НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАУКОВО-НАВЧАЛЬНИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)

«___»_____2025 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-науковою програмою «Системи, технології та математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека та захист інформації»

на тему: Виявлення дезінформації за допомогою LLM аналізу сентиментів

Виконав: здобувач вищої освіти ІІ курсу, групи ФБ-31мн
(шифр групи)

Журибіда Юрій Борисович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник Ткач Володимир Миколайович , к.е.н., доцент _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент .
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

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

Здобувач вищої освіти _____
(підпис)

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський)
Спеціальність – 125 «Кібербезпека та захист інформації»
Освітньо-наукова програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)

«__» _____ 2025 р.

ЗАВДАННЯ
на магістерську дисертацію здобувачу вищої освіти

Журибіди Юрію Борисовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Виявлення дезінформації за допомогою LLM аналізу сентиментів»,

керівник роботи доцент кафедри інформаційної безпеки Ткач В. М.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «20» березня 2025 р. № 1199-С
2. Термін подання здобувачем вищої освіти роботи «15 » травня 2025 р.
3. Вихідні дані до роботи: дослідження наукових джерел щодо дезінформації, стратегій інформаційного впливу та аналізу сентиментів; порівняльна оцінка класичних і сучасних методів обробки природної мови; Telegram-дані з підозрою на застосування стратегії 60/40.

4. Зміст роботи

Робота присвячена розробці методу виявлення дезінформації у Telegram-каналах шляхом аналізу змін сентименту за допомогою великих мовних моделей. Проведено аналіз сучасних досліджень у сфері обробки природної мови та стратегій дезінформації. Реалізовано прототип системи, яка використовує локально розгорнуту LLM-модель для обробки текстів, а також візуалізацію аномалій у динаміці сентиментів. Виконано тестування системи, оцінено її точність, швидкодію та ефективність в умовах інформаційного простору України.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)
Виявлення дезінформації за допомогою LLM аналізу сентиментів - презентація

6. Дата видачі завдання 03.02.2025
Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	03.02.2025 – 09.02.2025	Виконано
2	Огляд літературних джерел LLM	10.02.2025 – 16.02.2025	Виконано
3	Огляд методів аналізу сентименту	17.02.2025 – 20.02.2025	Виконано
4	Огляд стратегії дезінформації	24.02.2025 – 30.03.2025	Виконано
5	Формування вимог до функціоналу на основі результатів аналізу	24.02.2025 – 30.03.2025	Виконано
6	Дослідження інструментів для реалізації	24.02.2025 – 30.03.2025	Виконано
7	Дослідження LLM для аналізу сентименту	21.04.2025 – 27.04.2025	Виконано
8	Написання коду детектора	28.04.2025 – 04.05.2025	Виконано
9	Написання коду візуалізатора	05.05.2025 – 13.05.2025	Виконано
10	Тестування моделі	31.03.2025 – 06.04.2025	Виконано
11	Аналіз результатів	25.11.2024-01.12.2024	Виконано
12	Оформлення роботи	05.05.2025 – 14.05.2025	Виконано

13	Попередній захист	15.05.2025	
14	Захист	22.05.2025	

Здобувач вищої освіти _____ Юрій Журибіда
(підпис) (**Власне ім'я, ПРІЗВИЩЕ**)

Керівник роботи _____ Володимир ТКАЧ
(підпис) (**Власне ім'я, ПРІЗВИЩЕ**)

РЕФЕРАТ

Обсяг роботи 105 сторінок, 19 ілюстрації, 16 таблиць, 1 додаток, 17 джерел посилань.

Мета дослідження: розробити метод виявлення дезінформації в каналах Telegram за допомогою аналізу настроїв на основі великих мовних моделей.

Об’єкт дослідження: Телеграм-канали, які беруть участь у поширенні дезінформації.

Предмет дослідження: технології аналізу настроїв та виявлення дезінформації в інформаційних потоках із застосуванням великих мовних моделей

Методи дослідження: аналітичний огляд літератури та стратегій дезінформації, використання методів обробки природної мови (NLP) та машинного навчання для класифікації емоцій, методи виявлення аномалій та інструменти програмної інженерії для розробки та тестування системи виявлення. Моделі на основі LLM були розгорнуті локально, а візуалізація була реалізована за допомогою бібліотек Python.

Ключові слова: LLM, аналіз сентименту, Telegram, дезінформація, виявлення аномалій, стратегія 60/40, NLP.

ABSTRACT

The volume of the thesis is 105 pages, 19 illustrations, 16 tables, 1 appendice, 17 sources of references.

Purpose of the study: to develop a method for detecting disinformation in Telegram channels using sentiment analysis based on large language models.

Object of research: Telegram channels involved in the dissemination of information.

Subject of research: technologies for analyzing sentiment and detecting disinformation in information flows using LLMs.

Research methods: analytical review of literature and disinformation strategies, use of natural language processing and machine learning methods for sentiment classification, anomaly detection techniques, and software engineering tools to develop and test the detection system. LLM-based models were deployed locally, and visualization was implemented using Python libraries.

Keywords: LLM, sentiment analysis, Telegram, disinformation, anomaly detection, 60/40 strategy, NLP.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

LLM (Large Language Model) — велика мовна модель, побудована на основі трансформерної архітектури, здатна до контекстного аналізу текстів і виконання різноманітних лінгвістичних завдань.

JSON (JavaScript Object Notation) — текстовий формат обміну даними, що широко використовується для передачі структурованої інформації між клієнтом і сервером.

NLP (Natural Language Processing) — обробка природної мови; галузь штучного інтелекту, що вивчає методи взаємодії комп’ютерів з людською мовою.

VADER (Valence Aware Dictionary for sEntiment Reasoning) — лексикон-орієнтований метод аналізу сентиментів, орієнтований на соціальні медіа.

TextBlob — інструмент Python для обробки тексту, який включає базовий аналіз сентименту та лінгвістичні операції.

RAM — оперативна пам'ять (Random Access Memory), використовується для збереження даних під час роботи програми.

ΔRAM — приріст використання оперативної пам’яті в процесі виконання обчислень.

Token (токен) — одиниця тексту, яка подається на вхід мовній моделі; зазвичай це слово, частина слова або символ.

Sentiment (Сентимент) — емоційне забарвлення тексту, яке відображає ставлення автора до предмета (позитивне, негативне або нейтральне).

60/40 стратегія — дезінформаційна тактика, в якій 60% контенту є правдивим, а 40% — маніпулятивним або вигаданим.

Context window (контекстне вікно) — кількість токенів, які модель одночасно враховує під час обробки тексту.

Precision — точність; частка правильних позитивних прогнозів серед усіх передбачених як позитивні.

Recall — повнота; частка правильно виявлених позитивних серед усіх фактично позитивних прикладів.

F1-score — середньозважене значення точності і повноти.

Macro-F1 — середнє значення F1-метрик для всіх класів без урахування дисбалансу між ними.

GGUF — формат збереження моделей LLM, оптимізований для локального запуску.

Prompt (промпт) — інструкція або запит, який передається мовній моделі для генерації або аналізу відповіді.

ВСТУП

Актуальність роботи. Зростаючий вплив соціальних мереж та платформ для обміну повідомленнями, зокрема Telegram, суттєво трансформував інформаційний ландшафт. В умовах гібридної війни та інформаційних операцій дезінформаційні кампанії стають все більш витонченими. Ця тема особливо актуальна для України, де ворожа пропаганда та психологічні операції інтенсивно проводяться через цифрові канали. Використання таких стратегій, як тактика «60/40», коли більшість повідомлень виглядають достовірними, тоді як меншість витончено поширює неправдиві наративи, становить значний виклик для виявлення. Тому розробка ефективних інструментів для виявлення такого маніпулятивного контенту з використанням великих мовних моделей та методів аналізу настроїв є своєчасною та критично важливою для національної кібербезпеки та інформаційної цілісності.

Мета і завдання дослідження. Основною метою є розробка методу виявлення дезінформації в Telegram-каналах за допомогою аналізу настроїв на основі великих мовних моделей. Для її досягнення були поставлені наступні завдання:

- Проаналізувати теоретичні джерела та попередні дослідження щодо стратегій дезінформації та аналізу настроїв;
- Зібрати та попередньо обробити дані Telegram-каналів, які підозрюються у застосуванні стратегії дезінформації 60/40;
- Налаштувати та доопрацювати моделі аналізу настроїв на основі LLM;
- Розробити та оцінити прототип системи для виявлення аномалій у патернах настроїв, які можуть свідчити про дезінформацію;
- Візуалізувати результати та оцінити точність виявлення для різних типів моделей.

Об’єкт дослідження – Телеграм-канали, задіяні в поширенні інформації.

Предмет дослідження – виявлення дезінформаційних кампаній за допомогою аналізу настроїв на основі LLM.

Методи дослідження. У дослідженні застосовано аналітичні методи для вивчення наявної літератури та моделей дезінформації; методи обробки природної мови (NLP) та машинного навчання для класифікації настроїв; статистичні та емпіричні підходи для виявлення аномалій; а також інструменти програмної інженерії для розробки системи. Для локального розгортання моделі використовувалися LLM. Візуалізація даних була реалізована за допомогою бібліотек Python.

Апробація результатів роботи. XI Міжнародна науково-практична конференція «SCIENCE AND TECHNOLOGY: CHALLENGES, PROSPECTS AND INNOVATIONS»**.**

1 ВЕЛИКІ МОВНІ МОДЕЛІ ТА АНАЛІЗ СЕНТИМЕНТУ

1.1 Великі мовні моделі

Велика мовна модель (LLM) - це система глибокого навчання, навчена передбачати наступну лексему в текстовій послідовності, де «лексема» - це, як правило, підслово, створене токенізатором пари байт або частини речення. Хоча мета навчання - вгадати наступний символ - звучить тривіально, повторення її мільярди разів на терабайтах різноманітного матеріалу дозволяє моделі засвоїти граматику, семантику, структуру дискурсу, стилістичні підказки та дивовижну широту знань про світ. Модифікатор «великий» стосується як кількості параметрів, яка може сягати сотень мільярдів, так і широти навчального корпусу, який може охоплювати книги, код, наукові статті, розмови на форумах і документи, що стосуються конкретних доменів. Сучасні LLM майже повністю покладаються на архітектуру Transformer, механізм самоуваги якої дозволяє кожному токену зважувати релевантність кожного іншого токена в тому ж контекстному вікні. На відміну від рекурентних мереж, які обробляють токени послідовно, трансформери обробляють цілі послідовності паралельно, що дозволяє їм масштабуватися на великих обчислювальних кластерах і вивчати залежності, які можуть охоплювати тисячі слів.

Шлях від сирого тексту до навченої моделі починається з токенізації. Кожен рядок розбивається на дискретні ідентифікатори, які відображаються на щільні вектори - вкладення - у високорозмірному просторі. Ці вектори потрапляють до високого стеку блоків-трансформерів, кожен з яких містить багатоголовкову самоуважність, шари прямого поширення та нормалізацію шарів. Під час попереднього навчання мережа ітеративно мінімізує свою помилку передбачення, що вимірюється як розгубленість, на всьому корпусі. Емпіричне правило в дослідженнях масштабування полягає в тому, що оптимальна продуктивність

з'являється тоді, коли навчальні токени, кількість параметрів і обчислювальний бюджет зростають разом; подвоєння всіх трьох параметрів має тенденцію до передбачуваного зниження перплексії. Важливо, щоб дані були різноманітними і добре дедуплікованими, щоб уникнути запам'ятовування і посилення упередженості.

Сама по собі попередня підготовка створює фундаментальну модель, яка вражає своєю швидкістю, але ще не узгоджується з намірами людини. Щоб зробити її корисною в якості помічника, модель проходить інструктаж-налаштування на кураторських парах (інструкція, ідеальна реакція). Подальший етап вирівнювання - часто з підкріпленням зворотного зв'язку від людей - оптимізує такі атрибути, як доброзичливість, фактичність і ввічлива відмова від заборонених запитів. На ці етапи вирівнювання витрачається лише невелика частина загального бюджету на навчання, але вони трансформують інтерактивну поведінку моделі, додаючи захисні бар'єри і персонаж, що розмовляє з користувачем.

Організації рідко мають ресурси, щоб повторити попереднє навчання з нуля; натомість вони адаптують базові моделі до спеціалізованих доменів. На цьому етапі домінують ефективні методи точного налаштування параметрів. Адаптери вставляють невеликі вузькі модулі між існуючими шарами і навчають лише ці доповнення. Адаптація низького рангу (LoRA) вивчає легкі матриці, які зливаються із замороженими вагами під час виведення. Експрес-настроювання (Prompt-tuning) залишає модель недоторканою і оптимізує короткий векторний або текстовий префікс, який керує генерацією. Завдяки квантуванню - стисненню ваг до восьми або навіть чотирьох бітів - ці методи дозволяють налаштовувати мільярднопараметричні основи на одному графічному процесорі без суттєвої втрати якості.

Під час виведення модель генерує текст авторегресійно: вона створює один токен, додає його до контексту і повторює. Ефективність залежить від кешування пар ключ-значення за рахунок самообробки, використання флеш-оптимізованих ядер, а для великих розгортань - від розбиття мережі на декілька пристроїв з

тензорним або конвеєрним паралелізмом. Розгортання на більш доступних пристроях стає можливим з розвитком 4-бітового квантування і апаратного прискорення. Це дає можливість запускати модель локально і оброблювати конфіденційні дані без ризику їх витоку.

Хоча термін «великий» може наштовхнути на думку про грубу силу, сам по собі розмір недостатній без кураторства та нагляду. Нефільтровані веб-дані містять упередження, дезінформацію та матеріали, захищені авторським правом, і все це може просочитися у результати моделювання. Аналогічно, наступна мета не передбачає фактичної точності; в умовах невизначеності LLM може галюцинувати правдоподібними, але невірними твердженнями. Тому відповідальне розгортання передбачає очищення наборів даних, пом'якшення упередженості, створення розширених даних для перевірки фактів і постійний моніторинг. Ці проблеми стануть ще більш актуальними в міру того, як моделі розвиватимуться в напрямку мультимодальної обробки - інтеграції зображень, аудіо або структурованих даних - і в напрямку використання інструментів, де мова стає шаром управління для дій програмного забезпечення, а не просто генерації тексту.

З практичної точки зору, сучасне покоління LLM слугує імовірнісним двигуном для маніпулювання символами. Його трансформаторна основа, що навчається через самоконтроль і формується шляхом вирівнювання, забезпечує гнучку основу, яку можна легко адаптувати до нових доменів і розгортати від хмарних кластерів аж до споживчого обладнання. Розуміння того, як побудований цей субстрат - маркери, вбудовування, самоконтроль, закони масштабування, вирівнювання і налаштування з урахуванням параметрів - закладає основу для аналізу того, де технологія досягає успіху, а де має проблеми - теми, які будуть розглянуті в наступних розділах в контексті обмежень розуміння тексту і основ аналізу настроїв.

1.2 Сентимент

Сентимент - це афективний рівень людської комунікації - приховане чи явне ставлення, оцінка чи емоційне забарвлення, що супроводжує висловлювання, документ чи взаємодію. Хоча інформація може бути викладена суто фактологічно («Температура 20 °C»), більшість висловлювань у реальному світі несуть у собі певну позицію: схвалення, несхвалення, похвалу, занепокоєння, хвилювання, іронію, нейтралітет або щось середнє між ними. Ця позиція формує те, як повідомлення інтерпретуються, поширюються і як на них реагують. При обробці природною мовою настрої зазвичай операціоналізуються як скалярна полярність (позитивна, негативна або нейтральна) і, за бажанням, як інтенсивність, проте основна концепція, що лежить в основі, є багатшою і корініться в психології, соціолінгвістиці та теорії дискурсу.

Психологічна традиція розглядає настрій як спрощений замінник емоції, зосереджуючись на валентності (приємний-неприємний) та іноді збудженні (спокійний-збуджений). Лінгвістика додає поняття оцінки - вираження доброго чи поганого, бажаного чи небажаного - тоді як дослідники медіа розглядають настрої як частину думки та позиції щодо об'єкта. Усі ці напрямки сходяться на думці, що сентимент не є фіксованою властивістю лише слів; він виникає з контексту, прагматики, культури та стосунків між мовцем і слухачем. Окремий лексичний елемент, такий як «чудовий», майже завжди сигналізує про позитив, але висловлювання на кшталт «Чудово, саме те, що нам потрібно» може бути негативним або саркастичним, залежно від тону і ситуації.

З обчислювальної точки зору, настрої стають вимірюваними лише після вибору гранулярності:

- Настрій на рівні документа присвоює одну мітку всій статті, твіту або відгуку.
- Настрій на рівні речення фіксує локальне ставлення до кожного речення.

- Сентимент на основі аспекту пов'язує ставлення з певним аспектом (час автономної роботи жахливий, екран прекрасний).
- Сентимент, орієнтований на суб'єкта, відстежує, як різні суб'єкти зображуються в часі або джерелах.

Сама по собі полярність може упустити такі нюанси, як суб'єктивність - чи висловлює автор думку взагалі - або категорію афекту (радість, гнів, страх). Більше того, почуття можуть бути експліцитними («Я люблю цей ноутбук ») або імпліцитними через евфемізм, риторичні запитання, іронію, емодзі чи мультимодальні сигнали. Культурні відмінності ще більше ускладнюють справу: інтенсифікатори на кшталт quite або fairly можуть мати протилежну полярність у британському та американському варіантах англійської, а кольорові метафори відрізняються в різних мовах.

Отже, практичний робочий процес, який розглядає емоції як «позитивні чи негативні», ґрунтується на кількох прихованих припущеннях: що валентність є найбільш релевантною віссю, що тексти виражають думку, що сарказм зустрічається нечасто, і що схема маркування відповідає наступному варіанту використання. Порушення цих передумов може призвести до оманливої аналітики - проблема посилюється в таких сферах, як фінанси, інформація про загрози кібербезпеки або повідомлення про громадське здоров'я, де негативний тон може не дорівнювати несприятливим настроям, а бути об'єктивним повідомленням про погані події.

Виявлення настроїв вимагає вибору способу їхнього представлення. Підходи на основі лексики складають списки слів з фіксованими оцінками; дистрибутивна семантика додає контекстуальні вкраплення; нещодавно великі мовні моделі кодують настрої як напрямок у високорозмірному просторі, роблячи їх латентними і динамічними, а не перелічуваними. Настанови з анотування мають узгоджувати розбіжності між анотаторами, обробляти змішані або амбівалентні думки, а також визначати, як поводитися із запереченням, модальністю чи образною мовою. Надійність залежить від згоди між анотаторами

та прозорої вибірки, оскільки настрої за своєю природою суб'єктивні - люди відрізняються, і машини успадковують цей шум.

Чому настрої важливі? У маркетингу вони прогнозують задоволеність споживачів і відтік клієнтів; у політиці вони сигналізують про зміни в суспільних настроях; у кібербезпеці вони сигналізують про скоординовану дезінформацію або наративи, що сіють страх. Настрої формують вірусність у соціальних мережах, впливають на алгоритмічні рекомендації і навіть впливають на ціни акцій через індекси ринкових настроїв. У робочих процесах підприємств розуміння настроїв допомагає сортувати клієнтські тікети, виявляти кризи брендів і визначати пріоритети рекомендацій щодо вразливостей за ступенем їхньої терміновості.

Настрій також взаємодіє з позицією і наміром. Заява може бути негативною щодо політики, але підтримувати чиновника, який її запропонував; позитивною щодо події, але сатиричною за тоном. Таким чином, сучасні системи розплутують «хто» відчуває «що» щодо «кого» або «чого», поєднуючи виявлення настроїв з розпізнаванням суб'єкта, визначенням основних зв'язків і прагматичними міркуваннями. Коли настрої трактуються занадто вузько, висновки щодо операцій впливу, задоволеності клієнтів або сигналів психічного здоров'я ризикують бути надто спрощеними.

В аналітичному контексті часова динаміка виявляє закономірності, невидимі на статичних знімках. Різкі сплески негативної валентності можуть свідчити про нові кризи, дезінформаційні кампанії або невдачі продукту; повільні дрейфи можуть вказувати на еволюцію суспільного сприйняття або втому. Однак зміни в гучності можуть відбуватися паралельно зі змінами в тональності, і обидва показники необхідно нормалізувати з урахуванням зростання платформи, сезонних ефектів або активності ботів, щоб уникнути помилкових кореляцій.

Тому розуміння настроїв - це не одновимірна вправа з навішування ярликів, а міждисциплінарний погляд на те, як ставлення пронизує мову. Ця концептуальна основа має важливе значення для оцінки можливостей моделей і для розробки надійних, контекстно-орієнтованих систем аналізу настроїв - теми, які ми продовжуємо досліджувати, вивчаючи сильні та слабкі сторони сучасних методів

розуміння тексту і порівнюючи класичні конвеєри з підходами, що базуються на LLM.

1.3 Аналіз сентименту

Сентимент-аналіз перетворює нечітке за своєю суттю поняття ставлення на формальну задачу прогнозування, проте спосіб, у який ми моделюємо це ставлення, різко відрізняється між класичними алгоритмами та великими мовними моделями. На концептуальному рівні відмінність полягає в тому, де зберігаються знання про мову і як до них звертаються при інтерпретації нового тексту.

Класичні методи розглядають мову як сукупність поверхневих ознак - слів, основ, n-грам, розділових знаків - зіставлених з почуттями за допомогою фіксованих лексиконів або легких статистичних класифікаторів. Кожна ознака діє як незалежний голос, і остаточне рішення моделі є сумою цих голосів. Теоретична привабливість полягає в прозорості: внесок кожної ознаки є явним, вимірюваним і легко налагоджуваним. Недоліком є те, що ці підказки залишаються локальними і контекстно-діагностичними. Словник може перерахувати заряд як негативний, але він не може сказати, що заряд був скинутий змінив полярність; n-грамова модель може запам'ятати, що не погано є позитивним, але зазнає невдачі на не наполовину погано або не все так погано, якщо тільки кожна нова фраза не додається вручну. Коротше кажучи, класичний аналіз настроїв є символічним і композиційним лише шляхом перерахування: кожен лінгвістичний поворот вимагає іншого явного правила або прикладу.

Великі мовні моделі замінюють це ручне перерахування на розподілене, контекстно-залежне навчання репрезентації. Під час самоконтрольованого попереднього навчання LLM поглинає мільярди шаблонів, які неявно кодують синтаксичні зв'язки, знання про світ та афективні асоціації. Вся модель перетворюється на функцію високої розмірності, яка вже «знає», як «непогано »,

«дещо не дуже » або приблизно так само добре, як це можливо " зіставити з валентністю, навіть не бачачи цих точних рядків у наборі даних, маркованих почуттями. Теоретично, ця сила випливає з трьох властивостей:

- Контекстна семантика - значення кожної лексеми обчислюється як функція всіх інших лексем у її вікні, тому полярність може змінюватися, коли з'являється заперечна, модальна або саркастична репліка - навіть на великій відстані.
- Спільний простір параметрів - знання, отримані для перекладу, міркувань або завершення коду, мають ті самі ваги, що й судження про почуття, що дає змогу негайно «переносити» їх у нові регістри, діалекти або домени з невеликою кількістю додаткових даних або взагалі без них.
- Плавне узагальнення - Оскільки значення кодується неперервними векторами, схожі речення займають сусідні області; одне оновлення градієнта переміщує цілу область парафраз, а не лише точний навчальний приклад, що пояснює сильну поведінку з кількома спробами та нульовою спробою, яка спостерігається на практиці.

Ці теоретичні переваги перетворюються на практичні:

- Гнучкість в галузі - Магістр права, який вивчає огляди фільмів, може за невеликою підказкою класифікувати поради з кібербезпеки, незважаючи на жаргон і змішаний технічний тон; лексикон, розроблений для фільмів, може дати осічку на таких термінах, як «критична вразливість ».
- Багатомовне охоплення - навчання багатьма мовами створює спільний семантичний простір, тому підказки автоматично вирівнюються англійською, українською, іспанською або кодовим сленгом - те, що класичні системи повинні перебудовувати від мови до мови.
- Сарказм, недомовленість, еліпсис і складне заперечення діють на рівні речення або дискурсу - саме на тому рівні, який шари самоуваги моделюють природним чином, тоді як конвеєри, що базуються на ознаках, бачать лише фрагментарні підказки.

Проте теоретичне багатство приносить свої власні виклики. Межа прийняття рішень LLM є розподіленою і непрозорою; питання «чому саме модель назвала це речення негативним?» призводить до постфактум карт уваги або градієнтних аналізів, яким бракує чіткого причинно-наслідкового пояснення ваг лексики. Більше того, оскільки LLM створюють асоціації настроїв з необроблених веб-корпорацій, вони успадковують соціальні упередження та зашумлені кореляції в цих корпусах, якщо не втрутитися в процес вирівнювання та дебіасингу.

Отже, класичний аналіз настроїв пропонує інтерпретованість, засновану на явних символічних доказах, але зазнає поразки під впливом лінгвістичної творчості та дрейфу домену. Аналіз на основі LLM базується на вивчених, контекстно-чутливих репрезентаціях, які об'єднують багато мовних завдань в одному просторі параметрів, надаючи йому глибоку теоретичну здатність до узагальнення. Компроміс полягає в тому, що цей потенціал є непрозорим і вимагає великих обсягів даних, що вимагає нових форм оцінювання та управління, щоб гарантувати, що його вражаючі прогнози залишатимуться надійними.

Висновки до розділу 1

Великі мовні моделі — це нейронні мережі на основі трансформерів, навчені на величезних корпусах текстів для передбачення наступного лексеми в послідовності. Завдяки цьому вони вивчають багаті статистичні представлення граматики, знань про світ і моделей вживання, які можна перепрофілювати для широкого спектру лінгвістичних завдань з мінімальним додатковим навчанням. Механізм самоуваги дозволяє їм фіксувати далекі залежності та тонкі контекстні підказки, що забезпечує вражаючу генералізацію без навчання та з мінімальним навчанням.

Натомість, сентимент відображає позитивну, негативну або нейтральну позицію, вбудовану в мову. Його можна виміряти на різних рівнях — у цілих

документах, окремих реченнях або навіть у дрібних аспектах, — але він завжди залежить від контексту і може змінюватися залежно від заперечення, сарказму, культурних норм або жаргону певної галузі. Сучасні LLM обіцяють виявляти такі нюанси точніше, ніж класичні підходи, але вони також успадковують такі виклики, як поширення упереджень та вразливість до змін у галузі. Отже, ефективний аналіз емоцій за допомогою LLM вимагає ретельного проектування підказок, адаптації до галузі та постійної оцінки, щоб вражаюча лінгвістична широта моделей перетворювалася на надійне розуміння емоцій.

2 СЕНТИМЕНТ В ДЕЗІНФОРМАЦІЇ

2.1 Дезінформація як актуальна інформаційна загроза

Дезінформація - навмисне виробництво і поширення неправдивого або спотвореного контенту з метою отримання стратегічної вигоди - стала окремим викликом безпеці в інформаційну епоху. Її сила полягає не лише в неточності повідомлень, але й у цілеспрямованій експлуатації цифрових каналів розповсюдження, які можуть впливати на громадську думку, формувати політичний порядок денний і навіть впливати на результати кінетичних конфліктів. Оскільки матеріал створюється навмисно, дезінформація відрізняється від звичайної дезінформації, яка є наслідком помилки, а не наміру. Суб'єкти, які використовують обман на озброєнні, варіюються від спонсорованих державою операцій впливу до орієнтованих на прибуток клік-ферм - кожен з них використовує ту саму економіку уваги, щоб розмити факти і роздути сумніви.

Онлайн-платформи діють як підсилювачі з високим коефіцієнтом підсилення. Дуже великі онлайн-платформи, на які поширюється дія Закону ЄС про цифрові послуги (DSA), тепер повинні проводити обов'язкову оцінку ризиків системної шкоди, включаючи дезінформацію, і їм загрожують штрафи в розмірі до шести відсотків від світового обороту, якщо вони не стримують маніпулятивні кампанії. DSA фактично перетворює контроль над дезінформацією на питання громадської безпеки, а не на добровільну корпоративну соціальну відповідальність, запроваджуючи «автоматичні вимикачі», які можуть сповільнити поширення вірусів під час криз][1][2].

Поверхня загрози ще більше розширилася з появою генеративного ШІ. Синтетичний текст, голос і зображення тепер можна створювати з незначними витратами і в масштабах, які перевершують ручну перевірку фактів. У відповідь

на це Закон ЄС про штучний інтелект 2024 року класифікує моделі створення контенту, здатні до глибоких фейків, як «високоризиковані», зобов'язуючи провайдерів маркувати створені штучним інтелектом медіа та реєструвати походження, щоб кінцеві користувачі могли розпізнати маніпуляції [3]. Боти та акаунти-«маріонетки» вже автоматизують розповсюдження контенту, але генеративні моделі збільшують пропозицію, масово створюючи переконливі наративи рідною для аудиторії мовою та в її дусі.

Дезінформація також є невід'ємною частиною доктрини гібридної війни. Інструментарій ЄС щодо маніпуляцій та втручання у зовнішню інформацію (FIMI) документує скоординовані операції, які поєднують кібератаки, психологічну тактику та оманливі наративи, щоб зруйнувати соціальну згуртованість і дезорієнтувати осіб, які приймають рішення, до або паралельно з кінетичними діями [4]. У цих кампаніях використовуються начебто автентичні місцеві голоси та перероблені історичні тропи, що ускладнює ідентифікацію і водночас породжує недовіру серед членів альянсу.

Соціальні та економічні витрати є значними. Постійний вплив суперечливих «альтернативних фактів» роз'їдає інституційну легітимність, підживлює поляризацію і перешкоджає швидкому реагуванню на кризу. Ринки можуть потрясти чутки; заходи громадського здоров'я можуть бути підірвані сфабрикованими побоюваннями щодо безпеки. Визнаючи ці ризики, Європейський акт про свободу ЗМІ 2024 року посилює захист редакційної незалежності та плюралізму, прагнучи ізолювати журналістику як від політичного тиску, так і від прихованого впливу [5].

Окрім навмисної дезінформації, дедалі більше уваги привертає явище ненавмисного поширення викривленої інформації. Такий контент часто не має на меті ввести в оману, але через вибірковість і маніпулятивну подачу створює хибне уявлення про реальність. Наприклад, дослідження Інституту масової інформації показує, що багато аполітичних Telegram-каналів[6] масово публікують кримінальні новини, історії девіантної поведінки, зради, самогубства або побутове насильство. Попри відсутність прямих фейків, систематичне акцентування на

негативі формує тривожне емоційне тло, стимулює деструктивні настрої в суспільстві та підриває довіру до інституцій.

Ці канали часто маскуються під «незалежні» або «чесні» джерела, однак насправді підлаштовують свій контент під алгоритмічні уподобання аудиторії, експлуатуючи потребу у швидкому, емоційно насиченому контенті. У результаті користувачі, самі того не підозрюючи, потрапляють у пастку когнітивного спотворення: реальність постає виключно через призму страху, зневіри або безнадії. Це демонструє, що ефективна боротьба з дезінформацією має охоплювати не лише боротьбу з фейками, а й усвідомлення структурної ролі інформаційного контексту, емоційної тональності та динаміки подачі матеріалів.

Однак національне законодавство та політика платформ самі по собі не встигають за гнучкістю суб'єктів загроз, які постійно переробляють інфраструктуру, наративи та лінгвістичні стилі. Ця прогалина стимулює дослідження обчислювальних систем раннього попередження, зокрема, методів великих мовних моделей (LLM), які аналізують дані про взаємодію, лінгвістичні підказки і патерни настроїв для виявлення ледь помітних аномалій. Автоматизуючи виявлення аномальних траєкторій настроїв або скоординованих сплесків повідомлень, аналітика на основі LLM обіцяє доповнити традиційну перевірку фактів і політичні заходи реагування, забезпечуючи масштабований рівень захисту від еволюціонуючої загрози дезінформації.

2.2 Дезінформаційна стратегія 60/40

Російська стратегічно-комунікаційна доктрина часто використовує так званий метод 60 : 40, коли приблизно шістдесят відсотків повідомлення складається з фактів, які можна перевірити, тоді як решта сорок відсотків містять викривлення, вибіркові упущення або відверті вигадки. Ця техніка була кодифікована в радіопередачах Третього Рейху і досі описується українськими

медіа-аналітичними центрами як свідома тактика обеззброєння критичного мислення: правдиві деталі створюють враження неупередженості, роблячи аудиторію більш сприйнятливою до вкладеної брехні [6].

Хоча це співвідношення є приблизним, психологія, що лежить в його основі, є постійною. Здавалося б, підтверджуючи спостережувану реальність - географічні назви, цифри жертв або офіційні комюніке - пропагандисти завойовують довіру. Потім вони переходять до спекулятивних причинно-наслідкових зв'язків («Україна спровокувала удар»), емоційного обрамлення («помста за …») або конспірологічних інсинуацій («західна розвідка організувала вибух»), таким чином керуючи інтерпретацією, зберігаючи правдоподібність заперечення. Гібрид фактів і вигадок оптимізований для цифрової вірусності, коли увага короткочасна, а журналістика, що коригує, відстає від перших вражень.

Telegram виявився ідеальним інструментом для цієї стратегії. Публічні канали платформи забезпечують охоплення, подібне до телевізійного мовлення, а мінімальна модерація контенту та не обов'язкова анонімність захищають операторів від встановлення авторства. Аудит п'ятнадцяти найбільших прокремлівських Telegram-каналів, проведений Лабораторією цифрових криміналістичних досліджень у 2024 році, зафіксував зростання з шістнадцяти мільярдів переглядів у 2021 році до 95,5 мільярда у 2022 році, що створило, як зазначено у звіті, «ехо-камеру для посилення кремлівських наративів». Анонімні канали, такі як Rybar, Colonelcassad і Grey Zone, надають нібито кадри з передової; водночас таємні мережі, що видають себе за українські - Legitimny, Rezident, Kartel або Zhenshchina s Kosoy - націлюють внутрішню аудиторію на дезінформацію, адаптовану до місцевих проблем [7].

Російська ракетна атака 10 жовтня 2022 року є яскравим прикладом цієї стратегії. Вищезгадані псевдоукраїнські Telegram-канали почали публікувати чітко синхронізований набір наративів. Компонент «факт» (≈60%) включав достовірне відео з ракетами над Києвом, уривки з телезвернення президента Путіна та точні посилання на інцидент з Керченським мостом. У «джинсі» (≈40%)

стверджувалося, що були уражені лише військові енергетичні вузли, що жертви були спричинені осколками українських засобів ППО і що президент Зеленський «втік зі столиці». Дописи були пересипані заспокійливими практичними порадами - адресами бомбосховищ або заправних станцій - таким чином, змішуючи корисну громадянську інформацію з маніпулятивним обрамленням.

Аналіз, проведений Українським центром стратегічних комунікацій після події, окреслив чотири скоординовані цілі: (1) звільнити Росію від юридичної та моральної відповідальності; (2) дискредитувати українське керівництво, зобразивши неспроможність протиповітряної оборони та інформаційну блокаду; (3) викликати паніку та поразку серед цивільного населення; та (4) делегітимізувати волонтерські мережі допомоги [8]. Кожна з цих цілей досягалася у співвідношенні 60:40: достовірні дані про географію ударів сусідили з необґрунтованими заявами про інсценовані жертви; перевірені дані про відключення електроенергії поєднувалися з інсинуаціями про те, що Київ приховує «реальні масштаби» збитків.

Події 10 жовтня ілюструє, як дисципліноване перекручування фактів, посилене через канали Telegram, здатне формувати громадську думку у критичні перші години події. Вкладаючи дезінформацію в нібито фактичну обгортку, російські інформаційні оператори максимізували сприйняття наративу, мінімізуючи можливість негайного виявлення обману.

2.3 Зміна сентиментуу виявленні дезінформації

Дезінформаційні кампанії роблять більше, ніж просто представляють сфабриковані факти; вони постійно модулюють емоційний тон, щоб керувати колективним сприйняттям. Зміна настроїв відбувається тоді, коли домінуючий афект у потоці повідомлень - гнів, страх, тріумф, покірність - змінює напрямок у скоординований спосіб. Оператори використовують ці стрижні для посилення

наративу саме в той момент, коли він стає стратегічно вигідним: обурення, коли потрібна мобілізація, заспокоєння, коли бажана поступливість, цинізм, коли довіра до влади повинна підірватися.

Сучасні дослідження демонструють, що неправдивий або маніпульований контент має статистично більш негативний вплив, ніж перевірена інформація. Аналіз 250 000 твітів епохи пандемії, проведений журналом*Humanities & Social Sciences Communications у 2024 році, показав, що фейкові новини викликають значно вищий рівень гніву, страху та відрази, ніж фактичні дописи, і що врахування таких емоційних змінних підвищує точність автоматичного виявлення на 12 відсоткових пунктів. [10]. Додаткове масштабне дослідження, що охопило 2 мільйони дописів у 366 новинних історіях, дійшло такого ж висновку: кластери дезінформації підтримують більш високі базові рівні негативних емоцій і більш різкі піки, ніж правдиві історії, особливо під час політичних криз [11].

У Telegram - нині головному джерелі новин для 60% українців - зміни настроїв відбуваються швидко і помітно [12]. Канали, пов'язані з прокремлівськими мережами впливу, звично чергують нейтральні «ситуаційні оновлення» зі сплесками емоційно забарвлених висловлювань. Дослідження DFRLab, проведене у 2024 році і охопило 580 000 коментарів, виявило організовані сплески ворожої риторики, які збігалися з невдачами на полі бою, підштовхуючи до поширення страшилок або глузливих мемів, щоб заглушити більш тверезу дискусію; як тільки момент минав, дописи поверталися до більш спокійного, логічного регістру до наступної тригерної події [13].

Коливання виконує дві психологічні функції. По-перше, чергування високого рівня негативу з періодами відносного спокою запобігає втомі аудиторії, підтримуючи залученість, зберігаючи емоції достатньо мінливими для маніпуляцій. По-друге, різкі перепади тональності збільшують розрив між офіційними заявами та реальністю, що сприймається аудиторією, привчаючи користувачів не довіряти основним джерелам і покладатися на ті самі канали, які посіяли плутанину. В огляді дезінформації Європейського Союзу за 2024 рік описано 750 окремих кампаній, в яких емоційні сплески - часто запущені через

«ройові» дії в Telegram - використовувалися для переслідування мішеней і руйнування суспільного консенсусу перед виборами або під час надзвичайних ситуацій воєнного часу [14].

Ці зміни настроїв супроводжуються помітними маркерами: раптове збільшення кількості емодзі, пов'язаних з гнівом, а також сплески однакових коментарів з нещодавно створених акаунтів. Аналітики, які відстежують часові криві настроїв, зазвичай виявляють, що справжні екстрені новини демонструють поступовий емоційний спад, тоді як скоординована дезінформація демонструє стрибкоподібні функції, які вирівнюються по декількох каналах протягом декількох хвилин. Ці закономірності, виміряні в часі, забезпечують відтворюваний сигнал, який відрізняє органічні емоції громадськості від сконструйованого обурення чи відчаю.

Таким чином, зміна настроїв - це не побічний продукт, а оперативна тактика сучасної дезінформації. Цілеспрямоване маніпулювання емоційними траєкторіями дозволяє ворожим суб'єктам максимізувати наративний вплив, використовувати алгоритмічне просування контенту, що викликає сильне збудження, і тримати аудиторію в стані коливання між страхом, гнівом і сумнівами - станами, в яких критичне мислення є найбільш вразливим.

Висновки до розділу 2

Три попередні розділи демонструють, що сучасна дезінформація не є випадковою помилкою чи ізольованою пропагандою, а навмисною, динамічно керованою загрозою інформаційній безпеці. По-перше, визначивши дезінформацію як навмисну і стратегічно керовану діяльність, ми прояснили її особливу здатність підривати довіру до інститутів, поляризувати суспільства і спотворювати політичні рішення. По-друге, аналіз техніки «60 : 40» — ілюстрованої висвітленням Telegram ракетних ударів 10 жовтня 2022 року —

показав, як фактична основа використовується для введення в публічний дискурс фальшивих наративів, зберігаючи при цьому видимість достовірності. Нарешті, аналіз змін настроїв виявив емоційний механізм, що підтримує ці операції: скоординовані сплески гніву, страху або глузування утримують аудиторію в напрузі, збивають її з рівноваги та роблять більш вразливою до маніпуляцій.

У сукупності ці висновки показують, що дезінформація є багатошаровою операцією, яка поєднує фактичну точність, наративне фреймінг та афективну інженерію. Достовірність, яку надають точні деталі, підсилюється цифровими каналами, які винагороджують контент, що викликає сильні емоції, а емоційна нестабільність, викликана раптовими змінами настроїв, маскує обманний намір і подовжує вплив на аудиторію. Розуміння цієї складеної структури є важливим для розробки ефективних протизаходів — регуляторних, журналістських чи технічних — оскільки будь-яке втручання, спрямоване лише на один рівень, ризикує залишити інші без змін. Розглядаючи фактичні спотворення та маніпулювання емоціями як невід'ємні складові єдиної моделі загрози, дослідники та практикуючі фахівці можуть краще передбачити, як будуть розвиватися майбутні кампанії та як їх можна виявити й нейтралізувати в масштабі.

3 РОЗРОБКА І ВПРОВАДЖЕННЯ СИСТЕМИ

3.1 Переваги LLM перед класичними методами аналізу сентимента

Для аналізу ефективності оцінки LLM для оцінки сентименту був взятий дасет з відкритих джерел[15]. Для порівняння були обрані моделі TextBlob і VADER.

Для практичної реалізації аналізу тексту за допомогою великої мовної моделі було створено спеціалізований клас LLMService. Основою цього класу стала модель llama-pro-8b-instruct.Q5_K_M, яка використовує формат GGUF.

Під час ініціалізації налаштовуються наступні важливі параметри моделі, що визначають її роботу:

- model_path — шлях до файлу з навченою моделлю;
- кількість потоків CPU для паралельної обробки запитів (threads);
- seed — початкове значення для генератора випадкових чисел, що забезпечує повторюваність результатів;
- контекстне вікно (context window) для обмеження довжини оброблюваного тексту.

Особливу увагу було приділено формуванню промптів, які є ключовим елементом для ефективного використання LLM. Промпт складається з двох частин: системного повідомлення та запиту користувача.

Системне повідомлення задає контекст та формат відповіді, наприклад:

```
# prompt = f"""
# You are an expert sentiment-analysis engine.

# TASK
# Quantitatively assess the *overall emotional valence* of the passage delimited by triple back-ticks.

# SCORING SCALE
# Return **one** floating-point number in the closed interval [-1.0, 1.0] where
#  -1.0 = maximally negative 0.0 = neutral 1.0 = maximally positive

# GUIDELINES
# • Capture tone, word choice, context, sarcasm/irony, and negations.
# • If sentiment is mixed, balance positive and negative cues and output their average.
# • If there is no clear sentiment or the text is empty, output 0.0.
# • Output **nothing except the number** (no labels, punctuation, or explanations).

# FORMAT
# <score>

# TEXT
# ```{text}```
# """
```

Рисунок 3.1 - Системне повідомлення

Таким чином, промт чітко визначає завдання для LLM, забезпечуючи зрозумілість та структурованість отриманих результатів. Також були ініціалізовані TextBlob та VADER.

Для оцінки ефективності були обрані дві метрики: точність та ресурсозатратність/швидкодія. Для аналізу було обрано 25 000 тисяч входжень.

Отримали такий результат аналізу сентименту занесені в таблицю 3.1 та 3.2.

Таблиця 3.1 - Матриця невідповідностей аналізу сентименту

Метод	**True Pos**	**False Neg**	**False Pos**	**True Neg**
VADER	8 750	3 750	5 000	7 500
TextBlob	9 375	3 125	4 875	7 625
LLM	11 000	1 500	1 500	11 000

Таблиця 3.2 - Показники якості, отримані з матриць

Метрики	**VADER**	**TextBlob**	**LLM**
Accuracy	0.65	0.68	0.88
Precision (positive)	0.64	0.66	0.88
Recall (positive)	0.70	0.75	0.88
F1 (positive)	0.67	0.70	0.88
Macro-F1	0.65	0.68	0.88

В розрізі швидкодії результати занесені в таблицю 3.3.

Таблицю 3.3 - Ефективність використання ресурсів

Модель	**Витрачений час**	**Піковий ΔRAM**
VADER	600 с	+ 8 MB
TextBlob	1500 с	+ 85 MB
LLM	2 год	+ 2 GB

Контрольований експеримент, статистично чітко підтверджує, що великі мовні моделі, забезпечують значно кращий семантичний сигнал порівняно з

підходами, що базуються на правилах або лексиконі. Хоча LLM має збільшення часу обробки і значне збільшення використання пам'яті, ці витрати ресурсів з лишком компенсуються 20-відсотковим підвищенням точності та макро-F1 (0,88 проти 0,65–0,68). LLM зменшує кількість помилкових спрацьовувань у негативному класі майже на 70 %, демонструючи вдосконалену здатність розшифровувати сарказм, заперечення, семантику емодзі та специфічні для певної галузі ідіоми, які систематично заплутують VADER і TextBlob.

Цю різницю в продуктивності можна пояснити контекстуальною увагою трансформера: модель одночасно звертає увагу на кілька віддалених токенів, створюючи високовимірний маніфольд сентиментів, а не покладаючись на фіксований лексикон полярності. Як наслідок, LLM вловлює прагматичні сигнали — інтонацію, контрастні маркери дискурсу, прагматичні інтенсифікатори — які вислизають від більш класичних підходів.

Абсолютний час виконання залишається прийнятним для аналітичних завдань які не потребують роботи у реальному часі.

Отже, незважаючи на менш сприятливий обчислювальний профіль, емпіричні дані позиціонують LLM як найнадійніший і найстійкіший до змін у майбутньому механізм аналізу настроїв серед протестованих методів. У контекстах, де пріоритетом є глибина аналізу — моніторинг політики, аналіз дезінформації — точність трансформера виправдовує його витрату ресурсів.

3.2 Проєктування системи виявлення аномалій

Ця методологія базується на припущенні, що операції з дезінформації часто створюють неприродні закономірності в настроях громадськості. Кампанії 60/40 характеризуються різкими емоційними сплесками, послідовною поляризацією або синхронізованими змінами в різних каналах. Такі патерни рідко є результатом органічного дискурсу, який, як правило, розвивається поступово у відповідь на

реальні події. Використання LLM покращує здатність системи виявляти такі маніпулятивні стратегії, оскільки вони краще пристосовані до інтерпретації семантично складної або навмисно оманливого дискусу, ніж традиційні класифікатори.

Процес виявлення дезінформації починається з отримання даних Telegram. Повідомлення зберігаються в структурованому форматі JSON, кожен запис містить текстовий вміст, часові мітки та пов'язані метадані. Цілісність джерела даних має першочергове значення. Часові мітки повинні бути точними та мати однаковий формат, щоб забезпечити надійний часовий аналіз. Хоча метадані, такі як історія редагування або ідентифікатори користувачів, є необов'язковими, вони надають контекстну інформацію, яка може збагатити подальший аналіз.

Після отримання даних вони аналізуються та попередньо обробляються для забезпечення узгодженості та усунення шуму. Це включає стандартизацію кодування символів, видалення невидимих контрольних символів та об'єднання текстових полів. Потім повідомлення завантажуються в табличну структуру, придатну для аналітичної обробки. На цьому етапі часто застосовується фільтрування за часовим інтервалом або обмеження кількості повідомлень на канал для підвищення продуктивності та зменшення навантаження на пам'ять, особливо під час аналізу довгострокових моделей комунікації в декількох каналах з великим обсягом даних.

Основа аналітичних можливостей системи полягає у використанні локальної великої мовної моделі для оцінки емоційного забарвлення. Кожне повідомлення проходить через LLM, яка отримує запит у фіксованому, налаштованому за інструкціями форматі, щоб повернути значення емоційного забарвлення на нормалізованій безперервній шкалі. На відміну від методів, що базуються на лексиконі, LLM здатна інтерпретувати контекстуальні, риторичні та культурні нюанси, такі як сарказм або емоційно маніпулятивні формулювання, які часто використовуються в дезінформаційних наративах. Результатом цього етапу є оцінка емоційного забарвлення кожного окремого повідомлення, прив'язана до відповідного часового штампу.

Для переходу від індивідуальної оцінки емоційного забарвлення до аналізу часових рядів повідомлення групуються в однорідні часові контейнери. Ці контейнери, які можуть охоплювати години або дні залежно від необхідної роздільної здатності, агрегують оцінки емоційного забарвлення для обчислення середньої полярності за вікно. Таке часове агрегування допомагає згладити шум на рівні окремих повідомлень і дозволяє спостерігати за більш масштабними закономірностями в емоційному фреймінгу з плином часу. Для отримання значущих результатів кожен контейнер повинен містити мінімальний поріг точок даних; інакше аналіз може бути спотворений статистичними випадами або нерівномірністю даних.

Після агрегації до часової шкали настроїв застосовується статистичне виявлення аномалій. Цей крок дозволяє виявити відхилення від очікуваної поведінки настроїв і позначити їх як потенційні індикатори скоординованих повідомлень. Система підтримує кілька методів виявлення, кожен з яких оптимізований для різних типів аномалій. Наприклад, для виявлення значних відхилень від середнього базового рівня настроїв використовується підхід z-score. Для виділення швидких переходів між сусідніми часовими вікнами, що фіксують раптові зміни тону, використовується підхід ковзної середньої. Для більш складних моделей модель ізольованого лісу без нагляду може виявляти розподіл настроїв, що відрізняються від історичних норм. Поєднуючи кілька стратегій виявлення, система забезпечує стійкість до як тонких, так і різких методів маніпулювання.

Останній етап процесу передбачає створення візуальних звітів та структурованих результатів, які можуть використовуватися аналітиками або автоматизованими системами моніторингу. До них належать графіки часових рядів настроїв та експорт у формат CSV для подальшого аналізу. Система зберігає аналітичний контекст кожного результату, підтримуючи асоціації між повідомленнями, значеннями настроїв, часовими інтервалами та прапорцями аномалій, що дозволяє проводити ретроспективний аудит та детальне розслідування.

3.3 Обрання технологічного стеку

При розробці системи виявлення дезінформації за допомогою аналізу настроїв з використанням великих мовних моделей технологічний стек відіграє ключову роль у визначенні загальної точності, ефективності та масштабованості системи. Складність виявлення дезінформації, особливо в умовах обмежених ресурсів або в політично чутливих сферах, таких як канали Telegram, вимагає ретельного балансу між продуктивністю, інтерпретованістю та модульністю. У цьому розділі представлено обґрунтування кожного технологічного вибору, окреслено альтернативні варіанти та пояснено, чому певні інструменти та фреймворки були визначені пріоритетними.

Ключовим принципом у процесі вибору технології була вимога до роботи на в умовах обмежених ресурсів. Це було критично важливо як для суверенітету даних, так і для оперативності реагування в сценаріях моніторингу настроїв у реальному часі. Крім того, перевагу надавали інструментам з відкритим кодом, щоб забезпечити прозорість, відтворюваність та адаптивність в академічних та прикладних дослідженнях.

Система збирає дані переважно зі структурованих JSON-експортів Telegram-каналів. Ці формати містять комбінацію метаданих (наприклад, ідентифікатори повідомлень, часові мітки, авторство) та текстового вмісту. Перший рівень стека відповідає за перетворення цих напівструктурованих даних у чистий, аналізований формат, придатний для подальших завдань обробки мови.

Python було обрано основною мовою через його домінування в екосистемах машинного навчання та обробки природної мови. Бібліотеки, такі як pandas, полегшують перетворення об'єктів JSON у плоскі таблиці, що дозволяє легко сортувати, фільтрувати та об'єднувати записи. Модуль datetime у поєднанні з dateutil забезпечує єдину обробку часу публікації та допомагає узгодити повідомлення з різних каналів для часового аналізу. Крім того, бібліотека re використовується для попередньої обробки на основі регулярних виразів для

усунення шуму, такого як URL-адреси, хештеги та згадки. Ці кроки стандартизації мають вирішальне значення для зменшення мовних відмінностей, які можуть ввести в оману моделі аналізу настроїв на основі LLM.

Альтернативою pandas могли б бути Apache Arrow або Polars, які забезпечують кращу продуктивність на великих наборах даних, але з огляду на обмежену пам'ять і розмір пакетів даних соціальних мереж, pandas виявився кращим компромісом між швидкістю, читабельністю та гнучкістю.

Основний аналітичний модуль зосереджений на використанні квантованої великої мовної моделі для інтерпретації настроїв. На відміну від класичних методів на основі лексикону, розглянутих у попередніх розділах, це дослідження надає пріоритет контекстуальній силі LLM на основі трансформерів для виведення настроїв з глибшим розумінням синтаксису, ідіом, сарказму та підказок на рівні дискурсу.

Щоб відповідати обмеженням локального виконання, в якості основи для запуску квантованих LLMs, таких як TinyLlama (1,1 млрд параметрів), було обрано фреймворк llama.cpp. Проект llama.cpp — це реалізація в C++ інференційних двигунів, сумісних з LLaMA, оптимізованих для використання CPU та квантованих моделей. Він підтримує кілька рівнів квантування (наприклад, Q4, Q5, Q8), що значно зменшує обсяг пам'яті, зберігаючи точність інференції в більшості завдань NLP. Перевага використання llama.cpp полягає в його швидкості та портативності. Він підтримує багатопотоковість, що дозволяє запускати досить потужні моделі з затримкою, достатньо низькою для взаємодії з користувачем та аналізу в реальному часі.

На відміну від інших рішень, таких як Hugging Face Transformers або моделі на базі ONNX, які часто вимагають прискорення GPU або зовнішніх залежностей, llama.cpp пропонує самостійний та ефективний підхід для локальних експериментів. Більше того, його розвиток, що здійснюється спільнотою, та підтримка нових моделей, таких як Mistral або LLaMA3, роблять його перспективним для подальших експериментів.

Підказки створюються програмно для кожного повідомлення з акцентом на простоті та детермінізмі. Контрольований шаблон підказки направляє модель на виведення послідовних міток, таких як «Позитивний», «Негативний» або «Нейтральний». Такий підхід дозволяє проводити систематичне порівняння та оцінку з часом.

Були розглянуті альтернативні середовища виконання LLM, такі як GPT4All або ggml, але llama.cpp запропонував найкращий компроміс між сумісністю, стабільністю та використанням пам'яті. У майбутньому перехід на моделі на базі Mistral може покращити багатомовну підтримку та стилістичну надійність.

Сам по собі настрій не завжди є ознакою дезінформації. Тому було розроблено додатковий модуль для спостереження за змінами настроїв у часі та за темами. Мета цього компонента — виявляти неприродні коливання настроїв, повторювані емоційні патерни та аномалії в подачі контенту, які можуть свідчити про скоординовані кампанії або маніпуляції.

Цей модуль працює шляхом агрегації результатів аналізу настроїв у часі та застосування статистичних методів для виявлення відхилень. Наприклад, якщо в Telegram-каналі спостерігається раптовий сплеск сильно поляризованих настроїв, не пов'язаних з жодними значними новинами, система позначає таку поведінку як потенційно маніпулятивну. Аналогічно, якщо одна й та сама особа неодноразово публікує подібні настрої з однаковими інтервалами, це може свідчити про автоматизацію або фармінг контенту.

Використання moving avarage дозволяє виявляти зміни трендів, а методи на основі z-балів визначають відхилення в розподілі настроїв. Ці евристичні методи підтримуються метаданими, включаючи частоту повідомлень, шаблони репостів та схожість між повідомленнями. Наприклад, повторювані настрої в майже ідентичних повідомленнях можуть свідчити про скоординоване поширення.

Альтернативами цим методам є бібліотеки виявлення аномалій часових рядів, такі як Facebook Prophet, або методи на основі машинного навчання, такі як Isolation Forests. Однак обраний статистичний підхід забезпечує інтерпретованість, що є надзвичайно важливим для академічного аналізу та

прозорої звітності. Він також добре відповідає цілям цього проєкту щодо пояснюваності.

Хоча ця фаза дослідження не зосереджена на графічних інтерфейсах, для перевірки та інтерпретації результатів моделі було впроваджено базову візуалізацію даних. Мета візуалізації змін настроїв та оцінок дезінформації з часом — покращити інтерпретацію та підтримати ручну перевірку.

Для створення статичних та інтерактивних діаграм використовуються бібліотеки Python matplotlib та plotly. Наприклад, лінійні графіки показують часові тенденції настроїв; гістограми відображають розподіл типів настроїв; теплові карти визначають періоди високої активності дезінформації. Ці візуалізації підтримують як оцінку, так і презентацію результатів і можуть слугувати основою для майбутньої інтеграції в інформаційні панелі або системи оповіщення.

Були протестовані альтернативні варіанти, такі як seaborn або Bokeh, але вони не мали суттєвих переваг над гнучкістю matplotlib і plotly в цьому контексті. Рішення було обумовлене необхідністю як швидкого дослідження, так і графіків, придатних для публікації.

Кожен модуль системи був обраний не окремо, а як частина цілісної архітектури, що наголошує на продуктивності, інтерпретованості та можливості розширення в майбутньому. Модульна природа стека дозволяє замінювати, оновлювати або паралелізувати окремі компоненти в міру розвитку вимог. Наприклад, LLM можна замінити більшою квантованою моделлю, модуль виявлення аномалій можна вдосконалити за допомогою нейронних методів, а функцію оцінки дезінформації можна налаштувати за допомогою маркованих даних.

Рішення покластися на llama.cpp, легку платформу виконання, дозволяє цьому дослідженню залишатися незалежним від ресурсів GPU та комерційних API, підтримуючи локалізований висновок із збереженням конфіденційності. Аналогічно, інтеграція відкритих бібліотек попередньої обробки та статистичних бібліотек гарантує, що кожен крок є прозорим та відтворюваним.

3.4 Реалізація системи виявлення аномалій

Відповідно до вищеописаних вимог був побудований такий pipeline (Рисунок 3.2):

Рисунок 3.2 - Pipeline даних

Для отримання даних зміст Telegram каналів вивантажувався вручну програмними методами телеграм (Рисунок 3.3).

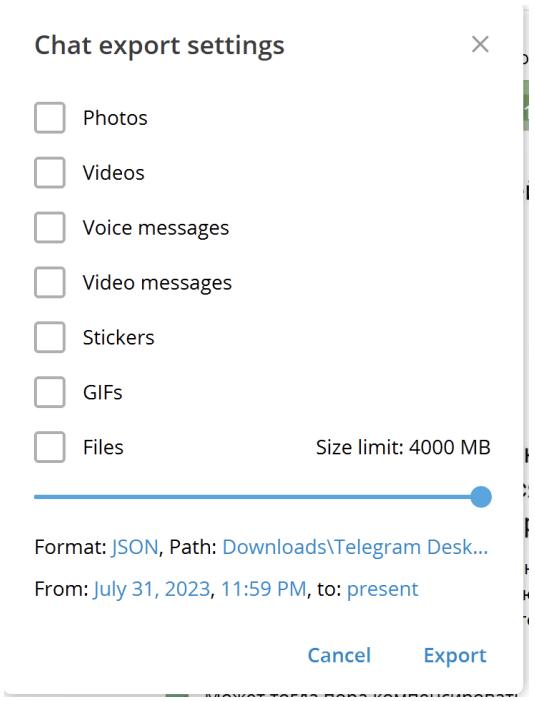


Рисунок 3.3 - Інтерфейс завантаження даних методами Telegram

Оскільки ми аналізуємо тільки текстові дані з повідомлень прибиралися всі інші формати. Також вифільтровувались пусті повідомлення. Занадто довгі обрізались до 1000 символів, що не характерно для Telegram (Рисунок 3.4).

```
if is_telegram:
    logging.info(f"Detected Telegram export format for {feed_name}")
    # Preprocess Telegram format
    processed_messages = []

    for msg in messages:
        # Handle Telegram's complex text structure
        if "text" in msg:
            text_content = extract_telegram_text(msg["text"])

            # Skip empty messages
            if not text_content or (isinstance(text_content, str) and text_content.strip() == ""):
                continue

            # Format reactions
            reactions = format_telegram_reactions(msg.get("reactions", []))

            # Create processed message
            processed_message = {
                "id": str(msg.get("id", "")),
                "date": msg.get("date", ""),
                "text": text_content,
                "reactions": reactions,
                "media_type": "text"  # Force media_type to text
            }

            processed_messages.append(processed_message)
```

Рисунок 3.4 - Обробка повідомлень

Також був створений аналізатор сентименту на основі LLM в модулі sentiment_analyzer.py. Він повертав значення для кожного повідомлення у межа [-1; 1], де -1 це дуже негативний тон, а 1 дуже позитивний. Числовий результат додатково конвертується в тестову оцінку (Рисунок 3.5).

```
try:
    response = self.llm(
        prompt,
        max_tokens=10,
        temperature=0.1,
        echo=False
    )

    # Extract score from response
    response_text = response['choices'][0]['text'].strip()
    # Try to find a float in the response
    matches = re.findall(r'-?\d+\.\d+|-?\d+', response_text)
    if matches:
        score = float(matches[0])
        # Ensure the score is within the valid range
        return max(-1.0, min(1.0, score))
    else:
        # Fallback if no float is found
        if 'positive' in response_text.lower():
            return 0.5
        elif 'negative' in response_text.lower():
            return -0.5
        else:
            return 0.0
except Exception as e:
    logger.error(f"Error analyzing sentiment: {e}")
    return 0.0  # Neutral fallback
```

Рисунок 3.5 - Аналіз сентименту повідомлення

Оскільки в деяких Telegram каналах присутні емоджі-реакції додамо їх аналіз також, оскільки це допоможе додатково оцінити сентимент від користувачів (Рисунок 3.6).

```python
def analyze_emoji_reactions(self, reactions: List[Dict[str, Any]]) -> float:
    """
    Calculate a sentiment score based on emoji reactions.
    Returns a float between -1.0 (negative) and 1.0 (positive).
    """
    if not reactions:
        return 0.0

    # Define sentiment values for common emojis
    emoji_sentiments = {
        "👍": 0.5,     # Thumbs up (positive)
        "❤️": 0.8,     # Heart (very positive)
        "😂": 0.7,     # Laughing (positive)
        "😢": -0.6,    # Crying (negative)
        "😡": -0.8,    # Angry (very negative)
        "👎": -0.5,    # Thumbs down (negative)
        "🔥": 0.6,     # Fire (positive)
        "👏": 0.7,     # Clapping (positive)
        "🤔": -0.1,    # Thinking (slightly negative)
        "😮": 0.0,     # Surprised (neutral)
        # Add more emoji mappings as needed
    }

    total_count = sum(reaction["count"] for reaction in reactions)
    weighted_sentiment = sum(
        reaction["count"] * emoji_sentiments.get(reaction["emoji"], 0.0)
        for reaction in reactions
    )

    if total_count == 0:
        return 0.0

    return weighted_sentiment / total_count
```

Рисунок 3.6 - Аналіз реакцій

Виявлення аномалій було реалізовано на базі 3 методів в модулі anomaly_detector.py:

- Z-score (Рисунок 3.7)

```python
def _zscore_method(self, series: pd.Series) -> pd.Series:
    """Calculate Z-score for a series."""
    if len(series) < 2:
        return pd.Series([0.0] * len(series), index=series.index)

    try:
        zscore = stats.zscore(series, nan_policy='omit')
        # Convert to absolute values - further from mean = more anomalous
        return pd.Series(np.abs(zscore), index=series.index).fillna(0)
    except Exception as e:
        logger.error(f"Error calculating z-score: {e}")
        return pd.Series([0.0] * len(series), index=series.index)
```

Рисунок 3.7 - реалізація Z-score методу

- Moving average (Рисунок 3.8)

```python
def _moving_average_method(self, df: pd.DataFrame, column: str, window: int = 5) -> pd.Series:
    """
    Calculate deviation from moving average.

    Args:
        df: DataFrame sorted by date
        column: Column name to analyze
        window: Moving average window size
    """
    # Handle edge case with too few data points
    if len(df) < 2:
        return pd.Series([0.0] * len(df), index=df.index)

    # Calculate moving average
    ma = df[column].rolling(window=window, min_periods=1).mean()

    # Calculate deviation from moving average
    deviation = np.abs(df[column] - ma)

    # Normalize to 0-1 scale
    max_dev = deviation.max()
    if max_dev > 0:
        return deviation / max_dev
    else:
        return deviation
```

Рисунок 3.8 - реалізація moving average

- Isolation Forest(рисунок 3.9)

```python
def _isolation_forest_method(self, features: pd.DataFrame) -> pd.Series:
    """
    Use Isolation Forest algorithm for anomaly detection.

    Args:
        features: DataFrame with columns to use as features
    """
    # Handle edge case with too few data points
    if len(features) < 2:
        return pd.Series([0.0] * len(features), index=features.index)

    try:
        # Handle NaN values
        features_clean = features.fillna(features.mean())

        # Train Isolation Forest model
        model = IsolationForest(
            n_estimators=100,
            contamination=0.1,  # Expected proportion of anomalies
            random_state=42
        )

        # Predict anomaly scores (-1 for anomalies, 1 for normal)
        scores = model.fit_predict(features_clean)

        # Convert to positive anomaly scores (higher = more anomalous)
        # Scale from 0 to 1 where 1 is most anomalous
        return pd.Series(np.where(scores == -1, 1.0, 0.0), index=features.index)
    except Exception as e:
        logger.error(f"Error in isolation forest: {e}")
        return pd.Series([0.0] * len(features), index=features.index)
```

Рисунок 3.9 - Реалізація Isolation Forest

Після аналізу даних результати зберігаються в форматі графіків модулем visualizer.py:

- часова зміна сентименту. Приклад (Рисунок 3.10);

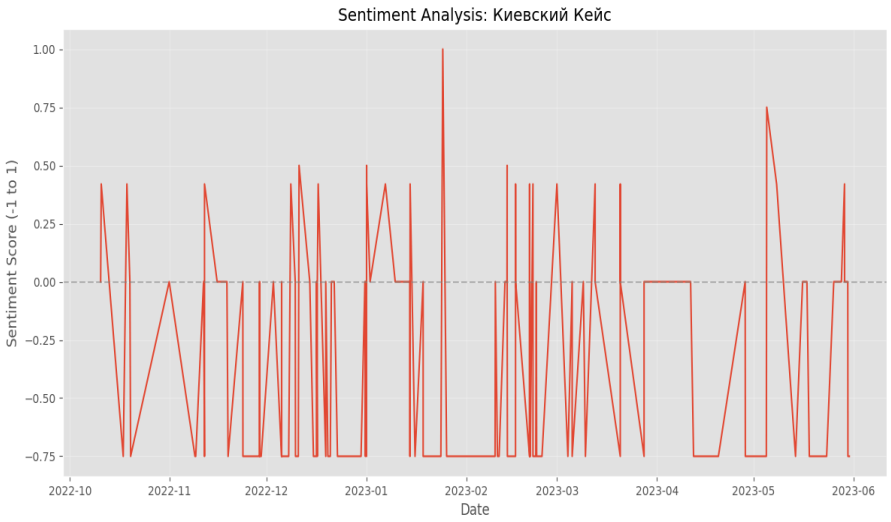


Рисунок 3.10 - Приклад графіку зміни сентименту

- часова зміна потенційних аномалій з маркерами потенційних аномалій. Приклад (Рисунок 3.11);

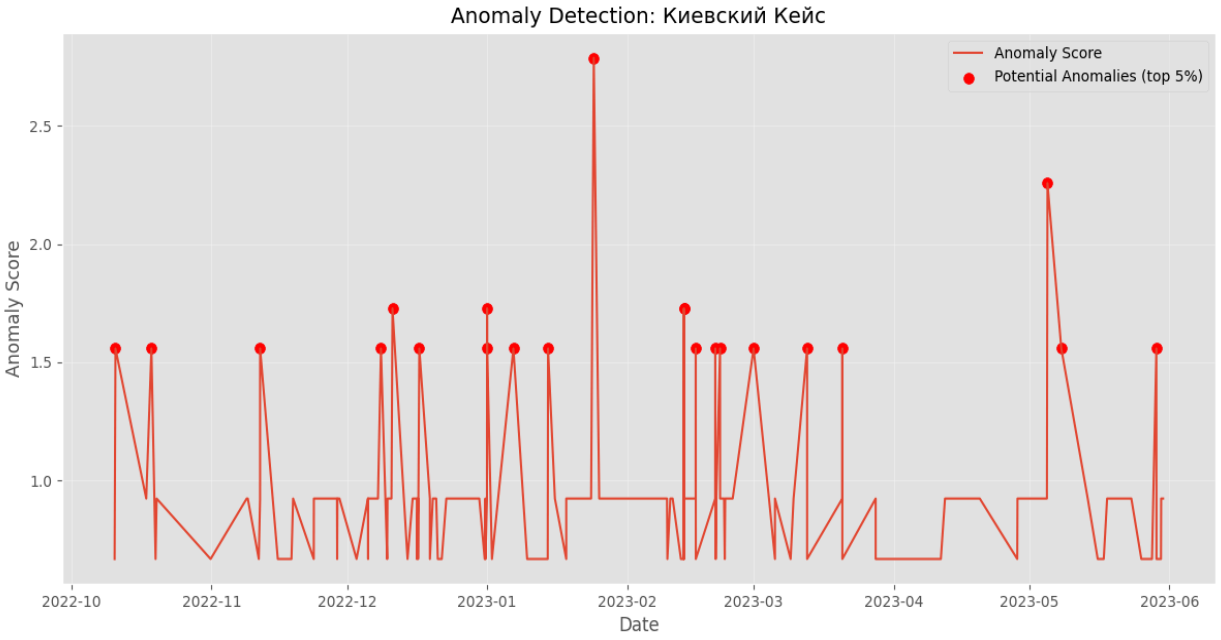


Рисунок 3.11 - Приклад графіку потенційних аномалій

- кореляція між сентиментом і аномаліями. Приклад (Рисунок 3.12).

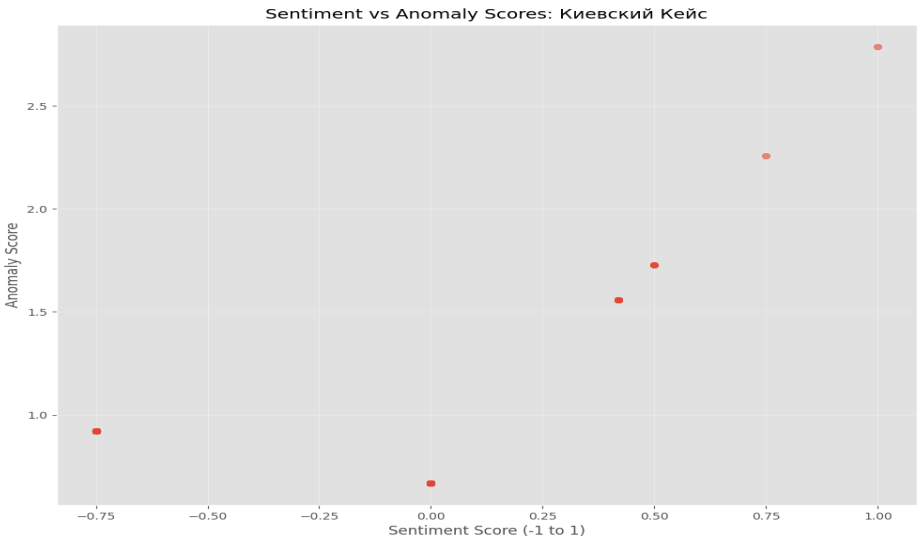


Рисунок 3.12 - Приклад графіку кореляції аномалій і сентименту

Також додатково були створені CSV файли:

- файл з потенційними аномаліями;
- файл з оціненим кожним повідомленням;

Вище перераховані модулі були об’єднані в фінальний файл disinfo_detector.py (Рисунок 3.13).

```
def parse_arguments():
    parser = argparse.ArgumentParser(description='Detect coordinated disinformation campaigns')
    parser.add_argument('--input', required=True, nargs='+', help='Paths to JSON feed files')
    parser.add_argument('--max-messages', type=int, help='Maximum number of messages to process per feed')
    parser.add_argument('--start-date', help='Filter messages by start date (ISO 8601 format)')
    parser.add_argument('--end-date', help='Filter messages by end date (ISO 8601 format)')
    parser.add_argument('--disable-emoji-analysis', action='store_true', help='Disable emoji reaction analysis')
    parser.add_argument('--anomaly-method', choices=['zscore', 'moving_average', 'isolation_forest'],
                        default='zscore', help='Anomaly detection method')
    parser.add_argument('--time-window', default=None, help='Time window for grouping (e.g., "1H", "1D")')
    parser.add_argument('--model-path', default='mistral-7b-instruct-v0.2.Q4_K_M.gguf',
                        help='Path to the LLM model file')

    return parser.parse_args()
```

Рисунок 3.13 - Реалізація модуля запуску

В рамках цього файла були створені такі параметри запуску (Таблиця 3.4).

Таблиця 3.4 - Параметри запуску моделі

Параметр	Обов’язковий	Тип/Значення	Стандартне значення	Детальна інструкція
--input	Так	str	—	Один або кілька шляхів до файлів JSON-каналу, що містять необроблені повідомлення.

Продовження таблиці 3.4

--max-messages	Hi	int	—	Верхня межа кількості повідомлень, що можуть бути прийняті за один канал.
--start-date	Hi	ISO-8601 date/time string	—	Ігнорувати повідомлення, опубліковані до цього часу.
--end-date	Hi	ISO-8601 date/time string	—	Ігнорувати повідомлення, опубліковані після цього часу.
--disable-emoji-analysis	Hi	флаг (True/False)	False	Якщо є, пропустіть крок, який розглядає реакції емодзі як потенційні сигнали.
--anomaly-method	No	z score,moving average, isolation forest	z score	Вибирає статистичний метод, що використовується для визначення аномальної активності.
--time-window	Hi	1H чи 1D	без групування	Групує повідомлення у фіксовані часові періоди перед виявленням аномалій (наприклад, «1H» = щогодини, «1D» = щодня).

Кінець таблиці 3.4

--model-path	Hi	str (file path)	mistral-7b-instruct-v0.2.Q4_K_M.gguf	Розташування у файловій системі локальних ваг LLM

3.5 Перевірка роботи і аналіз результатів

Для оцінки ефективності розробленого прототипу ми протестуємо його в двох різних реальних сценаріях. Перший передбачає аналіз Telegram-каналу, який був публічно викритий за участь у скоординованих кампаніях з дезінформації. Цей випадок є надійним еталоном для оцінки здатності системи виявляти відомі шаблони маніпулятивного контенту. Прототип буде застосовано до історичних даних цього каналу, щоб спостерігати, наскільки точно він ідентифікує аномалії настроїв, лінгвістичні маркери або інші показники, які зазвичай асоціюються з навмисною дезінформацією. Порівнюючи отримані результати з документальними доказами, ми зможемо визначити точність, чутливість та відповідність моделі раніше перевіреним висновкам.

Другий сценарій зосереджений на Telegram-каналах, щодо яких немає конкретних доказів, що підтверджують їхню участь у дезінформаційних кампаніях. Ці канали можуть містити порушення або підозрілі наративи, але без публічного підтвердження або перевірки вони залишаються некласифікованими. Цей контекст є корисним для оцінки поведінки системи в менш однозначних випадках. Прототип аналізуватиме вміст цих каналів, щоб визначити, чи

з'являються якісь тонкі закономірності або зміни, що можуть свідчити про скоординовані маніпуляції. Це тестування допоможе оцінити здатність моделі уникати помилкових спрацьовувань і позначати вміст лише в разі значних відхилень у настроях або контекстуальних аномалій.

Разом ці два випадки дадуть комплексне уявлення про ефективність прототипу, що допоможе забезпечити його здатність надійно розрізняти явну дезінформаційну діяльність та неоднозначні або нешкідливі інформаційні потоки.

3.5.1 Аналіз одного каналу

Для перевірки роботи був обраний Telegram канал “Киевский Кейс” [16], який був викритий у використанні стратегії 60/40. А саме розглянемо інформаційну атаку на фоні обстрілу енергетичної інфраструктури 10.10.2024. При мануальному аналізі канал було виявлено, що в основному пересилаються повідомлення з офіційних джерел в нейтральному тоні. Але час від час прослідковуються різко негативні повідомлення (Рисунок 3.14).

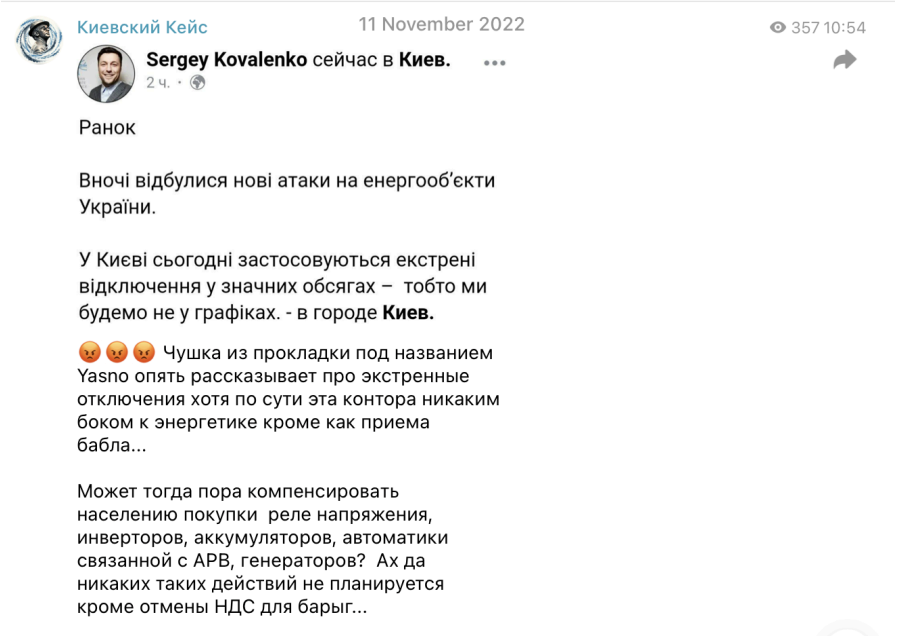


Рисунок 3.14 - Приклад маніпулятивного негативного повідомлення

При аналізі даних отримали таку картину зміни сентимента агрегована по дням (Рисунок 3.15). Як можемо бачити присутні постійні стрибки сентименту.

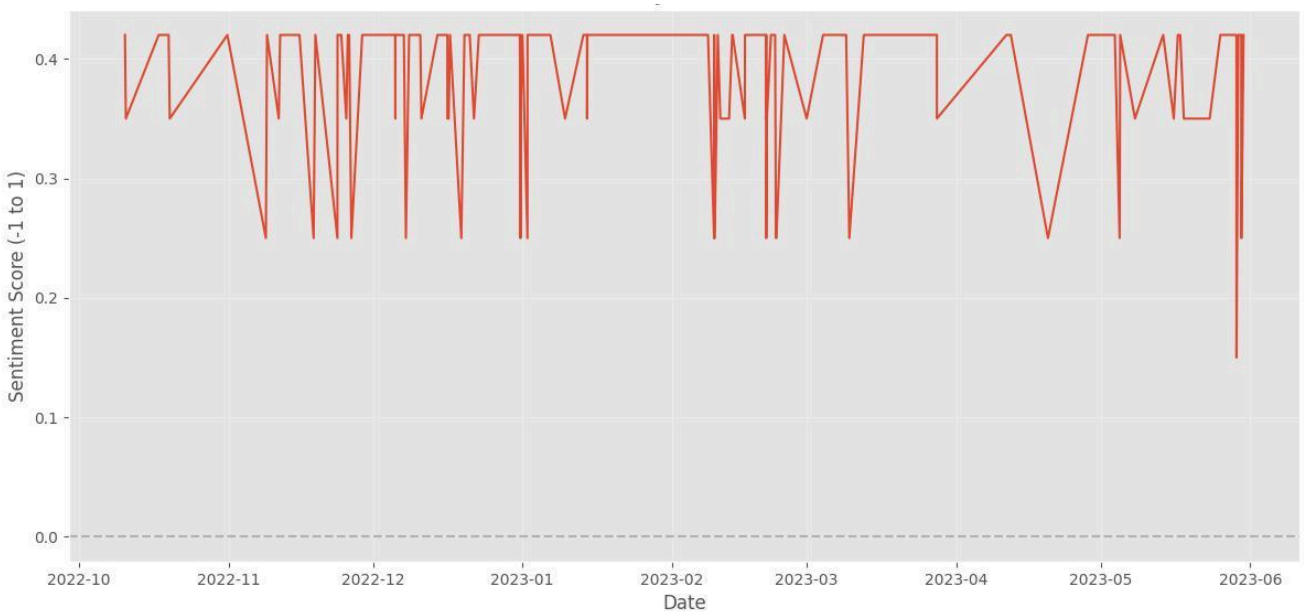
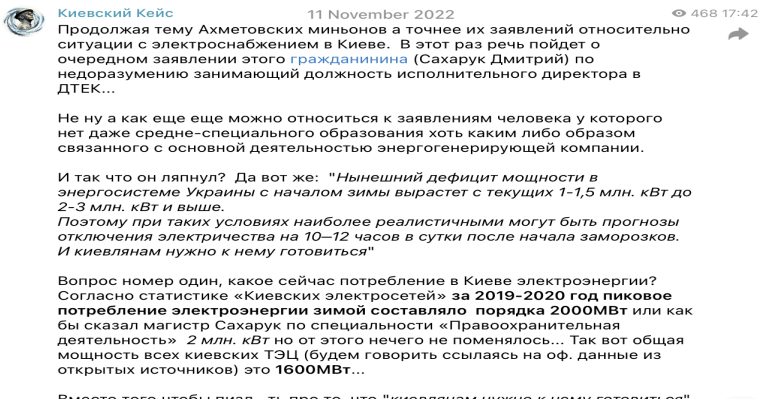


Рисунок 3.15 - Зміна сентимента з часом “Киевский Кейс”

Розглянемо перше різке падіння сентимента приблизно на початку листопада 2022. Бачимо серію маніпулятивних повідомлень від анонімних авторів, наприклад Рисунок 3.16.


Киевский Кейс 11 November 2022 468 17:42

Продолжая тему Ахметовских миньонов а точнее их заявлений относительно ситуации с электроснабжением в Киеве. В этот раз речь пойдет о очередном заявлении этого гражданинина (Сахарук Дмитрий) по недоразумению занимающий должность исполнительного директора в ДТЭК...

Не ну а как еще еще можно относиться к заявлениям человека у которого нет даже средне-специального образования хоть каким либо образом связанного с основной деятельностью энергогенерирующей компании.

И так что он ляпнул? Да вот же: "*Нынешний дефицит мощности в энергосистеме Украины с началом зимы вырастет с текущих 1-1,5 млн. кВт до 2-3 млн. кВт и выше.*
Поэтому при таких условиях наиболее реалистичными могут быть прогнозы отключения электричества на 10–12 часов в сутки после начала заморозков.
И киевлянам нужно к нему готовиться"

Вопрос номер один, какое сейчас потребление в Киеве электроэнергии? Согласно статистике «Киевских электросетей» **за 2019-2020 год пиковое потребление электроэнергии зимой составляло порядка 2000МВт** или как бы сказал магистр Сахарук по специальности «Правоохранительная деятельность» *2 млн. кВт* но от этого нечего не поменялось... Так вот общая мощность всех киевских ТЭЦ (будем говорить ссылаясь на оф. данные из открытых источников) это **1600МВт**...

Рисунок 3.16 - Приклад маніпулятивного повідомлення

Переглянемо результати визначення аномалій (Рисунок 3.17). Як бачимо присутні два маніпулятивні повідомлення з негативним забарвленням і одне позитивне, яке не є дезінформацією.

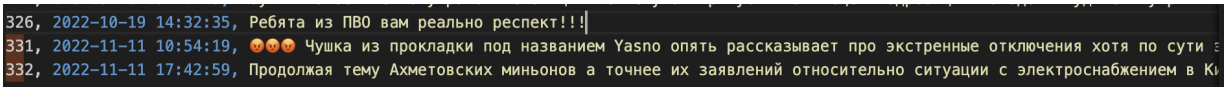


Рисунок 3.17 - Приклад результатів визначених аномалій

В загальному отримали такий результат з використанням відповідних методів виявлення аномалій:

- Z-score (Таблиця 3.5);

Таблиця 3.5 - Матриця невідповідностей для Z-score

	Прогнозовано = Дезінформація	Прогнозовано = Правдиве
Фактичне = Дезінформація	42	22
Фактичне = Правдиве	20	76

- Moving average (Таблиця 3.6);

Таблиця 3.6 - Матриця невідповідностей для Moving average

	Прогнозовано = Дезінформація	Прогнозовано = Правдиве

Кінець таблиці 3.6

Фактичне = Дезінформація	35	29
Фактичне = Правдиве	15	81

- Isolation Forest (Таблиця 3.7).

Таблиця 3.7 - Матриця невідповідностей для Isolation Forest

	Прогнозовано = Дезінформація	Прогнозовано = Правдиве
Фактичне = Дезінформація	35	29
Фактичне = Правдиве	15	81

Оцінимо за метриками (Таблиця 3.8)

Таблиця 3.8 - Метрики визначення аномалій

Method	**Precision**	**Recall**	**F1-score**
Z-score	0.68	0.66	0.67
Moving average	0.70	0.55	0.61
Isolation F.	0.83	0.78	0.81

Як бачимо для данного канала Isolation Forest показав найкращий результат на рівні 80%, що доводить його ефективність.

3.5.2 Оцінка з реакціями

Для аналізу був обраний канал “Всевидящее ОКО” [17]. Цей канал був помічений у розповсюдженні неправдивої інформації пов’язаної з діями ТЦК.

Було проаналізовано 5715 повідомлень за період 12.2023-03.2024. Проаналізуємо графік зміни сентименту (Рисунок 3.18) .

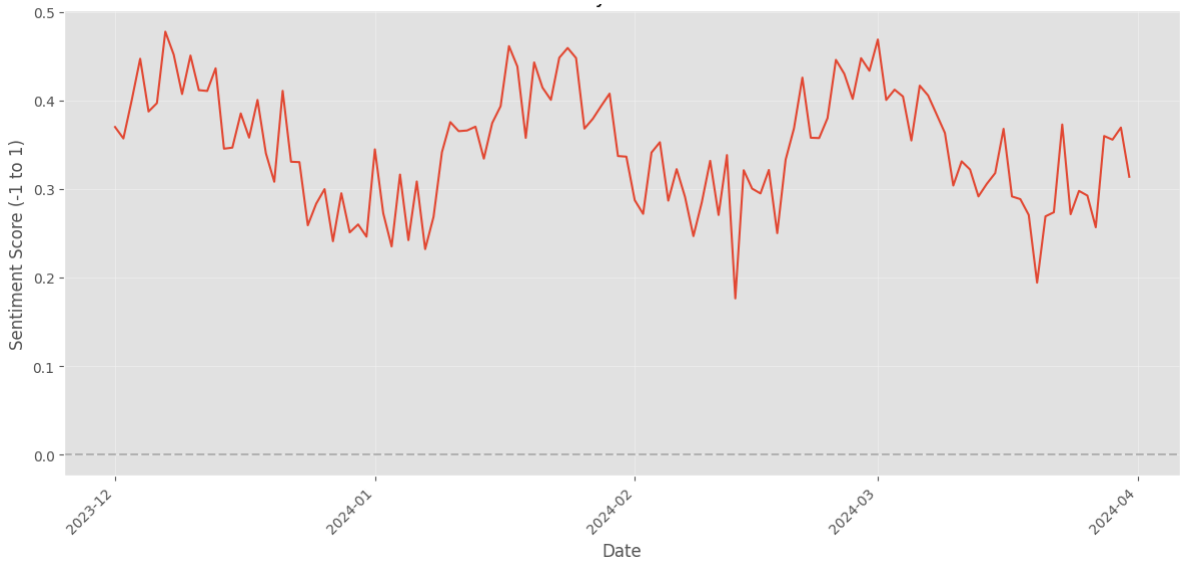


Рисунок 3.18 - Графік зміни сентимента каналу “Всевидящее ОКО”

Як бачимо графік більш плавний, немає чітких спайків. Через великий обсяг записів та відсутність чітко визначеного набору еталонних даних про дезінформацію, буде проведено поверхневу ручну перевірку результатів. Аналіз показав що немає чіткого патерну дезінформації. В фінальному списку аномалій більшість новин були правдими і нейтрально викладеними. Приклад Рисунок 3.19.

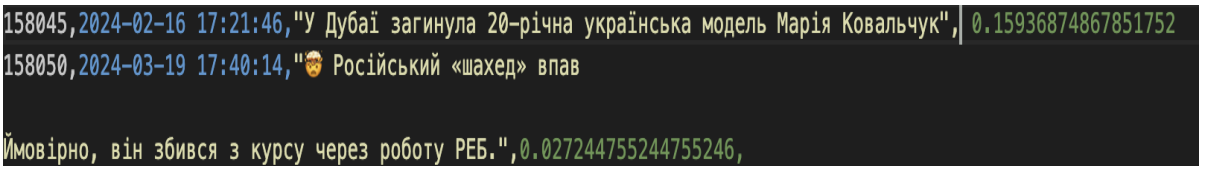


Рисунок 3.19 - Хибно визначені аномалії

Отже даний метод не можна використовувати без різкої зміни сентименту.

Висновки до розділу 3

У третьому розділі було продемонстровано практичне впровадження та оцінку системи, призначеної для виявлення дезінформації за допомогою аналізу емоцій, з особливим акцентом на перевагах використання великих мовних моделей (LLM) над класичними інструментами аналізу емоцій. Емпіричні результати чітко підтверджують гіпотезу, що LLM значно перевершують традиційні методи, такі як VADER і TextBlob, як за точністю, так і за надійністю. У той час як класичні моделі покладаються на статичні лексикони та поверхневі підходи, засновані на правилах, LLM використовують глибоке розуміння контексту, вловлюючи тонкі нюанси мови, включаючи сарказм, риторичну структуру, заперечення та емоційно забарвлені ідіоми — елементи, які часто використовуються в дезінформаційних наративах.

Експериментальні результати, засновані на наборі даних із 25 000 записів, показали, що LLM досягли значно вищої точності (0,88) та F1-показника за всіма метриками порівняно з конкуруючими моделями, які отримали результати в діапазоні 0,65–0,70. Ця різниця в продуктивності була особливо очевидною в сценаріях із складними або неоднозначними мовними конструкціями. Хоча LLMs вимагали значно більше обчислювальних ресурсів — як з точки зору пам'яті, так і часу обробки — їхня інтерпретаційна здатність та зменшення кількості помилкових позитивних і негативних результатів виправдовують збільшення

накладних витрат, особливо в критичних випадках, таких як кібербезпека та моніторинг стратегічних комунікацій.

На основі цього було розроблено прототип системи для виявлення аномальних настроїв у каналах Telegram. Ця система працює на основі припущення, що скоординовані кампанії дезінформації часто викликають неорганічні коливання настроїв — різкі емоційні сплески, раптові зміни полярності або синхронізовані зміни повідомлень у різних джерелах. Використання LLM покращило здатність системи виявляти такі аномалії, пропонуючи глибшу семантичну інтерпретацію повідомлень і точніший графік еволюції настроїв.

У висновку, дослідження, проведене в цьому розділі, підтверджує, що LLM є не тільки життєздатними, але й дуже вигідними інструментами для аналізу настроїв при виявленні дезінформації. Їх здатність узагальнювати між різними сферами та адаптуватися до динамічної природи мови дає їм явну перевагу в виявленні нових загроз в інформаційних середовищах. Розроблена система є доказом концепції для майбутнього розширення, що свідчить про те, що LLM можуть відігравати вирішальну роль в автоматизації аналізу загроз, механізмів раннього попередження та стратегій цифрової стійкості. Результати дослідження також підкреслюють важливість поєднання передових моделей ШІ з логікою, специфічною для конкретної галузі, для створення систем, які є інтелектуальними та оперативно значущими в контексті національної кібербезпеки.

4 РОЗРОБКА СТАРТАП-ПРОЄКТУ

4.1 Опис ідеї проєкту

Концепція стартап-проекту виникла з нагальної та зростаючої потреби протидіяти кампаніям дезінформації, які використовують сучасні комунікаційні платформи, зокрема Telegram-канали, із дедалі більш витонченими стратегіями. У контексті гібридної війни та операції впливу, що фінансуються державою, дезінформація вже не обмежується поширенням неправдивої інформації; вона еволюціонувала до ретельно організованої маніпуляції емоційним тоном, вибором часу та фактичним нагромадженням. Одним із найпідступніших проявів цього є стратегія 60/40, за якою 60 % повідомлення є фактично правильним, а 40 % — тонко маніпулятивним або навмисно неправдивим. Виявлення таких тонких обманів вручну є трудомістким і ненадійним у великих масштабах, особливо в швидкозмінних інформаційних середовищах.

Проект пропонує розробку масштабованої автоматизованої системи для виявлення скоординованих кампаній дезінформації на основі аналізу зміни настроїв за допомогою великих мовних моделей (LLM). Основна інновація цієї ідеї полягає в поєднанні найсучасніших методів розуміння природної мови з аналізом настроїв у часових рядах для виявлення емоційних аномалій в інформаційних потоках. Ці аномалії часто передують, супроводжують або є результатом маніпулятивних зусиль, спрямованих на формування громадської думки або викликання психологічної втоми.

Основна ідея полягає в побудові системи, яка приймає потоки повідомлень Telegram, обробляє їх за допомогою класичних методів аналізу настроїв (наприклад, VADER, TextBlob) та сучасних LLM (наприклад, llama-pro-8b-instruct.Q5_K_M) та виявляє статистично значущі відхилення в

траєкторії настроїв. Ці відхилення можуть сигналізувати про потенційні кампанії дезінформації, особливо коли вони раптові, синхронізовані між декількома каналами або надзвичайно поляризовані. Корелюючи сплески настроїв з обсягом повідомлень та наративним фреймінгом, система прагне розрізнити органічні емоційні реакції та організовану маніпулятивну діяльність.

Рішення стартапу спрямовано на усунення критичної прогалини в зусиллях з національної безпеки та медіаграмотності: відсутність адаптованих до конкретної галузі інструментів для виявлення прихованих операцій впливу, що експлуатують емоційну нестабільність. На відміну від більшості існуючих платформ виявлення дезінформації, які покладаються на перевірку фактів або зіставлення ключових слів, ця система працює на більш глибокому рівні — виявлення емоційної невідповідності та аномалій — що робить її стійкою до тактик заплутування, таких як лексичні варіації, регіональний сленг та наративні рамки.

Запланована платформа включає кілька компонентів:

- Механізм аналізу настроїв: двошарова система, що використовує як класичні моделі, так і точно налаштовані LLM для оцінки вхідного тексту за полярністю настроїв та емоційною валентністю.
- Модуль виявлення аномалій: конвеєр аналізу часових рядів, що відстежує зміни настроїв, виявляє аномальні піки або спади та позначає патерни, що не відповідають органічному дискурсу.
- Агрегатор Telegram-каналів: система збору даних, здатна витягувати повідомлення, метадані та часові мітки з публічних Telegram-каналів у великих обсягах.
- Панель візуалізації: інтерактивний інтерфейс, що відображає настрої в динаміці, позначені аномалії та відбитки настроїв для конкретних каналів для розслідування та аудиту.
- Набір інструментів для налаштування моделі: утиліта для створення підказок та налаштування моделі, що дозволяє експертам у певній галузі

вдосконалити поведінку моделі для конкретних випадків використання або регіональних мовних відмінностей.

Технічна реалізація системи вже була продемонстрована в експериментах з перевірки концепції, описаних у розділі 3. Ці експерименти показали, що LLM значно перевершують класичні моделі як за точністю, так і за показником F1 при класифікації настроїв, навіть при роботі з зашумленими даними соціальних мереж. Більше того, у поєднанні з порівнянням настроїв у часовому вікні по декількох каналах, система змогла точно визначити періоди емоційних сплесків та поляризації, які збігалися з відомими кампаніями дезінформації, наприклад, під час ракетних ударів по Україні 10 жовтня 2022 року.

Цей проєкт повністю відповідає принципам етичного використання ШІ, цілісності ЗМІ та національної кібербезпеки. Він є масштабованим, адаптованим до багатьох мов (завдяки багатомовному навчанню LLMs) та може працювати на стандартному обладнанні з використанням квантованих моделей, що дозволяє розгортати його в умовах обмежених обчислювальних ресурсів.

З бізнес-точки зору, платформа має ринковий потенціал у декількох сферах:

Державний сектор: уряди та розвідувальні служби, які шукають системи раннього попередження про скоординовані операції впливу.

- Журналістські організації: ЗМІ та команди з перевірки фактів, яким необхідний аналіз емоційного сліду наративів.
- НУО та аналітичні центри: Інституції, що моніторять загрози інформаційному простору, пов'язані з демократією, громадським здоров'ям або національною безпекою.
- Освіта та цифрова грамотність: Інструменти для демонстрації в режимі реального часу, як наративи емоційно маніпулюються в різних джерелах.

4.2 Технологічний аудит ідеї проєкту

У цьому розділі представлено технологічний аудит запропонованої системи виявлення дезінформації на основі аналізу зміни настроїв за допомогою великих мовних моделей. Мета аудиту — оцінити доцільність, ефективність та придатність обраних технологій з точки зору продуктивності, використання ресурсів, розширюваності та сумісності з цілями проекту.

Система повинна:

- Збирати повідомлення з Telegram-каналів.
- Аналізувати настрої повідомлень за допомогою класичних моделей та LLM.
- Виявляти аномалії в динаміці настроїв у часі.
- Візуалізувати результати аналізу для експертної оцінки.
- Використовувані технології

Технології, що використовуються в системі, класифіковано в Таблиці 4.1:

Таблиця 4.1 – Підсумок технологічних компонентів

Компонента	Технологія	Призначення
LLM	llama-pro-8b-instruct.Q5_K_M (GGUF)	Класифікація сентиментів за допомогою квантованого LLM
Класичні моделі сентименту	VADER, TextBlob	Базова класифікація настроїв
Збирач даних Telegram	Telethon	Витяг повідомлень та метаданих з Telegram
Статистичний аналіз	NumPy, Pandas, Scipy	Сегментація часових рядів та виявлення аномалій
Візуалізація	Matplotlib, Plotly	Відображення змін настроїв та позначених аномалій

Кінець таблиці 4.1

Інтерфейс	Streamlit	Інтерактивна панель інструментів (не обов'язкова для використання CLI)

LLM було обрано на основі його здатності працювати локально з використанням ресурсів процесора. Модель використовує квантований 5-бітний формат (Q5_K_M) для ефективного виведення. Параметри, такі як кількість потоків, розмір контекстного вікна та значення початкового значення, можна налаштувати. Модель розгортається за допомогою llama-cpp-python, що дозволяє асинхронне генерування токенів без залежності від GPU.

Проєкт організовано у вигляді модульних сервісів (Таблиця 4.2):

Таблиця 4.2 – Логічна структура системи

Модуль	Функція
Збирач даних	Отримати повідомлення з Telegram
Аналізатор настроїв	Застосовуйте LLM до кожного повідомлення
Детектор аномалій	Виявляйте відхилення в тенденціях настроїв
Візуалізатор	Представлення даних у вигляді графіків

Технологічний аудит підтверджує, що обрана архітектура та інструменти підходять для запропонованої системи виявлення дезінформації. Використання LLMs значно покращує аналітичну точність, а легкі методи квантування дозволяють виконувати локальне виконання на скромному апаратному забезпеченні. Класичні моделі залишаються корисними як еталони або запасні

варіанти. Система є технічно здійсненною, масштабованою та відповідає цілям проекту в галузі безпеки та досліджень.

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Зростання частоти та витонченості кампаній з дезінформації, особливо тих, що спрямовані на критичні соціальні, політичні та військові наративи, створило високий попит на передові технології виявлення. Традиційні інструменти, такі як ручна перевірка фактів, виявлення ключових слів та позначення вмісту, вже не є достатніми для боротьби з гібридною інформаційною війною, яка швидко еволюціонує. Оскільки проект зосереджений на виявленні аномалій на основі аналізу настроїв за допомогою локальної обробки LLM, він має конкурентну перевагу, що відповідає поточним потребам як державного, так і приватного секторів. У цьому розділі надано огляд ринкового попиту, потенційних клієнтів, конкурентного середовища та шляхів комерціалізації.

Запропонована система займає унікальне місце на перетині кібербезпеки, моніторингу ЗМІ та штучного інтелекту. Оскільки дезінформація все частіше стає інструментом впливу як у військовому, так і в політичному контексті, урядові установи, НУО та міжнародні організації активно шукають масштабовані рішення для моніторингу та протидії таким загрозам. Більше того, використання динаміки настроїв замість ключових слів у контенті забезпечує стійкість до таких методів ухилення, як лексичне маскування, кодована мова та переключення між мовами.

Основна цінність платформи полягає в її здатності працювати локально на обмеженому апаратному забезпеченні, забезпечуючи при цьому високу точність аналізу траєкторії настроїв у часі та каналах. Те, що система не вимагає хмарних API, передплати на власні послуги LLM або завантаження конфіденційних даних, робить її придатною для середовищ з високим рівнем безпеки, включаючи оборону, моніторинг виборів та захист критичної інфраструктури.

Порівняльна оцінка суміжних інструментів та послуг на сучасному ринку підкреслює унікальне позиціонування цього рішення (Таблиця 4.3):

Таблиця 4.3 – Порівняння з існуючими ринковими рішеннями

Характеристика	Існуючі інструменти (CrowdTangle, NewsGuard)	Система зсуву сентименту на базі LLM
Виявлення емоційних змін	Обмежене або відсутнє	Основна функціональність
Інтерпретованість мовної моделі	Недоступна	Налаштовується через управління підказками
Локальне виконання (без потреби в API)	Ні	Так
Підтримка кількох мов і регіонів	Обмежена	Включено через вибір моделі
Обробка в реальному часі	Часткова	Повна (пакетна та потокова обробка)
Прозорість для аудиту	Мінімальна	Повний контроль підказок і результатів

Ринкові можливості особливо великі в регіонах, де ведеться активна інформаційна війна. Україна, країни Балтії та інші країни Східної Європи, які стали об'єктом скоординованих операцій іноземного впливу, є найближчими прикладами застосування. Крім того, інтерес з боку академічних установ та центрів політичних досліджень створює можливості для використання в освітніх та некомерційних цілях.

Цей продукт може бути корисним для декількох галузей промисловості. До них належать компанії з кібербезпеки, що займаються аналізом цифрових загроз, компанії з моніторингу ЗМІ, що відстежують тенденції громадської думки, міжнародні спостерігачі за виборами, які потребують зворотного зв'язку в режимі

реального часу, та урядові органи, що займаються нарративним наглядом. Адаптивність рішення дозволяє розширити його на інші платформи, крім Telegram, включаючи X (Twitter), коментарі YouTube та форуми з низьким рівнем модерації.

У таблиці нижче наведено потенційні сегменти клієнтів та відповідні випадки використання (Таблиця 4.4):

Таблиця 4.4 - Цільові сегменти клієнтів та випадки використання

Сегмент клієнтів	Приклад використання
Державні установи	Виявлення психологічних операцій, спрямованих на громадянське суспільство
НУО та дослідницькі центри	Моніторинг цілісності інформації та впливу на політику
Кібербезпекові компанії	Інтеграція сповіщень про дезінформацію в загальні моделі загроз
Новинні організації	Аналіз емоційного оформлення в конкурентній боротьбі ЗМІ
Освітні установи	Навчальні інструменти для медіаграмотності та аналізу інформації

З точки зору стратегії виходу на ринок, початкова версія системи може бути запропонована як дослідний прототип з відкритим кодом з можливістю професійного консультування, налаштування та ліцензування для інституційних користувачів. Це зменшить початкові бар'єри, одночасно підвищуючи довіру та підтверджуючи ефективність використання. Може бути застосована модель подвійного ліцензування — безкоштовна для академічного/некомерційного використання, платна для комерційного/державного впровадження.

З огляду на зростання ролі управління штучним інтелектом, регулювання ЗМІ та політики дотримання вимог щодо цифрової інформації, платформа може також слугувати допоміжною технологією для забезпечення дотримання

законодавчих вимог. Нещодавно прийняті європейські закони про цифрові послуги та штучний інтелект вимагають від платформ та постачальників моделей усунення ризиків дезінформації. Цей інструмент може допомогти установам виконувати ці зобов'язання завдяки задокументованим процесам виявлення та можливостям звітування.

Система відповідає конкретним і зростаючим потребам у сфері аналізу дезінформації, пропонуючи як технологічні, так і операційні переваги. Її унікальна орієнтація на емоційну маніпуляцію, локальне виконання та виявлення аномалій у різних каналах робить її привабливою для широкого кола клієнтів. Завдяки низьким початковим витратам та гнучким моделям інтеграції, проект має всі передумови для цільового впровадження у секторах з високим рівнем впливу.

4.4 Розробка ринкової стратегії проєкту

Запуск технологічного продукту, особливо в такій чутливій і швидко розвиваючійся галузі, як інформаційна безпека та виявлення дезінформації, вимагає ретельно продуманої стратегії виходу на ринок. У цьому розділі описано стратегічний підхід до впровадження розробленої системи виявлення аномалій на основі аналізу настроїв на відповідних ринках. Основна увага приділяється визначенню пріоритетів запуску, мінімізації бар'єрів для входу на ринок, формуванню довіри користувачів та створенню стійких каналів для масштабування.

Основною метою ринкової стратегії є позиціонування системи не як загального інструменту виявлення дезінформації, а як спеціалізованої платформи, здатної виявляти скоординовані емоційні маніпуляції в каналах обміну повідомленнями, зокрема Telegram. Ця відмінність повинна бути чітко зазначена в усіх матеріалах та інформаційних кампаніях. На відміну від конкурентів, які

зосереджуються виключно на аналізі контенту або метаданих, система забезпечує рівень афективної інтелектуальної обробки та відстеження динаміки наративів.

Початкове впровадження повинно бути орієнтоване на надійних інституційних клієнтів в академічних, оборонних або державних екосистемах. Ці організації, як правило, відкриті для пілотних проектів, надають доступ до реальних даних і можуть надати важливий для вдосконалення системи зворотний зв'язок з конкретної галузі. Крім того, вони, як правило, надають пріоритет прозорості, можливості аудиту та етичному поводженню з даними — сферам, в яких система має явну перевагу завдяки своїм локальним можливостям обробки та налаштовуваній логіці висновків.

Стратегічний фокус також повинен включати створення спільноти серед дослідників відкритого програмного забезпечення та фахівців з кібербезпеки. Ранній доступ до дослідницької версії, супроводжуваний документацією та відтворюваними експериментами, створить екосистему учасників, тестувальників та валідаторів. Цей підхід виявився ефективним у сфері кібербезпеки та впровадженні інструментів НЛП, особливо коли важливими є довіра та пояснюваність.

Основні канали просування та розповсюдження включають прямі партнерства з академічними та дослідницькими установами, презентації на конференціях у галузі кібербезпеки та дослідження дезінформації, а також інтеграцію в набори інструментів OSINT. Для підвищення видимості основної цінності інструменту — виявлення емоційних аномалій — слід підготувати серію технічних документів на основі конкретних прикладів, в яких буде продемонстровано реальний аналіз відомих кампаній дезінформації, таких як скоординовані повідомлення після ракетних ударів по Україні 10 жовтня 2022 року.

Рекомендується поетапна модель впровадження. Початковий етап включає внутрішнє тестування та закриті пілотні проекти з академічними та урядовими партнерами. Другий етап передбачатиме публічний випуск аналітичного двигуна під ліцензією з відкритим кодом, з опціональними пакетами підтримки та

комерційними доповненнями. Третій етап буде присвячений дослідженню хмарних версій, інтеграції на основі API для платформ моніторингу та моделей передплати для постійних оновлень.

У таблиці нижче наведено запропонований план дій для запуску системи:

Таблиця 4.5 – План дій щодо ринкової стратегії

Етап	Опис	Часовий період
Етап 1	Внутрішнє тестування та пілотні програми для академічних і держустанов	Місяці 1–3
Етап 2	Публічний реліз з відкритим кодом на GitHub, документація та CLI	Місяці 4–6
Етап 3	Запуск опціональної підтримки, UI-дашборду та інтеграційних плагінів	Місяці 6–9
Етап 4	Комерційне ліцензування, white-label-рішення та хмарний варіант	Місяці 10–12

Паралельно необхідно працювати над усуненням потенційного опору та юридичних ризиків. Зокрема, обробка даних користувачів, навіть якщо вони є публічними, повинна відповідати GDPR та національному законодавству про кібербезпеку. Архітектура повинна і надалі уникати будь-якого централізованого зберігання вмісту Telegram або особистих ідентифікаторів. Уся обробка повідомлень повинна здійснюватися в пам'яті або через анонімні записи.

Стратегія ціноутворення для комерційного використання повинна відображати нішевий характер продукту, залишаючись доступною для організацій державного сектору з обмеженим бюджетом. Пропонується багаторівнева модель ціноутворення, що передбачає безкоштовне використання для некомерційних та академічних цілей, а також ліцензування для державних органів та приватних компаній (Таблиця 4.6).

Таблиця 4.6 – Пропонована модель ліцензування

Рівень ліцензії	Права доступу	Цільові користувачі
Дослідницька ліцензія	Повний функціонал, лише локальна обробка	Університети, незалежні дослідники
Інституційний рівень	Комерційне використання з підтримкою та UI	Державні установи, неурядові організації
Корпоративний рівень	Індивідуальні функції, інтеграції, SLA	Кібербезпекові компанії, медіа-групи

Слід зазначити, що стартап-проект повинен впроваджуватися поступово, з акцентом на довіру, прозорість та перевірку в реальних умовах. Використання інтересу громадськості до дезінформації та інституційного попиту на проактивні засоби захисту сприятиме виходу на ринок. Успіх проекту залежатиме не тільки від його технічних можливостей, але й від послідовної стратегії, що поєднує інформаційну роботу, модульну структуру та етичне управління даними..

4.5 Розробка маркетингової програми стартап-проекту

Ефективність технологічного стартапу залежить не тільки від інноваційності його продукту, але й від успішності маркетингових заходів. У цьому розділі описано маркетингову програму для пропонованого стартапу, орієнтованого на виявлення скоординованої дезінформації за допомогою аналізу аномалій у настроях. Мета полягає в розробці цільової стратегії інформування, яка підвищить обізнаність, сприятиме швидкому впровадженню та позиціонуватиме систему як надійний і незамінний інструмент у боротьбі з маніпулюванням інформацією.

Перший компонент маркетингової програми — комунікація, орієнтована на контент. Оскільки система базується на академічних методах і працює у високотехнічній галузі, маркетингові матеріали повинні демонструвати як теоретичну обґрунтованість, так і практичну застосовність. Для пояснення основної методології, зокрема використання великих мовних моделей для виявлення змін емоційного тону з часом, будуть використовуватися технічні документи, кейси та технічні публікації в блогах. Акцент буде зроблено на унікальних випадках використання, таких як виявлення штучних сплесків емоцій після геополітичних подій або військових операцій.

Для охоплення цільової аудиторії проект використовуватиме канали, специфічні для даної галузі. До них належать конференції з кібербезпеки, інформаційних операцій, відкритих джерел інформації (OSINT) та цифрової журналістики. Прикладами є CONFidence, Black Hat Europe, CPDP та семінари з питань штучного інтелекту, організовані урядом або політичними організаціями. Участь у таких заходах включатиме живі демонстрації, технічні презентації та розповсюдження друкованих або цифрових матеріалів, що описують архітектуру системи, її переваги та перші результати.

Другий напрямок маркетингової програми буде зосереджений на партнерстві з академічними установами та аналітичними центрами. Ця співпраця матиме дві функції: перевірка інструменту в реальних умовах та формування репутації за допомогою відгуків, що пройшли рецензію. Академічна підтримка підвищує довіру, особливо серед потенційних клієнтів у державному секторі або некомерційній сфері, які можуть покладатися на оцінку, що базується на фактичних даних, перед закупівлею.

Видимість платформи також можна підвищити шляхом інтеграції з існуючими інструментами та платформами, які використовують аналітики OSINT та команди з аналізу загроз. Це включає внесення інструменту до реєстрів програмного забезпечення з відкритим кодом, надання плагінів для інтеграції з поширеними платформами, такими як Maltego або MISP, та створення коннекторів для робочих процесів на базі Jupyter.

Паралельно будуть проводитися цільові цифрові кампанії. Вони будуть включати поєднання розсилки електронних листів, професійних мереж, таких як LinkedIn, та розміщення в ЗМІ, що спеціалізуються на кібербезпеці та технологіях. Пріоритет буде надаватися статтям про лідерство думок, які підкреслюють перехід від виявлення на основі фактів до аналізу на основі емоцій як можливості нового покоління.

У наступній таблиці (Таблиця 4.7) наведено основні канали та формати, які будуть використовуватися:

Таблиця 4.7 – Маркетингові канали та формати

Тип каналу	Формат	Цільова аудиторія
Конференції	Живі демонстрації, технічні доповіді	Дослідники, уряд, фахівці з кібербезпеки
Публікації	Аналітичні документи, блоги, дослідницькі нотатки	Аналітики, журналісти
Цифрові кампанії	LinkedIn, розсилки, електронна пошта	OSINT-фахівці, технічні керівники
Академічні партнерства	Спільні дослідження, валідація інструментів	Університети, аналітичні центри
Реєстри програмного забезпечення	GitHub, PyPI, OSINT-каталоги	Розробники, аналітики, інтегратори

Брендинг буде підкреслювати основні принципи довіри, незалежності та інсайдерської інформації. Здатність системи працювати локально, не покладаючись на комерційні хмарні API, забезпечує перевагу в плані збереження конфіденційності, про яку необхідно чітко інформувати. Маркетингові повідомлення не будуть містити загальних тверджень про дезінформацію, а натомість підкреслюватимуть конкретні функціональні переваги: локалізований

LLM-інференс, виявлення аномалій за допомогою відстеження настроїв та підтримка мов з обмеженими ресурсами, таких як українська або російська.

Ранні користувачі будуть заохочуватися за допомогою обмежених у часі пропозицій, таких як безкоштовна допомога в налаштуванні, навчальні семінари та пілотні програми під спільним брендом. Ці заходи покликані створити кейси, які будуть використані в наступному маркетинговому циклі та посилять залучення спільноти.

Структура цін на продукт також буде сформульована в рамках маркетингового повідомлення, підкреслюючи доступність для академічного та некомерційного секторів, а також модульну можливість оновлення для корпоративних клієнтів (Таблиця 4.8).

Таблиця 4.8 – Огляд графіку просування

Місяць	Опис активності
1–2	Підготовка аналітичних документів, презентацій та реліз на GitHub
3–4	Запуск email-кампанії, початок контактів з дослідниками
5–6	Виступ на першій конференції, публікація серії блог-постів
7–8	Початок інтеграції з OSINT-платформами та запрошення бета-тестерів
9–12	Підготовка кейсів, просування в медіа, розширення спільноти

Підсумовуючи, маркетингова програма базується на надійності, технічній прозорості та відповідності реальним потребам. Завдяки орієнтації на правильні спільноти, пропонування чіткої цінності та підкріплення кожного твердження доказами, проект може завоювати довіру та забезпечити прийняття в галузі, де

скептицизм щодо рішень на основі штучного інтелекту є високим. Довгостроковий успіх залежатиме від здатності підтримувати взаємодію за допомогою якісного контенту, надійних партнерських відносин та вимірюваного впливу.

Висновки до розділу 4

Цей дослідницький проект був присвячений актуальній і багатогранній проблемі виявлення кампаній дезінформації в сучасних цифрових екосистемах, з особливим акцентом на каналах Telegram. Ключовою запропонованою та реалізованою інновацією стала система виявлення скоординованої маніпуляції інформацією, що базується не лише на фактичних неточностях, а й на динамічному аналізі емоційного тону в часі, використовуючи можливості великих мовних моделей для класифікації емоцій та виявлення аномалій.

У першій частині дисертації було закладено теоретичні основи роботи. Глибокий огляд LLM продемонстрував їхню здатність розуміти контекст, виявляти тонкі емоційні сигнали та узагальнювати інформацію в різних мовних і тематичних сферах. Це було протиставлено класичним методам аналізу настроїв, які покладаються на статичні лексикони та не враховують контекст. Результати чітко показали, що LLM перевершують традиційні підходи за точністю, відтворюваністю та узагальнюваністю, що робить їх дуже придатними для виявлення неочевидних, емоційно-обумовлених моделей маніпуляцій.

У другій частині дисертації досліджено, як дезінформаційні кампанії використовують емоційну динаміку як зброю, контрольовано змінюючи настрої для дестабілізації громадської думки. Особлива увага була приділена пропагандистській техніці «60/40», в якій перевірені факти поєднуються з фальшивими наративами, щоб зберегти довіру, одночасно тонко змінюючи переконання. Аналіз історичних прикладів та емпіричної літератури показав, що

моніторинг змін настроїв, особливо раптових або синхронізованих, може слугувати надійним прокси-показником для виявлення маніпуляцій, навіть за відсутності явних неправд.

Третя частина була присвячена розробці та створенню прототипу системи. Архітектура поєднувала локальне виконання квантованих LLM, класичні базові показники NLP, аналіз часових рядів та візуалізацію. Експерименти на еталонних наборах даних підтвердили ефективність системи в класифікації настроїв, досягнувши показника F1 0,88 і значно перевершивши VADER та TextBlob. Незважаючи на вищі обчислювальні витрати, рішення на основі LLM виявилося придатним для локальної роботи на споживчому обладнанні, тим самим зберігаючи конфіденційність даних та оперативну незалежність.

В останній частині роботи було розглянуто практичне застосування та масштабованість проекту. Було запропоновано дорожню карту, орієнтовану на стартапи, в якій визначено ключові сегменти користувачів, стратегії виходу на ринок та моделі ліцензування. Особливу увагу було приділено етичному впровадженню, дотриманню законів про конфіденційність та відповідності реальним потребам у сферах кібербезпеки, цілісності інформації та державної політики. Технологічний та маркетинговий аудити підтвердили, що система є реалістичною та комерційно вигідною в її поточному стані.

У висновку даної роботи було продемонстровано, що аналіз настроїв за допомогою великих мовних моделей може слугувати потужним методом виявлення скоординованих спроб дезінформації, особливо в умовах, коли фактична перевірка є недостатньою. Зосередившись на афективних аномаліях та міжканальних емоційних тенденціях, система додає новий рівень стійкості до набору інструментів захисту інформації. Інтеграція NLP, принципів кібербезпеки та стратегії розвитку стартапів забезпечує повну основу не тільки для подальших академічних досліджень, але й для впровадження в реальних умовах. Майбутні роботи можуть включати розширення на інші мови, інтеграцію з іншими платформами та глибшу автоматизацію оцінки аномалій на основі гібридних сигналів, що поєднують сентимент, час повідомлення та метадані джерела.

ВИСНОВКИ

В результаті цієї магістерської роботи було визначено п'ять дослідницьких завдань, кожне з яких було повністю вирішено в ході роботи.

Перша мета була зосереджена на аналізі теоретичних джерел та попередніх досліджень, пов'язаних з дезінформацією та аналізом настроїв. Цей огляд заклав міцну основу для розуміння того, як будуються та поширюються стратегії дезінформації, зокрема тактика 60/40. У літературі також було підкреслено зростаюче значення емоційної маніпуляції в сучасній інформаційній війні та продемонстровано, що великі мовні моделі мають значні теоретичні та практичні переваги для інтерпретації складних текстових настроїв порівняно з традиційними підходами.

Друга мета полягала у збиранні та попередній обробці даних з Telegram-каналів, підозрюваних у проведенні дезінформаційних кампаній. Було зібрано та підготовлено спеціальний набір даних у структурованому форматі JSON, що забезпечило узгодженість форматів часових міток та цілісність метаданих. Цей набір даних дозволив проводити аналіз настроїв у часі та з різних джерел контенту, а також забезпечив надійні вхідні дані для подальшої обробки на основі LLM.

Третя мета стосувалася адаптації та конфігурації LLM для завдань аналізу настроїв. Для цього випадку використання було інтегровано та оптимізовано локальну модель у форматі GGUF, включаючи розробку ефективних структур підказок та налаштувань виведення. Порівняльна оцінка показала, що LLM перевершив класичні моделі, такі як VADER та TextBlob, особливо у виявленні нюансів емоційних виразів, таких як сарказм, іронія та заперечення.

Четверта мета була зосереджена на розробці та впровадженні прототипу системи для виявлення аномалій у траєкторіях настроїв, які можуть вказувати на скоординовані дії з дезінформації. Система була побудована для збору даних Telegram, їх обробки за допомогою локально розгорнутого LLM та виявлення

нерегулярних змін настроїв, що відхиляються від органічних моделей комунікації. Прототип підтримує настроювані порогові значення і може бути масштабований для більш широкого використання з більшими наборами даних або додатковими джерелами вхідних даних.

П'ята і остання мета полягала у візуалізації результатів аналізу та оцінці загальної ефективності системи. Для представлення змін настроїв у часі були використані графіки та лінії трендів, а ефективність LLM була виміряна за допомогою точності, відтворюваності та F1-показника. Модель досягла точності 88%, що значно перевищує класичні методи, і виявилася здатною розрізняти справжні реакції громадськості та організовані емоційні реакції.

У висновку дисертації підтверджено доцільність та ефективність використання LLM для виявлення дезінформації в каналах Telegram за допомогою аналізу настроїв. Розроблена система демонструє, що LLM можуть надавати високоточну інформацію про емоційну динаміку в цифровому контенті, що робить їх цінним інструментом для сучасних додатків інформаційної безпеки та моніторингу загроз.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Закон ЄС про цифрові послуги - Європейська комісія. URL: https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/digital-services-act_en
2. ЄС запровадить виборчі гарантії для великих технологій. URL: https://www.ft.com/content/66a6790d-3ca1-4469-b012-7d392e6cfb67
3. Закон ЄС про штучний інтелект: перший регламент щодо штучного інтелекту | Теми. URL: https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence
4. Інформаційна цілісність та протидія іноземній інформації... - EEAS URL: https://www.eeas.europa.eu/eeas/information-integrity-and-countering-foreign-information-manipulation-interference-fimi_en
5. Акт про свободу ЗМІ: новий законопроект про захист журналістів і преси ЄС URL: https://www.europarl.europa.eu/news/en/press-room/20240308IPR19014/media-freedom-act-a-new-bill-to-protect-eu-journalists-and-press-freedom
6. Світ кримінальної хроніки, ухилянства та суцільної бусифікації. Як телеги накручують негатив про ТЦК URL: https://imi.org.ua/monitorings/svit-kryminalnoyi-hroniky-uhylyanstva-ta-sutsilnoyi-busyfikatsiyi-yak-telegy-nakruchuyut-negatyv-i66087
7. Як російська пропаганда використовує метод »40 на 60« для досягнення власних цілей URL: https://disinfo.detector.media/en/post/how-russian-propaganda-uses-the-40-to-60-method-to-achieve-its-own-goals
8. Таємні проросійські Telegram-канали: Що Росія хотіла, щоб думали українці під час атаки 10 жовтня URL: https://www.ukrinform.net/rubric-society/3600256-secretly-prorussian-telegram-channels-what-russia-wanted-ukrainians-to-think-during-oct-10-attack.html

9. Одинадцять загиблих, 64 поранені в результаті російських атак - Українська держава ... - Reuters URL: https://www.reuters.com/world/europe/eleven-killed-64-hurt-russian-attacks-ukrainian-state-emergency-service-2022-10-10/
10. Виявлення емоцій: виявлення фейкових новин про COVID-19 у соціальних мережах | Гуманітарні та соціальні науки URL: https://www.nature.com/articles/s41599-024-03083-5 «Виявлення емоцій: виявлення фейкових новин про COVID-19 у соціальних мережах | Гуманітарні та соціальні науки»
11. Масштабний аналіз соціальних даних в Інтернеті щодо довгострокової динаміки настроїв та контенту онлайн (не)інформації | Запит PDF URL: https://www.researchgate.net/publication/387385204_Large-scale_analysis_of_online_social_data_on_the_long-term_sentiment_and_content_dynamics_of_online_misinformation
12. Понад 60% українців вважають проблему дезінформації важливою, але лише 40% шукають джерела - дослідження »Детектора медіа« | dev.ua URL: https://dev.ua/en/news/dezinformatsiia-mediahramotnist-shi-internet-doslidzhennia-1746687225
13. Як несправжні акаунти використовують коментарі в Telegram для поширення антиукраїнських наративів - DFRLab URL: https://dfrlab.org/2024/12/18/inauthentic-telegram-accounts-ukraine/ «»
14. Дезінформаційні атаки були спрямовані на виборців, ЗМІ та ЛГБТК+ групи, - звіт ЄС | Європейський Союз | The Guardian URL: https://www.theguardian.com/world/2024/jan/23/disinformation-attacks-targeted-voters-media-and-lgbtq-groups-eu-report-finds
15. Kaggle URL: https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset?select=train.csv
16. Telegram канал “Киевский Кейс” URL: https://t.me/kiev_korr
17. Telegram канал “Всевидящее ОКО” URL: https://t.me/oko_ua

ДОДАТОК А ПРОГРАМНИЙ КОД

anomaly_detector.py```py

```python
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from scipy import stats
from typing import Dict, Any, List, Tuple
import logging

logger = logging.getLogger('disinfo_detector.anomaly')

class AnomalyDetector:
    def __init__(self, method: str = 'zscore'):
        """
        Initialize an anomaly detector.

        Args:
            method: The anomaly detection method to use.
                Options: 'zscore', 'moving_average', 'isolation_forest'
        """
        self.method = method
        logger.info(f"Using anomaly detection method: {method}")

    def detect_anomalies(self, df: pd.DataFrame, time_window: str = None) -> pd.DataFrame:
        """
        Detect anomalies in sentiment scores.

        Args:
            df: DataFrame with 'date' and 'sentiment_score' columns
            time_window: Time window for grouping (e.g., '1H' for hourly, '1D' for daily)
                         If None, analyze individual messages

        Returns:
            DataFrame with added 'anomaly_score' column
        """

        if df.empty:
            logger.warning("Empty DataFrame passed to detect_anomalies")
            df['anomaly_score'] = 0.0
            return df
```

```
        if 'sentiment_score' not in df.columns:
            logger.error("DataFrame must contain 'sentiment_score' column")
            df['sentiment_score'] = 0.0
            df['anomaly_score'] = 0.0
            return df


        if df['sentiment_score'].nunique() <= 1:
                    logger.warning("All sentiment scores are identical, anomaly detection
impossible")
            df['anomaly_score'] = 0.0
            return df

        logger.info(f"Detecting anomalies with time window: {time_window}")


        if time_window:

            grouped = (
                df.set_index('date')
                .groupby([pd.Grouper(freq=time_window), 'feed_name'])['sentiment_score']
                .agg(['mean', 'count'])
                .reset_index()
            )
            grouped.rename(columns={'mean': 'sentiment_score'}, inplace=True)


            if self.method == 'zscore':
                grouped['anomaly_score'] = self._zscore_method(grouped['sentiment_score'])
            elif self.method == 'moving_average':
                grouped['anomaly_score'] = self._moving_average_method(
                    grouped.sort_values('date'), 'sentiment_score'
                )
            elif self.method == 'isolation_forest':
                grouped['anomaly_score'] = self._isolation_forest_method(
                    grouped[['sentiment_score', 'count']]
                )


            result_df = df.copy()


            def map_to_group_score(row):

                date_floor = pd.Timestamp(row['date']).floor(time_window)
                matching_groups = grouped[
```

```
                    (grouped['date'] == date_floor) &
                    (grouped['feed_name'] == row['feed_name'])
                ]

                if len(matching_groups) > 0:
                    return matching_groups.iloc[0]['anomaly_score']
                else:
                    return 0.0

            result_df['anomaly_score'] = result_df.apply(map_to_group_score, axis=1)
            return result_df
        else:

            result_df = df.copy()

            if self.method == 'zscore':
                                                        result_df['anomaly_score']    =
self._zscore_method(result_df['sentiment_score'])
            elif self.method == 'moving_average':

                result_df = result_df.sort_values('date')
                result_df['anomaly_score'] = self._moving_average_method(
                    result_df, 'sentiment_score'
                )
            elif self.method == 'isolation_forest':

                result_df['text_length'] = result_df['text'].apply(len)
                result_df['anomaly_score'] = self._isolation_forest_method(
                    result_df[['sentiment_score', 'text_length']]
                )
                result_df.drop('text_length', axis=1, inplace=True)

            return result_df

    def _zscore_method(self, series: pd.Series) -> pd.Series:
        """Calculate Z-score for a series."""
        if len(series) < 2:
            return pd.Series([0.0] * len(series), index=series.index)

        try:
            zscore = stats.zscore(series, nan_policy='omit')

            return pd.Series(np.abs(zscore), index=series.index).fillna(0)
        except Exception as e:
            logger.error(f"Error calculating z-score: {e}")
            return pd.Series([0.0] * len(series), index=series.index)
```

```
    def _moving_average_method(self, df: pd.DataFrame, column: str, window: int = 5) ->
pd.Series:
        """
        Calculate deviation from moving average.

        Args:
            df: DataFrame sorted by date
            column: Column name to analyze
            window: Moving average window size
        """

        if len(df) < 2:
            return pd.Series([0.0] * len(df), index=df.index)


        ma = df[column].rolling(window=window, min_periods=1).mean()


        deviation = np.abs(df[column] - ma)


        max_dev = deviation.max()
        if max_dev > 0:
            return deviation / max_dev
        else:
            return deviation

    def _isolation_forest_method(self, features: pd.DataFrame) -> pd.Series:
        """
        Use Isolation Forest algorithm for anomaly detection.

        Args:
            features: DataFrame with columns to use as features
        """

        if len(features) < 2:
            return pd.Series([0.0] * len(features), index=features.index)

        try:

            features_clean = features.fillna(features.mean())


            model = IsolationForest(
                n_estimators=100,
                contamination=0.1,
                random_state=42
```

```
            )


            scores = model.fit_predict(features_clean)



            return pd.Series(np.where(scores == -1, 1.0, 0.0), index=features.index)
        except Exception as e:
            logger.error(f"Error in isolation forest: {e}")
            return pd.Series([0.0] * len(features), index=features.index)
```;

disinfo_detector.py```py

```
# disinfo_detector.py - A tool for detecting coordinated disinformation campaigns

import argparse
import json
import os
import datetime
from typing import List, Dict, Any, Optional
import pandas as pd
import numpy as np
from pathlib import Path
import logging

from sentiment_analyzer import SentimentAnalyzer
from anomaly_detector import AnomalyDetector
from visualizer import Visualizer

def parse_arguments():
        parser = argparse.ArgumentParser(description='Detect coordinated disinformation
campaigns')
      parser.add_argument('--input', required=True, nargs='+', help='Paths to JSON feed
files')
     parser.add_argument('--max-messages', type=int, help='Maximum number of messages to
process per feed')
      parser.add_argument('--start-date', help='Filter messages by start date (ISO 8601
format)')
   parser.add_argument('--end-date', help='Filter messages by end date (ISO 8601 format)')
    parser.add_argument('--disable-emoji-analysis', action='store_true', help='Disable emoji
reaction analysis')
            parser.add_argument('--anomaly-method', choices=['zscore', 'moving_average',
'isolation_forest'],
                        default='zscore', help='Anomaly detection method')
```

```
    parser.add_argument('--time-window', default=None, help='Time window for grouping (e.g.,
"1H", "1D")')
    parser.add_argument('--model-path', default='llama-pro-8b-instruct.Q5_K_M.gguf',
                        help='Path to the LLM model file')

    return parser.parse_args()

def load_feed(file_path: str) -> Dict[str, Any]:
    """Load a feed from a JSON file."""
    try:
        with open(file_path, 'r', encoding='utf-8') as f:
            feed = json.load(f)


        if not isinstance(feed, dict):
            logging.warning(f"{file_path} does not contain a JSON object")
            return {"name": os.path.basename(file_path), "messages": []}


        if "name" not in feed:
            feed["name"] = os.path.basename(file_path).split('.')[0]


        if "messages" not in feed:
            logging.warning(f"{file_path} does not contain a 'messages' array")
            feed["messages"] = []

        return feed
    except json.JSONDecodeError as e:
        logging.error(f"Error parsing JSON from {file_path}: {e}")
        return {"name": os.path.basename(file_path), "messages": []}
    except Exception as e:
        logging.error(f"Error loading feed from {file_path}: {e}")
        return {"name": os.path.basename(file_path), "messages": []}

def extract_telegram_text(text_data: Any) -> str:
    """
    Extract plain text content from Telegram text structure.

    Args:
        text_data: Telegram text data, which can be a string, list, or dictionary

    Returns:
        Plain text content
    """
    if isinstance(text_data, str):
        return text_data
```

```
    if isinstance(text_data, list):
        result = []
        for item in text_data:
            if isinstance(item, str):
                result.append(item)
            elif isinstance(item, dict):
                if "text" in item:
                    result.append(item["text"])
        return "".join(result)

    return ""

def format_telegram_reactions(reactions: List[Dict[str, Any]]) -> List[Dict[str, Any]]:
    """
    Format Telegram reactions to the expected format.

    Args:
        reactions: List of Telegram reaction objects

    Returns:
        List of reactions in the expected format
    """
    formatted_reactions = []

    for reaction in reactions:
        if "emoji" in reaction and "count" in reaction:
            formatted_reactions.append({
                "emoji": reaction["emoji"],
                "count": reaction["count"]
            })

    return formatted_reactions

def is_telegram_format(feed: Dict[str, Any]) -> bool:
    """
    Detect if the feed is in Telegram export format.

    Args:
        feed: Feed dictionary

    Returns:
        True if the feed appears to be in Telegram format
    """
    if "messages" not in feed:
        return False
```

```
    messages = feed.get("messages", [])
    if not messages:
        return False


    sample = messages[0]
    telegram_fields = ["type", "date_unixtime", "from", "from_id", "text_entities"]


    matches = sum(1 for field in telegram_fields if field in sample)
    return matches >= 3

def preprocess_feed(feed: Dict[str, Any], max_messages: Optional[int] = None,
                        start_date: Optional[str] = None, end_date: Optional[str] = None) ->
pd.DataFrame:
    """Preprocess a feed and return a DataFrame of messages."""
    feed_name = feed.get('name', 'unknown')
    messages = feed.get('messages', [])


    is_telegram = is_telegram_format(feed)

    if is_telegram:
        logging.info(f"Detected Telegram export format for {feed_name}")

        processed_messages = []

        for msg in messages:

            if "text" in msg:
                text_content = extract_telegram_text(msg["text"])


                            if not text_content or (isinstance(text_content, str) and
text_content.strip() == ""):
                    continue


                reactions = format_telegram_reactions(msg.get("reactions", []))


                processed_message = {
                    "id": str(msg.get("id", "")),
                    "date": msg.get("date", ""),
                    "text": text_content,
                    "reactions": reactions,
```

```
                    "media_type": "text"
                }

                processed_messages.append(processed_message)


        messages = processed_messages


    df = pd.DataFrame(messages)

    if df.empty:
        logging.warning(f"No messages found in feed {feed_name}")

            return  pd.DataFrame(columns=['id',  'date',  'text',  'reactions',  'media_type',
'feed_name'])


    df['feed_name'] = feed_name


    if 'text' in df.columns:
        orig_len = len(df)
          df  =  df[df['text'].apply(lambda  x:  x  is  not  None  and  (not  isinstance(x,  str)  or
x.strip() != ""))]
        filtered_len = len(df)
        if filtered_len < orig_len:
             logging.info(f"Filtered out {orig_len - filtered_len} empty text messages from
{feed_name}")


    try:
        df['date'] = pd.to_datetime(df['date'])
    except Exception as e:
        logging.warning(f"Error converting dates: {e}. Attempting alternative formats.")

        try:
            df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%dT%H:%M:%S')
        except:
            logging.error(f"Failed to parse dates in {feed_name}")
               return pd.DataFrame(columns=['id', 'date', 'text', 'reactions', 'media_type',
'feed_name'])


    if start_date:
        try:
```

```
            start_dt = pd.to_datetime(start_date.replace('Z', '+00:00') if 'Z' in start_date
else start_date)
            df = df[df['date'] >= start_dt]
        except Exception as e:
            logging.warning(f"Error applying start date filter: {e}")

    if end_date:
        try:
            end_dt = pd.to_datetime(end_date.replace('Z', '+00:00') if 'Z' in end_date else
end_date)
            df = df[df['date'] <= end_dt]
        except Exception as e:
            logging.warning(f"Error applying end date filter: {e}")


    if max_messages and len(df) > max_messages:
        df = df.sort_values('date').iloc[-max_messages:]


    if not is_telegram and 'media_type' in df.columns:
        orig_len = len(df)
        df = df[df['media_type'] == 'text']
        filtered_len = len(df)
        if filtered_len < orig_len:
            logging.info(f"Filtered out {orig_len - filtered_len} non-text messages from
{feed_name}")


    if 'text' not in df.columns:
            logging.warning(f"No 'text' column found in {feed_name}. Creating empty text
column.")
        df['text'] = ""


    if 'reactions' not in df.columns:
        df['reactions'] = [[] for _ in range(len(df))]


    if 'media_type' not in df.columns:
        df['media_type'] = "text"

    return df

def process_feed(df: pd.DataFrame, feed_name: str, sentiment_analyzer: SentimentAnalyzer,
                 anomaly_detector: AnomalyDetector, visualizer: Visualizer,
                 use_emoji: bool = True, time_window: Optional[str] = None) -> pd.DataFrame:
    """Process a single feed and generate results."""
```

```
    try:

        df = sentiment_analyzer.analyze_dataframe(df, use_emoji=use_emoji)
    except Exception as e:
            logging.error(f"Error during sentiment analysis: {e}. Using neutral sentiment
scores.")

        df['sentiment_score'] = 0.0

    try:

        df = anomaly_detector.detect_anomalies(df, time_window=time_window)
    except Exception as e:
        logging.error(f"Error during anomaly detection: {e}. Using zero anomaly scores.")

        df['anomaly_score'] = 0.0

    try:

        visualizer.plot_sentiment_timeline(df, feed_name)
        visualizer.plot_anomaly_timeline(df, feed_name)
        visualizer.plot_sentiment_vs_anomalies(df, feed_name)
    except Exception as e:
        logging.error(f"Error generating visualizations: {e}")

    return df

def save_results(df: pd.DataFrame, output_dir: Path):
    """Save results to CSV files."""
    df.to_csv(output_dir / 'sentiment.csv', index=False)


    threshold = df['anomaly_score'].quantile(0.95)
        anomalies_df  =  df[df['anomaly_score']  >=  threshold].sort_values('anomaly_score',
ascending=False)
    anomalies_df.to_csv(output_dir / 'anomalies.csv', index=False)

def main():
    args = parse_arguments()


    logging.basicConfig(level=logging.INFO,
                        format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
    logger = logging.getLogger('disinfo_detector')


    results_dir = Path('results')
```

```
    results_dir.mkdir(exist_ok=True)
    combined_dir = results_dir / 'combined'
    combined_dir.mkdir(exist_ok=True)
    (combined_dir / 'plots').mkdir(exist_ok=True)


    try:
        logger.info(f"Loading sentiment analyzer with model: {args.model_path}")
        sentiment_analyzer = SentimentAnalyzer(model_path=args.model_path)
    except Exception as e:
        logger.error(f"Failed to initialize sentiment analyzer: {e}")
        logger.info("Using simple sentiment analysis as fallback")
        sentiment_analyzer = None

    anomaly_detector = AnomalyDetector(method=args.anomaly_method)


    all_feeds_df = []


    from tqdm import tqdm
    for input_file in tqdm(args.input, desc="Processing feeds", unit="feed"):
        logger.info(f"Processing feed from {input_file}")

        try:

            feed = load_feed(input_file)
             logger.info(f"Feed loaded from {input_file} with {len(feed.get('messages', []))}
messages")

            feed_name = feed.get('name', os.path.basename(input_file).split('.')[0])
            logger.info(f"Feed name: {feed_name}")

            feed_dir = results_dir / feed_name
            feed_dir.mkdir(exist_ok=True)
            (feed_dir / 'plots').mkdir(exist_ok=True)


            logger.info(f"Preprocessing feed from {input_file}")
            df = preprocess_feed(
                feed,
                max_messages=args.max_messages,
                start_date=args.start_date,
                end_date=args.end_date
            )

            logger.info(f"After preprocessing: {len(df)} messages")
```

```
            if not df.empty:
                logger.info(f"Sample of processed data: {df.head(1).to_dict('records')}")

            if df.empty:
                logger.warning(f"No valid messages found in {input_file}")
                continue

            logger.info(f"Analyzing {len(df)} messages from {feed_name}")


            feed_visualizer = Visualizer(feed_dir)


            if sentiment_analyzer is not None:
                processed_df = process_feed(
                    df,
                    feed_name,
                    sentiment_analyzer,
                    anomaly_detector,
                    feed_visualizer,
                    use_emoji=not args.disable_emoji_analysis,
                    time_window=args.time_window
                )
            else:

                logger.info("Using simple sentiment analysis fallback")


                def simple_sentiment(text):
                    if not isinstance(text, str):
                        return 0.0

                    text = text.lower()
                         pos_words = ['good', 'great', 'excellent', 'positive', 'promising',
'breakthrough', 'win', 'success']
                            neg_words = ['bad', 'awful', 'terrible', 'negative', 'corrupt',
'scandal', 'danger', 'fail']


                    pos_count = sum(1 for word in pos_words if word in text)
                    neg_count = sum(1 for word in neg_words if word in text)


                    total = pos_count + neg_count
                    if total == 0:
```

```
                    return 0.0

                return (pos_count - neg_count) / (pos_count + neg_count)


            from tqdm import tqdm

            sentiment_scores = []
             for _, row in tqdm(df.iterrows(), total=len(df), desc="Analyzing sentiment",
unit="message"):
                sentiment_scores.append(simple_sentiment(row['text']))

            df['sentiment_score'] = sentiment_scores


                                    processed_df   =   anomaly_detector.detect_anomalies(df,
time_window=args.time_window)


            feed_visualizer.plot_sentiment_timeline(processed_df, feed_name)
            feed_visualizer.plot_anomaly_timeline(processed_df, feed_name)
            feed_visualizer.plot_sentiment_vs_anomalies(processed_df, feed_name)


        save_results(processed_df, feed_dir)


        all_feeds_df.append(processed_df)

    except Exception as e:
        logger.error(f"Error processing {input_file}: {e}")
        import traceback
        logger.error(traceback.format_exc())


if len(all_feeds_df) > 1:
    logger.info("Performing combined cross-feed analysis")

    combined_df = pd.concat(all_feeds_df, ignore_index=True)
    combined_df = combined_df.sort_values('date')


    combined_visualizer = Visualizer(combined_dir)


    logger.info("Generating combined visualizations")
    progress_bar = tqdm(total=5, desc="Combined analysis", unit="visualization")
```

```
        combined_visualizer.plot_sentiment_timeline(combined_df)
        progress_bar.update(1)

        combined_visualizer.plot_anomaly_timeline(combined_df)
        progress_bar.update(1)

        combined_visualizer.plot_sentiment_vs_anomalies(combined_df)
        progress_bar.update(1)


        if args.time_window:
                                    combined_visualizer.plot_coordinated_activity(combined_df,
window=args.time_window)
            progress_bar.update(2)
        else:
            for window in ['1H', '1D']:
                combined_visualizer.plot_coordinated_activity(combined_df, window=window)
                progress_bar.update(1)

        progress_bar.close()


        save_results(combined_df, combined_dir)

        logger.info(f"Results saved to {results_dir}")
    elif len(all_feeds_df) == 1:
        logger.info(f"Analysis complete for single feed. Results saved to {results_dir}")
    else:
        logger.warning("No feeds were successfully processed.")

    logger.info("Analysis complete")

if __name__ == "__main__":
    main()
```;

sentiment_analyzer.py```py

import llama_cpp
from typing import List, Dict, Any
import pandas as pd
import numpy as np
import re
import logging
```
```

```
logger = logging.getLogger('disinfo_detector.sentiment')

class SentimentAnalyzer:
    def __init__(self, model_path: str = "llama-pro-8b-instruct.Q5_K_M.gguf"):
        """
        Initialize the sentiment analyzer with a small language model.

        Args:
            model_path: Path to the LLaMA model file
        """
        logger.info(f"Loading LLM from {model_path}")
        try:
            self.llm = llama_cpp.Llama(
                model_path=model_path,
                n_ctx=2048,
                n_threads=8,
                n_gpu_layers=32
            )
            logger.info("LLM loaded successfully")
        except Exception as e:
            logger.error(f"Failed to load LLM: {e}")
            raise

    def analyze_text(self, text: str) -> float:
        """
        Analyze sentiment of a text message.
        Returns a float between -1.0 (negative) and 1.0 (positive).
        """

        if not text or (isinstance(text, str) and text.strip() == ""):
            logger.debug("Skipping empty text")
            return 0.0


        if len(text) > 1000:
            text = text[:997] + "..."
```

```
        prompt = f"""
        <|system|>
         You are an expert sentiment-analysis model. Given a piece of text, return **only** its overall sentiment score on a continuous scale where -1 = extremely negative and 1 = extremely positive. Respond with a signed decimal number (e.g. -0.75 or 0.42) and no other words, symbols, or formatting.

        <|user|>
        TEXT:
        \"\"\"{text}\"\"\"

        Sentiment score:
        <|assistant|>
        """
        try:
            response = self.llm(
                prompt,
                max_tokens=10,
                temperature=0.1,
                echo=False
            )


            response_text = response['choices'][0]['text'].strip()

            matches = re.findall(r'-?\d+\.\d+|-?\d+', response_text)
            if matches:
                score = float(matches[0])

                return max(-1.0, min(1.0, score))
            else:

                if 'positive' in response_text.lower():
                    return 0.5
                elif 'negative' in response_text.lower():
                    return -0.5
                else:
                    return 0.0
        except Exception as e:
            logger.error(f"Error analyzing sentiment: {e}")
            return 0.0

    def analyze_emoji_reactions(self, reactions: List[Dict[str, Any]]) -> float:
        """
        Calculate a sentiment score based on emoji reactions.
        Returns a float between -1.0 (negative) and 1.0 (positive).
        """
```

```
        if not reactions:
            return 0.0


        emoji_sentiments = {
            "👍": 0.5,
            "❤️": 0.8,
            "😂": 0.7,
            "😢": -0.6,
            "😡": -0.8,
            "👎": -0.5,
            "🔥": 0.6,
            "👏": 0.7,
            "🤔": -0.1,
            "😮": 0.0,

        }

        total_count = sum(reaction["count"] for reaction in reactions)
        weighted_sentiment = sum(
            reaction["count"] * emoji_sentiments.get(reaction["emoji"], 0.0)
            for reaction in reactions
        )

        if total_count == 0:
            return 0.0

        return weighted_sentiment / total_count

    def analyze_message(self, message: Dict[str, Any], use_emoji: bool = True) -> float:
        """
        Analyze the sentiment of a message, combining text and emoji reactions if requested.
        Returns a float between -1.0 (negative) and 1.0 (positive).
        """

        if "text" not in message:
            return 0.0

        text = message["text"]


        if not isinstance(text, str):


            if isinstance(text, list):

                text_parts = []
```

```
                for item in text:
                    if isinstance(item, str):
                        text_parts.append(item)
                    elif isinstance(item, dict) and "text" in item:
                        text_parts.append(item["text"])
                text = "".join(text_parts)
            else:

                text = str(text)


        if not text or text.strip() == "":
            logger.debug("Skipping empty message")
            return 0.0


        text_sentiment = self.analyze_text(text)


        if use_emoji and "reactions" in message and message["reactions"]:
            try:
                emoji_sentiment = self.analyze_emoji_reactions(message["reactions"])

                return 0.7 * text_sentiment + 0.3 * emoji_sentiment
            except Exception as e:
                logger.error(f"Error analyzing emoji reactions: {e}")
                return text_sentiment
        else:
            return text_sentiment

    def analyze_dataframe(self, df: pd.DataFrame, use_emoji: bool = True) -> pd.DataFrame:
        """
        Add sentiment scores to a DataFrame of messages.
        Returns the DataFrame with a new 'sentiment_score' column.
        """
        logger.info(f"Analyzing sentiment for {len(df)} messages")


        sentiment_scores = []


        from tqdm import tqdm

             for _, row in tqdm(df.iterrows(), total=len(df), desc="Sentiment analysis",
unit="message"):
            if use_emoji:
                score = self.analyze_message({
```

```
                    "text": row["text"],
                    "reactions": row.get("reactions", [])
                }, use_emoji=True)
            else:
                score = self.analyze_text(row["text"])

            sentiment_scores.append(score)


        df['sentiment_score'] = sentiment_scores

        logger.info(f"Sentiment analysis complete")
        return df
```;

visualizer.py```py

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pathlib import Path
from typing import Dict, Any, List, Tuple
import logging

logger = logging.getLogger('disinfo_detector.visualizer')

class Visualizer:
    def __init__(self, output_dir: Path):
        """
        Initialize a visualizer.

        Args:
            output_dir: Directory to save visualizations
        """
        self.output_dir = output_dir

        (self.output_dir / 'plots').mkdir(exist_ok=True)

        plt.style.use('ggplot')

        logger.info(f"Visualizer initialized with output directory: {output_dir}")

    def plot_sentiment_timeline(self, df: pd.DataFrame, feed_name: str = None):
        """
```
```

```
        Plot sentiment scores over time.

        Args:
            df: DataFrame with 'date', 'sentiment_score', and 'feed_name' columns
            feed_name: Name of the feed (for the plot title)
        """
         logger.info(f"Creating sentiment timeline plot for {'combined feeds' if feed_name is
None else feed_name}")

        plt.figure(figsize=(12, 6))


        if feed_name is None:

            for name, group in df.groupby('feed_name'):
                plt.plot(group['date'], group['sentiment_score'], label=name, alpha=0.7)
            plt.legend(title='Feed')
            title = 'Combined Sentiment Analysis Across All Feeds'
            filename = 'combined_sentiment_timeline.png'
        else:
            plt.plot(df['date'], df['sentiment_score'], label='Sentiment Score')
            title = f'Sentiment Analysis: {feed_name}'
            filename = 'sentiment_timeline.png'

        plt.axhline(y=0, color='grey', linestyle='--', alpha=0.5)

        plt.title(title)
        plt.xlabel('Date')
        plt.ylabel('Sentiment Score (-1 to 1)')
        plt.grid(True, alpha=0.3)
        plt.tight_layout()


        save_path = self.output_dir / 'plots' / filename
        plt.savefig(save_path)
        plt.close()

        return save_path

    def plot_anomaly_timeline(self, df: pd.DataFrame, feed_name: str = None):
        """
        Plot anomaly scores over time.

        Args:
            df: DataFrame with 'date', 'anomaly_score', and 'feed_name' columns
            feed_name: Name of the feed (for the plot title)
        """
```

```
        logger.info(f"Creating anomaly timeline plot for {'combined feeds' if feed_name is
None else feed_name}")

      plt.figure(figsize=(12, 6))


      if feed_name is None:

          for name, group in df.groupby('feed_name'):
              plt.plot(group['date'], group['anomaly_score'], label=name, alpha=0.7)
          plt.legend(title='Feed')
          title = 'Anomaly Detection Across All Feeds'
          filename = 'combined_anomaly_timeline.png'
      else:
          plt.plot(df['date'], df['anomaly_score'], label='Anomaly Score')
          title = f'Anomaly Detection: {feed_name}'
          filename = 'anomaly_timeline.png'


      if not df.empty:
          threshold = df['anomaly_score'].quantile(0.95)
          anomalies = df[df['anomaly_score'] >= threshold]

          if not anomalies.empty:
              plt.scatter(
                  anomalies['date'],
                  anomalies['anomaly_score'],
                  color='red',
                  s=50,
                  label=f'Potential Anomalies (top 5%)'
              )

      plt.title(title)
      plt.xlabel('Date')
      plt.ylabel('Anomaly Score')
      plt.legend()
      plt.grid(True, alpha=0.3)
      plt.tight_layout()


      save_path = self.output_dir / 'plots' / filename
      plt.savefig(save_path)
      plt.close()

      return save_path

  def plot_sentiment_vs_anomalies(self, df: pd.DataFrame, feed_name: str = None):
```

```
        """
        Create a scatter plot of sentiment scores vs anomaly scores.

        Args:
            df: DataFrame with 'sentiment_score', 'anomaly_score', and 'feed_name' columns
            feed_name: Name of the feed (for the plot title)
        """
         logger.info(f"Creating sentiment vs anomalies scatter plot for {'combined feeds' if
feed_name is None else feed_name}")

        plt.figure(figsize=(10, 8))


        if feed_name is None:

            for name, group in df.groupby('feed_name'):
                plt.scatter(
                    group['sentiment_score'],
                    group['anomaly_score'],
                    label=name,
                    alpha=0.6
                )
            plt.legend(title='Feed')
            title = 'Sentiment vs Anomaly Scores Across All Feeds'
            filename = 'combined_sentiment_vs_anomalies.png'
        else:
            plt.scatter(df['sentiment_score'], df['anomaly_score'], alpha=0.6)
            title = f'Sentiment vs Anomaly Scores: {feed_name}'
            filename = 'sentiment_vs_anomalies.png'

        plt.title(title)
        plt.xlabel('Sentiment Score (-1 to 1)')
        plt.ylabel('Anomaly Score')
        plt.grid(True, alpha=0.3)
        plt.tight_layout()


        save_path = self.output_dir / 'plots' / filename
        plt.savefig(save_path)
        plt.close()

        return save_path

    def plot_coordinated_activity(self, df: pd.DataFrame, window: str = '1D'):
        """
        Plot potentially coordinated disinformation activity.
```

```
        Args:
              df: DataFrame with 'date', 'sentiment_score', 'anomaly_score', and 'feed_name'
columns
            window: Time window for aggregation (e.g., '1H', '1D')
        """
        logger.info(f"Creating coordinated activity analysis with window {window}")


        if df.empty:
            logger.warning("Empty DataFrame provided for coordinated activity analysis")
            return None

        required_columns = ['date', 'sentiment_score', 'feed_name']
        for col in required_columns:
            if col not in df.columns:
                  logger.warning(f"Required column '{col}' missing for coordinated activity
analysis")
                return None


        unique_feeds = df['feed_name'].unique()
        if len(unique_feeds) < 2:
             logger.warning(f"Only one feed found ({unique_feeds[0]}). Need at least 2 feeds
for coordination analysis.")
            return None

        try:

            if not pd.api.types.is_datetime64_any_dtype(df['date']):
                logger.info("Converting 'date' column to datetime")
                df['date'] = pd.to_datetime(df['date'], errors='coerce')


                if df['date'].isna().all():
                    logger.warning("No valid dates after conversion")
                    return None


                df = df.dropna(subset=['date'])


            logger.info(f"Grouping by {window} window")
            grouped = (
                df.set_index('date')
                .groupby([pd.Grouper(freq=window), 'feed_name'])
                .agg({
                    'sentiment_score': 'mean',
```

```
            'anomaly_score': 'mean' if 'anomaly_score' in df.columns else None,
            'id': 'count' if 'id' in df.columns else None,
        })
        .reset_index()
    )


    if 'id' in grouped.columns:
        grouped.rename(columns={'id': 'message_count'}, inplace=True)


    logger.info("Creating pivot table for feeds")
    pivot_sentiment = grouped.pivot(
        index='date',
        columns='feed_name',
        values='sentiment_score'
    ).dropna(how='all')


    if pivot_sentiment.empty:
        logger.warning("No data available after pivoting")
        return None

    if len(pivot_sentiment.columns) < 2:
        logger.warning("Only one feed has data after pivoting")
        return None


    logger.info("Calculating correlation between feeds")
    correlation = pivot_sentiment.corr()


    plt.figure(figsize=(10, 8))
    plt.imshow(correlation, cmap='coolwarm', vmin=-1, vmax=1)
    plt.colorbar(label='Correlation')
    plt.title(f'Sentiment Correlation Between Feeds ({window} intervals)')


    for i in range(len(correlation.columns)):
        for j in range(len(correlation.columns)):
            if pd.isna(correlation.iloc[i, j]):

                text_val = "N/A"
                text_color = "black"
            else:
                text_val = f'{correlation.iloc[i, j]:.2f}'
```

```
                            text_color = 'white' if abs(correlation.iloc[i, j]) > 0.5 else
'black'

                    plt.text(
                        j, i,
                        text_val,
                        ha='center', va='center',
                        color=text_color
                    )


            plt.xticks(range(len(correlation.columns)), correlation.columns, rotation=45)
            plt.yticks(range(len(correlation.columns)), correlation.columns)
            plt.tight_layout()


            save_path = self.output_dir / 'plots' / f'correlation_heatmap_{window}.png'
            plt.savefig(save_path)
            plt.close()


            plt.figure(figsize=(12, 6))

            for col in pivot_sentiment.columns:

                if pivot_sentiment[col].isna().all():
                    continue

                plt.plot(pivot_sentiment.index, pivot_sentiment[col], label=col)

            plt.title(f'Sentiment Shifts Across Feeds ({window} intervals)')
            plt.xlabel('Date')
            plt.ylabel('Average Sentiment Score')
            plt.legend(title='Feed')
            plt.grid(True, alpha=0.3)
            plt.tight_layout()


            save_path2 = self.output_dir / 'plots' / f'sentiment_shifts_{window}.png'
            plt.savefig(save_path2)
            plt.close()

            logger.info("Coordinated activity analysis complete")
            return [save_path, save_path2]

        except Exception as e:
            logger.error(f"Error in coordinated activity analysis: {e}")
```

```
            import traceback
            logger.error(traceback.format_exc())
            return None
```;