напрямки для подальшого вдосконалення та розвитку.

Обмеження поточного рішення:

1. Відсутність формальної оцінки точності: Головним обмеженням є відсутність кількісної оцінки точності (Precision, Recall, F1-score) на великому, розміченому тестовому наборі даних. Це не дозволяє об'єктивно порівняти ефективність системи з іншими рішеннями за стандартними метриками.

2. Залежність від промпту: Якість аналізу LLM сильно залежить від формулювання промпту. Хоча промпт був оптимізований, завжди існує потенціал для його подальшого вдосконалення.
3. Мономодальність: Система аналізує лише текстовий контент. Сучасна дезінформація часто є мультимодальною (текст + зображення/відео), що не враховується поточним рішенням.
4. Динамічність джерел: Зміни у верстці новинних сайтів можуть вимагати періодичного оновлення правил веб-скрапінгу.
5. Відсутність "глобального" знання: LLM, хоч і має обширні знання з попереднього навчання, не має доступу до актуальної інформації "в реальному часі" поза текстом, що аналізується, або до специфічних фактчекінгових баз даних для перевірки фактів. Вона покладається на свої "внутрішні" знання та здатність до логічного висновку.
6. Мовні нюанси та діалекти: Хоча LLM підтримує українську мову, для дуже специфічних діалектів або регіональних нюансів дезінформації її ефективність може бути нижчою.

Потенційні напрямки для покращення:

1. Створення та використання розміченого датасету: Розробка або збір великого, якісного розміченого датасету українських новин з мітками дезінформації дозволить провести формальну кількісну оцінку ефективності системи, а також потенційно тонко налаштувати LLM для українського контексту.
2. Вдосконалення промпту та методів промпт-інженерінгу: Постійна оптимізація промпту, експерименти з різними стилями та структурами інструкцій для LLM, а також використання технік, таких як Chain-of-Thought prompting, для підвищення глибини аналізу.
3. Інтеграція мультимодального аналізу: Розширення системи для аналізу зображень, відео та аудіо, пов'язаних з новинним контентом, що дозволить виявляти дезінформацію у візуальному та аудіоформаті. Це

може включати використання спеціалізованих моделей для аналізу зображень (наприклад, для виявлення ознак фотошопу або маніпуляцій).

4. Розширення джерел та адаптивність скрапінгу: Додавання більшої кількості новинних джерел та розробка більш гнучких, можливо, ШІ-орієнтованих, скраперів, які автоматично адаптуються до змін у верстці сайтів.
5. Інтеграція з базами фактів: Підключення до зовнішніх фактчекінгових баз даних або знань для підвищення точності перевірки фактів.
6. Розробка користувацьких фільтрів та сповіщень: Додавання функціоналу для користувачів, щоб вони могли фільтрувати новини за рівнем ризику, джерелом, тематикою, а також налаштовувати сповіщення про нову дезінформацію.
7. Оптимізація бази даних: У разі значного зростання обсягів даних, міграція з SQLite3 на більш масштабовані бази даних (наприклад, PostgreSQL, MongoDB, або розподілені NoSQL рішення) для забезпечення довгострокової продуктивності.

Врахування цих напрямків дозволить суттєво покращити функціональність, точність та масштабованість розробленої системи у майбутньому.

ВИСНОВКИ

У рамках даної дипломної роботи було успішно розроблено та реалізовано систему виявлення дезінформації в новинах на основі нейронних мереж, зокрема, з використанням великої мовної моделі Gemini.

Протягом роботи було послідовно виконано низку ключових етапів, що забезпечили побудову функціонального рішення. Розроблена модульна архітектура системи, що включає компоненти для збору даних, їх аналізу LLM, зберігання інформації та візуалізації, демонструє гнучкість та розширюваність. Реалізований модуль збору даних ефективно комбінує парсинг RSS-стрічок та веб-скрапінг повного тексту статей з популярних українських новинних сайтів, забезпечуючи актуальний потік інформації. Центральним елементом системи став модуль аналізу дезінформації на основі Gemini API, який, використовуючи ретельно розроблений промпт англійською мовою з вимогою української відповіді, показав високу якість якісного аналізу. Це включає ідентифікацію необ'єктивності та маніпуляцій, а також генерацію зрозумілих пояснень, що значно підвищує цінність отриманих результатів. Для надійного зберігання структурованої інформації про новини та результатів їхнього аналізу було створено модуль зберігання даних на базі SQLite3, використовуючи ефективну реляційну схему. Розроблений веб-інтерфейс на GitHub Pages дозволяє візуалізувати результати аналізу в майже реальному часі завдяки автоматизованому експорту даних у JSON та синхронізації через Git. Експериментальні дослідження підтвердили оперативність системи у зборі (до 50-70 новин за годину) та аналізі даних (близько 2 секунд на запит LLM), а також забезпечили актуальність інформації на сайті з затримкою до 20-25 хвилин.

Таким чином, поставлена мета – розробка системи виявлення дезінформації в новинах на основі нейронних мереж – була успішно досягнута, а всі сформульовані на початковому етапі завдання виконані. Система

функціонує згідно із задекларованою архітектурою та демонструє практичну здатність виявляти та пояснювати ознаки дезінформації. Наукова новизна роботи полягає у розробці інтегрованого підходу, який поєднує автоматизований збір даних та можливості сучасної великої мовної моделі для якісного, пояснюваного аналізу, що є важливим внеском у сферу пояснюваного штучного інтелекту (XAI) в контексті боротьби з дезінформацією. Практична цінність системи є значною, оскільки вона може слугувати ефективним інструментом для підвищення медіаграмотності звичайних користувачів, допомагати фактчекінговим організаціям як перший рівень фільтрації та сприяти швидкому реагуванню на поширення дезінформації.

Розроблена система є функціональною основою, яка має значний потенціал для подальшого вдосконалення. Перспективи розвитку включають розширення джерел даних та інтеграцію з платформами соціальних мереж, додавання мультимодального аналізу (зображень, відео, аудіо), тонке налаштування LLM на специфічних розмічених датасетах української дезінформації, а також вдосконалення методів промпт-інженерінгу. Крім того, подальші напрямки охоплюють розробку механізму зворотного зв'язку для користувачів, впровадження нових метрик для формальної кількісної оцінки ефективності, покращення інтерфейсу та користувацького досвіду, а також оптимізацію бази даних для масштабування у разі значного зростання обсягів даних. Реалізація цих напрямків дозволить створити ще більш потужний та всеосяжний інструмент для боротьби з дезінформацією в сучасному інформаційному просторі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python Software Foundation. Python Language Reference, version 3.12 [Електронний ресурс]. — Режим доступу: https://www.python.org/ — Дата звернення: 08.06.2025.
2. Richardson L. Beautiful Soup Documentation [Електронний ресурс]. — Режим доступу: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ — Дата звернення: 08.06.2025.
3. SQLite Consortium. SQLite Documentation [Електронний ресурс]. — Режим доступу: https://www.sqlite.org/docs.html — Дата звернення: 08.06.2025.
4. Google. Gemini API Documentation [Електронний ресурс]. — Режим доступу: https://ai.google.dev/gemini-api/docs — Дата звернення: 08.06.2025.
5. Github Docs. Managing your repositories [Електронний ресурс]. — Режим доступу: https://docs.github.com/en/repositories — Дата звернення: 08.06.2025.
6. RSS Advisory Board. RSS 2.0 Specification [Електронний ресурс]. — Режим доступу: https://www.rssboard.org/rss-specification — Дата звернення: 08.06.2025.
7. JSON.org. Introducing JSON [Електронний ресурс]. — Режим доступу: https://www.json.org/json-en.html — Дата звернення: 08.06.2025.
8. Diagrams.net. Інструмент для побудови блок-схем [Електронний ресурс]. — Режим доступу: https://www.diagrams.net/ — Дата звернення: 08.06.2025.
9. PEP 8 – Style Guide for Python Code [Електронний ресурс]. — Режим доступу: https://peps.python.org/pep-0008/ — Дата звернення: 08.06.2025.
10. Van Rossum G., Warsaw B., Coghlan N. PEP 257 – Docstring Conventions [Електронний ресурс]. — Режим доступу: https://peps.python.org/pep-0257/ — Дата звернення: 08.06.2025.

ДОДАТКИ

ДОДАТОК А

МОДУЛЬ ПАРСИНГУ RSS-СТРІЧОК

```
import feedparser

def parse_rss_feeds(rss_urls): # функція для обробки rss-стрічок
    all_news = [] # список для збереження новин

    for url in rss_urls:
        feed = feedparser.parse(url) # парсинг стрічки з перебором
     url адрес
        source_title = feed.feed.get('title', 'Невідоме джерело')
     # отримання назви джерела

        count = 0
        for entry in feed.entries:
            if count >= 2: # тимчасове обмеження до 2 новин на
     джерело щоб не парсити повністю весь сайт
                break
            news_item = { # формування словника новини
                'source': source_title,
                'title': entry.get('title', 'Без заголовка'),
                'published': entry.get('published', 'Без дати'),
                'link': entry.get('link', 'Без посилання'),
                'summary': entry.get('summary', 'Без опису')
            }
            all_news.append(news_item) # додавання новини до
     списку
            count += 1
    return all_news # повернення всіх зібраних новин
```

ДОДАТОК Б

МОДУЛЬ ВИЛУЧЕННЯ ПОВНОГО ТЕКСТУ СТАТТІ

```
import requests # для http-запитів
from readability import Document # для вилучення основного
     контенту
from bs4 import BeautifulSoup # для парсингу html
import re # для регулярних виразів

def get_clean_article_text(url): # функція для отримання чистого
     тексту статті
    try:
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64;
     x64)' # імітація браузера

        }
        response = requests.get(url, headers=headers, timeout=10)
     # виконання get-запиту
        response.raise_for_status() # перевірка статусу відповіді

        # основний контент сторінки
        doc = Document(response.text)
        summary_html = doc.summary() # вилучення основного html-
     контенту

        # парс текста з HTML
        soup = BeautifulSoup(summary_html, 'html.parser')
        text = soup.get_text(separator=' ', strip=True)

        # чистка тексту подвійних пробілів і переносів
        clean_text = re.sub(r'\s+', ' ', text)

        return clean_text # повернення очищеного тексту

    except Exception as e: # обробка можливих винятків
        print(f"Помилка при завантаженні {url}: {e}")
        return "Не вдалося отримати повний текст."
```

ДОДАТОК В

МОДУЛЬ АНАЛІЗУ КОНТЕНТУ ЗА ДОПОМОГОЮ AI GEMINI API

```
import base64
import os # для доступу до змінної середовища
from google import genai
from google.genai import types

# промпт

FULL_PROMPT_TEMPLATE = """
You are an expert media analyst.
Your task is to analyze the following news article and provide a
     structured JSON output.

Input Article:
Title: "{title}"
Full Text: "{full_text}"
Publication Date: "{publication_date}"


Your analysis should include:
1.  **Topic Identification**: Determine the main topic of the news
     article (one word or a very short phrase).
2.  **Sentiment Analysis**: Determine the overall sentiment/tone
     of the news article (positive, negative, or neutral).
3.  **Critical Content Analysis**: Analyze the article for the
     presence of:
    * **Clickbait headline**: Determine if the headline is
     designed to exploit curiosity or sensationalism to attract
     clicks, rather than accurately reflect the content.
    * **Manipulative statements**: Identify any loaded language,
     generalizations, emotional appeals, or other rhetorical
     devices used to influence opinion.
    * **False claims (misinformation/disinformation)**: Detect any
     claims that are factually incorrect or misleading.
4.  **Overall Quality & Style Assessment**: Evaluate the article's
     writing quality and adherence to journalistic standards,
     specifically:
    * **Is it well-written?**: Assess its clarity, grammar, and
     overall readability.
    * **Is it objective?**: Determine if it maintains journalistic
     objectivity, presenting facts fairly without strong personal
     opinions or obvious bias.
    * **Style notes**: Briefly describe the general writing style
     (e.g., formal, informal, sensational, neutral).

Output Guidelines:
-   **JSON Format**: The output must be a valid JSON object
     strictly adhering to the provided JSON schema.
```

```
-   **Explanations**: For clickbait, manipulation, and false
     claims, if 'is_present' is true, provide a very brief
     explanation (1-2 short phrases) of *what* is present and
     *why* you think so. If false, leave the explanation empty.
-   **Language of Output**: Provide all string values in the JSON
     output, including explanations and summaries, **strictly in
     Ukrainian**.
-   **Conciseness**: Keep all descriptions, explanations, and
     summaries concise and to the point.
-   **Clarity**: Use clear, straightforward language.

Final summary:
-   **Overall Summary Explanation**: Provide a concise summary of
     all analysis findings (one to two short sentences). This
     should also explicitly mention *why* manipulation or false
     claims were identified, if applicable.
"""

def prepare_full_input_text(title, full_text, pub_date):
    """
    Формує повний текст запиту для Gemini, поєднуючи промпт та
     дані новини
    """
    return FULL_PROMPT_TEMPLATE.format(
        title=title,
        full_text=full_text,
        publication_date=pub_date
    )

def generate_analysis(news_title, news_full_text,
     news_publication_date):
    """
    Основна функція для генерації аналізу новини за допомогою
     Gemini API
    """
    client = genai.Client(
        api_key=os.environ.get("GEMINI_API_KEY"), # отримує api
     ключ з змінних середовища для безпеки
    )

    model_name = "gemini-2.0-flash-lite" # визначає назву моделі
     gemini для використання


    # комбінує промпт та вхідні дані статті
    full_request_text = prepare_full_input_text(
        news_title, news_full_text, news_publication_date
    )

    contents = [
        types.Content(
            role="user",
```

```
            parts=[
                types.Part.from_text(text=full_request_text),
            ],
        ),
    ]
    # конфігурація для генерації контенту, включаючи json-схему
     для вивод
    generate_content_config = types.GenerateContentConfig(
        response_mime_type="application/json", # вказує, що
     відповідь має бути в json форматі
        response_schema=genai.types.Schema(
        type=genai.types.Type.OBJECT,
        required=[ # обов'язкові поля в json-відповіді
            "topic",
            "sentiment",
            "clickbait",
            "manipulation",
            "false_claims",
            "quality_analysis",
            "overall_summary",
        ],
        properties={ # визначення властивостей json-схеми
            "topic": genai.types.Schema(
                type=genai.types.Type.STRING,
                description="Main topic of the news article (one
     word or a very short phrase).",
            ),
            "sentiment": genai.types.Schema(
                type=genai.types.Type.STRING,
                description="Overall sentiment/tone of the news
     article.",
                enum=["positive", "negative", "neutral"], #
     дозволені значення для сентименту
            ),
            "clickbait": genai.types.Schema(
                type=genai.types.Type.OBJECT,
                required=["is_present"],
                properties={
                    "is_present": genai.types.Schema(
                        type=genai.types.Type.BOOLEAN,
                        description="True if the headline is
     clickbait, false otherwise.",
                    ),
                    "explanation": genai.types.Schema(
                        type=genai.types.Type.STRING,
                        description="Brief explanation if
     clickbait is present, otherwise an empty string.",
                        default="",
                    ),
                },
            ),
            "manipulation": genai.types.Schema(
```

```
            type=genai.types.Type.OBJECT,
            required=["is_present"],
            properties={
                "is_present": genai.types.Schema(
                    type=genai.types.Type.BOOLEAN,
                    description="True if manipulative
statements or emotional manipulation are present, false
otherwise.",
                ),
                "explanation": genai.types.Schema(
                    type=genai.types.Type.STRING,
                    description="Brief explanation if
manipulation is present, otherwise an empty string.",
                    default="",
                ),
            },
        ),
        "false_claims": genai.types.Schema(
            type=genai.types.Type.OBJECT,
            required=["is_present"],
            properties={
                "is_present": genai.types.Schema(
                    type=genai.types.Type.BOOLEAN,
                    description="True if false claims are
present, false otherwise.",
                ),
                "explanation": genai.types.Schema(
                    type=genai.types.Type.STRING,
                    description="Brief explanation if false
claims are present, otherwise an empty string.",
                    default="",
                ),
            },
        ),
        "quality_analysis": genai.types.Schema(
            type=genai.types.Type.OBJECT,
            required=["is_well_written", "is_objective"],
            properties={
                "is_well_written": genai.types.Schema(
                    type=genai.types.Type.BOOLEAN,
                    description="True if the article is well-
written, clear, and grammatically correct.",
                ),
                "is_objective": genai.types.Schema(
                    type=genai.types.Type.BOOLEAN,
                    description="True if the article maintains
journalistic objectivity and avoids strong personal
opinions.",
                ),
                "style_notes": genai.types.Schema(
                    type=genai.types.Type.STRING,
```

```
                        description="Brief notes on the writing
 style (e.g., formal, informal, sensational, neutral). Empty
 if not applicable.",
                        default="",
                    ),
                },
                description="Assessment of the article's writing
 quality, clarity, and objectivity based on journalistic
 standards.",
            ),
            "overall_summary": genai.types.Schema(
                type=genai.types.Type.STRING,
                description="A concise summary of the analysis
 findings (one to two short sentences), including brief
 explanations if manipulation or false claims were
 identified.",
            ),
        },
        ),
    )

    print(f"Відправлення запиту до моделі: {model_name}...")
    result_text = ""
    try:
        # отримує відповідь від моделі частинами
        for chunk in client.models.generate_content_stream(
            model=model_name,
            contents=contents,
            config=generate_content_config,
        ):
            result_text += chunk.text or "" # додає кожну частину
 до загального тексту.

    except Exception as e:
        result_text = f"Сталася помилка при генерації контенту:
 {e}"

    return result_text # повертає згенерований текст аналізу

def get_overall_risk_level(analysis_result: dict) -> str:
    """
    Визначає загальний рівень ризику або достовірності новини на
     основі результатів аналізу від Gemini
    """
    if analysis_result.get("false_claims_present") == 1:
        return "Фейк / Дезінформація"
    elif analysis_result.get("manipulation_present") == 1:
        return "Маніпуляція / Упередженість"
    elif analysis_result.get("clickbait_present") == 1:
        return "Потребує перевірки (Клікбейт)"
    elif analysis_result.get("is_objective") == 0:
        return "Потребує перевірки (Необ'єктивно)"
```

```
    else:
        return "Підтверджено"
```

ДОДАТОК Г

МОДУЛЬ УПРАВЛІННЯ БАЗОЮ ДАНИХ

```
from analyzer.gemini_api import generate_analysis
from analyzer.gemini_api import get_overall_risk_level
import sqlite3
import json
import datetime

def create_table(db_path='news.db'):
    conn = sqlite3.connect(db_path)
    c = conn.cursor()
    # створення таблиці news_analysis, якщо вона ще не існує
    c.execute('''
        CREATE TABLE IF NOT EXISTS news_analysis (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            title TEXT NOT NULL UNIQUE,
            link TEXT,
            source TEXT,
            published DATETIME NOT NULL,
            topic TEXT,
            sentiment TEXT,
            clickbait_present BOOLEAN,
            clickbait_explanation TEXT,
            manipulation_present BOOLEAN,
            manipulation_explanation TEXT,
            false_claims_present BOOLEAN,
            false_claims_explanation TEXT,
            is_well_written BOOLEAN,
            is_objective BOOLEAN,
            style_notes TEXT,
            overall_summary TEXT,
            risk_level TEXT,
            created_at DATETIME DEFAULT CURRENT_TIMESTAMP
        );
    ''')
    conn.commit()
    conn.close()

def save_analysis(news_item: dict, analysis_result: dict,
     db_path='news.db'):
    """
    Зберігає результат аналізу новини в базу даних.
    news_item: оригінальний словник з новиною (звідки беремо title,
     published, link, source)
    analysis_result: словник з результатом аналізу від Gemini
    """
    conn = sqlite3.connect(db_path)
    c = conn.cursor()

    # розрахунок загального рівня ризику на основі результатів аналізу
```

```
risk_level = get_overall_risk_level({
    "clickbait_present": analysis_result.get('clickbait',
 {}).get('is_present'),
    "manipulation_present": analysis_result.get('manipulation',
 {}).get('is_present'),
    "false_claims_present": analysis_result.get('false_claims',
 {}).get('is_present'),
    "is_objective": analysis_result.get('quality_analysis',
 {}).get('is_objective')
})

# вставка або ігнорування запису, якщо новина з таким заголовком
 вже існує (unique constraint)
try:
    c.execute('''
        INSERT OR IGNORE INTO news_analysis (
            title, published, link, source, topic, sentiment,
            clickbait_present, clickbait_explanation,
            manipulation_present, manipulation_explanation,
            false_claims_present, false_claims_explanation,
            is_well_written, is_objective, style_notes,
            overall_summary, risk_level, created_at
        ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
 ?, ?)
    ''', (
        news_item.get('title'),
        news_item.get('published'),
        news_item.get('link'),
        news_item.get('source'),
        analysis_result.get('topic'),
        analysis_result.get('sentiment'),
        analysis_result.get('clickbait', {}).get('is_present'),
        analysis_result.get('clickbait', {}).get('explanation'),
        analysis_result.get('manipulation', {}).get('is_present'),
        analysis_result.get('manipulation', {}).get('explanation'),
        analysis_result.get('false_claims', {}).get('is_present'),
        analysis_result.get('false_claims', {}).get('explanation'),
        analysis_result.get('quality_analysis',
 {}).get('is_well_written'),
        analysis_result.get('quality_analysis',
 {}).get('is_objective'),
        analysis_result.get('quality_analysis',
 {}).get('style_notes'),
        analysis_result.get('overall_summary'),
        risk_level,
        datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    ))
    conn.commit()
    print(f"Аналіз для новини '{news_item.get('title')}' успішно
 збережено в БД з рівнем ризику: {risk_level}")
except sqlite3.IntegrityError as e:
    if "UNIQUE constraint failed" in str(e):
```

```
            print(f"Новина '{news_item.get('title')}' вже існує в базі
    даних. Пропускаємо.")
        else:
            print(f"Помилка цілісності при збереженні в БД: {e}")
    except Exception as e:
        print(f"Невідома помилка при збереженні аналізу в БД: {e}")
    finally:
        conn.close()

def news_exists(title, db_path='news.db'):
    conn = sqlite3.connect(db_path)
    c = conn.cursor()
    # перевірка, чи існує новина з заданим заголовком у базі даних
    c.execute('SELECT id FROM news_analysis WHERE title = ?', (title,))
    result = c.fetchone()
    conn.close()
    return result is not None

def export_news_to_json(db_path, json_path):
    try:
        conn = sqlite3.connect(db_path)
        conn.row_factory = sqlite3.Row # дозволяє отримувати доступ до
    стовпців за назвою
        c = conn.cursor()

        c.execute('SELECT * FROM news_analysis')
        rows = c.fetchall()

        news_data_for_json = []
        for row in rows:
            row_dict = dict(row) # перетворюємо Row-об'єкт на словник
            # збір даних для json-експорту, з обробкою відсутніх
    значень
            news_data_for_json.append({
                "id": row_dict.get("id"),
                "title": row_dict.get("title"),
                "link": row_dict.get("link", "#"),
                "source": row_dict.get("source", "Невідоме джерело"),
                "published": row_dict.get("published"),
                "topic": row_dict.get("topic"),
                "sentiment": row_dict.get("sentiment"),
                "clickbait_present": row_dict.get("clickbait_present"),
                "clickbait_explanation":
    row_dict.get("clickbait_explanation"),
                "manipulation_present":
    row_dict.get("manipulation_present"),
                "manipulation_explanation":
    row_dict.get("manipulation_explanation"),
                "false_claims_present":
    row_dict.get("false_claims_present"),
                "false_claims_explanation":
    row_dict.get("false_claims_explanation"),
```

```
                "is_well_written": row_dict.get("is_well_written"),
                "is_objective": row_dict.get("is_objective"),
                "style_notes": row_dict.get("style_notes"),
                "overall_summary": row_dict.get("overall_summary"),
                "risk_level": row_dict.get("risk_level"),
                "created_at": row_dict.get("created_at")
            })
        # запис зібраних даних у json-файл
        with open(json_path, 'w', encoding='utf-8') as json_file:
            json.dump(news_data_for_json, json_file,
 ensure_ascii=False, indent=4)

        print(f"Дані успішно експортовано у {json_path}")

    except Exception as e:
        print(f"Помилка при експорті: {e}")

    finally:
        if conn:
            conn.close()
```

ДОДАТОК Д

МОДУЛЬ АВТОМАТИЧНОЇ СИНХРОНІЗАЦІЇ З GIT

```
import subprocess # для виконання зовнішніх команд

def push_to_git(repo_dir, file_path_rel, commit_message="Auto Push
     JSON"):
    """
    repo_dir — шлях до кореневої папки гіт-репозиторію
    file_path_rel — відносний шлях до файлу відносно repo_dir
    commit_message - коміт повідомлення
    """

    try:
        print("Performing git pull to synchronize with remote...")
        result_pull = subprocess.run(
            ['git', 'pull', 'origin', 'main'],
            cwd=repo_dir,
            text=True,
            capture_output=True
        )
        print("git pull output:\n", result_pull.stdout)
        if result_pull.stderr:
            print("git pull errors/warnings:\n", result_pull.stderr)

        # перевірка успішності pull або виявлення конфліктів
        if result_pull.returncode != 0 and "Already up to date." not in
     result_pull.stdout and "Fast-forward" not in result_pull.stdout:
            print("Помилка під час git pull. Можливо, є конфлікти
     об'єднання або інші проблеми.")
            print("Будь ласка, перевірте вивід вище та вирішіть їх
     вручну перед повторним запуском.")
            return

        # Git add
        print("git add output:")
        result_add = subprocess.run(
            ['git', 'add', file_path_rel],
            cwd=repo_dir,
            text=True,
            capture_output=True,
            check=True # викликає виняток, якщо команда повертає
     ненульовий код
        )
        print(result_add.stdout, result_add.stderr)

        # Git commit
        result_commit = subprocess.run(
            ['git', 'commit', '-m', commit_message],
            cwd=repo_dir,
            text=True,
```

```
            capture_output=True
        )

        if result_commit.returncode == 0:
            print("git commit output:", result_commit.stdout)
        elif "nothing to commit" in result_commit.stderr.lower() or
 "nothing to commit" in result_commit.stdout.lower():
            print("Немає нових змін для коміту.")
            return # якщо немає змін, немає сенсу пушити
        else:
            print("git commit error:", result_commit.stderr)
            return

        # Git push
        result_push = subprocess.run(
            ['git', 'push', 'origin', 'main'],
            cwd=repo_dir,
            text=True,
            capture_output=True,
            check=True
        )
        print("git push output:", result_push.stdout)

        print(f"Файл '{file_path_rel}' успішно відправлено в
 репозиторій.")

    except subprocess.CalledProcessError as e: # обробка помилок
 виконання git-команд
        print(f"Помилка при виконанні git-команди: {e}")
        print("STDOUT:", e.stdout)
        print("STDERR:", e.stderr)
    except FileNotFoundError: # обробка випадку, коли git не знайдено
        print("Помилка: Команда 'git' не знайдена. Переконайтеся, що
 Git встановлений і доданий до PATH.")
    except Exception as e: # обробка будь-яких інших непередбачених
 помилок
        print(f"Виникла неочікувана помилка: {e}")
```

ДОДАТОК Е

ГОЛОВНИЙ МОДУЛЬ УПРАВЛІННЯ ТА КООРДИНАЦІЇ

```
import time
from parsers.rss_parsers import parse_rss_feeds
from parsers.full_text_parsers import get_clean_article_text
from analyzer.gemini_api import generate_analysis
from data_base.data_base import save_analysis
from data_base.data_base import news_exists
from data_base.data_base import create_table
from json_push.push import push_to_git
from data_base.data_base import export_news_to_json
import json
import logging

# налаштування логування
logging.basicConfig(level=logging.INFO, format='%(asctime)s -
    %(levelname)s - %(message)s')
# визначення шляхів до репозиторію, файлів та бд
repo_dir = r'C:\Users\maks3\source\repos\news_checker'
file_path_rel = 'data/news.json'
db_path = r'C:\Users\maks3\source\repos\news_checker\news.db'
json_path = r'C:\Users\maks3\source\repos\news_checker\data\news.json'

# вся логіка в одній функцію
def run_news_checker(): # головна функція для запуску перевірки новин
    logging.info("Початок циклу перевірки новин...")
    rss_urls = [
        'https://www.pravda.com.ua/rss/',
        'https://tsn.ua/rss',
        'https://dailycaller.com/feed/'
    ]
    create_table(db_path)
    news_items = parse_rss_feeds(rss_urls)

    processed_count = 0
    for news_item in news_items:
        logging.info("="*80)
        logging.info(f"Джерело: {news_item['source']}")
        logging.info(f"Заголовок: {news_item['title']}")
        logging.info(f"Дата: {news_item['published']}")
        logging.info(f"Посилання: {news_item['link']}")
        # logging.info(f"Опис: {news_item['summary']}") # мусорить

        if news_exists(news_item['title'], db_path): # перевірка, чи
     новина вже оброблена
            logging.info("Новина вже оброблена, пропускаємо...")
            continue

        full_text = get_clean_article_text(news_item['link']) #
     отримання повного тексту статті
```

```
    if not full_text: # перевірка якщо повний текст не вдалося
 отримати
        logging.warning(f"Не вдалося отримати повний текст для:
 {news_item['link']}. Пропускаємо аналіз.")
        continue

    try:
        # генерація аналізу за допомогою gemini api
        gemini_response_json_string = generate_analysis(
            news_item['title'],
            full_text,
            news_item['published']
        )
        gemini_analysis_dict =
 json.loads(gemini_response_json_string) # парсинг json-відповіді
    except (json.JSONDecodeError, TypeError) as e: # обробка
 помилок парсингу json
        logging.error(f"Помилка парсингу JSON або генерації аналізу
 від Gemini для '{news_item['title']}': {e}")
        logging.error(f"Отриманий рядок, що викликав помилку:
 {gemini_response_json_string}")
        continue
    except Exception as e: # загальний виняток для проблем з Gemini
 API
        logging.error(f"Невідома помилка під час аналізу Gemini для
 '{news_item['title']}': {e}")
        continue

    save_analysis(news_item, gemini_analysis_dict, db_path) #
 збереження результатів аналізу в бд
    processed_count += 1
    logging.info(f"Новина '{news_item['title']}' успішно оброблена
 та збережена.")

logging.info(f"Завершено обробку новин. Оброблено {processed_count}
 нових новин.")

# експорт та Git push після обробки всіх нових новин
if processed_count > 0: # пуш тільки якщо були якісь зміни
    logging.info("Експорт новин в JSON...")
    export_news_to_json(db_path, json_path)
    logging.info("JSON експортовано.")

    logging.info("Відправлення змін на Git...")

    push_to_git(repo_dir, file_path_rel, "Auto: New news analysis
 pushed")
    logging.info("Зміни відправлено на Git.")
else:
    logging.info("Нових новин не знайдено для обробки. Git push не
 потрібен.")
```

```
    logging.info("Цикл перевірки новин завершено.")


if __name__ == '__main__':

    interval_seconds = 1200 # скріпт буде працювати раз в 20 хв

    while True: # безкінечний цикл для постійної роботи скрипта
        try:
            run_news_checker() # запуск основної логіки
        except Exception as e: # обробка критичних помилок головного
 циклу
            logging.critical(f"Критична помилка у головному циклі:
 {e}", exc_info=True)

        logging.info(f"Очікування {interval_seconds / 60} хвилин до
 наступної перевірки...")
        time.sleep(interval_seconds) # пауза перед наступним запуском
```

ДОДАТОК Є

ВЕБ ІНТЕРФЕЙС HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="css/style.css">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
     scale=1.0">
    <title>FactWatch - News Monitoring & Analysis</title>
</head>
<body>
    <div class="container">
        <header class="header">
            <h1>FactWatch</h1>
            <p>Моніторинг новин і аналіз дезінформації в реальному
     часі</p>
        </header>

        <aside class="sidebar">
            <h3>Новинні сайти:</h3>
            <ul class="nav-list">
                <li><a href="index.html">Українська
     правда</a></li>
                <li><a href="tsn.html">Новини на tsn.ua</a></li>
                <li><a href="dailycaller.html">The Daily
     Caller</a></li>
            </ul>
        </aside>

        <main class="main">
            <div class="content-header">
                <h2>Аналіз останніх новин:</h2>
            </div>

            <div class="news-grid">
                </div>
        </main>
    </div>

    <script src="js/script.js"></script>
    </body>
</html>
```

ДОДАТОК Ж

ГЛОБАЛЬНІ СТИЛІ ІНТЕРФЕЙСУ

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family:   'Inter',   -apple-system,   BlinkMacSystemFont,
     'Segoe UI', Roboto, sans-serif;
    line-height: 1.6;
    color: #E0E2E5;
    background-color: #071e2f;
}

.container {
    max-width: 1400px;
    margin: 0 auto;
    min-height: 100vh;
    display: grid;
    grid-template-columns: 1fr 280px;
    grid-template-rows: auto 1fr;
    grid-template-areas:
        "header header"
        "main sidebar";
    gap: 0;
}

/* Header */
.header {
    grid-area: header;
    background: linear-gradient(135deg, #1a365d 0%, #2d5a87 100%);
    color: white;
    padding: 1.5rem 2rem;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

    .header h1 {
        font-size: 2.5rem;
        font-weight: 700;
        margin-bottom: 0.25rem;
    }

    .header p {
        font-size: 1.25rem;
        opacity: 0.9;
        font-weight: 300;
    }
```

```
/* Sidebar */
.sidebar {
    grid-area: sidebar;
    background: #1a365d;
    padding: 2rem 1.5rem;
    /*border-left: 1px solid #2d5a87;  deleted  */
    box-shadow: -2px 0 4px rgba(0,0,0,0.2);
}

    .sidebar h3 {
        font-size: 1.1rem;
        font-weight: 600;
        color: #CBD5E0;
        margin-bottom: 1.5rem;
        padding-bottom: 0.5rem;
        border-bottom: 2px solid #2d5a87;
    }

.nav-list {
    list-style: none;
}

    .nav-list li {
        margin-bottom: 0.75rem;
    }


    .nav-list a {
        color: #E0E2E5;
        text-decoration: none;
        font-size: 0.9rem;
        padding: 0.5rem 0.75rem;
        border-radius: 6px;
        display: block;
        transition: all 0.2s ease;
    }

        .nav-list a:hover {
            background-color: #2d5a87;
            color: #FFFFFF;
            transform: translateX(4px);
        }

        .nav-list a.active {
            background-color: #4A5568;
            color: white;
        }

/* Main Content */
.main {
    grid-area: main;
    padding: 2rem;
```

```
    background-color: #0d2d44;
}

.content-header {
    margin-bottom: 2rem;
}

    .content-header h2 {
        font-size: 1.5rem;
        color: #FFFFFF;
        margin-bottom: 0.5rem;
        font-weight: 600;
    }

    .content-header p {
        color: #CBD5E0;
        font-size: 0.95rem;
    }

.news-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(350px, 1fr));
    gap: 1.5rem;
}

.news-card {
    background: #1a365d;
    border-radius: 12px;
    padding: 1.5rem;
    box-shadow: 0 4px 6px rgba(0,0,0,0.15);
    /*border: 1px solid #2d5a87; deleted*/
    transition: all 0.3s ease;
}

    .news-card:hover {
        transform: translateY(-3px); /* ,, */
        box-shadow: 0 8px 25px rgba(0,0,0,0.3);
    }

/* дублікат */
/* .nav-list a:hover {
    text-decoration: underline;
    color: #f0f0f0;
} */

/* дублікат */
/* .news-card:hover {
    transform: translateY(-30px);
    box-shadow: 0 8px 25px rgba(0,0,0,0.3);
} */

.source-name {
```

```
    font-size: 1.1rem;
    font-weight: 700;
    color: #FFFFFF;
    margin-bottom: 1rem;
    padding-bottom: 0.5rem;
    border-bottom: 2px solid #e2e8f0;
}

.headline {
    font-size: 1rem;
    font-weight: 600;
    color: #E0E2E5;
    margin-bottom: 0.75rem;
    line-height: 1.4;
}


    .headline a {
        color: #E0E2E5;
        text-decoration: underline;/*додав підкреслення */
    }

        .headline a:hover {
            color: #FFFFFF;
            text-decoration: underline;
        }

.news-date {
    font-size: 0.85rem;
    color: #A0AEC0;
    margin-bottom: 1rem;
    font-weight: 500;
}

.analysis {
    background: #0d2d44;
    padding: 1rem;
    border-radius: 8px;
    border-left: 4px solid #2d5a87;
}

.analysis-label {
    font-size: 0.8rem;
    font-weight: 600;
    text-transform: uppercase;
    letter-spacing: 0.5px;
    margin-bottom: 0.5rem;
    color: #CBD5E0;
}

.analysis-text {
    font-size: 0.9rem;
```

```
    color: #CBD5E0;
    line-height: 1.5;
}

/* Status indicators */
.status-verified {
    border-left-color: #15BD65;
}

    .status-verified .analysis-label {
        color: #15BD65;
    }

.status-questionable {
    border-left-color: #ed8936;
}

    .status-questionable .analysis-label {
        color: #ed8936;
    }

.status-false {
    border-left-color: #D03939;
}

    .status-false .analysis-label {
        color: #D03939;
    }

/* Responsive Design */
@media (max-width: 1024px) {
    .container {
        grid-template-columns: 1fr;
        grid-template-areas:
            "header"
            "sidebar"
            "main";
    }

    .sidebar {
        border-left: none;
        border-bottom: 1px solid #e2e8f0;
        box-shadow: 0 2px 4px rgba(0,0,0,0.05);
    }

    .news-grid {
        grid-template-columns: 1fr;
    }
}

@media (max-width: 768px) {
    .header {
```

```
        padding: 1rem;
    }

    .main {
        padding: 1rem;
    }

    .sidebar {
        padding: 1rem;
    }

    .news-card {
        padding: 1rem;
    }
}
```

ДОДАТОК З

ЛОГІКА ВЕБ ІНТЕРФЕЙСУ

```
// js/script.js

// Функція для отримання класу CSS на основі risk_level
function getAnalysisStatusClass(riskLevel) {
    if (riskLevel === null || riskLevel === undefined) return
     'status-neutral'; // Додаємо перевірку
    const lowerRiskLevel = riskLevel.toLowerCase(); // Приводимо
     до нижнього регістру для гнучкості

    if (lowerRiskLevel.includes('підтверджено') ||
     lowerRiskLevel.includes('verified') ||
     lowerRiskLevel.includes('правдиво')) {
        return "status-verified";
    }
    if (lowerRiskLevel.includes('маніпуляція') ||
     lowerRiskLevel.includes('упередженість') ||
     lowerRiskLevel.includes('questionable') ||
     lowerRiskLevel.includes('оманлива') ||
     lowerRiskLevel.includes('сумнівно')) {
        return "status-questionable";
    }
    if (lowerRiskLevel.includes('неправдиві') ||
     lowerRiskLevel.includes('фейк') ||
     lowerRiskLevel.includes('false') ||
     lowerRiskLevel.includes('дезінформація')) {
        return "status-false";
    }
    return "status-neutral"; // для інших випадків
}

// Функція для отримання тексту для analysis-label
function getAnalysisLabel(riskLevel) {
    if (riskLevel === null || riskLevel === undefined) return
     'Аналіз невідомий'; // Додаємо перевірку
    const lowerRiskLevel = riskLevel.toLowerCase();

    if (lowerRiskLevel.includes('підтверджено') ||
     lowerRiskLevel.includes('verified') ||
     lowerRiskLevel.includes('правдиво')) {
        return "Перевірена інформація";
    }
    if (lowerRiskLevel.includes('маніпуляція') ||
     lowerRiskLevel.includes('упередженість')) {
        return "Маніпуляція / Упередженість";
    }
    if (lowerRiskLevel.includes('фейк') ||
     lowerRiskLevel.includes('дезінформація') ||
     lowerRiskLevel.includes('неправдиві')) {
```

```
        return "Виявлено дезінформацію";
    }
    if (lowerRiskLevel.includes('потребує перевірки') &&
     lowerRiskLevel.includes('клікбейт')) {
        return "Потребує перевірки (Клікбейт)";
    }
    if (lowerRiskLevel.includes('потребує перевірки') &&
     lowerRiskLevel.includes('необ\'єктивно')) {
        return "Потребує перевірки (Необ'єктивно)";
    }
    if (lowerRiskLevel.includes('потенційно оманлива')) {
        return "Потенційно оманлива";
    }
    return "Аналіз невідомий";
}


// Функція для отримання назви джерела з URL сторінки
function getSourceFromUrl() {
    const path = window.location.pathname;
    const fileName = path.substring(path.lastIndexOf('/') + 1);

    switch (fileName) {
        case 'index.html':
            return 'Українська правда';
        case 'tsn.html':
            return 'Новини на tsn.ua';
        case 'dailycaller.html':
            return 'The Daily Caller';
        default:
            return 'Українська правда';
    }
}

// Функція для встановлення активного класу в бічній панелі
function setActiveNavLink() {
    const navLinks = document.querySelectorAll('.nav-list a');
    const currentPath = window.location.pathname;
    const currentFileName =
     currentPath.substring(currentPath.lastIndexOf('/') + 1);

    navLinks.forEach(link => {
        link.classList.remove('active'); // Видаляємо клас active
     з усіх посилань
        // Порівнюємо href посилання з поточною назвою файлу
        if (link.getAttribute('href') === currentFileName) {
            link.classList.add('active');
        }
    });
}

// Функція для форматування дати
```

```
function formatNewsDate(isoDateString) {
    if (!isoDateString) return "Невідома дата";
    try {
        const date = new Date(isoDateString);
        if (isNaN(date.getTime())) {
            // Якщо не парситься "Mon, 02 Jun 2025 21:35:26 +0300"
            const parts = isoDateString.split(' ');
            if (parts.length >= 5) {
                const day = parts[1];
                const month = parts[2];
                const year = parts[3];
                const time = parts[4];
                const dateUtc = new Date(`${day} ${month} ${year}
 ${time} UTC`);
                return dateUtc.toLocaleString('uk-UA', {
                    year: 'numeric',
                    month: 'long',
                    day: 'numeric',
                    hour: '2-digit',
                    minute: '2-digit',
                    hourCycle: 'h23'
                });
            }
            return isoDateString;
        }
        return date.toLocaleString('uk-UA', {
            year: 'numeric',
            month: 'long',
            day: 'numeric',
            hour: '2-digit',
            minute: '2-digit',
            hourCycle: 'h23'
        });
    } catch (e) {
        console.error("Помилка форматування дати:", e);
        return isoDateString;
    }
}

// головна функція: Завантаження та відображення новин
async function fetchAndDisplayNews() {
    const newsGrid = document.querySelector('.news-grid');
    if (!newsGrid) {
        console.error("Елемент .news-grid не знайдено на
 сторінці.");
        return;
    }
    newsGrid.innerHTML = ''; // чистка сітки перед завантаженням
 нових даних

    const requestedSource = getSourceFromUrl(); // отримуємо
 джерело для фільтрації
```

```
try {
    const response = await fetch('data/news.json'); // шлях до
 JSON-файлу
    if (!response.ok) {
        throw new Error(`HTTP error! status:
 ${response.status}`);
    }
    const allNewsData = await response.json();

    // Фільтр новин за джерелом
    const filteredNews = allNewsData.filter(newsItem => {
        // Перевірте: newsItem.source має точно відповідати
 requestedSource
        return requestedSource === 'all' || newsItem.source
 === requestedSource;
    });

    if (filteredNews.length === 0) {
        newsGrid.innerHTML = '<p>Новин для цього джерела поки
 що немає.</p>';
        return;
    }

    // Рендеринг відфільтрованих новин
    filteredNews.forEach(newsItem => {
        const newsCard = document.createElement('article');
        newsCard.className = 'news-card';

        const analysisStatusClass =
 getAnalysisStatusClass(newsItem.risk_level);
        const analysisLabelText =
 getAnalysisLabel(newsItem.risk_level);
        const formattedDate =
 formatNewsDate(newsItem.published);

        newsCard.innerHTML = `
            <div class="source-name">${newsItem.source ||
 'Невідоме джерело'}</div>
            <h3 class="headline"><a href="${newsItem.link}"
 target="_blank">${newsItem.title}</a></h3>
            <div class="news-date">${formattedDate}</div>
            <div class="analysis ${analysisStatusClass}">
                <div class="analysis-
 label">${analysisLabelText}</div>
                <div class="analysis-text">
                    ${newsItem.overall_summary || 'Резюме
 аналізу відсутнє.'}
                    ${newsItem.manipulation_present ?
 `<br>Маніпуляція: ${newsItem.manipulation_explanation}` : ''}
```

```
                        ${newsItem.false_claims_present ?
     `<br>Неправдиві твердження:
     ${newsItem.false_claims_explanation}` : ''}
                        ${newsItem.clickbait_present ?
     `<br>Клікбейт: ${newsItem.clickbait_explanation}` : ''}
                    </div>
                </div>
            `;
            newsGrid.appendChild(newsCard);
        });

    } catch (error) {
        console.error("Помилка завантаження або відображення
     новин:", error);
        newsGrid.innerHTML = '<p style="color: red;">Помилка
     завантаження новин. Спробуйте пізніше.</p>';
    }
}

// Єдиний DOMContentLoaded слухач
document.addEventListener('DOMContentLoaded', () => {
    setActiveNavLink(); // Встановити активне посилання
    fetchAndDisplayNews(); // Завантажити та відобразити новини
});
```

ДОДАТОК І

СКРИНШОТ ВЕБ-ІНТЕРФЕЙСУ (ФІНАЛЬНА РОБОТА)