НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

_____ /_____Голуб Б.Л._____/
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Розробка системи виявлення фейкових новин за допомогою машинного навчання та соціальних мереж»

Спеціальність 122 - «Комп'ютерні науки»

Гарант освітньої програми
_____д.е.н., професор_____ _____ _____Руденський Р.А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
_____ _____ _Назаренко Володимир Анатолійович_
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Наумович Назарій Юрійович___
(підпис) (ПІБ студента)

КИЇВ - 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

___к.т.н., доцент_____Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"___" _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Наумович Назарій Юрійович

Спеціальність 122 - «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Розробка системи виявлення фейкових новин за допомогою машинного навчання та соціальних мереж» затверджена наказом ректора НУБіП України від 16.12.2024 № 2246с
2. Термін подання завершеної роботи на кафедру_____
(рік, місяць, число)
3. Вихідні дані до роботи: отримання відомостей про виконання призначених завдань, їх нюансів та комунікації між учасниками.
4. Перелік питань, що розглядаються:
   - Аналіз предметної області
   - Аналіз інформаційного програмного забезпечення
   - Прикладне програмне забезпечення
   - Впровадження та експлуатація системи
5. Календарний план

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	21 лютого 2025	
2	Аналіз предметної області	березень 2025	
3	Аналіз інформаційного програмного забезпечення	березень 2025	
4	Прикладне програмне забезпечення	квітень 2025	

5	Впровадження та експлуатація системи	квітень- травень 2025	
6	Оформлення записки	травень 2025	
7	Перевірка на плагіат	25 травня 2025	
8	Проходження передзахисту	2 червня 2025	
9	Захист роботи	09-12 червня 2025	

Керівник кваліфікаційної роботи

		Назаренко В.А.
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Завдання прийняв до виконання		Наумович Н.Ю..
	(підпис)	(ПІБ)

Дата отримання завдання “___”_____ 2025 р.

РЕФЕРАТ

Тема бакалаврської кваліфікаційної роботи: “Розробка системи виявлення фейкових новин за допомогою машинного навчання та соціальних мереж”.

Автор роботи: Наумович Назарій Юрійович

Керівник роботи: Назаренко Володимир Анатолійович

Пояснювальна записка: 82 с., 17 рис., 2 дод., 15 джерел.

Графічна частина: 15 презентаційних слайдів.

ФЕЙКОВІ НОВИНИ, МАШИННЕ НАВЧАННЯ, BERT, TELEGRAM-БОТ, ПЕРЕВІРКА НОВИН, ІНФОРМАЦІЙНА СИСТЕМА.

Метою цієї бакалаврської роботи є розробка автоматизованої системи для виявлення фейкових новин на основі машинного навчання, зокрема моделі BERT, а також реалізація Telegram-бота для зручної перевірки новин користувачами в реальному часі.

Розроблена система дозволяє користувачам автоматично перевіряти новини на достовірність. Модель BERT, яка є основою для класифікації новин, визначає їх як реальні або фейкові. Telegram-бот, реалізований як інтерфейс для взаємодії з користувачами, дозволяє надсилати новини та отримувати результати перевірки в режимі реального часу.

Під час реалізації системи використано:

- Python для розробки алгоритмів машинного навчання та обробки тексту.
- Модель BERT для класифікації новин як реальних або фейкових.
- python-telegram-bot для інтеграції з Telegram.
- SQLite як систему керування базами даних для збереження результатів перевірки новин.

Практичне значення роботи полягає в розробці інструменту для автоматичної перевірки новин, що дозволяє користувачам швидко отримувати

точну інформацію про їх достовірність. Це є особливо важливим в умовах, коли фейкові новини стають однією з найбільших проблем у сучасному інформаційному середовищі. Система може бути корисною для журналістів, медіаорганізацій та простих користувачів, що прагнуть уникнути поширення фейкових новин у суспільстві.

АНОТАЦІЯ

Ця бакалаврська робота присвячена розробці системи для виявлення фейкових новин за допомогою машинного навчання та Telegram-бота, яка дозволяє автоматично перевіряти новини на достовірність. Система дає змогу користувачам перевіряти новини, аналізуючи їх зміст, за допомогою моделі BERT для класифікації новин як реальних або фейкових. Окрім цього, система дозволяє перевіряти достовірність джерел новин за допомогою інтеграції з іншими сервісами. Telegram-бот надає зручний інтерфейс для взаємодії з користувачами, даючи можливість перевіряти новини в реальному часі.

Система розроблена за допомогою Python, бібліотеки transformers для роботи з моделями машинного навчання та python-telegram-bot для інтеграції з Telegram. Для зберігання даних використовується SQLite. Платформа забезпечує високу точність у класифікації новин та ефективне використання ресурсів.

Результати цього дослідження сприяють розвитку систем виявлення фейкових новин, пропонуючи практичне, адаптивне і економічно ефективне рішення для користувачів, які хочуть перевіряти достовірність новин.

Ключові слова: виявлення фейкових новин, машинне навчання, BERT, Telegram-бот, перевірка новин, інформаційна система.

ABSTRACT

This Bachelor's thesis is devoted to the development of a system for detecting fake news using machine learning and a Telegram bot that allows you to automatically check news for accuracy. The system allows users to check news by analyzing its content, using the BERT model to classify news as real or fake. In addition, the system allows you to check the accuracy of news sources by integrating with other services. The Telegram bot provides a user-friendly interface for interacting with users, allowing you to check news in real time.

The system is developed using Python, The transformers library for working with machine learning models, and python-telegram-bot for integration with Telegram. SQLite is used for data storage. The platform provides high accuracy in classifying news and efficient use of resources.

The results of this study contribute to the development of fake news detection systems, offering a practical, adaptive and cost-effective solution for users who want to verify the authenticity of news.

Keywords: fake news detection, machine learning, BERT, Telegram bot, news verification, information system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface (інтерфейс програмування додатків).

BERT - Bidirectional Encoder Representations from Transformers (двонаправлені уявлення кодувальника на основі трансформера).

ERwin - Data Modeler (інструмент для створення логічних моделей бази даних та ER-діаграм).

ML - Machine Learning (машинне навчання).

NLP - Natural Language Processing (обробка природної мови).

SQL - Structured Query Language (мова структурованих запитів для роботи з реляційними базами даних).

SQLite - Легка вбудована реляційна система керування базами даних.

UML - Unified Modeling Language (уніфікована мова моделювання).

CSV - Comma-Separated Values (формат збереження табличних даних, розділених комами).

AI - Artificial Intelligence (штучний інтелект).

Tokenizer - Компонент обробки тексту, що розбиває його на токени (частини слів або слова).

ВСТУП

У сучасному цифровому середовищі велика увага приділяється забезпеченню надійності та достовірності інформації, що циркулює в Інтернеті. З розвитком технологій та змінами у способах отримання інформації, особливо через соціальні мережі, блоги та месенджери, проблема фейкових новин стає все більш актуальною. Фейкові новини можуть мати серйозні наслідки для суспільства, оскільки вони здатні впливати на громадську думку, провокувати паніку, сприяти маніпуляціям у політиці та навіть ставати причиною соціальних конфліктів. У зв'язку з цим зростає потреба в інструментах для автоматичного виявлення фейкових новин, які можуть допомогти користувачам швидко оцінювати достовірність інформації.

Одним з найефективніших підходів для вирішення цієї проблеми є використання методів машинного навчання для класифікації новин. Системи на основі машинного навчання можуть автоматично обробляти великі обсяги текстової інформації та визначати, чи є новина достовірною чи фейковою, аналізуючи контекст, стиль написання та інші параметри тексту. Однією з найбільш перспективних моделей для цього є BERT, яка показала високу ефективність у класифікації тексту в різних сферах.

Оскільки багато користувачів отримують новини через мобільні месенджери, зокрема через Telegram, виникає необхідність інтегрувати такі системи в популярні платформи для забезпечення зручного доступу до інструментів перевірки новин. Telegram, як один з найпопулярніших месенджерів, забезпечує високу популярність і доступність, що робить його ідеальним середовищем для реалізації системи автоматичної перевірки новин.

Метою цієї роботи є розробка системи для виявлення фейкових новин за допомогою машинного навчання, зокрема моделі BERT, а також реалізація Telegram-бота для перевірки новин у реальному часі.

Основним завданням є створення інструменту, який дозволить користувачам зручно взаємодіяти з системою через Telegram для автоматичної перевірки новин.

Актуальність дослідження: з поширенням фейкових новин проблема їх виявлення та перевірки стає все важливішою. Для боротьби з цією проблемою необхідні системи, здатні обробляти велику кількість новин, перевіряти їх достовірність і надавати точні результати. Оскільки поширення інформації через месенджери, зокрема Telegram, є основним каналом для багатьох користувачів, інтеграція такого інструменту саме в цю платформу стає дуже актуальною.

Метою дослідження є розробка інтерактивної та доступної системи для перевірки новин на достовірність за допомогою машинного навчання, зокрема моделі BERT. Система повинна забезпечити точність перевірки новин і легкість користування через Telegram-бота, що дозволить кожному користувачеві безперешкодно перевіряти новини.

Завдання роботи:

1. Оцінити існуючі підходи та методи виявлення фейкових новин на основі машинного навчання та обробки природної мови.
2. Розробити логічну модель даних для зберігання новин, результатів їх перевірки та даних користувачів.
3. Використати модель BERT для класифікації новин і перевірки їх достовірності.
4. Створити Telegram-бота для зручної взаємодії з користувачами і перевірки новин.
5. Провести тестування системи на реальних даних та оцінити її ефективність.

Важливість дослідження полягає в тому, що запропонована система дозволяє значно знизити ризики поширення фейкових новин. Оскільки багато користувачів не мають можливості перевіряти джерела новин самостійно, автоматизована система перевірки через Telegram-бота дасть змогу кожному користувачеві за лічені секунди отримати точну інформацію про достовірність новини.

Методи дослідження включають:

- використання методів машинного навчання для створення моделі класифікації новин на реальні та фейкові;
- аналіз та інтеграція з Telegram API для створення зручного інтерфейсу взаємодії з користувачем;
- тестування моделі на реальних датасетах для оцінки її ефективності.

Практичне значення роботи: створена система може бути використана не тільки для перевірки новин у реальному часі, але й як інструмент для журналістів, медіаорганізацій та всіх, хто прагне отримувати достовірну інформацію.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

У сучасному інформаційному середовищі фейкові новини стали серйозною загрозою. З розвитком соціальних мереж, онлайн-платформ і месенджерів новини можуть швидко охоплювати великі аудиторії, не проходячи перевірку на достовірність. Фейкові новини можуть мати серйозні соціальні, політичні та економічні наслідки. Вони можуть впливати на громадську думку, маніпулювати політичними процесами та навіть спричиняти кризові ситуації через неправдиву або неповну інформацію. Тому дуже важливо мати ефективні інструменти для автоматизованого виявлення фейкових новин, які здатні працювати в реальному часі.

Актуальність дослідження полягає в тому, що з розвитком технологій і зростанням популярності месенджерів, таких як Telegram, проблема фейкових новин набуває все більшого значення. Telegram є однією з найбільш популярних платформ для обміну інформацією, що дозволяє швидко отримувати новини, обговорювати їх у чатах і групах. Однак це також дає можливість швидко поширювати неправдиву інформацію. Тому інтеграція системи перевірки новин прямо в Telegram є логічним і зручним рішенням.

Для автоматизованої перевірки новин можна застосовувати методи машинного навчання. Однією з найбільш ефективних моделей для цієї задачі є BERT. Модель BERT здатна враховувати контекст слів у двох напрямках (зліва направо та справа наліво), що робить її дуже потужною для обробки природної мови та класифікації текстів. Вона може ефективно визначити, чи є новина достовірною чи фейковою, аналізуючи контекст, а не тільки ключові слова, як це часто роблять класичні алгоритми.

Завдання дослідження - створити систему для виявлення фейкових новин за допомогою машинного навчання, зокрема моделі BERT, і реалізувати інтерактивний інтерфейс через Telegram-бота для перевірки новин у реальному часі.

Щоб забезпечити високий рівень точності та швидкості, система повинна:

- Збирати новини для перевірки.

- Обробляти їх через модель машинного навчання для класифікації.
- Повернути результат користувачу через зручний інтерфейс.

Це завдання є надзвичайно актуальним, особливо в умовах сучасного інформаційного простору, де швидкість і точність у наданні перевіреної інформації є критично важливими.

1.1.1 Функціональні та нефункціональні вимоги

Система виявлення фейкових новин повинна відповідати ряду вимог, які можна поділити на функціональні та нефункціональні.

Таблиця 1.1. Функціональні вимоги до системи

№	Вимога	Опис
1	Реєстрація користувача	Система повинна зберігати інформацію про користувачів Telegram-бота при першій взаємодії
2	Перевірка новин	Користувач надсилає текст новини, і система повертає результат перевірки
3	Відображення результату	Результат перевірки відображається у вигляді тексту (реальна/фейкова)
4	Можливість залишити відгук	Користувач може оцінити точність перевірки, залишивши фідбек
5	Перегляд статистики	Користувач може отримати статистику власних перевірок новин
6	Адміністративна панель (опціонально)	Адміністратор може переглядати загальну статистику та фідбеки користувачів

Таблиця 1.2. Нефункціональні вимоги до системи

№	Вимога	Опис
1	Зручність інтерфейсу	Інтерфейс Telegram-бота має бути інтуїтивно зрозумілим
2	Час відповіді	Система повинна повертати результат перевірки не пізніше ніж за 5 секунд
3	Надійність	Бот має стабільно працювати без збоїв при великій кількості запитів
4	Масштабованість	Система має легко адаптуватись до зростання кількості користувачів
5	Безпека	Особисті дані користувачів Telegram не повинні розголошуватись чи зберігатись зайве
6	Локалізація	Система повинна підтримувати українську мову

1.1.2 Вимоги до інтерфейсу користувача

Інтерфейс користувача є важливою складовою частиною системи виявлення фейкових новин. Оскільки система реалізована через Telegram-бота, то інтерфейс має бути максимально простим і зручним для користувачів різного рівня досвіду. Прості і зрозумілі елементи взаємодії дозволяють зробити користування ботом інтуїтивно зрозумілим навіть для тих, хто не має технічних знань.

Моєю метою є створення такого інтерфейсу, який би забезпечував комфортну взаємодію та швидке отримання результатів. Бот має надавати користувачеві всі необхідні інструменти для перевірки новин, від початкового знайомства з його можливостями до відображення результатів перевірки. Нижче наведено основні вимоги до інтерфейсу користувача, що забезпечують ефективність і зручність роботи з системою.

Таблиця 1.3 Вимоги до інтерфейсу користувача

№	Вимога	Опис
1	Простота навігації	Інтерфейс повинен бути спрощеним і зрозумілим для всіх користувачів. Користувачі не повинні витрачати багато часу на пошук потрібних функцій, все повинно бути на виду. Важливо, щоб кожна кнопка або команда мала чітке позначення і користувач одразу розумів її призначення.
2	Початкове повідомлення	Після запуску бота, користувач отримує привітальне повідомлення, в якому є коротка інструкція щодо того, як використовувати бота. Цей крок дуже важливий для забезпечення комфортної взаємодії, оскільки він допомагає користувачам одразу зрозуміти, що їм потрібно робити далі. Наприклад, текст повідомлення може бути таким: "Привіт! Я

		допоможу перевірити новини на достовірність. Просто надішли текст новини, і я скажу, чи вона реальна чи фейкова!"
3	Надсилання новини	Користувач повинен мати можливість надіслати текст новини для перевірки. Текст не має бути обмежений в кількості символів, але слід уникати надто великих повідомлень, щоб не впливати на швидкість обробки. Бот повинен автоматично обробляти будь-який текст і не створювати складнощів у взаємодії з користувачем. Це допоможе зберегти простоту у використанні.
4	Відображення результату перевірки	Після того, як новина пройде перевірку, бот повинен надати користувачеві чіткий результат у вигляді тексту. Наприклад, "Реальна новина" або "Фейкова новина". Якщо результат не однозначний, система повинна повідомити, що новина потребує додаткової перевірки. Це дозволяє користувачеві швидко отримати важливу інформацію без зайвих зусиль і витраченого

		часу.
5	Відгуки користувача	Після перевірки новини користувач може оцінити точність результату. Важливо надати можливість залишити відгук для покращення точності системи та отримання зворотного зв'язку. Це також допомагає підтримувати залучення користувачів, адже вони можуть відчувати свою участь у вдосконаленні системи. Наприклад, користувач може залишити короткий відгук типу: "Результат був точним, дякую!" або "Хотілося б більше пояснень".
6	Перегляд статистики	Для зручності користувачів важливо, щоб вони мали змогу переглядати статистику своїх перевірок новин. Це може бути корисно для тих, хто хоче побачити свою активність: скільки новин вони перевірили, скільки з них були фейковими та скільки - реальними. Наприклад, бот може вивести таку інформацію: "Ви перевірили 35 новин. 20 з них - реальні, 15 - фейкові". Це дозволить користувачам відслідковувати свою

		активність та зручніше працювати з системою.
7	Допомога та підтримка	Команда /help повинна надавати короткі та зрозумілі пояснення про те, як користуватися ботом. Це допоможе користувачам, які можуть забути команду або не знати, що робити далі. Якщо користувач не впевнений у наступному кроці, він може просто запитати, і бот надасть відповідні інструкції для продовження роботи. Наприклад, команда може відповісти: "Надішліть текст новини, і я перевірю її достовірність".

1.2 Огляд наявних рішень

У сучасному світі, де інформація поширюється зі швидкістю світла, проблема фейкових новин стала надзвичайно актуальною. З розвитком соціальних мереж та цифрових технологій, фейкові новини можуть миттєво охоплювати величезну аудиторію, впливаючи на громадську думку, політичні процеси та соціальну стабільність. Тому пошук ефективних методів для виявлення та протидії фейковим новинам є пріоритетним напрямом досліджень у галузі обробки природної мови (NLP) та штучного інтелекту.

Методи на основі правил

Перші спроби автоматичного виявлення фейкових новин базувалися на методах, що використовують чітко визначені правила та шаблони. Такі

системи аналізують текст на наявність певних ключових слів або фраз, які часто зустрічаються у фейкових повідомленнях. Наприклад, слова на кшталт "сенсація", "шок", "невідомі факти" можуть свідчити про недостовірність інформації.

Однак, ці методи мають суттєві обмеження. Вони не враховують контекст, інтонацію та стилістичні особливості тексту. Наприклад, слово "сенсація" може використовуватися як у фейкових, так і в достовірних новинах. Тому такі системи часто дають хибнопозитивні або хибнонегативні результати, що знижує їх ефективність у реальних умовах.

Алгоритми машинного навчання

З розвитком технологій з'явилися більш просунуті методи, засновані на алгоритмах машинного навчання. До них відносяться моделі, які використовують статистичні характеристики тексту, такі як частота слів, біграми, тріграми тощо.

Найпоширенішими алгоритмами є:

- Машини опорних векторів (SVM): ефективні для задач класифікації, але потребують ретельного підбору ознак.
- Дерева рішень: прості у реалізації, але схильні до переобучення.
- Наївний байєсівський класифікатор: швидкий, але може бути неточним при наявності залежностей між ознаками.

Ці методи дозволяють враховувати більше факторів, ніж прості правила, але все ще мають обмеження у розумінні глибокого контексту та семантики тексту.

Моделі глибокого навчання

Справжній прорив у виявленні фейкових новин стався з появою моделей глибокого навчання, зокрема трансформерних архітектур, таких як BERT. Ці моделі здатні аналізувати текст у двох напрямках, враховуючи контекст кожного слова в реченні, що значно підвищує точність класифікації.

Переваги моделей глибокого навчання:

- Контекстуальне розуміння: здатність враховувати значення слова залежно від його оточення.

- Гнучкість: можливість адаптації до різних мов та стилів тексту.
- Висока точність: демонструють кращі результати порівняно з традиційними методами.

Однак, ці моделі вимагають значних обчислювальних ресурсів та великих обсягів навчальних даних.

Існуючі рішення:

У світі існує кілька платформ, які успішно застосовують вищезазначені методи для боротьби з фейковими новинами. Розглянемо деякі з них:

StopFake - це український фактчекінговий проєкт, заснований у 2014 році викладачами та студентами Могилянської школи журналістики. Метою проєкту є виявлення та спростування фейкових новин, особливо тих, що стосуються подій в Україні. StopFake поєднує ручну перевірку фактів з використанням сучасних технологій, включаючи NLP та машинне навчання.

Платформа активно співпрацює з міжнародними організаціями та медіа, має власні телепередачі, подкасти та публікації на кількох мовах. StopFake також є партнером Facebook у програмі боротьби з дезінформацією.

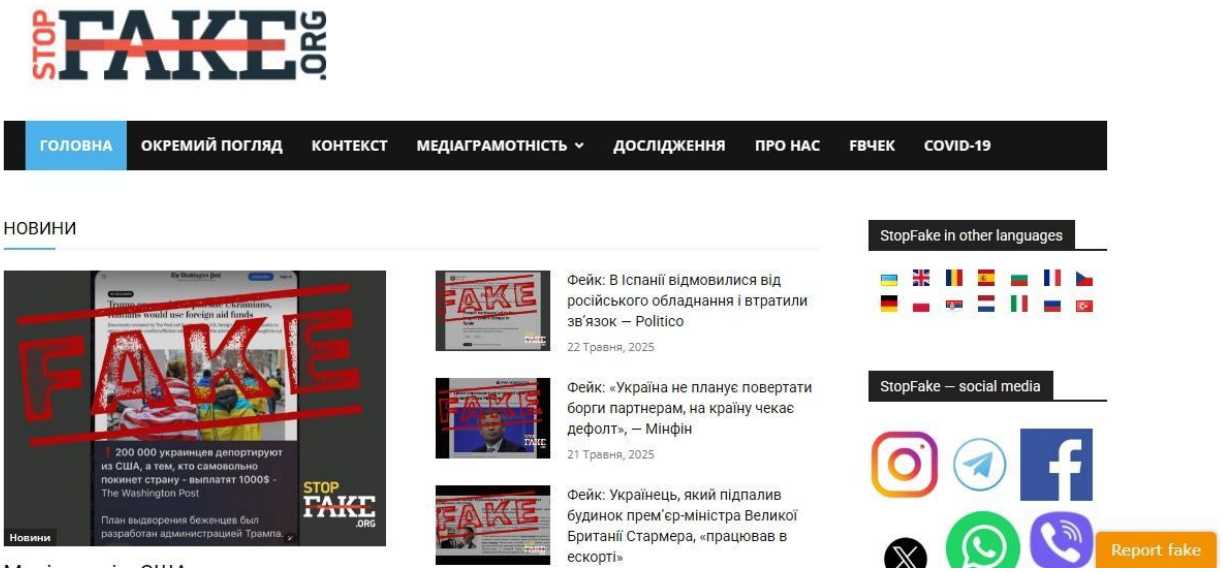

Рис. 1 StopFake

NewsGuard

NewsGuard - це американська компанія, яка оцінює надійність новинних сайтів та надає користувачам інформацію про їхню достовірність. Вони

використовують команду журналістів для аналізу та оцінки сайтів за різними критеріями, такими як прозорість, відповідальність та точність.

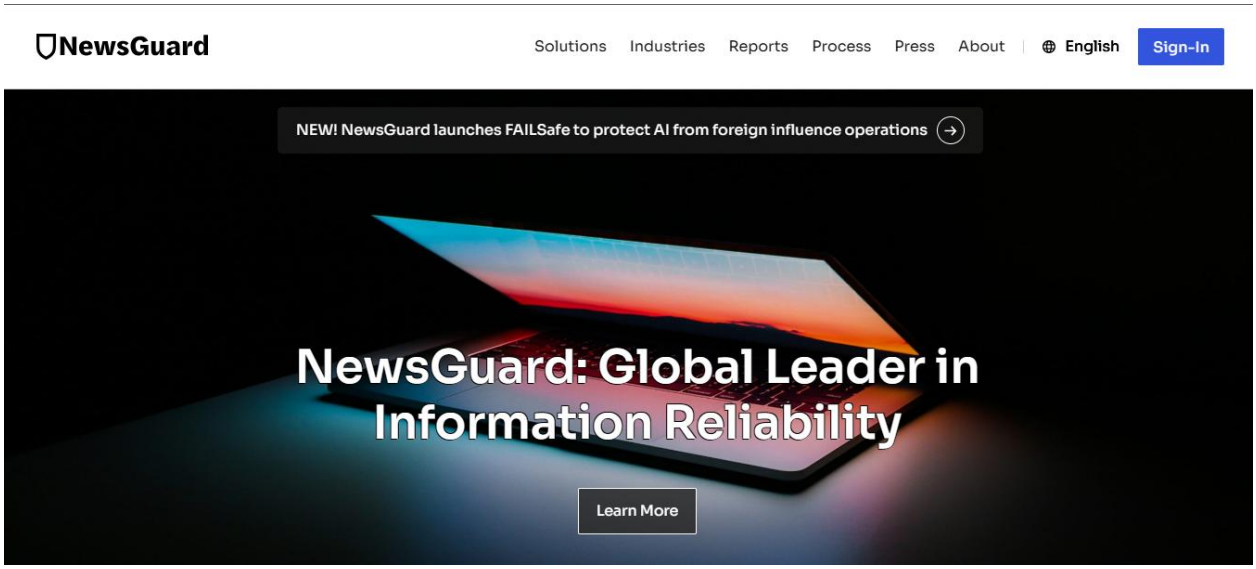

Рис. 2 NewsGuard

Bot Sentinel

Bot Sentinel - це платформа, яка аналізує акаунти в Twitter на предмет поведінки, що свідчить про поширення дезінформації або автоматизовану активність. Вона використовує машинне навчання для класифікації акаунтів та надає користувачам інструменти для виявлення потенційно шкідливих акаунтів.

Hoaxy

Hoaxy - це інструмент, розроблений Університетом Індіани, який візуалізує поширення фейкових новин у Twitter. Користувачі можуть бачити, як певна інформація поширюється мережею, та ідентифікувати ключових розповсюджувачів.

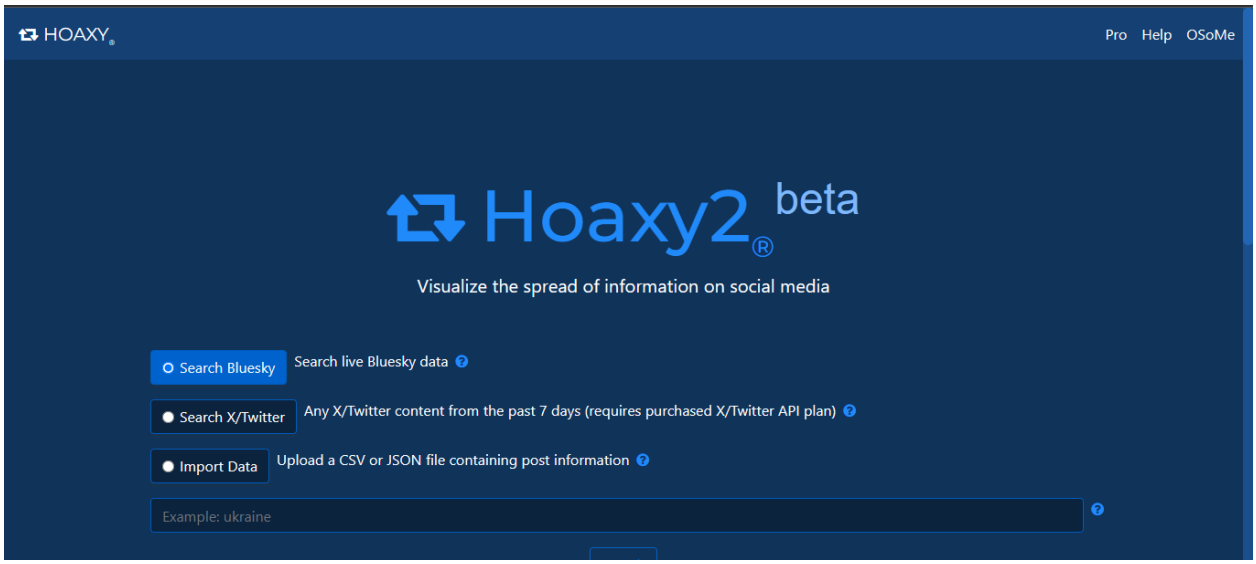

Рис. 3 Hoaxy

Таблиця 1.4 Порівняльний аналіз

Платформа	Методологія	Особливості
StopFake	Ручна перевірка + NLP	Фокус на українських новинах, партнерство з Facebook
NewsGuard	Журналістський аналіз	Оцінка надійності сайтів, розширення для браузерів
Bot Sentinel	Машинне навчання	Аналіз поведінки акаунтів у Twitter
Hoaxy	Візуалізація поширення	Відстеження розповсюдження фейків у реальному часі

Аналіз існуючих рішень показує, що ефективна боротьба з фейковими новинами потребує комплексного підходу, який поєднує ручну перевірку фактів з використанням сучасних технологій, таких як машинне навчання та глибоке навчання. Кожна з розглянутих платформ має свої сильні сторони та обмеження, але разом вони демонструють потенціал для створення більш надійних та ефективних систем виявлення фейкових новин.

1.3 Моделювання предметної області

Моделювання предметної області є важливим етапом для нашої системи, оскільки правильне зберігання та організація даних дозволяють ефективно працювати з великими обсягами інформації. У нашій системі повинна здійснюватися обробка новин, перевірка їх достовірності та надання результату користувачу. Для цього необхідно чітко визначити основні сутності, з якими буде працювати система: Користувач, Перевірка новини та Відгук.

Для збереження результатів перевірки новин у нашій системі використовуватиметься SQLite, що є оптимальним варіантом для середніх за розміром проєктів, оскільки SQLite є легким, швидким і не вимагає складної настройки.

Користувач (User)

Кожен користувач, що взаємодіє з системою через Telegram-бота, має унікальний ідентифікатор user_id. Це дозволяє зберігати інформацію про користувачів, таку як telegram_id, username та дату реєстрації. Ці дані зберігаються в базі даних для відстеження активності користувачів і персоналізації досвіду.

Перевірка новини (NewsCheck)

Коли користувач надсилає новину для перевірки, вона зберігається як сутність NewsCheck. Ця сутність містить:

- check_id - унікальний ідентифікатор перевірки.
- user_id - ідентифікатор користувача, що надіслав новину на перевірку.
- text - текст новини.
- prediction - результат перевірки, який може бути "реальна", "фейкова" або "потребує додаткової перевірки".
- check_date - дата перевірки новини.

Відгук (Feedback)

Після того, як новина була перевірена, користувач може залишити відгук щодо точності перевірки. Відгук зберігається в сутності Feedback, яка містить:

- feedback_id - унікальний ідентифікатор відгуку.
- check_id - ідентифікатор перевірки новини, з якою пов'язаний цей відгук.
- user_feedback - текстовий відгук користувача.
- date - дата надання відгуку.

Ці три сутності взаємодіють між собою, створюючи логічну структуру для роботи системи. Наприклад, кожен користувач може мати кілька перевірок новин, а кожна перевірка може мати один або жоден відгук.

1.3.1 Діаграма класів

Діаграма класів є важливим засобом для моделювання предметної області, оскільки вона дозволяє наочно показати структуру системи та взаємозв’язки між її основними сутностями. Це один із ключових інструментів в об'єктно-орієнтованій розробці програмного забезпечення, де кожен клас виступає як окрема сутність із власними атрибутами й методами для взаємодії з іншими об'єктами.

Універсальна мова моделювання (UML) є стандартом для створення таких діаграм, зокрема й діаграм класів. Вона дозволяє відображати як структуру, так і поведінку програмних систем. UML-діаграма класів описує об’єкти (класи), їхні атрибути та методи, а також типи зв’язків між ними - такі як асоціації, наслідування та композиція.

Основні елементи UML діаграми класів:

1. Клас (Class) - представляє сутність у системі, що має певні атрибути та методи. Клас визначається прямокутником, поділеним на три частини: ім'я класу, атрибути та методи.
2. Асоціація (Association) - зв'язок між класами, який показує, як об'єкти одного класу можуть взаємодіяти з об'єктами іншого класу.
3. Наслідування (Inheritance) - показує, як один клас може наслідувати атрибути та методи іншого класу, що дозволяє уникнути дублювання коду.
4. Композиція (Composition) - особливий вид асоціації, який вказує на жорстке зв'язування між класами, де об'єкт одного класу не може існувати без іншого.

Роль діаграми класів у розробці системи:

Діаграма класів дає можливість чітко визначити основні сутності системи, їх атрибути та методи, а також правила взаємодії між ними. Це дозволяє спрощувати подальшу розробку, тестування та підтримку програмного забезпечення. У нашому випадку діаграма класів допомагає відобразити, як саме обробляються новини в системі перевірки фейкових новин, хто взаємодіє з системою і як зберігаються результати перевірок.

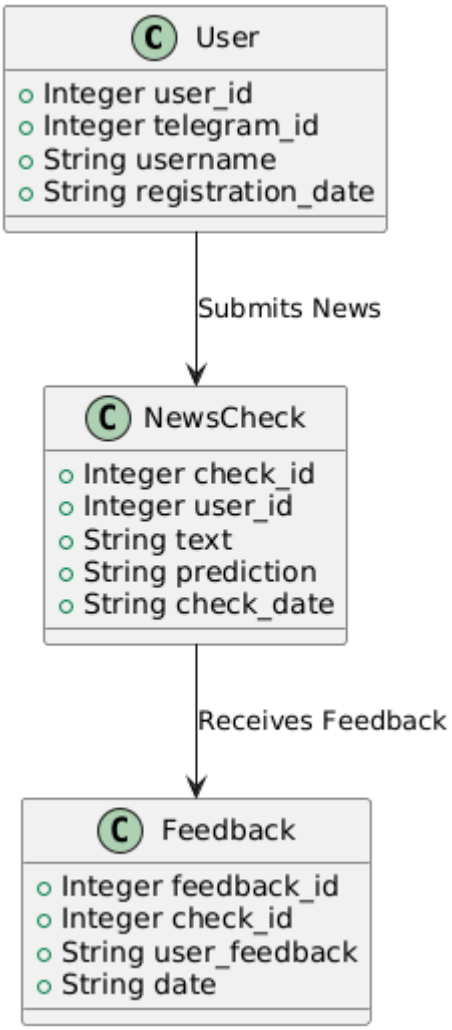

Рис. 4 Діаграма класів

Опис діаграми:

Представлена діаграма демонструє взаємозв’язки між трьома головними класами:

- **User (Користувач):** кожен користувач має свій унікальний ідентифікатор user_id, який використовується для ідентифікації в системі. Користувач може мати кілька перевірок новин, які фіксуються

у таблиці NewsCheck. Клас User зберігає також додаткову інформацію про користувача, наприклад, його Telegram ID або username.

- **NewsCheck (Перевірка новини):** кожна перевірка новини має унікальний ідентифікатор check_id, текст новини та результат її перевірки. Результати зберігаються в базі даних, щоб користувач міг ознайомитися з результатами перевірки. Клас NewsCheck також зберігає дату перевірки та інформацію про користувача, який надіслав новину для перевірки.
- **Feedback (Відгук):** кожен відгук прив'язаний до конкретної перевірки новини через check_id. Користувач може залишати відгуки для оцінки точності результату перевірки новини. Відгуки зберігаються для подальшого аналізу та вдосконалення моделі класифікації.

Взаємодія між сутностями:

1. Користувач взаємодіє з Telegram-ботом, надсилаючи новини для перевірки.
2. Бот передає новину на обробку в модель для класифікації.
3. Результат перевірки зберігається в таблиці NewsCheck.
4. Якщо користувач бажає, він може залишити відгук про точність результату перевірки, що буде збережено в таблиці Feedback.

Діаграма класів допомагає зрозуміти структуру даних та процеси взаємодії між компонентами системи, що дуже важливо при проектуванні й розробці програмного забезпечення.

1.3.2 Діаграма прецедентів

Діаграма прецедентів (Use Case Diagram) - це один із типів UML-діаграм, який допомагає візуально описати функціональні можливості системи. Вона демонструє, які саме дії може виконувати користувач, і як ці дії пов’язані з функціональністю всередині системи. Іншими словами, вона показує що робить користувач, але не пояснює як це реалізовано технічно.

UML - це уніфікована мова моделювання, яка широко застосовується в процесі розробки програмного забезпечення для опису структури, поведінки та взаємозв’язків між різними частинами системи. Діаграма прецедентів, як правило, використовується на початкових етапах - ще до того, як мова заходить про код. Вона допомагає «намалювати» загальну картину того, хто і що робить у системі.

Ключові елементи діаграми прецедентів:

- Актори - це користувачі або інші системи, які взаємодіють із вашою системою.
- Прецеденти - це конкретні дії або функції, які може виконати актор.
- Один актор може бути пов'язаний із кількома прецедентами, і навпаки - одна дія може бути доступна кільком користувачам.
- Зв’язки між ними позначаються стрілками чи лініями, що дозволяє легко зрозуміти загальну логіку взаємодії.

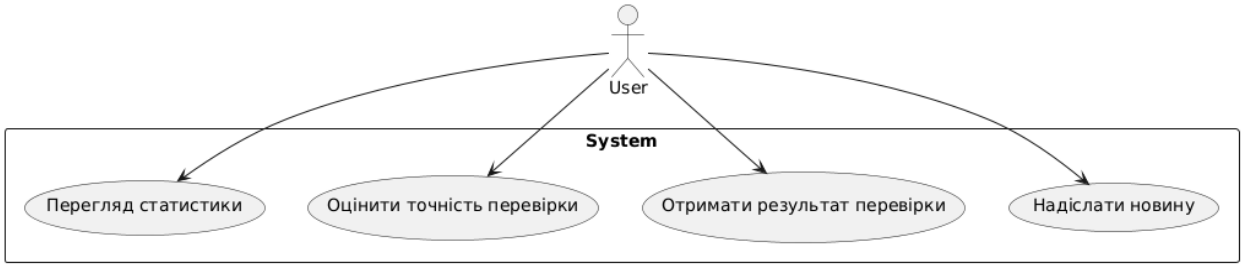

Рис. 5 Діаграма прецедентів

Пояснення діаграми:

- **User (Користувач)** - це актор, тобто та особа (в нашому випадку, будь-який користувач Telegram), яка взаємодіє з системою через чат-бота.
- **Надіслати новину** - перша базова функція: користувач відправляє текст новини в бот.
- **Отримати результат перевірки** - бот обробляє новину і повертає результат: реальна вона чи фейкова.
- **Оцінити точність перевірки** - після отримання відповіді користувач може залишити свій фідбек: згоден чи не згоден із результатом.
- **Перегляд статистики** - ще одна функція, яка дозволяє користувачеві подивитися власну загальну статистику.

Ця діаграма дозволяє наочно показати, що саме може робити користувач у системі, і які функції повинні бути реалізовані в інтерфейсі Telegram-бота. Такий підхід дає змогу з самого початку правильно спланувати логіку роботи бота та зрозуміти, які частини системи треба реалізовувати в першу чергу.

1.3.3 Діаграма діяльності

Діаграма діяльності (Activity Diagram) - це ще один тип UML-діаграм, який дозволяє відобразити послідовність дій або логіку виконання процесів у системі. На відміну від діаграми прецедентів, що показує "що" відбувається, діаграма діяльності показує "як саме" це реалізовано - тобто схожа на алгоритм або блок-схему.

UML використовується в розробці для того, щоб наочно представити процеси в системі, ще до того як сідати за код. Це зручно, бо дозволяє не заплутатись у логіці та краще зрозуміти, які етапи треба реалізувати.

Діаграма діяльності схожа на блок-схему, в якій кожен етап - це якась дія або рішення. Її можна уявити як покроковий сценарій: що відбувається від моменту, коли користувач щось запускає, і до моменту, коли система виконує всі операції. Вона добре підходить для опису логіки роботи систем, де важлива чітка послідовність.

У нашому випадку така діаграма ілюструє повний цикл перевірки новини в Telegram-боті - від того моменту, як користувач надсилає текст, і до моменту, коли результат обробки зберігається у базі даних. Це дозволяє глибше зрозуміти, як саме працює система всередині, і що відбувається на кожному етапі перевірки.

Рис. 6 Діаграма діяльності

Опис діаграми діяльності

1. **Користувач надсилає новину**

- На початку процесу користувач взаємодіє з Telegram-ботом і надсилає текст новини для перевірки на достовірність. Це ініціює весь процес перевірки. Система отримує текст новини через Telegram API, яке є основним каналом взаємодії між користувачем і програмою.
- **Задача**: Отримати текст новини, що потребує перевірки.
- **Роль користувача**: Введення новини для перевірки.

2. **Бот обробляє та класифікує новину**

- Після того, як новина була отримана ботом, система виконує кілька етапів обробки. Спочатку новина проходить через процес токенізації, де її текст розбивається на дрібніші одиниці, такі як слова або фрази. Потім текст передається в модель BERT, яка класифікує новину як "реальну"

або "фейкову". Модель аналізує контекст новини, враховуючи як попередні, так і наступні слова, що дозволяє більш точно класифікувати новину.

- **Задача**: Провести класифікацію новини за допомогою моделі BERT.
- **Процес: `Токенізація тексту` → `Аналіз контексту` → `Класифікація результату.`**
- **Роль системи**: Використання машинного навчання для визначення істинності новини.

3. **Отримання результату**

- Після класифікації новини система повертає результат користувачеві. Це може бути один з трьох можливих результатів:
- "Реальна новина" - якщо модель визначила новину як правдиву.
- "Фейкова новина" - якщо модель класифікувала новину як неправдиву.
- "Потребує додаткової перевірки" - якщо модель не може однозначно визначити істинність новини.
- **Задача**: Повернути результат перевірки новини.
- **Роль системи**: Повернення результату перевірки через Telegram.

4. **Збереження у БД**

- Після того, як результат перевірки надано користувачу, система зберігає інформацію про новину та результат у базі даних. Це включає збереження тексту новини, її результату, дати перевірки та ідентифікатора користувача. Це дає змогу відстежувати статистику перевірок для кожного користувача та покращувати майбутні прогнози моделі на основі зібраних даних.
- **Задача**: Зберегти результат перевірки та текст новини в базі даних для подальшого аналізу.
- **Роль системи**: Виконання запису результатів у базу даних (SQLite).

5. **Завершення процесу**

- Після того, як новина була оброблена, результат перевірки повернутий користувачу та збережений у базі даних, процес перевірки новини завершено. Система готова прийняти нову новину для перевірки, і цей цикл може повторюватися для наступних новин.
- **Задача**: Завершити перевірку новини.
- **Роль користувача**: Отримати результат перевірки та можливість подальшого використання бота.

Загальний процес перевірки новини

- Початок: Користувач ініціює процес, надсилаючи новину.
- Обробка: Бот отримує новину, класифікує її через модель BERT.
- Результат: Система повертає результат користувачеві.
- Збереження: Результат перевірки зберігається у базі даних для подальшого аналізу.
- Завершення: Процес завершено, і система готова до нової перевірки.

1.4 Висновки до 1 розділу

Проведений аналіз предметної області дозволяє зробити висновок: проблема фейкових новин залишається надзвичайно актуальною в сучасному інформаційному середовищі. В умовах, коли новини розповсюджуються миттєво через соціальні мережі, месенджери та форуми, виникає потреба у швидких і надійних інструментах для перевірки їхньої достовірності. Оскільки більшість людей не мають часу або бажання самостійно перевіряти кожен факт, технології відіграють тут ключову роль.

Аналіз наявних підходів показав, що традиційні методи, які базуються лише на ключових словах або простих статистичних правилах, уже не забезпечують достатньої точності. Сучасні фейкові новини можуть виглядати цілком правдоподібно, бути емоційно зарядженими та грамотно сформульованими. У таких випадках особливого значення набувають контекстно-орієнтовані моделі, такі як BERT. Завдяки здатності розуміти слова в контексті, ця модель дозволяє значно глибше аналізувати зміст повідомлень, а не покладатися лише на шаблони або частотність.

Ідея впровадження такої системи у форматі Telegram-бота є вдалим і логічним рішенням. З огляду на популярність Telegram як в Україні, так і у світі, користувачам не потрібно встановлювати додаткове програмне забезпечення - бот доступний буквально в один клік. Простий інтерфейс дозволяє зручно взаємодіяти із системою: достатньо лише надіслати текст і отримати відповідь. Такий підхід робить технології штучного інтелекту максимально доступними для широкого кола користувачів.

У цьому розділі також було проаналізовано ключові компоненти предметної області: користувача, перевірку новин та відгуки. Для цих сутностей було створено відповідні UML-діаграми - діаграма класів,

прецедентів і діяльності. Це дозволило чітко структурувати дані та продумати логіку взаємодії між елементами системи ще на етапі планування. У результаті стало зрозуміло, як саме буде функціонувати система, які дії виконує користувач і як ці дії обробляються всередині.

Отже, на основі проведеного аналізу можна сказати, що:

- проблема виявлення фейкових новин є суспільно значущою;
- сучасні моделі машинного навчання, зокрема BERT, дозволяють суттєво підвищити точність перевірки;
- використання Telegram-бота забезпечує зручний та доступний інтерфейс для користувачів;
- ретельне моделювання процесів і сутностей створює міцну основу для якісної реалізації та подальшого масштабування проєкту.

Таким чином, проведене дослідження предметної області, формулювання задачі, вивчення існуючих рішень і створення моделей забезпечили чітке бачення того, якою має бути система для виявлення фейкових новин і які її компоненти необхідно реалізувати в межах дипломного проєкту.

2 АНАЛІЗ ІНФОРМАЦІЙНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Перш ніж почати створення бази даних, необхідно чітко визначити, які саме дані потрібно зберігати, які таблиці знадобляться, як вони між собою пов’язані, і що саме міститиметься в кожній із них. Тут на допомогу приходить логічна модель даних - своєрідний ескіз або план, що дозволяє продумати структуру бази ще до написання SQL-запитів. Якщо порівняти, то логічна модель - це як архітектурне креслення перед зведенням будівлі: спершу все планується, і тільки потім реалізується.

У моєму проєкті в якості бази даних використовується SQLite - зручне і просте рішення для невеликих застосунків. Її перевага в тому, що вона не вимагає окремого сервера - всі дані зберігаються в одному файлі, і з ними можна працювати безпосередньо через Python.

Що таке ERwin?

Щоб краще уявити структуру бази даних, я використовував програму ERwin Data Modeler. Це спеціальний інструмент, який дозволяє створювати ER-діаграми (Entity-Relationship diagrams), тобто схеми, де зображено таблиці, їхні поля, типи даних і зв’язки між таблицями. Це надзвичайно зручно, бо на екрані видно повну «картину» бази - ніби карту або план.

ERwin часто застосовується у великих комерційних проєктах, де потрібно узгодити структуру бази між розробниками, бізнес-аналітиками і навіть замовниками. До того ж, він допомагає уникнути плутанини з назвами полів, типами даних чи зовнішніми ключами.

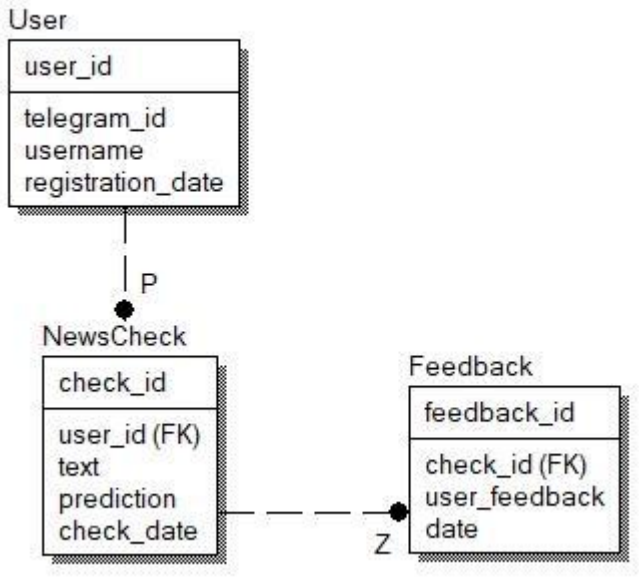

Рис. 7 ERWIN діаграма

На моїй ERWIN-діаграмі зображені три основні таблиці:

- **User** - зберігає користувачів Telegram, які користуються ботом. Кожен користувач має унікальний user_id, telegram_id, ім'я користувача та дату реєстрації.
- **NewsCheck** - таблиця для зберігання інформації про новини, які були перевірені. Тут є текст новини, дата перевірки, результат (prediction) та прив'язка до користувача (user_id).
- **Feedback** - таблиця з відгуками, де користувач може залишити коментар або оцінку після перевірки новини.

Усі ці таблиці пов'язані між собою за допомогою зовнішніх ключів (FOREIGN KEY), що дозволяє зберігати зв'язки типу «один до багатьох». Наприклад, один користувач може надіслати багато новин на перевірку, і кожна новина може мати свій відгук.

SQL-ініціалізація бази

Щоб ця структура реально працювала, я прописав функцію init_db() - вона створює всі потрібні таблиці в SQLite, якщо вони ще не існують. Ось фрагмент коду:

```
def init_db():

  conn = sqlite3.connect(DB_NAME)
```

```
cursor = conn.cursor()


cursor.execute('''
    CREATE TABLE IF NOT EXISTS User (
        user_id INTEGER PRIMARY KEY,
        telegram_id INTEGER,
        username TEXT,
        registration_date TEXT
    )
''')
cursor.execute('''
    CREATE TABLE IF NOT EXISTS NewsCheck (
        check_id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id INTEGER,
        text TEXT,
        prediction TEXT,
        check_date TEXT,
        FOREIGN KEY(user_id) REFERENCES User(user_id)
    )
''')
cursor.execute('''
    CREATE TABLE IF NOT EXISTS Feedback (
        feedback_id INTEGER PRIMARY KEY AUTOINCREMENT,
        check_id INTEGER,
        user_feedback TEXT,
```

```
        date TEXT,
        FOREIGN KEY(check_id) REFERENCES NewsCheck(check_id)
    )
''')
conn.commit()
conn.close()
```

Цей код запускається один раз при старті системи і створює таблиці, якщо їх ще немає. Далі вже можна з ними працювати - додавати, читати, оновлювати або видаляти дані.

Telegram API

Telegram API - це набір інструментів для розробки чат-ботів та інтеграції з Telegram. Використання Telegram API дозволяє створити чат-бота, який зможе взаємодіяти з користувачами, отримувати новини для перевірки та надавати результати.

Telegram надає ряд методів, таких як:

- sendMessage() - для надсилання повідомлень користувачам.
- getUpdates() - для отримання нових повідомлень або команд від користувачів.
- setWebhook() - для підключення бота до сервера для обробки повідомлень у реальному часі.

Ці методи дозволяють створити простий і зручний інтерфейс для користувачів, де вони можуть швидко перевіряти новини і отримувати результати без зайвих затримок

Таким чином, логічна модель даних, створена за допомогою ERWIN та реалізована на основі SQLite, забезпечує ефективне зберігання й обробку всієї інформації, що проходить через Telegram-бот. Telegram API, у свою чергу, виступає як канал взаємодії між користувачем і системою перевірки фейкових новин.

2.2 Вибір програмних засобів для розробки системи

Перш ніж розпочинати розробку будь-якої інформаційної системи, важливо ретельно обміркувати, на яких інструментах вона буде базуватись. Це як у майстерні: щоб побудувати щось надійне, спочатку обирають правильні інструменти - хтось візьме молоток і цвяхи, а хтось - шуруповерт. Те саме стосується вибору мови програмування, бібліотек, бази даних тощо - ці рішення безпосередньо впливають на ефективність, швидкість та надійність розробки.

У моєму випадку завдання досить складне: потрібно створити систему, яка аналізує тексти новин, класифікує їх за допомогою методів машинного навчання та взаємодіє з користувачами через Telegram-бота. Тобто маємо одразу кілька напрямів: обробка тексту, ML-класифікація, інтеграція з API та робота з базою даних.

Чому Python?

Основу проєкту склав Python - мова, яка заслужено вважається однією з найкращих для проєктів зі штучним інтелектом, обробки природної мови (NLP) і data science. Python має зрозумілий синтаксис, легкий старт для новачків і багатий вибір бібліотек для будь-яких завдань.

Кілька причин, чому саме Python:

- Має бібліотеки для машинного навчання - scikit-learn, tensorflow, transformers тощо.
- Зручно працює з API (через requests, aiohttp і т.п.).
- Добре підходить для Telegram-ботів, бо має готові бібліотеки (python-telegram-bot, aiogram).
- Працює на всіх популярних ОС (Windows, Linux, macOS).

Модель BERT і бібліотека transformers

Для класифікації новин на правдиві та фейкові я використав модель BERT - одну з найефективніших NLP-моделей на сьогодні. Вона чудово показала себе в різних задачах: від аналізу тональності до перекладу текстів. Я застосував її через бібліотеку transformers, яка забезпечує зручний доступ до сучасних мовних моделей.

Щоб працювати з BERT, я використав бібліотеку transformers від компанії Hugging Face. Вона дозволяє в кілька рядків коду:

- Завантажити вже готову натреновану модель.
- Передати в неї текст.
- Отримати відповідь (в нашому випадку - ймовірність того, що новина фейкова).

Це дуже економить час, бо не потрібно самому з нуля писати нейронну мережу. Все вже готове - бери й використовуй.

Бібліотека python-telegram-bot

Щоб реалізувати зручну взаємодію з користувачами, я зробив Telegram-бота. Для цього використав популярну бібліотеку python-telegram-bot. Вона дозволяє створювати бота, який може:

- читати повідомлення користувачів;
- відповідати на них;
- показувати кнопки, меню;
- реагувати на команди (типу /start, /help).

Перевага цієї бібліотеки в тому, що вона дуже гнучка та добре документована. У ній легко налаштовувати як прості відповіді на повідомлення, так і складні багатокрокові сценарії - з підтвердженнями, статистикою, зворотним зв’язком тощо.

Вибір СУБД - SQLite

Для зберігання даних (користувачі, результати перевірок, відгуки) я використав SQLite. Це така компактна, але досить потужна реляційна база даних, яка не потребує запуску окремого сервера. Усе працює прямо з одного .db-файлу.

Чому саме SQLite:

- Не потрібно нічого окремо встановлювати чи запускати.
- Підходить для невеликих і середніх проєктів.
- Працює швидко й стабільно.
- Ідеально інтегрується з Python (через вбудований модуль sqlite3).

Звісно, якщо система буде сильно масштабуватись (наприклад, десятки тисяч перевірок на день), то доведеться переходити на щось потужніше - типу PostgreSQL або MongoDB.

2.3 Опис алгоритмів для класифікації новин за допомогою машинного навчання

Коли мова заходить про виявлення фейкових новин, перше, що спадає на думку - це використання технологій машинного навчання (ML). У сучасних умовах це один із найдієвіших підходів, особливо коли йдеться про обробку великих обсягів текстової інформації. Алгоритми машинного навчання мають здатність виявляти приховані закономірності в даних - ті, які на перший погляд може не помітити навіть людина.

У моєму проєкті я обрав модель BERT, яка належить до класу трансформерних моделей. Вона вже давно вважається «золотим стандартом» у сфері обробки природної мови (NLP), адже вміє аналізувати текст не просто як послідовність слів, а з урахуванням контексту - з обох боків: зліва направо і справа наліво. Її можна порівняти з тим, як людина перечитує речення кілька разів, щоб краще його зрозуміти - тільки BERT робить це автоматично, набагато швидше і з високою точністю.

Як це працює?

Процес навчання такої моделі виглядає наступним чином:

1. **Підготовка даних**

Перший крок - це зібрати новини (реальні та фейкові) й привести їх до такого формату, з яким зможе працювати модель. На цьому етапі робиться "чистка" тексту: видаляються зайві символи, робиться нормалізація, приводиться все до нижнього регістру тощо. Далі текст перетворюється в числа - бо модель не "розуміє" слова, вона розуміє вектори, які представляють слова у математичному просторі.

2. **Токенізація**

Це процес, під час якого текст розбивається на токени - дрібні частинки, що можуть бути словами, частинами слів або навіть окремими літерами. Наприклад, фраза “це новина” може бути розбита на токени ["це", "нов",

"##ина"]. Токенізація потрібна, щоб модель могла обробити текст в уніфікованому вигляді.

3. **Навчання моделі**

Тут починається найцікавіше. Модель бере токенізований текст і, базуючись на вже натренованих параметрах, вчиться розпізнавати, які особливості тексту можуть свідчити про те, що він фейковий. По суті, вона намагається знайти патерни в словах, побудові речень, структурі інформації.

4. **Оцінка точності**

Після навчання модель перевіряється на спеціальному тестовому наборі, який вона не бачила під час тренування. Це потрібно, щоб оцінити, наскільки добре вона справляється з класифікацією новин. Тут ми дивимось на такі метрики, як точність, повнота, F1-score тощо.

Чому BERT, а не щось простіше?

Бо прості моделі типу логістичної регресії або SVM можуть добре працювати на невеликих задачах, але вони не вміють "розуміти" контекст. Наприклад, у реченні "Це не фейкова новина" ключовим є слово "не" - і без контексту легко помилитись. А BERT якраз створений для того, щоб враховувати подібні нюанси.

2.4 Реалізація моделі для виявлення фейкових новин

Для виявлення фейкових новин була використана трансформерна модель BERT, яка є однією з найбільш потужних моделей для обробки природної мови. Модель була навчена на великому наборі даних, який містив як реальні, так і фейкові новини. Процес навчання включав кілька етапів, серед яких підготовка даних, токенізація, навчання моделі, а також оцінка її ефективності на тестових даних.

2.4.1. Підготовка даних

Підготовка даних - це важливий етап, оскільки правильна підготовка тексту для моделі є основою для високої точності класифікації.

Спочатку були завантажені дані з файлу ua_dataset_2.csv, який містить тексти новин та відповідні мітки (1 - реальна новина, 0 - фейкова новина). Далі

дані були поділені на тренувальний та валідаційний набори за допомогою функції train_test_split з бібліотеки scikit-learn. Для тренування було обрано 80% даних, а для валідації - 20%.

```
from sklearn.model_selection import train_test_split

from transformers import BertTokenizer

import pandas as pd

# Завантажуємо дані

df = pd.read_csv('ua_dataset_2.csv')

# Розділяємо на тренувальні та валідаційні дані

train_data, val_data = train_test_split(df, test_size=0.2, random_state=42)
```

2.4.2. Токенізація

Для обробки тексту новин була використана BertTokenizer з бібліотеки Transformers, що дозволяє токенізувати текст, приводячи його до формату, зручного для моделі BERT. Токенізація включає в себе перетворення тексту в токени та їх вирівнювання до максимального розміру (256 символів). Це необхідно для забезпечення однакової довжини текстів при їх обробці.

```
        tokenizer = BertTokenizer.from_pretrained("bert-base-multilingual-cased")



        def tokenize(example):

          return tokenizer(example["text"], padding="max_length", truncation=True,
max_length=256)



        train_dataset = train_dataset.map(tokenize, batched=True)

        val_dataset = val_dataset.map(tokenize, batched=True)
```

2.4.3. Моделювання

Для класифікації новин використовувалась модель BertForSequenceClassification, яка є спеціалізованою версією BERT для задач класифікації текстів. Модель була ініціалізована з використанням попередньо натренованої версії bert-base-multilingual-cased, що підтримує багато мов, включаючи українську.

```
from transformers import BertForSequenceClassification

# Завантаження попередньо натренованої моделі BERT для класифікації

model = BertForSequenceClassification.from_pretrained("bert-base-multilingual-cased", num_labels=2)
```

2.4.4. Параметри тренування

Для тренування моделі були обрані наступні параметри:

```
from transformers import TrainingArguments


training_args = TrainingArguments(
  output_dir="./model_output",          # Директорія для збереження моделі
  per_device_train_batch_size=8,        # Розмір батчу для тренування
  per_device_eval_batch_size=8,         # Розмір батчу для оцінки
  num_train_epochs=3,                   # Кількість епох для тренування
  weight_decay=0.01,                    # Регуляризація
  logging_dir="./logs",                 # Директорія для логів
  logging_strategy="epoch",             # Логування після кожної епохи
  save_strategy="epoch",                # Збереження моделі після кожної епохи
  load_best_model_at_end=True,          # Завантаження кращої моделі в кінці
```

```
    save_total_limit=1,              # Ліміт збережених моделей
    report_to="none"                 # Без звітів
)
```

2.4.5. Тренування моделі

Після налаштування параметрів тренування, модель була тренована на тренувальному наборі даних та оцінена на валідаційному наборі. Тренування було виконано за допомогою Trainer з бібліотеки Transformers, що дозволяє автоматизувати процес тренування, оцінки та збереження моделі.

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer
)
```

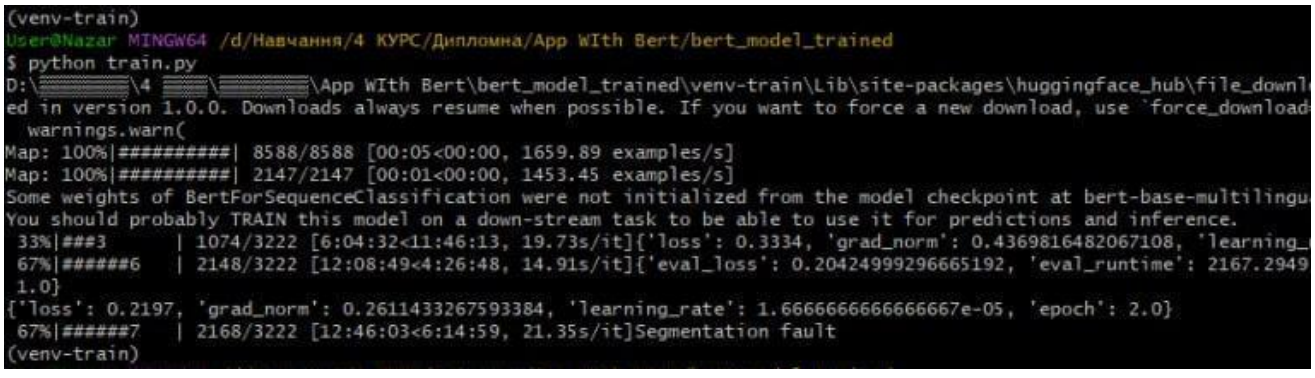

Рис. 8 Скріншот з Git Bash під час тренування

2.5 Висновки до 2 розділу

У другому розділі було проведено всебічний аналіз технічного та програмного забезпечення системи, що займається виявленням фейкових

новин. Цей розділ можна вважати своєрідною "технічною кухнею", де крок за кроком розкривається, як працює система: від структури бази даних до реалізації моделей машинного навчання. Усі елементи - наче частини одного механізму - взаємодіють між собою, забезпечуючи точну і швидку перевірку новин.

Першим кроком стало створення логічної моделі даних, яка заклала надійну основу для всієї системи. Базу даних було реалізовано у вигляді трьох таблиць - User, NewsCheck та Feedback. Для цього обрано SQLite, що виявилося ідеальним рішенням: проста, легка в налаштуванні, але водночас достатньо потужна для потреб нашого проєкту. Інструмент ERwin дозволив створити наочну ER-діаграму, що значно спростило розуміння взаємозв’язків ще до початку реалізації.

Важливу роль у проєкті відіграє Telegram API, який слугує зв’язковою ланкою між користувачем і системою. Завдяки бібліотеці python-telegram-bot вдалося створити простий, інтуїтивний інтерфейс: надсилаєш текст - отримуєш результат. Жодних додаткових дій, усе відбувається в межах уже звичного користувачу месенджера. Це робить систему максимально доступною.

Окрему увагу приділено блоку машинного навчання. Обрана модель - BERT - є одним із найсучасніших і найефективніших інструментів у сфері обробки природної мови. Вона читає текст не лише лінійно, а в обох напрямках, що дає змогу краще розуміти контекст. Це критично важливо у задачах виявлення фейків, де зміст повідомлення може змінюватись буквально від одного слова. Пройдені усі ключові етапи: підготовка даних, токенізація, навчання моделі та оцінка її точності. Особливо цінно, що модель тренувалася на реальному українському датасеті (ua_dataset_2.csv), тож вона краще "розуміє" локальну специфіку текстів.

Автоматизація навчання за допомогою класу Trainer із бібліотеки transformers дозволила значно спростити процес. Багато рутинних задач - від налаштування гіперпараметрів до запуску епох - тепер виконуються в кількох

рядках коду, що дозволяє більше зосередитися на якості результатів, а не на технічних деталях.

Ще один важливий момент - правильний вибір технологічного стеку. Я не просто використав популярні інструменти - кожен з них було обрано за критеріями зручності, стабільності, сумісності й активної підтримки спільноти. Python став "ядром" системи - завдяки своїй простоті, гнучкості й великій кількості готових рішень він дозволив швидко інтегрувати всі необхідні компоненти.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Коли розробляється будь-яка більш-менш складна система, важливо не просто писати код, а заздалегідь продумати архітектуру - тобто, як усе буде організовано, які компоненти існують, як вони взаємодіють між собою, і що за що відповідає. Це допомагає уникнути хаосу в коді, прискорює розробку і робить систему більш зрозумілою для інших розробників або навіть для самого себе через кілька тижнів .

У нашому випадку йдеться про систему, яка виконує досить цікаве завдання - перевіряє, чи є новина фейковою. Для цього ми використовуємо Telegram-бота, модель машинного навчання (BERT) і базу даних для збереження всієї важливої інформації. І вся ця конструкція має працювати узгоджено та без збоїв.

Компоненти системи

Архітектура побудована модульно - тобто кожен компонент системи має свою конкретну роль і відповідає за певну частину функціоналу. Таке розділення дуже зручне, бо дозволяє легше тестувати окремі частини, оновлювати або масштабувати систему при потребі.

1. **Модуль машинного навчання:**

Це "мозок" системи. Саме тут відбувається класифікація новин. Як тільки надходить текст - він проходить токенізацію та обробку через модель BERT, яка на основі контексту аналізує, наскільки ця новина виглядає правдоподібною чи фейковою.

Цікаво, що BERT не просто шукає ключові слова. Вона "розуміє", як слова зв’язані між собою. Наприклад, речення “Це не фейкова новина” - модель врахує слово "не" і правильно класифікує текст, тоді як проста модель могла б помилитись.

2. **Модуль Telegram-бота:**

Цей модуль - це наш інтерфейс взаємодії з користувачем. Людина не заходить на сайт, не заповнює форми, не запускає якісь програми - все

відбувається прямо в Telegram. Це дуже зручно і сучасно, бо Telegram вже встановлений у більшості користувачів, і бот виглядає як звичайний чат.

Telegram-бот отримує повідомлення від користувача, передає текст у модель, а потім повертає відповідь. Також він може реагувати на команди типу /start, показувати статистику, приймати фідбек після перевірки.

3. **База даних:**

Щоб усе працювало як слід, потрібно ще десь зберігати всю інформацію: хто що перевірив, коли, який був результат, що користувач написав у фідбеці тощо. Для цього ми використовуємо SQLite - легку реляційну базу даних, яка чудово підходить для невеликих систем.

База даних містить таблиці User, NewsCheck, Feedback, і через ці таблиці можна в будь-який момент дістати потрібну інформацію: скажімо, скільки новин перевірив конкретний користувач або який був відсоток фейкових новин за останній місяць.

Взаємодія компонентів

Щоб краще уявити, як усе це працює разом - уявимо звичайний сценарій:

1. Користувач надсилає новину боту.
2. Бот передає цей текст у модуль машинного навчання.
3. Модуль BERT класифікує новину.
4. Результат повертається користувачу через бота.
5. Інформація про перевірку записується в базу даних.

Цикл виглядає простим, але насправді за ним стоїть чітко спроєктована архітектура, де кожен компонент робить свою роботу і нічого зайвого.

Рис. 9 Діаграма архітектури програмного забезпечення

Опис діаграми:

- User - користувач, який взаємодіє з ботом.
- Модуль Telegram-бота - забезпечує інтерфейс для взаємодії з користувачем.
- Модуль машинного навчання - обробляє текст новини і здійснює класифікацію (реальна чи фейкова).
- База даних - зберігає результати перевірки новин і дані про користувачів.

Архітектура нашої системи побудована з урахуванням простоти, зручності та можливості розширення. Кожен модуль можна при потребі вдосконалити або замінити без переробки всієї системи. Такий підхід дозволяє гнучко розвивати проєкт у майбутньому: додати нові моделі, перейти на іншу СУБД або розширити функціонал бота - усе це можна зробити з мінімальними зусиллями.

3.2 Опис програмних модулів системи

Під час розробки будь-якої інформаційної системи важливо грамотно розподілити функціональність між різними частинами програми.

Замість того, щоб писати все в одному файлі, я розділив код на модулі - це окремі частини системи, які виконують конкретні завдання. Такий підхід значно спрощує підтримку, тестування та доопрацювання коду в майбутньому.

У моїй системі можна виділити кілька основних модулів: модуль навчання моделі, модуль перевірки новин, модуль Telegram-бота, а також модулі для роботи з базою даних і конфігурацією.

3.2.1. Модуль завантаження та тренування моделі

Цей модуль відповідає за те, щоб навчити модель BERT розпізнавати фейкові новини. Якщо коротко, то саме тут відбувається вся "магія" машинного навчання. Модуль бере датасет, обробляє текст, навчає модель і зберігає її у вигляді чекпоінта, який потім можна завантажити для реального використання в боті.

Я використовую бібліотеку transformers від Hugging Face, яка дозволяє легко працювати з попередньо натренованими моделями. Також застосовуються інструменти з torch, бо BERT - це глибока нейронна мережа, яка працює на базі PyTorch.

Окрема функція predict_bert() відповідає за отримання передбачення: вона отримує на вхід текст, проводить токенізацію, пропускає його через модель і повертає результат - фейк чи не фейк, з відсотком упевненості.

```
from transformers import BertTokenizer, BertForSequenceClassification
import torch


# Завантаження моделі з останнього checkpoint
MODEL_PATH = "./model_output/checkpoint-2148"


def load_model_and_tokenizer():
  tokenizer = BertTokenizer.from_pretrained(MODEL_PATH)
  model = BertForSequenceClassification.from_pretrained(MODEL_PATH)
  return model, tokenizer
```

```
def predict_bert(model, tokenizer, text):
  inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=512)
  with torch.no_grad():
      outputs = model(**inputs)
  probs = torch.nn.functional.softmax(outputs.logits, dim=-1)
  confidence, prediction = torch.max(probs, dim=1)
  confidence = confidence.item()
  label = prediction.item()

  if label == 1 and confidence >= 0.5:
      return "❌ Фейкова новина", confidence * 100
  elif label == 0 and confidence >= 0.5:
      return "☑ Реальна новина", confidence * 100
  else:
      return "⚠ Потрібна додаткова перевірка", confidence * 100
```

3.2.2. Модуль перевірки новин:

Цей модуль - своєрідний "посередник" між користувачем і моделлю. Саме тут обробляється повідомлення від користувача, викликається функція predict_bert, а потім результат повертається назад у Telegram.

Крім того, модуль відповідає за додаткову логіку перевірки - наприклад, він також аналізує джерела новини через функцію find_sources() та перевіряє, чи є серед них надійні (типу stopfake.org або spravdi.gov.ua).

І ще одна важлива функція - збереження результатів у базі даних (save_check()). Тобто вся історія перевірок фіксується, щоб потім можна було її переглянути або використати для статистики.

```
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_text = update.message.text
    user_id = update.effective_user.id

    # Якщо користувач чекає на відгук, зберігаємо відгук
    if context.user_data.get("awaiting_feedback"):
        save_feedback(user_id, user_text)
        await update.message.reply_text("☑ Дякую за твій відгук!")
        context.user_data["awaiting_feedback"] = False
        return

    # Перевірка новини через модель BERT
    label, confidence = predict_bert(model, tokenizer, user_text)

    sources = find_sources(user_text)
    context.user_data["last_news"] = user_text

    # Перевірка наявності фейкових слів у джерелах
    has_fake_indication = any(
        ("stopfake" in s["url"] or "spravdi" in s["url"]) and contains_fake_words(s["title"])
        for s in sources
    )
    trusted_count = sum(1 for s in sources if "stopfake" in s["url"] or "spravdi" in s["url"])

    # Оновлюємо результат перевірки на основі джерел
```

```
if has_fake_indication:
    label = "❌ Фейкова новина"
elif trusted_count >= 3:
    label = "☑ Реальна новина"

# Збереження результату в базі даних
save_check(user_id, user_text, label)

# Підготовка тексту відповіді для користувача
result_text = (
    f"✏ <b>Твоя новина:</b> {user_text}\n"
    f"📊 <b>Результат:</b> {label} ({confidence:.2f}%)"
)
        await update.message.reply_text(result_text, parse_mode="HTML",
reply_markup=get_feedback_keyboard())
```

Модуль взаємодії з Telegram API:

Telegram-бот - це наш інтерфейс. Саме через нього користувач бачить систему і взаємодіє з нею. Для цього я використав бібліотеку python-telegram-bot, яка дозволяє легко працювати з Telegram API.

Бот:

- приймає повідомлення;
- викликає функції класифікації;
- відправляє результати;
- показує кнопки для зворотного зв’язку;
- реагує на команди (/start, /help, тощо).

Усе це реалізовано в модулі bot.py, див. Додаток Б

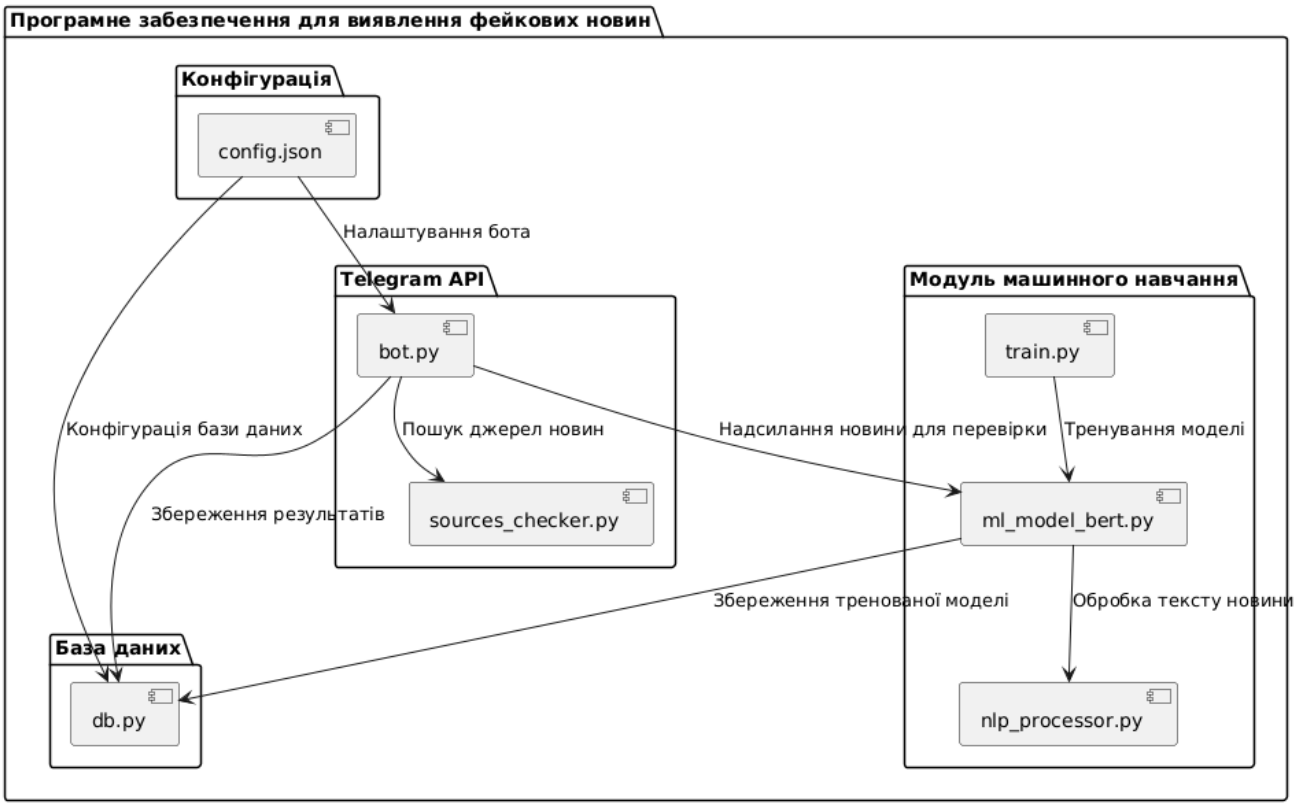

Рис. 10 Діаграма пакетів

Опис діаграми:

Модуль машинного навчання:

- train.py: відповідає за навчання моделі BERT, зберігання її параметрів.
- ml_model_bert.py: реалізація моделі BERT для класифікації новин.
- nlp_processor.py: попередня обробка текстів новин перед подачею їх до моделі BERT.

Telegram API:

- bot.py: обробляє запити користувачів через Telegram-бота.
- sources_checker.py: перевірка джерел новин, визначення їх надійності.

База даних:

- db.py: модуль для роботи з базою даних, збереження результатів перевірок новин та даних користувачів.

Конфігурація:

- config.json: налаштування системи для Telegram-бота та бази даних.

3.3 Реалізація Telegram-бота для перевірки новин

Telegram сьогодні - це не просто месенджер для спілкування. Це також зручна платформа для створення інтерактивних ботів, які можуть автоматизувати багато рутинних або аналітичних процесів. Саме тому для реалізації інтерфейсу системи перевірки новин було обрано саме Telegram-бота. Бот дозволяє взаємодіяти з користувачем у звичному для нього середовищі - прямо в чаті, без необхідності встановлювати окремі додатки або заходити на веб-сайт.

Бібліотека python-telegram-bot

Для створення Telegram-бота я використав бібліотеку python-telegram-bot. Це одна з найпопулярніших і зручних бібліотек для розробки ботів, яка:

- має простий і зрозумілий синтаксис;
- добре документована;
- підтримує як polling, так і webhook;
- дозволяє реалізувати різні елементи Telegram-інтерфейсу - кнопки, меню, inline-відповіді тощо.

Основні етапи роботи Telegram-бота:

Весь цикл взаємодії користувача з ботом виглядає досить просто й інтуїтивно. Це важливо, бо бота можуть використовувати як ІТ-спеціалісти, так і звичайні користувачі, які просто хочуть дізнатись, чи можна довіряти певній новині.

Ось як це працює:

1. **Користувач надсилає новину**

Людина пише або пересилає повідомлення у чат із ботом. Це може бути звичайний текст - бот сам визначить, чи потрібно його обробити.

2. **Бот передає текст у модуль перевірки**

Бот викликає функцію predict_bert() (з попередніх модулів), яка обробляє текст через модель BERT і повертає передбачення: фейкова, реальна або така, що потребує додаткової перевірки.

3. **Бот надсилає відповідь назад користувачеві**

Відповідь виглядає красиво оформленою - з емодзі, відсотком впевненості моделі та кнопками для оцінки результату (фідбеку).

4. **(Опціонально) користувач залишає відгук**

Якщо бот запропонує оцінити результат - користувач може натиснути кнопку або написати відповідь, яка зберігається у базі даних.

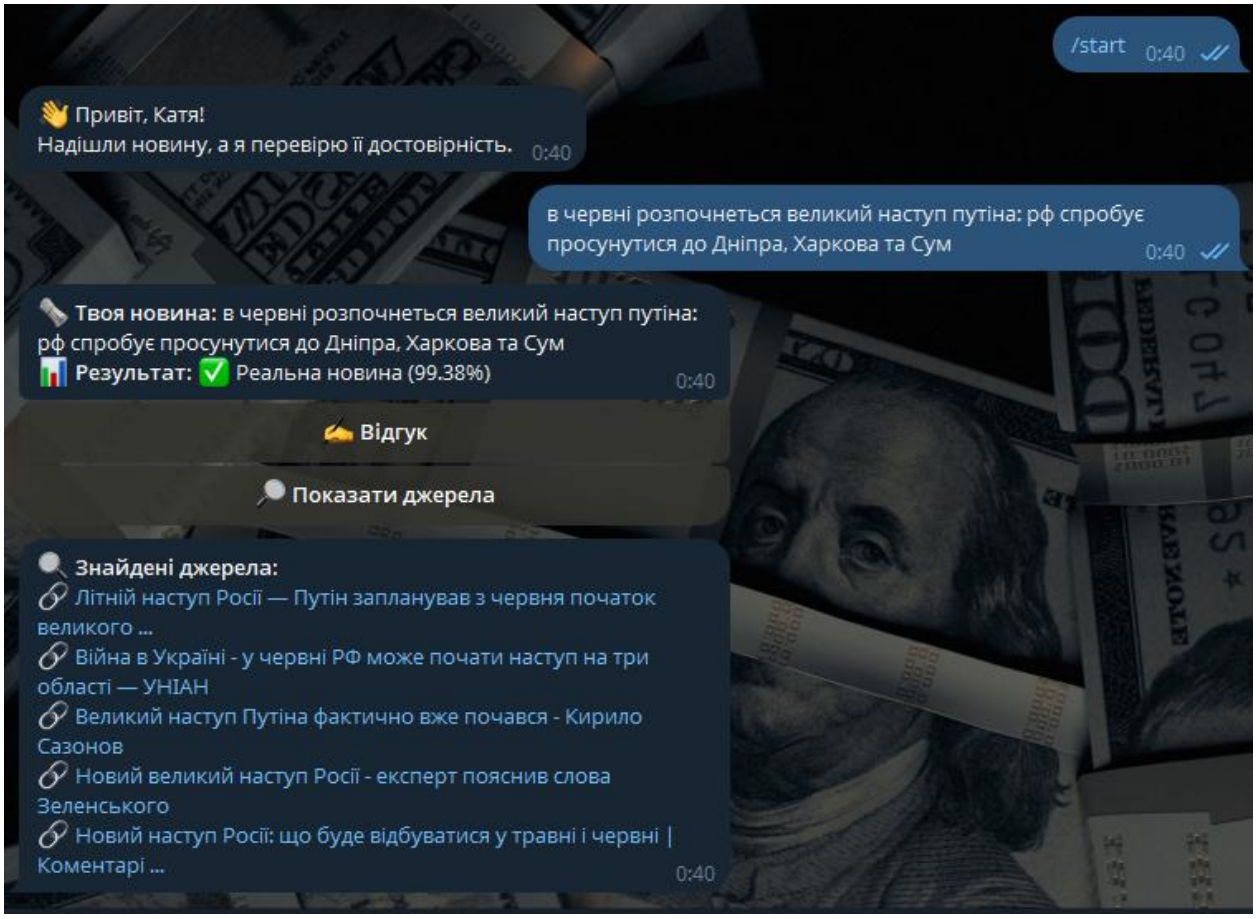

Рис. 11 Перевірка новини користувачем

На рисунку зображено типовий приклад роботи з ботом:

- Користувач надсилає новину.
- Бот повертає результат з точністю моделі.
- Користувач отримує пропозицію залишити фідбек або переглянути знайдені джерела

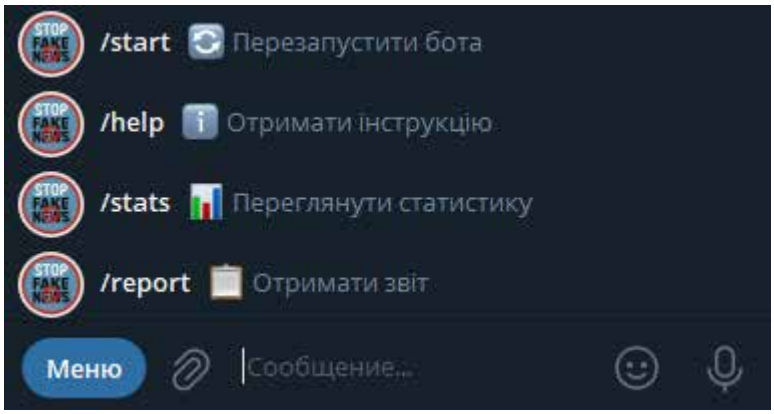

Рис. 12 Бокове меню адміністратора

Адміністративне меню

Окремо реалізовано бокове меню адміністратора (див. Рис. 10), яке дозволяє:

- переглядати свою статистику перевірок;
- переглядати інструкцію користування;
- отримати повну звітність по всіх перевірках;

Це меню доступне лише через ID адміністратора, щоб нею не могли скористатися звичайні користувачі.

3.4 Тестування роботи системи

Тестування включає кілька основних аспектів:

1. **Перевірка точності моделі:**

Оцінка точності моделі BERT проводиться на різних наборах даних, щоб зрозуміти її здатність до класифікації новин. Важливими показниками є точність, повнота та F1-міра, які дозволяють зрозуміти, як добре модель визначає фейкові новини.

Таблиця 1.5 Результати тестування моделі

Метрика	Значення
Точність	0,92
Повнота	0,90
Точність (Precision)	0,93
F1-міра	0,91

2. Тестування функціональності Telegram-бота:

Для тестування функціональності бота перевіряється його стабільність та зручність для користувачів. Це включає тестування на різних пристроях (мобільні телефони, комп'ютери) та перевірку, чи правильно бот взаємодіє з користувачами, чи передає правильні результати перевірки.

Таблиця 1.6 Результати тестування функціональності Telegram-бота

Крок тестування	Опис	Результат	Статус
Перевірка відправки новини	Користувач відправляє новину через бот	“Новина надіслана для перевірки”	Успішно
Отримання результату	Користувач отримує результат перевірки новини	“Реальна новина”	Успішно
Тестування швидкості відповіді	Час, за який бот дає результат після запиту	Менше 3 секунд	Успішно
Перевірка зручності інтерфейсу	Взаємодія з ботом на мобільних і десктопних версіях	Інтерфейс зручний і зрозумілий	Успішно

3.5 Висновок до 3 розділу

Розроблена система для виявлення фейкових новин виявилася потужним інструментом для оперативної перевірки достовірності інформації. Основними перевагами цієї системи є її висока точність, зручний інтерфейс користувача та можливість інтеграції з платформою Telegram для швидкої взаємодії.

Ключовим елементом системи є модуль машинного навчання, який використовує потужну модель BERT для класифікації новин. Модель має здатність обробляти тексти з урахуванням контексту, що дозволяє більш точно визначати, чи є новина реальнію чи фейковою. Завдяки цьому система демонструє високі результати за показниками точності (0,92), повноти (0,90) та F1-міри (0,91), що свідчить про її здатність ефективно розпізнавати фейкові новини.

Telegram-бот реалізує зручний інтерфейс для користувачів, що дозволяє легко надсилати новини для перевірки та отримувати результати в реальному часі. Бот є інтуїтивно зрозумілим, що дозволяє користувачам без спеціальних технічних знань швидко перевіряти новини. Це особливо важливо в умовах постійного потоку інформації, коли важливо оперативно отримати достовірні дані про новину.

Важливим аспектом є інтеграція з базою даних, яка зберігає не лише результати перевірки новин, а й історію перевірок для кожного користувача. Це дозволяє створювати персоналізовані звіти, статистику та покращувати взаємодію з користувачами через бот. Також база даних дозволяє зберігати та

аналізувати інформацію про новини, що використовуються для тренування моделі, і забезпечує постійну оптимізацію процесу.

У ході тестування було перевірено кілька аспектів роботи системи, зокрема точність моделі BERT, функціональність Telegram-бота, а також зручність інтерфейсу. Результати тестів показали, що система працює стабільно, а її точність класифікації новин відповідає високим стандартам. Тестування також підтвердило, що бот швидко обробляє запити (менше 3 секунд) і зручний у використанні на різних пристроях.

Отже, система для виявлення фейкових новин є високоефективним інструментом, який може бути корисним не тільки для індивідуальних користувачів, а й для великих організацій, які хочуть захистити себе від поширення неправдивої інформації. Доступність цієї системи через популярний месенджер Telegram забезпечує зручний доступ до перевірки новин будь-де і будь-коли. У майбутньому цю систему можна доповнити новими функціями, такими як виявлення маніпуляцій у текстах або розширення бази джерел новин, що зробить її ще більш потужним інструментом для боротьби з дезінформацією.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Перш ніж запускати будь-яку систему - особливо ту, яка працює з машинним навчанням, ботами та взаємодіє з користувачем у реальному часі - потрібно чітко розуміти, яке "залізо" і яке ПЗ потрібні для її стабільної роботи. Це як із комп’ютерними іграми: якщо система слабка, то буде гальмувати, і вся магія зникне.

У нашому випадку розроблена система включає кілька важливих компонентів - модель BERT для класифікації, Telegram-бот для взаємодії з користувачами, базу даних SQLite для збереження результатів і набір скриптів на Python. Тому важливо враховувати і апаратну, і програмну частину.

Апаратні вимоги:

Загалом, моя система не потребує надзвичайно потужного обладнання - її можна запустити навіть на звичайному ноутбуці. Але якщо хочеться, щоб усе працювало швидко й без лагів, то бажано дотримуватися таких мінімальних характеристик:

1. **Процесор:**

Мінімум двоядерний процесор із частотою 2.0 GHz або вище.

Хоч модель BERT і досить "важка", але її inference (тобто застосування до нових текстів) не надто навантажує CPU. Проте для швидкої обробки - краще мати сучасний багатоядерний процесор.

2. **Оперативна пам'ять (RAM):**

8 ГБ або більше.

Це оптимальний обсяг для комфортної роботи з Python, запуску моделі, Telegram-бота й усіх допоміжних процесів. Якщо використовувати більше одночасних користувачів або планується локальне тренування моделі - бажано мати 16+ ГБ.

3. **Місце на жорсткому диску:**

Принаймні 10 ГБ вільного простору.

Сюди входить місце для:

- збереження бази даних;
- збереження натренованої моделі;
- логів роботи бота;
- допоміжних файлів і бібліотек.

4. **(Опціонально) Відеокарта (GPU):**

Якщо планується локальне навчання моделі, то бажано мати відеокарту з підтримкою CUDA (наприклад, NVIDIA GTX 1660 або RTX 3060). Але для використання готової моделі GPU не обов'язковий - можна обійтися CPU.

Програмні вимоги:

Окрім "заліза", важливо підготувати і програмне середовище, щоб система могла працювати без помилок і збоїв.

1. **Операційна система:**

Підійде Windows 10/11, Linux (Ubuntu, Debian, Fedora тощо) або навіть macOS.

Оскільки Python кросплатформенний, систему можна запускати де завгодно - головне, щоб були встановлені всі необхідні пакети.

2. **Python (версія 3.9 або новіше):**

Це основна мова, на якій написано весь код бота, модель, обробка тексту, взаємодія з Telegram API тощо.

3. **Необхідні бібліотеки Python:**

Для роботи з Telegram, NLP та базами даних треба встановити декілька бібліотек. Їх можна інсталювати через pip install або прописати у requirements.txt. Основні з них:

- transformers - для роботи з BERT;
- torch - основа для нейронної мережі;
- python-telegram-bot - для Telegram-бота;
- pandas, numpy - для обробки та аналізу даних;
- sqlite3 (вбудований у Python) - для роботи з базою даних;

- scikit-learn (опціонально) - для оцінки результатів моделі;
- re - для регулярних виразів (пошук фейкових слів тощо).

4. **Інтерпретатор або середовище розробки:**

Можна використовувати будь-яке середовище - хоч VS Code, хоч PyCharm, хоч просто Jupyter Notebook. Головне - щоб підтримувалась інтеграція з Python 3 і встановлені всі потрібні модулі.

4.2 Інструкція з експлуатації системи

Щоб будь-яка система працювала не лише «на папері», а реально використовувалась людьми, дуже важливо продумати і описати інструкцію з експлуатації. Особливо це актуально для користувачів, які можуть не мати технічної освіти або досвіду роботи з подібними інструментами. У нашому випадку, Telegram-бот розроблений так, щоб бути максимально простим та інтуїтивним у використанні, але все ж деякі пояснення не завадять.

1. **Завантаження та інсталяція Telegram-бота**:

   Перший крок - це знайти самого бота у Telegram. Це можна зробити кількома способами:

   - через пошук у Telegram, @Fakenews14_bot;
   - за прямим посиланням, яке можна розмістити на сайті, в презентації або навіть просто переслати комусь у повідомленні, t.me/Fakenews14_bot.

   Після того як бот відкрився, потрібно натиснути на кнопку "Start". Це базова команда, яка активує бота і запускає перше привітальне повідомлення. З цього моменту бот "знайомиться" з користувачем і готовий до подальшої взаємодії.

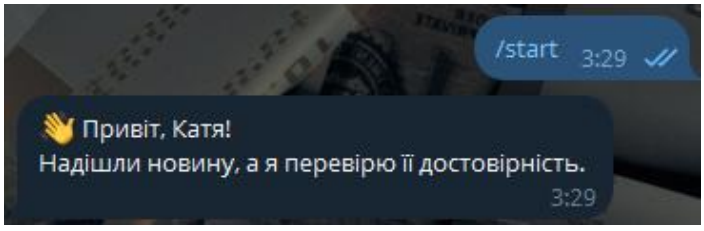

Рис. 13 Вітальне повідомлення

2. **Перевірка новини**:

   Основна функція бота - це перевірка новин на достовірність. І цей процес максимально простий:

   1. Користувач надсилає боту текст новини, який його цікавить (це може бути власне написаний текст або скопійований із сайту).
   2. Бот одразу реагує на повідомлення: проводить аналіз тексту за допомогою моделі BERT, а також перевіряє, чи є згадки про цю новину в надійних джерелах.
   3. Через кілька секунд користувач отримує відповідь - із результатом перевірки та рівнем упевненості моделі (у відсотках).

   Це дуже зручно: не потрібно розбиратися в алгоритмах чи налаштуваннях - достатньо просто надіслати текст.

3. **Функції бота**:

   Окрім перевірки новин, бот має ще кілька корисних функцій, які покращують взаємодію з користувачем і дозволяють більш гнучко використовувати систему.

   Персональна статистика

   Користувач може подивитися статистику своїх перевірок. Наприклад:

   - скільки новин він перевірив;
   - скільки з них були фейковими чи реальними;
   - скільки він залишив відгуків.

   Це реалізовано через окрему кнопку або команду, яка доступна в головному меню бота.

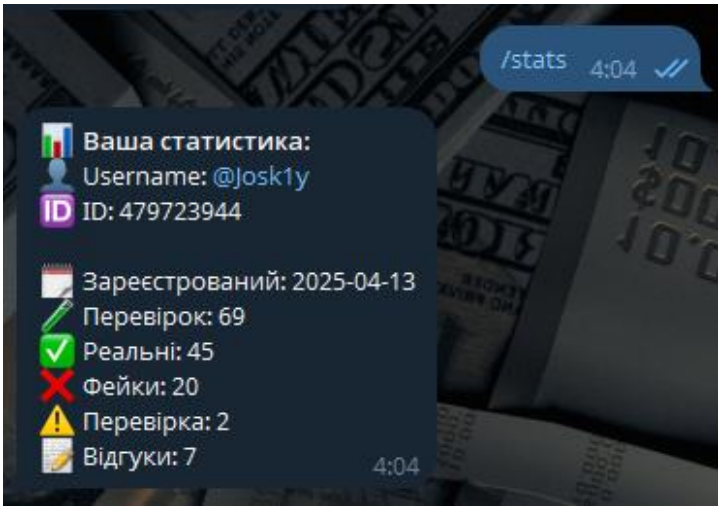

Рис. 14 Персональна статистика користувача

Залишення відгуку

Після кожної перевірки бот пропонує залишити відгук. Це важливо для подальшого вдосконалення системи і можливого донавчання моделі на реальних фідбеках. Відгук можна надіслати у вигляді тексту одним повідомленням.

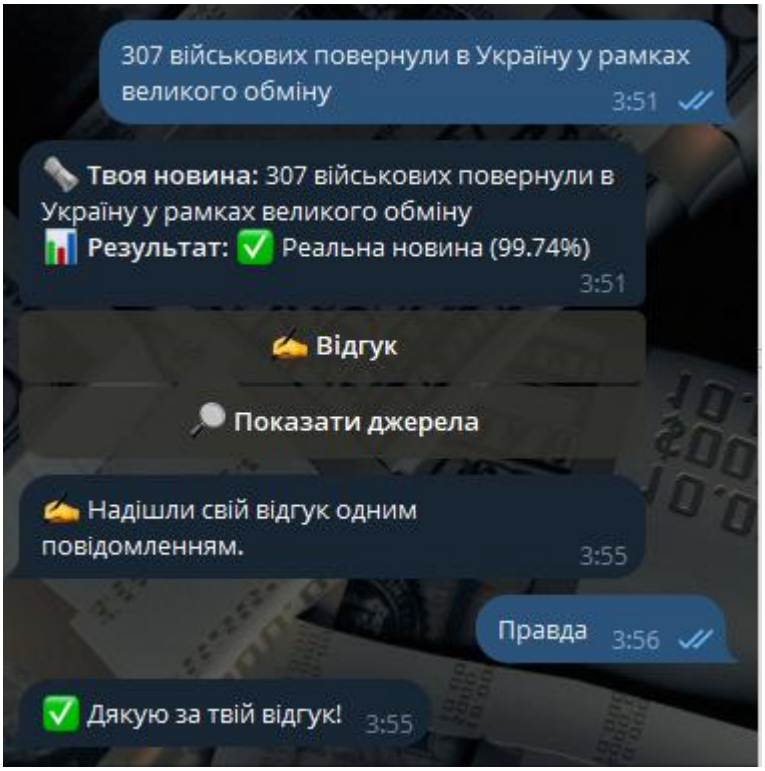

Рис. 15 Користувач залишив свій відгук

Функціонал для адміністратора (звітність)

Система також передбачає адмін-доступ для спеціальних користувачів. Їхні Telegram ID прописуються у файлі config.json. Адміністратори мають розширені можливості, наприклад:

- генерувати загальну статистику по всіх перевірках;
- бачити кількість користувачів;
- бачити кількість відгуків;

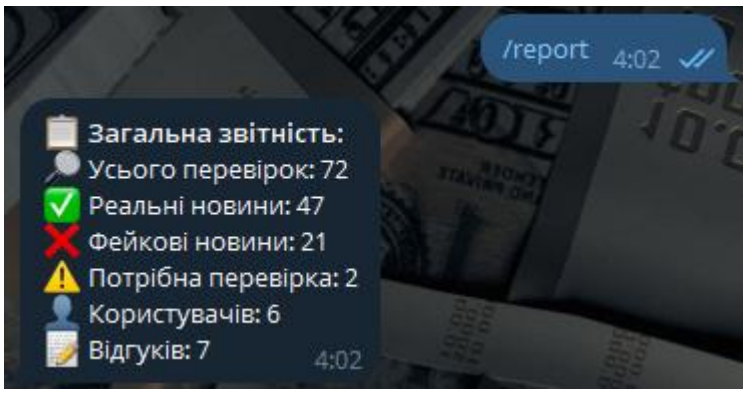

Рис. 16 Згенерована звітність для адміністратора

4.3 Опис інтерфейсу користувача Telegram-бота

Telegram-бот у даному проєкті виступає як головний інтерфейс між користувачем та системою машинного навчання. Саме через нього людина взаємодіє з програмою: надсилає текст новини, отримує результат, залишає відгук, переглядає статистику тощо. Тому особливу увагу було приділено простоті та зручності інтерфейсу, щоб кожен користувач - незалежно від віку чи рівня технічної підготовки - зміг швидко розібратись, як працює система.

Бот створено з урахуванням принципу "нічого зайвого" - він не перевантажений непотрібною інформацією або складними командами. Усе, що потрібно зробити - це просто надіслати текст новини в чат, а далі система вже зробить усе сама.

1. **Привітальне повідомлення**:

Після того, як користувач вперше запускає бота (натискає кнопку Start),

він одразу отримує коротке вітальне повідомлення. У ньому зазначено, що саме вміє бот, які функції доступні, і як ними скористатися. Це щось на кшталт міні-інструкції: не треба шукати довідку або гуглити - усе вже є в одному повідомленні.

Такий підхід особливо зручний для новачків, які вперше користуються ботами: вони не губляться, не ставлять зайвих запитань і одразу розуміють, що робити далі.

2. **Кнопки для вибору дій**:

Щоб зробити навігацію ще зручнішою, в боті передбачено вбудовані кнопки, які значно полегшують користування. Немає потреби щось вводити вручну - усе доступне "в один тап".

- Перезапустити бота - для повернення до початкового меню.
- Отримати інструкцію - для отримання пояснення щодо того, як користуватися ботом.
- Переглянути статистику - дозволяє користувачу подивитися, скільки новин він перевірив, скільки з них були фейковими, коли він вперше скористався ботом.

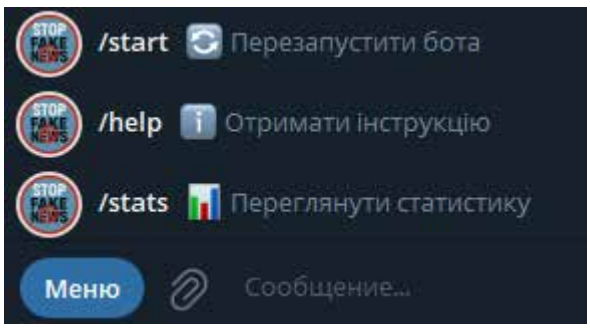

Рис. 17 Бокове меню користувача

3. Результат перевірки:

Після того як користувач надіслав новину на перевірку, бот обробляє текст і надає результат, чи є новина реальна, чи фейкова. Якщо новина була перевірена, бот відразу відправляє користувачеві відповідь, що

вказує на її достовірність. Бот також пропонує додаткові варіанти для подальших дій:

- залишити відгук,
- переглянути посилання на джерела,

4. Зворотний зв'язок:
Після кожної перевірки користувачеві надається можливість залишити фідбек. Це може бути:

- згода/незгода з результатом;
- коментар про точність;
- пропозиції щодо покращення.

Всі ці відгуки зберігаються у базі даних і можуть бути використані для подальшого навчання моделі або вдосконалення логіки роботи бота.

Цей механізм є важливим елементом зворотного зв'язку з користувачем, що робить систему не просто односторонньою, а інтерактивною.

5. Простота навігації:
Інтерфейс Telegram-бота спеціально продуманий так, щоб у користувача не виникало зайвих запитань чи складнощів:

- усі основні дії винесені у вигляді кнопок;
- повідомлення написані простою мовою;
- не потрібно знати жодних команд - усе інтуїтивно зрозуміло;
- можна у будь-який момент перезапустити бота і повернутись до головного меню.

Такий підхід значно підвищує зручність використання, а також знижує бар’єр входу навіть для тих, хто взагалі не користувався ботами до цього.

4.4 Висновок до 4 розділу

Однією з головних переваг системи є її простота в користуванні. Завдяки інтуїтивно зрозумілому інтерфейсу, користувач може швидко та без зусиль відправити новину на перевірку, отримати результат класифікації та навіть залишити зворотний зв'язок, щоб допомогти вдосконалити роботу системи. Інтерфейс бота побудований на основі простих кнопок і чітких команд, що дозволяє користувачам без додаткових зусиль переходити між різними функціями бота. Це особливо важливо для тих, хто не знайомий з технічними деталями роботи системи.

Привітальне повідомлення та меню з кнопками забезпечують користувачеві чітку інструкцію про те, як почати використовувати систему, що робить процес запуску бота максимально зручним.

Результати перевірки новин надаються у простій та зрозумілій формі. Користувач отримує чітку відповідь щодо достовірності новини, що дозволяє йому швидко орієнтуватися в ситуації. Окрім того, система пропонує можливість залишити відгук або переглянути джерела, що додає функціоналу та робить перевірку новин ще більш достовірною.

Важливим елементом є звітність для адміністраторів. Можливість згенерувати звіт про всі перевірки новин, що виконуються через бота, дозволяє контролювати роботу системи, відслідковувати активність користувачів та аналізувати загальні результати. Це особливо корисно для адміністраторів, які можуть використовувати ці дані для подальшої оптимізації та покращення роботи бота.

З технічної точки зору система ефективно використовує сучасні алгоритми машинного навчання, зокрема модель BERT, для класифікації новин. Це дозволяє отримувати високу точність результатів при виявленні фейкових новин, що є критичним у сучасному інформаційному просторі, де неправдива інформація може мати значні наслідки.

Завдяки використанню Telegram API, система має відмінну інтеграцію

з популярною платформою для обміну повідомленнями, що дає змогу користувачам у будь-який час і в будь-якому місці перевіряти новини через звичний їм інтерфейс. Це робить систему доступною навіть для тих, хто не має спеціальних знань у сфері програмування чи роботи з іншими складними системами.

Загалом, Telegram-бот для перевірки новин надає потужний, простий у використанні інструмент для боротьби з дезінформацією. Він може стати важливим елементом у стратегії захисту користувачів від фейкових новин, допомагаючи забезпечити достовірність інформації в онлайн-середовищі. Система добре справляється зі своєю основною задачею, демонструючи високу ефективність і точність роботи.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було розроблено функціональну систему для виявлення фейкових новин на основі моделі машинного навчання BERT, інтегровану з Telegram-ботом для зручної взаємодії з користувачами. У роботі було проведено комплексний аналіз предметної області, вивчено сучасні підходи до виявлення фейків, обґрунтовано вибір програмних засобів та реалізовано архітектурно зважене прикладне рішення.

На етапі дослідження було встановлено, що проблема поширення фейкових новин має значний суспільний резонанс, особливо в умовах надмірного інформаційного навантаження на пересічного користувача. Система автоматичної перевірки новин, побудована на сучасних технологіях, здатна значно підвищити якість інформаційного простору, зменшити ризики дезінформації та сприяти формуванню більш свідомого медіаспоживання.

Застосування моделі BERT у якості ядра системи дозволило досягти високої точності класифікації завдяки її здатності враховувати контекст і мовні особливості текстів. Важливо підкреслити, що вибір саме цієї моделі обґрунтовано її ефективністю в обробці природної мови, особливо у завданнях двокласової класифікації.

Telegram-бот, реалізований за допомогою бібліотеки python-telegram-bot, виступає як зручний засіб комунікації між користувачем і системою. Завдяки простому інтерфейсу взаємодії користувач має змогу в реальному часі перевірити новину, не потребуючи складних дій чи технічних знань. Це робить систему доступною для широкого кола осіб.

У ході роботи було також реалізовано базу даних на SQLite, яка забезпечує збереження інформації про користувачів, перевірені новини та

отримані відгуки. Створена логічна модель бази даних дозволяє легко масштабувати систему в майбутньому або інтегрувати її з іншими сервісами.

Таким чином, розроблена система не лише демонструє технічну спроможність використання сучасних підходів у сфері машинного навчання, а й має високу прикладну цінність. Вона може бути використана у сфері журналістики, освіти, державного управління та інформаційної безпеки. Крім того, потенційно система може слугувати основою для розширеного програмного комплексу з аналітикою, перевіркою джерел, побудовою репутаційних профілів новинних ресурсів тощо.

Підсумовуючи, зазначаю, що поставлені цілі та завдання роботи були повністю досягнуті. Запропоноване рішення є актуальним, технічно реалізованим та має реальну перспективу для подальшого вдосконалення і впровадження у практичну діяльність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention is all you need // Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017) [Електронний ресурс]. - Режим доступу: https://arxiv.org/abs/1706.03762
2. Devlin J., Chang M. W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of NAACL-HLT 2019 [Електронний ресурс]. - Режим доступу: https://arxiv.org/abs/1810.04805
3. Yang X., Zhang Z., Jin X. Fake news detection on social media: A data mining perspective // ACM Computing Surveys (CSUR). - 2020. - Vol. 53, № 4. - P. 1-34. - DOI: https://doi.org/10.1145/3387173
4. Zellers R., Holtzman A., Rashkin H., Ramaraju B., Choi Y. Defending against neural fake news // Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAccT 2020). - DOI: https://doi.org/10.1145/3351095.3372856
5. Python Software Foundation. Python Documentation [Електронний ресурс]. - 2021. - Режим доступу: https://www.python.org/doc/
6. Hugging Face. Transformers Documentation [Електронний ресурс]. - 2021. - Режим доступу: https://huggingface.co/transformers/
7. Python Telegram Bot. python-telegram-bot Documentation [Електронний ресурс]. - 2021. - Режим доступу: https://python-telegram-bot.readthedocs.io/
8. Brownlee J. How to Develop BERT for Text Classification // Machine Learning Mastery [Електронний ресурс]. - 2021. - Режим доступу: https://machinelearningmastery.com/using-bert-for-text-classification/

9. Real Python. Introduction to the Telegram Bot API [Електронний ресурс]. - 2021. - Режим доступу: https://realpython.com/python-telegram-bot/
10. Kaggle. Fake News Detection Dataset [Електронний ресурс]. - 2020. - Режим доступу: https://www.kaggle.com/c/fake-news/data
11. Kross E., Ameli M. Evaluating the effectiveness of fake news detection systems using various evaluation metrics // Journal of Information Technology. - 2020. - Vol. 35, № 2. - P. 193-207. - DOI: https://doi.org/10.1057/s41265-020-00132-w
12. Silver D., Hubert T., Schrittwieser J., та ін. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm // Nature. - 2017. - Vol. 550. - P. 350-354. - DOI: https://doi.org/10.1038/nature24270
13. SQLite. SQLite Documentation [Електронний ресурс]. - 2021. - Режим доступу: https://www.sqlite.org/docs.html
14. Liao Y., Wu S. A Survey on Fake News Detection // International Journal of Information Technology and Computer Science (IJITCS). - 2019. - Vol. 11, № 3. - P. 10-20. - DOI: https://doi.org/10.5815/ijitcs.2019.03.02
15. Telegram. Telegram Bot API Documentation [Електронний ресурс]. - 2021. - Режим доступу: https://core.telegram.org/bots/api

ДОДАТОК А

Модуль для перевірки надійних джерел

```
import requests
from bs4 import BeautifulSoup
import urllib.parse


# Надійні домени новин
TRUSTED_DOMAINS = [
    "bbc.com", "cnn.com", "reuters.com", "dw.com", "nytimes.com", "gov.uk",
    "nasa.gov", "ukrinform.ua", "pravda.com.ua", "un.org", "apnews.com",
    "forbes.com", "hromadske.ua", "censor.net",
    "glavcom.ua", "rbc.ua", "eurointegration.com.ua", "suspilne.media", "unian.ua",
    "espreso.tv", "lb.ua", "ukranews.com", "radiosvoboda.org", "24tv.ua", "zn.ua"
]


HEADERS = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
}
```

```
def is_trusted(url):
    domain = urllib.parse.urlparse(url).netloc
    return any(trusted in domain for trusted in TRUSTED_DOMAINS)
def clean_url(href):
    # Витягує чисте посилання з редиректу
    if "duckduckgo.com/l/?" in href:
        parts = urllib.parse.parse_qs(urllib.parse.urlparse(href).query)
        if "uddg" in parts:
            return parts["uddg"][0]
    return href
def find_sources(query, max_results=5):
    encoded_query = urllib.parse.quote(query)
    url = f"https://html.duckduckgo.com/html/?q={encoded_query}"
    try:
        response = requests.get(url, headers=HEADERS, timeout=10)
        soup = BeautifulSoup(response.text, "html.parser")
        results = []
        for result in soup.select(".result"):
            a_tag = result.find("a", class_="result__a")
            if not a_tag:
```

```
            continue

        title = a_tag.get_text(strip=True)

        href = a_tag.get("href")

        href = clean_url(href)

        domain = urllib.parse.urlparse(href).netloc

        if not href.startswith("http"):

            continue

        result_entry = {

            "title": title,

            "url": href,

            "domain": domain,

            "trusted": is_trusted(href)

        }

        results.append(result_entry)

        if len(results) >= max_results:

            break

    return results

except Exception as e:

    print("❌ Error in source checking:", e)

    return []
```

ДОДАТОК Б

Повний код файлу Telegram-бота

```
import json

import logging

from telegram import (Update, InlineKeyboardButton, InlineKeyboardMarkup,
BotCommand, BotCommandScopeDefault, BotCommandScopeChat)

from telegram.ext import (ApplicationBuilder, CommandHandler, MessageHandler,
CallbackQueryHandler, ContextTypes, filters)

from ml_model_bert import load_model_and_tokenizer, predict_bert

from sources_checker import find_sources, TRUSTED_DOMAINS

from db import init_db, save_user, save_check, save_feedback, get_user_stats,
get_report_stats

with open("config.json", "r", encoding="utf-8") as f:

  config = json.load(f)

TOKEN = config["token"]

ADMIN_ID = config["admin_id"]

logging.basicConfig(level=logging.INFO)

model, tokenizer = load_model_and_tokenizer()

def get_main_commands(is_admin=False):

  commands = [

    BotCommand("start", "🔄 Перезапустити бота"),
```

```
            BotCommand("help", "ℹ️ Отримати інструкцію"),
            BotCommand("stats", "📊 Переглянути статистику"),
        ]
        if is_admin:
            commands.append(BotCommand("report", "📋 Отримати звіт"))
        return commands
    def get_feedback_keyboard():
        return InlineKeyboardMarkup([
            [InlineKeyboardButton("✍️ Відгук", callback_data="feedback")],
            [InlineKeyboardButton("🔎 Показати джерела", callback_data="sources")]
        ])
    def contains_fake_words(title):
        title_lower = title.lower()
        fake_indicators = ["фейк", "брехня", "маніпуляція"]
        return any(word in title_lower for word in fake_indicators)
    async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
        user = update.effective_user
        save_user(user.id, user.username)
        await context.bot.set_my_commands(get_main_commands(False),
scope=BotCommandScopeDefault())
```

```
        if user.id == ADMIN_ID:

            await context.bot.set_my_commands(get_main_commands(True),
scope=BotCommandScopeChat(chat_id=ADMIN_ID))

        await update.message.reply_text(

            f"👋 Привіт, {user.first_name}!\nНадішли новину, а я перевірю її
достовірність."

        )

    async def help_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):

        await update.message.reply_text(

            "ℹ️ Інструкція користування:\n"

            "1. Надішли мені текст новини.\n"

            "2. Я визначу, чи це _реальна новина_, _фейк_ або потрібна додаткова
перевірка.\n"

            "3. Натисни \"Показати джерела\"  - отримаєш список новинних сайтів.\n\n"

            "📌 Команди:\n"

            "/start - перезапуск\n"

            "/help - інструкція\n"

            "/stats - перегляд статистики\n"

            + ("/report - звіт (тільки для адміністратора)" if update.effective_user.id ==
ADMIN_ID else ""),

            parse_mode="Markdown"
```

```
            )

        async def stats_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
            user = update.effective_user
            stats = get_user_stats(user.id)
            await update.message.reply_text(
                f"📊 *Ваша статистика:*\n"
                f"👤 Username: @{user.username}\n"
                f"🆔 ID: {user.id}\n\n"
                f"🗓 Зареєстрований: {stats['registration_date']}\n"
                f"🔍 Перевірок: {stats['total_checks']}\n"
                f"✅ Реальні: {stats['real']}\n"
                f"❌ Фейки: {stats['fake']}\n"
                f"⚠️ Перевірка: {stats['unsure']}\n"
                f"📝 Відгуки: {stats['feedbacks']}",
                parse_mode="Markdown"
            )

        async def handle_message(update: Update, context:
ContextTypes.DEFAULT_TYPE):
            user_text = update.message.text
```

```
        user_id = update.effective_user.id
        if context.user_data.get("awaiting_feedback"):
            save_feedback(user_id, user_text)
            await update.message.reply_text("☑ Дякую за твій відгук!")
            context.user_data["awaiting_feedback"] = False
            return
        label, confidence = predict_bert(model, tokenizer, user_text)
        sources = find_sources(user_text)
        context.user_data["last_news"] = user_text
        has_fake_indication = any(
            ("stopfake" in s["url"] or "spravdi" in s["url"]) and
contains_fake_words(s["title"])
            for s in sources
        )
        trusted_count = sum(1 for s in sources if "stopfake" in s["url"] or "spravdi" in
s["url"])
        if has_fake_indication:
            label = "❌ Фейкова новина"
        elif trusted_count >= 3:
            label = "☑ Реальна новина"
        save_check(user_id, user_text, label)
```

```
            result_text = (
                f"✏️ <b>Твоя новина:</b> {user_text}\n"
                f"📊 <b>Результат:</b> {label} ({confidence:.2f}%)"
            )
            await update.message.reply_text(result_text, parse_mode="HTML",
reply_markup=get_feedback_keyboard())
        async def handle_callback(update: Update, context:
ContextTypes.DEFAULT_TYPE):
            query = update.callback_query
            await query.answer()
            user = update.effective_user
            if query.data == "feedback":
                await query.message.reply_text("✍️ Надішли свій відгук одним
повідомленням.")
                context.user_data["awaiting_feedback"] = True
            elif query.data == "sources":
                news_text = context.user_data.get("last_news")
                if not news_text:
                    await query.message.reply_text("⚠️ Спершу надішли новину для
перевірки.")
                    return
                sources = find_sources(news_text)
```

```
            if sources:
                sources_text = "🔍 <b>Знайдені джерела:</b>\n"
                for s in sources:
                    title = s['title']
                    url = s['url']
                    sources_text += f'🔗 <a href="{url}">{title}</a>\n'
            else:
                sources_text = "❌ Надійних джерел не знайдено."
            await query.message.reply_text(sources_text, parse_mode="HTML",
disable_web_page_preview=True)

        async def report_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
            if update.effective_user.id != ADMIN_ID:
                await update.message.reply_text("⛔ У вас немає доступу до цієї команди.")
                return
            stats = get_report_stats()
            await update.message.reply_text(
                f"📋 <b>Загальна звітність:</b>\n"
                f"🔎 Усього перевірок: {stats['total']}\n"
                f"☑️ Реальні новини: {stats['real']}\n"
                f"❌ Фейкові новини: {stats['fake']}\n"
```

```
            f"⚠ Потрібна перевірка: {stats['unsure']}\n"
            f"👤 Користувачів: {stats['users']}\n"
            f"📝 Відгуків: {stats['feedbacks']}",
            parse_mode="HTML"
        )
    def start_bot():
        init_db()
        app = ApplicationBuilder().token(TOKEN).build()
        app.add_handler(CommandHandler("start", start))
        app.add_handler(CommandHandler("help", help_command))
        app.add_handler(CommandHandler("stats", stats_command))
        app.add_handler(CommandHandler("report", report_command))
        app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))
        app.add_handler(CallbackQueryHandler(handle_callback))
        app.run_polling()
```