Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра комп’ютерних інтелектуальних технологій та систем
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
## Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Інтелектуальна система оцінки достовірності новин на базі Telegram-бота із застосуванням технологій машинного навчання
(тема)


Виконав:
здобувач IV курсу, групи КІУКІ-21-10
Яким СОКОЛОВ
(прізвище, ініціали)


Спеціальність 123 – Комп’ютерна інженерія
(код і повна назва спеціальності)

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

Освітня програма Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: асист. каф. КІТС Кирило ОЛІЙНИК
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТС _____ Олег РУДЕНКО
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра комп'ютерних інтелектуальних технологій та систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Соколову Якиму Борисовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система оцінки достовірності новин на базі Telegram-бота із застосуванням технологій машинного навчання

затверджена наказом університету від 21.05.2025 р. № 399Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 24 червня 2025 р.

3. Вихідні дані до роботи Текстові новини, провокативність тексту, джерело інформації, мова програмування Python

4. Перелік питань, що потрібно опрацювати в роботі
Яким чином чат-бот буде інтегровано з месенджером Telegram?
Які функції має виконувати розроблений чат-бот?
Які інструменти та технології будуть використані для розробки?
Яким чином буде здійснюватися перевірка новин на достовірність?
За якими критеріями будуть оцінюватись новини?

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
Скріншоти взаємодії з чат-ботом
Слайди презентації (13 слайдів)

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Строк / терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Видача та узгодження теми проєкту | 26.05.2025 | виконано |
| 2 | Огляд і аналіз сучасного стану поставленої задачі | 30.05.2025 | виконано |
| 3 | Аналіз літератури за напрямком дослідження | 01.06.2025 – 04.06.2025 | виконано |
| 4 | Огляд та аналіз існуючих рішень для оцінки новин | 05.06.2025 | виконано |
| 5 | Вибір та розробка мультимодального підходу для оцінки новини на надійність | 06.06.2025 – 08.06.2025 | виконано |
| 6 | Розробка алгоритму оцінки провокативності тексту | 09.06.2025 – 11.06.2025 | виконано |
| 7 | Оформлення матеріалів кваліфікаційної роботи | 12.06.2025 – 18.06.2025 | виконано |
| 8 | Розробка презентації | 19.06.2025 | виконано |
| 9 | Подання до ЕК | 24.06.2025 | виконано |
| 10 | Захист проєкту | 25.06.2025 | виконано |

Дата видачі завдання 26 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ асист. каф. КІТС Олійник К.О.
(підпис) (посада, власне ім'я, прізвищ

# РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 68 с., 2 табл., 5 рис., 2 дод. та 45 джерел.

НЕЙРОННІ МЕРЕЖІ, ФЕЙКИ, НОВИНИ, ІНФОРМАЦІЙНІ ДЖЕРЕЛА, МАШИННЕ НАВЧАННЯ, МОДЕЛЬ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ІНФОРМАЦІЙНА БЕЗБЕКА

Метою роботи є дослідження методів виявлення фейкових новин використовуючи мультимодальний підхід.

Методи дослідження - проведення аналізу літератури, попередньо проведених робіт на цю тему та практична реалізація проекту.

Об’єтом дослідження є  методи виявлення фейкових та провокативних новин та інформації в мережі Інтернет.

Предметом дослідження є система виявлення та класифікації фейкових новин, визначення надійності інформаційного джерела в Інтернеті.

# ABSTRACT

Qualification thesis: 68 pages, 2 tables, 5 figures, 2 appendices and 45 sources.

NEURAL NETWORKS, FAKE NEWS, INFORMATION SOURCES, MACHINE LEARNING, MODEL, SOFTWARE, INFORMATION SECURITY

The purpose of the study is to explore methods for detecting fake news using a multimodal approach.

Research methods – analysis of literature, previously conducted works on the topic, and practical implementation of the project.

The object of the research is the methods for detecting fake and provocative news and information on the Internet.

The subject of the research is a system for detecting and classifying fake news, as well as assessing the reliability of information sources on the Internet.

ЗМІСТ

# ВСТУП

У сучасному інформаційному просторі питання достовірності новин стає надзвичайно актуальним. З поширенням цифрових медіа та соціальних мереж, дезінформація поширюється швидше, ніж будь-коли раніше, що створює значні загрози для стабільності суспільства.

Особливої актуальності набуває створення інструментів, здатних автоматично визначати фейкові новини, оцінювати надійність джерел та виявляти провокаційний характер повідомлень.

Метою кваліфікаційної роботи є створення програмної системи для автоматизованого аналізу новин з використанням методів машинного навчання, зокрема логістичної регресії та TF-IDF. Розроблена система реалізована у вигляді Telegram-бота, який збирає, аналізує та оцінює інформацію з новинних повідомлень.

Об’єктом дослідження є процеси поширення інформації в цифровому середовищі. Предметом дослідження — методи визначення достовірності новин, надійності джерел та провокаційності тексту.

Відповідно до мети, у дипломній роботі вирішуються наступні задачі:

– виконати аналіз предметної області та існуючих рішень;

– розробити архітектуру системи з окремими модулями для збору даних, аналізу тексту, оцінки достовірності та провокаційності;

– реалізувати Telegram-бота для взаємодії з користувачем;

– впровадити логістичну регресію для класифікації новин;

– провести експериментальне тестування системи;

– підготувати інструкцію користувача.

Результатом є інтелектуальна система, яка допомагає виявляти дезінформацію та підтримує критичне сприйняття новин.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Огляд існуючих рішень для аналізу новин

У контексті зростання обсягів цифрової інформації, що циркулює в медіа-просторі, виникає об’єктивна потреба в розробці ефективних інструментів для виявлення та фільтрації недостовірних повідомлень. У цьому розділі розглядаються сучасні підходи до перевірки новин та визначаються переваги запропонованого рішення у форматі Telegram-бота.

Існуючі рішення та їхні обмеження

– Платформи фактчекінгу. Сайти на кшталт PolitiFact, Full Fact та Snopes здійснюють незалежну перевірку фактів, орієнтуючись на журналістські стандарти достовірності. Їхня ефективність базується на експертному аналізі, однак така ручна перевірка є ресурсозатратною та повільною, що обмежує її застосування в умовах великого потоку інформації [1].

– Системи з використанням штучного інтелекту. Сервіси на кшталт AdVerif.ai застосовують алгоритми машинного навчання для виявлення фейкових новин, а Alto Analyzer — для моніторингу соціальних дискусій та оцінки наративів. Однак, як засвідчує дослідження Meta AI, ці алгоритми можуть бути вразливими до маніпуляцій текстом, які навмисне обходять критерії класифікації [2].

– Браузерні розширення та рейтингові системи. NewsGuard пропонує систему оцінювання надійності новинних сайтів на основі прозорих критеріїв: наявність редакційної політики, відокремлення фактів від думок тощо. Проте рейтингова система не завжди адаптується до локальних джерел або не має повного охоплення інформаційного простору [3].

– Наукові моделі та дослідження. Серед останніх робіт у цій сфері — моделі на основі трансформерів (наприклад, BERT або RoBERTa), які

показують високу точність в класифікації фейкових новин [4]. Однак їхнє використання у реальному часі потребує суттєвих обчислювальних ресурсів та якісних наборів даних для навчання.

## 1.2 Актуальність розробки Telegram-бота

У сучасному інформаційному середовищі, де обсяги даних зростають експоненційно, виникає нагальна потреба в ефективних інструментах для оцінки достовірності новин. Особливо це актуально в умовах гібридної війни, де дезінформація використовується як інструмент впливу на громадську думку та підриву довіри до офіційних джерел [5]

Telegram, як один із найпопулярніших месенджерів, став платформою для швидкого поширення інформації, включаючи фейкові новини та пропаганду. Через відсутність ефективної модерації контенту, Telegram використовується для розповсюдження дезінформації, що створює загрозу для інформаційної безпеки [6].

У відповідь на ці виклики, розробка Telegram-бота для автоматизованої перевірки новин набуває особливої актуальності. Такий бот може використовувати методи штучного інтелекту та обробки природної мови для аналізу тексту, виявлення фейкових новин та оцінки надійності джерел [7].

Таким чином, розробка Telegram-бота для перевірки достовірності новин є актуальним завданням, що сприяє підвищенню інформаційної безпеки, розвитку медіаграмотності та протидії дезінформації в сучасному цифровому середовищі. Дане рішення дозволяє автоматизувати процес перевірки інформації та забезпечити користувачів ефективним інструментом для швидкої та зручної перевірки новин в режимі реального часу.

## 1.3 Переваги Telegram-бота

Розроблений Telegram-бот враховує недоліки зазначених рішень і поєднує низку функцій, що забезпечують ефективність аналізу новинного контенту:

– Інтеграція з месенджером. Telegram — одна з найбільш популярних платформ для споживання інформації в Україні. Вбудована реалізація бота дозволяє користувачу взаємодіяти безпосередньо в знайомому середовищі, не покидаючи месенджер [8].

– Автоматизований аналіз. Бот використовує методи NLP і класифікаційні моделі для визначення фейковості повідомлень, що дозволяє зменшити час обробки запиту в десятки разів у порівнянні з ручною перевіркою [9].

– Оцінка надійності джерел. Система має вбудовану базу перевірених джерел, яка постійно оновлюється. На її основі бот виконує порівняльний аналіз новин, виявляючи розбіжності в подачі фактів.

– Аналіз емоційного фону. Через аналіз емоційної лексики бот може виявити потенційно маніпулятивні або провокаційні повідомлення, що є важливим для розпізнавання навмисної дезінформації [10].

– Масштабованість. Telegram-бот може обробляти сотні запитів одночасно, зберігаючи історію запитів користувача, що робить його придатним для широкого застосування в освітніх, журналістських та дослідницьких практиках.

Таким чином, запропонований підхід дозволяє вирішити основні проблеми, притаманні існуючим системам, та забезпечити швидку, точну й доступну перевірку достовірності новин у мобільному середовищі.

## 1.4 Постановка задачі

Виконання кваліфікаційної роботи спрямоване на створення інтелектуальної системи у вигляді Telegram-бота, призначеного для

автоматизованого аналізу новинного контенту з метою виявлення фейкових повідомлень, оцінки достовірності джерел та виявлення провокаційного змісту.

Основна ідея проєкту полягає у розробці багатофункціонального Telegram-бота, який дозволить користувачам надсилати текстові новини безпосередньо через месенджер та отримувати оперативну оцінку їх достовірності. Система має забезпечувати аналіз новинного тексту за допомогою методів обробки природної мови (NLP) та алгоритмів машинного навчання.

На основі проведеного аналізу існуючих рішень, а також визначених їх переваг і недоліків, було сформульовано основні вимоги до функціоналу та технічної реалізації системи. Зокрема, Telegram-бот повинен підтримувати попередню обробку тексту, векторизацію, класифікацію за рівнем достовірності, перевірку джерела на основі бази надійних ресурсів, а також виявлення маніпулятивних та провокативних фрагментів.

Запропонована система має бути здатною працювати в реальному часі, забезпечуючи масштабованість, доступність та зручність для кінцевого користувача.

## 1.5 Системні вимоги

Для взаємодії з Telegram-ботом користувачеві не потрібно мати спеціалізоване обладнання або додаткове програмне забезпечення. Достатньо наявності пристрою з підтримкою месенджера Telegram — смартфона, планшета або персонального комп’ютера.

Рекомендовані умови використання:

– встановлений клієнт Telegram (мобільний додаток, десктопна програма або веб-версія);

– стабільне підключення до Інтернету, що забезпечує надсилання запитів і отримання відповідей у режимі реального часу;

– можливість введення тексту (через віртуальну або фізичну клавіатуру).

Оскільки всі обчислення, включно з обробкою тексту та роботою мовної моделі, виконуються на стороні сервера, вимоги до апаратної частини пристрою користувача є мінімальними. Це забезпечує доступність системи для широкого кола користувачів.

## 1.6 Опис функцій Telegram-бота

Розроблювана система у вигляді Telegram-бота передбачає реалізацію ряду функціональних можливостей, спрямованих на забезпечення зручного та ефективного інструменту для перевірки новинного контенту. Основні функції, які реалізовані або передбачені до впровадження, наведено нижче:

– Перевірка новин за URL-посиланням. Користувач має можливість надіслати боту гіперпосилання на новинну публікацію. Система автоматично здійснює парсинг вмісту сторінки, витягує текст новини, після чого виконує його аналіз щодо достовірності та потенційної провокаційності. Також у цьому сценарії можлива перевірка надійності джерела, з якого отримано матеріал. На рисунку 1.1 зображено результат аналізу новини за URL-посиланням.

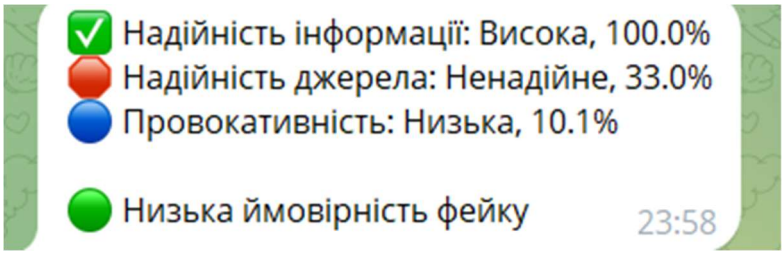


Рисунок 1.1 – Приклад аналізу URL-посилання

– Перевірка новини за її текстовим вмістом. Користувач може вручну ввести або вставити текст новини без посилання на джерело. У такому випадку бот аналізує повідомлення за двома критеріями:

– Достовірність змісту (на основі мовних моделей та статистичних ознак);

– Провокативність (зокрема, наявність емоційно забарвлених або маніпулятивних фраз).

Оцінка надійності джерела у цьому режимі не проводиться. Приклад аналізу новини за її текстом зображено на рисунку 1.2.

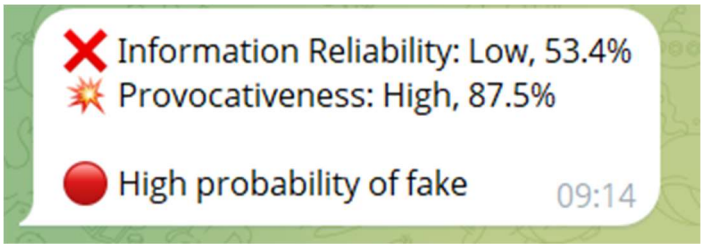


Рисунок 1.2 – Приклад аналізу новини за її текстом

– Отримання довідкової інформації. Telegram-бот має функцію інформування користувача про свої можливості. Команда допомоги (/help) викликає структуроване повідомлення з описом усіх доступних функцій, прикладами запитів і форматом введення.

Приклад довідкової інформації на англійській мові наведено на рисунку 1.3.

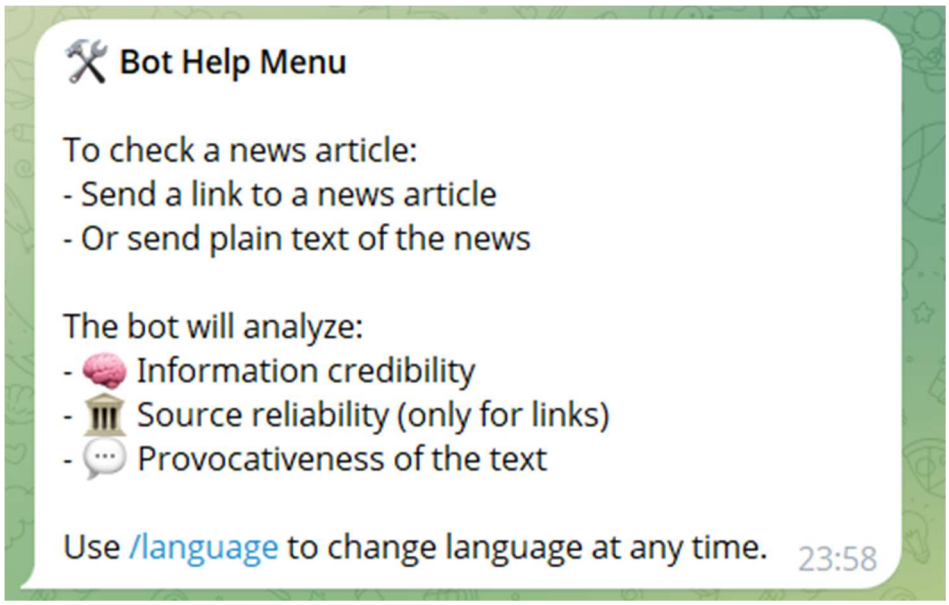


Рисунок 1.3 – Приклад довідкової інформації на англійській мові.

Також для поліпшення інформативності користувача присутнє додаткове привітальне повідомлення, що дозволяє ознайомитись з призначенням бота. Привітальне повідомлення зображено на рисунку 1.4.


Рисунок 1.4. – Привітальне повідомлення.

– Мультимовна підтримка. Інтерфейс бота адаптований для кількох мов. Користувач може змінити мову взаємодії (у тому числі на українську та англійську) за допомогою відповідної команди або кнопки в меню. Приклад меню для зміни мови інтерфейсу зображений на рисунку 1.5.

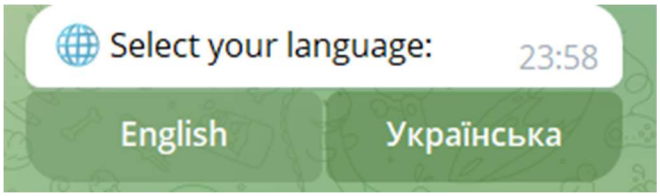


Рисунок 1.5 – Інтерфейс меню для зміни мови інтерфейсу.

Отже, розроблений чат-бот забезпечує базовий функціонал для зручної роботи двома різними мовами, що створює комфортні умови для користувачів із різномовним середовищем. У разі потреби перелік підтримуваних мов може бути розширено без суттєвих змін у загальній архітектурі системи. Крім того, користувач має змогу змінювати мову інтерфейсу та отримувати розширену інформацію щодо можливостей бота, що підвищує рівень персоналізації та зручності взаємодії з системою.

# 2 АНАЛІЗ ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

## 2.1. Алгоритми та методи класифікації текста

Класифікація тексту є ключовим завданням в обробці природної мови (NLP), що знаходить широке застосування у виявленні фейкових новин, аналізі тональності, категоризації та модерації контенту. Ефективність таких систем залежить від коректного представлення тексту та застосування відповідного алгоритму машинного навчання.

Одним з базових методів представлення тексту є модель «мішка слів» (Bag-of-Words), яка перетворює текст у вектор, що відображає кількість входжень кожного слова з фіксованого словника. Хоча цей підхід ігнорує контекст і порядок слів, він залишається популярним через свою простоту та швидкодію [11].

Поширеним удосконаленням є TF-IDF (Term Frequency – Inverse Document Frequency), що враховує не лише частоту терміну в окремому документі, але й його поширеність у всьому корпусі. Це дозволяє зменшити вплив загальновживаних слів і підвищити значущість ключових термінів [12]. Згідно з [13], TF-IDF показує високу ефективність у поєднанні з лінійними класифікаторами, зокрема логістичною регресією, оскільки створює розріджені, інтерпретовані вектори ознак.

Серед класичних алгоритмів класифікації тексту логістична регресія посідає одне з провідних місць. Це статистичний метод, що оцінює ймовірність належності об’єкта (тексту) до певного класу. У випадку з фейковими новинами, модель визначає ймовірність того, що новина є правдивою або недостовірною. Однією з переваг логістичної регресії є її інтерпретованість: кожна вага, що прив’язана до окремого терміну у TF-IDF-векторі, відображає його внесок у позитивну чи негативну класифікацію [14].

На думку дослідників [13], логістична регресія демонструє стабільні результати в задачах бінарної класифікації, не потребує великих обчислювальних ресурсів і легко масштабовується. У поєднанні з TF-IDF вона дозволяє швидко будувати легкі, але ефективні системи для аналізу тексту — що є особливо актуальним у випадку реалізації на мобільних або обмежених платформах, зокрема в Telegram-ботах.

Інші методи, зокрема наївний байєс, метод опорних векторів (SVM), або глибокі нейромережі, мають свої переваги, однак потребують більшого налаштування або ресурсів [15].

Рекурентні нейронні мережі (RNN), наприклад, ефективно працюють з послідовністю слів, а їх модифікація LSTM (Long Short-Term Memory) — дозволяє моделювати залежності у довгих текстах [16]. Конволюційні нейронні мережі (CNN) можуть виділяти локальні шаблони та ефективно працювати із короткими повідомленнями [17].

Найбільш сучасні підходи базуються на трансформерах, зокрема моделі BERT. Вона дозволяє враховувати контекст у всьому реченні завдяки двонаправленому аналізу, що робить її надзвичайно ефективною у завданнях семантичної класифікації [18].

У контексті Telegram-бота доцільним є комбінування логістичної регресії на TF-IDF векторах для базового аналізу й експериментальне впровадження BERT-моделей для складніших випадків. Це дозволяє зберігати високу швидкість і забезпечити точність у класифікації новин у реальному часі.

## 2.2. Аналіз етапів оцінки достовірності новини

Оцінка новинного повідомлення як достовірного або сумнівного є багатоступеневим процесом, що передбачає аналіз не лише змісту тексту, але й джерела, з якого отримано інформацію, а також загального тону та маніпулятивності викладу. У межах розроблюваного Telegram-бота

запропоновано реалізувати три послідовні етапи перевірки, які дозволяють комплексно оцінити новину.

### 2.2.1 Оцінка надійності новини

На першому етапі аналізується зміст новинного повідомлення. Метою є виявлення фактологічних неточностей, перекручень або ознак фейковості. Для цього застосовується метод порівняння вхідної новини з подібними статтями, отриманими з надійних джерел. Надійні джерела підбираються згідно з мовою та регіоном оригінального тексту, що дозволяє уникати помилок, пов’язаних з машинним перекладом, та забезпечити точніше семантичне порівняння.

Пошук статей із перевірених джерел здійснюється через Google Search Console — сервіс, що дозволяє здійснювати пошук через Google у межах обраного списку сайтів або по всій мережі. API підтримує кастомізацію запитів, включно з мовою, регіоном, доменами, типами файлів тощо [19]. Безкоштовна квота становить до 10000 запитів на добу, після чого діє система оплати.

Отримані матеріали та вихідний текст новини проходять попередню обробку: лемматизацію та векторизацію. Далі модель MPNET виконує порівняння новин за допомогою метрики косинусної подібності. Встановлюється пороговий рівень схожості (від 0 до 1), де 0 означає повну невідповідність, а 1 — ідентичність. Якщо схожість перевищує встановлений поріг, новина вважається достовірною. У результатах аналізу цей показник конвертується у відсоткову шкалу (0–100%) та демонструється користувачу.

### 2.2.2 Оцінка надійності джерела

На другому етапі здійснюється перевірка доменного імені джерела, з

якого отримано новину. Telegram-бот звертається до внутрішньої бази даних, яка містить класифікацію джерел за кількома категоріями:

– умовно надійні джерела — верифіковані ресурси, що мають репутацію стабільних постачальників перевіреної інформації;

– умовно ненадійні джерела — ресурси, що мають історію публікацій фейкових, маніпулятивних або непідтверджених матеріалів;

– сатиричні джерела — інформаційні майданчики, що створюють вигаданий або жартівливий контент з розважальною метою;

– інші джерела — ресурси, надійність яких обчислюється автоматично на основі попередніх аналізів.

Для кожного джерела з категорії "інші" у базі даних зберігається числовий показник надійності. Цей показник формується статистично: що частіше ресурс публікує правдиві новини згідно з результатами змістовної перевірки, тим вищим є його рейтинг. У випадку, якщо джерело виявлено вперше, йому присвоюється нейтральне значення до моменту накопичення достатньої кількості перевірок.

Такий механізм дозволяє автоматизувати процес оцінювання, уникнути ручного втручання та враховувати динамічні зміни поведінки джерел з часом. Підсумковий показник надійності джерела включається до аналітичного звіту, який користувач отримує після надсилання запиту на перевірку новини.

### 2.2.3 Оцінка провокативності тексту

Останній етап включає виявлення маніпулятивних або емоційно забарвлених формулювань у новині. Застосовується попередньо навчена модель, що поєднує TF-IDF векторизацію та логістичну регресію.

TF-IDF дозволяє виявляти терміни з найвищим інформаційним навантаженням, зменшуючи вагу загальновживаних слів [20]. На основі зважених TF-IDF-значень модель логістичної регресії оцінює ймовірність

того, що текст є провокаційним — подає конфліктно забарвлену, маніпулятивну або емоційно насичену інформацію [21].

Критерії, за якими визначається провокаційність, включають:

– використання надмірно емоційної лексики;

– заклики до агресії, протистояння чи насильства;

– спроби викликати страх, гнів, паніку;

– формулювання, що містять упереджені оцінки або надмірну категоричність.

На виході користувач отримує оцінку в шкалі від 0 до 100 % із супровідним коментарем (наприклад, "висока ймовірність провокаційності").

## 2.3. Огляд технологій та інструментів для розробки Telegram-бота

### 2.3.1 Мова програмування Python

Python — це високорівнева, інтерпретована, об'єктно-орієнтована мова програмування з динамічною типізацією, яка здобула популярність завдяки своїй простоті, читабельності та широкому спектру застосувань. Вона була створена Гвідо ван Россумом і вперше випущена в 1991 році як наступник мови ABC, з акцентом на зручність використання та розширюваність [22].

Однією з ключових особливостей Python є його синтаксис, який наголошує на читабельності коду та дозволяє розробникам виражати концепції в меншій кількості рядків порівняно з іншими мовами програмування. Це досягається завдяки використанню відступів для визначення блоків коду, а не фігурних дужок чи ключових слів, як у багатьох інших мовах [23].

Python підтримує кілька парадигм програмування, включаючи об'єктно-орієнтоване, процедурне та функціональне програмування. Це

забезпечує гнучкість у розробці програмного забезпечення та дозволяє розробникам вибирати найбільш відповідний стиль програмування для конкретного завдання [24].

Мова також відома своєю великою стандартною бібліотекою, яка включає модулі для роботи з різноманітними задачами, такими як обробка тексту, робота з файлами, мережеве програмування, веб-розробка, тестування та багато іншого. Крім того, Python має активну спільноту розробників, яка створила безліч сторонніх бібліотек і фреймворків, таких як Django для веб-розробки, NumPy для наукових обчислень та TensorFlow для машинного навчання [25].

Завдяки своїй простоті, гнучкості та потужності, Python став однією з найпопулярніших мов програмування у світі та широко використовується в різних галузях, включаючи веб-розробку, науку про дані, штучний інтелект, автоматизацію та багато іншого [26].

### 2.3.2 Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) — це безкоштовне, кросплатформне середовище розробки, створене компанією Microsoft для операційних систем Windows, macOS та Linux. Воно поєднує в собі легкість текстового редактора з функціональністю повноцінного IDE, надаючи потужні інструменти для розробки програмного забезпечення [27].

Серед основних можливостей VS Code:

– Підтримка багатьох мов програмування. VS Code забезпечує вбудовану підтримку для JavaScript, TypeScript та Node.js, а також дозволяє додавати розширення для Python, C++, Java, Go, Rust, PHP тощо.

– Інтелектуальне автодоповнення (IntelliSense). Редактор надає розумні підказки під час написання коду, включаючи автозавершення, параметри функцій, опис типів тощо.

– Вбудований налагоджувач. VS Code дозволяє виконувати

налагодження безпосередньо з редактора, встановлювати точки зупину, переглядати стек викликів, змінні тощо.

– Інтеграція з Git. Програма підтримує системи контролю версій, включно з Git та GitHub, дозволяючи переглядати зміни, виконувати коміти, зливання гілок та інші операції.

– Підтримка розширень. Через Visual Studio Code Marketplace користувачі можуть встановлювати теми, інструменти для тестування, інтерпретатори мов, налагоджувачі та інші функції.

Завдяки простоті, гнучкості, швидкості та підтримці величезної кількості інструментів VS Code став одним із найпопулярніших середовищ розробки у світі [27].

### 2.3.3 Парадигма ООП

У процесі розробки Telegram-бота використовувалося об’єктно-орієнтоване програмування (ООП) як основна парадигма побудови логіки додатку. Python, як мова з повноцінною підтримкою ООП, дозволяє організовувати код у вигляді класів, які інкапсулюють дані (атрибути) та поведінку (методи), що безпосередньо пов’язана з цими даними.

У рамках реалізованого проєкту були створені окремі класи для обробки повідомлень, взаємодії з базою даних, управління сесіями користувачів і викликів до мовних моделей.

Такий підхід дозволив досягти високого рівня модульності, що спростило підтримку та масштабування системи.

Основними перевагами ООП є:

– Модульність: кожен клас реалізує окрему логіку, що дозволяє легко змінювати або розширювати функціональність.

– Повторне використання коду: завдяки спадкуванню можна створювати нові класи на основі існуючих без дублювання коду.

– Інкапсуляція: приховування внутрішньої реалізації класу зменшує

залежність між компонентами системи.

– Зручність тестування та налагодження: логіка, ізольована в класах, легше піддається юніт-тестуванню.

Водночас ООП має і деякі недоліки. Зокрема, у невеликих проєктах застосування класів може ускладнювати код без суттєвої користі. Крім того, надмірне використання спадкування призводить до складної ієрархії класів, що ускладнює підтримку коду.

Незважаючи на ці обмеження, для системи середньої складності, якою є Telegram-бот з NLP-функціональністю, використання ООП є виправданим і дозволяє забезпечити чітку архітектуру проєкту та спрощує розширення функціоналу в майбутньому [28].

### 2.3.4 Мовна модель MPNet

MPNet (Masked and Permuted Pre-trained Network) — це сучасна трансформерна мовна модель, яка поєднує в собі переваги двох провідних підходів до попереднього навчання: маскування токенів (як у BERT) та перестановки порядку слів (як у XLNet). На відміну від своїх попередників, MPNet реалізує модифіковану архітектуру на основі encoder-only трансформера, в якій одночасно застосовується інформація про позицію та контекст, що забезпечує глибше семантичне розуміння тексту [29].

Однією з ключових особливостей MPNet є попереднє тренування з одночасним використанням маскування та перестановки токенів, що дозволяє моделі краще засвоювати як локальні, так і глобальні залежності у тексті. Завдяки цьому MPNet демонструє вищу точність у таких завданнях, як класифікація текстів, оцінка схожості речень, семантичний пошук та виявлення дублікатів.

У межах реалізованого Telegram-бота модель MPNet застосовувалася у складі бібліотеки sentence-transformers для отримання векторних

представлень вхідного тексту. Кожне речення або повідомлення перетворюється у вектор фіксованої довжини, який відображає його семантичний зміст. Далі порівняння між новинами здійснюється за допомогою косинусної подібності.

Косинусна подібність є поширеною метрикою векторної подібності у просторі ознак. Вона дозволяє оцінити ступінь схожості між двома текстами, представленими у вигляді векторів. Значення косинусної подібності варіюється від 0 до 1, де 1 означає повну ідентичність напрямків, а 0 — відсутність подібності. Формально ця метрика визначається наступною формулою [30]:

$$\cos(A, B) = \frac{A * B}{|\bar{A}| * |\bar{B}|}, \quad (2.1)$$

де $A, B$ — векторні представлення речень;

$A * B$ — скалярний добуток;

$|\bar{A}|, |\bar{B}|$ — евклідові норми векторів.

У реалізованому проєкті обчислення косинусної подібності дозволяє порівнювати вхідну новину з новинами з достовірних джерел. Якщо подібність перевищує заданий поріг, повідомлення вважається підтвердженим; якщо ні — бот попереджає користувача про потенційно фейковий зміст.

Такий підхід дозволяє ефективно застосовувати семантичне порівняння у реальному часі та забезпечує баланс між точністю й швидкодією, що особливо важливо для застосування в Telegram-ботах.

### 2.3.5 Система управління баз даних PostgreSQL

Для зберігання налаштувань, стану роботи Telegram-бота та інформації про користувачів була використана реляційна система

управління базами даних PostgreSQL. Вона забезпечує надійне збереження структурованих даних, обмеження цілісності, механізми безпеки, а також роботу з великими обсягами інформації у багатокористувацькому середовищі.

PostgreSQL є системою з відкритим кодом, яка відзначається високою стабільністю та відповідністю стандартам SQL. Вона підтримує складні запити, індексацію, тригери, процедури, зберігання JSON-структур, а також асинхронну обробку даних, що робить її придатною для інтеграції в системи реального часу. У контексті розробленого Telegram-бота база використовується для фіксації історії повідомлень, результатів перевірки новин, даних про довіру до джерел і параметрів обробки тексту [31].

### 2.3.6 Інструмент пошуку Google Search Console

У межах реалізації Telegram-бота для перевірки достовірності новин важливою складовою є отримання актуальної інформації з відкритих джерел. Для цього в проєкті було інтегровано Google Search Console, який надає можливість автоматизованого пошуку інформації в Інтернеті через надсилання HTTP-запитів. Результати повертаються у форматі JSON, що дозволяє легко обробляти їх у межах програмного коду.

Основне завдання API — знаходження релевантних джерел, які підтверджують або спростовують новину, надіслану користувачем. На основі вхідного тексту формуються пошукові запити, які надсилаються до Google Search Console. У відповідь надходять найбільш релевантні результати (5–10 посилань), з яких окремо витягується текстовий вміст сторінок. Далі ці тексти порівнюються з оригіналом за допомогою векторних представлень на основі моделі MPNet. Для оцінки схожості використовується метрика косинусної подібності.

Використання Google Search Console має низку переваг, зокрема:

висока якість результатів завдяки алгоритмам ранжування Google, гнучке налаштування запитів, можливість обмеження доменів, а також зручний формат відповіді. Завдяки цьому API забезпечує швидкий і точний доступ до релевантної інформації в режимі реального часу.

Разом з тим існують певні обмеження: кількість безкоштовних запитів є обмеженою, потрібна попередня конфігурація кастомного пошукового рушія (CSE), а також є часткові обмеження щодо доступу до контенту деяких соціальних мереж, зокрема Telegram і Facebook, через відсутність їхньої повної індексації.

Таким чином, Google Search API виконує ключову роль у системі автоматичної перевірки фактів, забезпечуючи пошук та збір текстової інформації з надійних джерел, що дозволяє підвищити точність класифікації новин та загальну ефективність Telegram-бота [32].

### 2.3.7 Використані бібліотеки

У процесі розробки Telegram-бота було використано низку бібліотек мови програмування Python, кожна з яких виконувала окрему функціональну роль у реалізації системи класифікації тексту та інтеграції з Telegram. Нижче подано опис основних бібліотек.

Pandas — це високорівнева бібліотека для аналізу та обробки даних, яка забезпечує структури даних (зокрема, DataFrame) та функціональні засоби для виконання різноманітних операцій із табличними, часовими та категоріальними даними. Вона активно застосовується в задачах попередньої обробки даних перед проведенням аналізу або навчанням моделей [33].

NumPy — бібліотека для ефективної роботи з багатовимірними масивами та матрицями. Крім основних операцій над масивами, надає широкий набір функцій для лінійної алгебри, статистики та трансформацій. NumPy є фундаментом для багатьох наукових бібліотек

Python, зокрема Pandas, SciPy та Scikit-learn [34].

NLTK (Natural Language Toolkit) — одна з найбільш розвинених бібліотек для обробки природної мови. Вона включає інструменти для токенізації, лематизації, POS-тегування, стемінгу, а також доступ до великої кількості корпусів і лексичних баз, зокрема WordNet. NLTK активно використовується в академічних дослідженнях та освітніх цілях, а також підходить для реалізації базових прототипів NLP-систем [35].

joblib — інструмент для серіалізації Python-об’єктів та кешування результатів обчислень. Найбільш доцільний для збереження об’ємних моделей машинного навчання, особливо створених з використанням Scikit-learn. Забезпечує ефективну роботу з великими масивами даних [36].

Scikit-learn — потужна бібліотека для реалізації алгоритмів машинного навчання, яка включає методи класифікації, регресії, кластеризації, зменшення розмірності, а також засоби для оцінки якості моделей. Scikit-learn підтримує численні моделі, зокрема логістичну регресію, дерева рішень, SVM та нейронні мережі. Завдяки простому API та активній спільноті вона стала стандартом де-факто для прототипування ML-рішень [37].

sentence-transformers — бібліотека, що базується на архітектурі трансформерів (BERT, RoBERTa, DistilBERT) і дозволяє генерувати векторні уявлення для речень. Векторизація виконується таким чином, що схожі за змістом речення мають близькі вектори. Ця властивість робить sentence-transformers незамінною у задачах семантичного пошуку, кластеризації та дуплікативного аналізу [38].

langdetect — легка бібліотека для визначення мови тексту. Підтримує понад 50 мов і не вимагає навчання, працюючи на основі евристичних правил. Застосовується як попередній етап для вибору відповідної мовної моделі [39].

psycopg2 — один із найнадійніших драйверів для взаємодії Python-програм із системою управління базами даних PostgreSQL. Забезпечує

підтримку транзакцій, підключення за допомогою курсорів, асинхронні запити та сумісність зі стандартом DB API 2.0 [40].

telebot (pyTelegramBotAPI) — обгортка над Telegram Bot API для Python, яка спрощує створення ботів, обробку повідомлень, реалізацію інлайн-кнопок, обробників команд та інших елементів взаємодії. Завдяки зручному інтерфейсу та активній підтримці, ця бібліотека широко застосовується у практичних проектах, зокрема для створення чат-ботів у сфері обслуговування, автоматизації, модерації та ін. [41].

dotenv (python-dotenv) — інструмент для зберігання та завантаження змінних середовища з файлу .env. Особливо корисна для збереження конфіденційної інформації, такої як токени доступу до API або облікові дані баз даних, окремо від основного коду [42].

json — стандартний модуль Python, що забезпечує кодування та декодування даних у форматі JSON. Є ключовим інструментом для обміну структурованими даними в клієнт-серверних додатках та збереження конфігурацій [43].

asyncio — модуль для створення асинхронних програм у Python. Дозволяє реалізовувати неблокуючі I/O-операції, що особливо корисно в системах реального часу або при роботі з великою кількістю одночасних підключень, як у випадку з Telegram-ботами [44].

Stanza — бібліотека для обробки природної мови, розроблена командою Стенфордського університету. Вона підтримує понад 60 мов і надає інструменти для токенізації, морфологічного аналізу, лематизації, частиномовного тегування та побудови синтаксичних дерев залежностей. Особливістю Stanza є використання нейронних мереж у всіх модулях, що забезпечує високу точність аналізу. У межах проєкту вона може використовуватися як альтернатива або доповнення до NLTK для поглибленого мовного аналізу [45].

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Архітектура проекту

Розроблена система Telegram-бота для оцінки достовірності новин реалізована з урахуванням принципів модульності та об’єктно-орієнтованого програмування. Структура проєкту передбачає чітке розділення логіки за окремими компонентами, кожен з яких виконує спеціалізовану функцію в загальному процесі аналізу тексту, обробки даних і взаємодії з користувачем.

Проєкт складається з дванадцяти основних елементів, розміщених у кореневій директорії. Папка venv є віртуальним середовищем Python, що забезпечує ізоляцію залежностей проєкту від глобальної системи. Це дозволяє уникати конфліктів версій бібліотек та полегшує розгортання застосунку. Папка messages містить шаблони відповідей різними мовами, що забезпечує підтримку багатомовного інтерфейсу. У директорії models зберігаються попередньо навчені класифікаційні моделі, створені на основі підготовлених корпусів із використанням бібліотеки Scikit-learn. Крім того, там знаходиться модуль model.py, який забезпечує можливість створення нових моделей.

Основна функціональна логіка проєкту реалізована у низці скриптів. Файл main.py відповідає за запуск Telegram-бота, ініціалізацію основних компонентів та забезпечення інтеграції всіх частин системи. Саме у цьому модулі відбувається створення об’єктів, таких як TextProcessor, MessageProcessor, Database, SearchEngine, Analyzer, та ініціалізація об’єкта TeleBot. Модуль analyzer.py виконує основні обчислення, пов’язані з оцінкою новин, зокрема містить реалізацію методів verify_news_by_text та verify_news_by_link. Модуль db.py забезпечує підключення та взаємодію з базою даних PostgreSQL. У message_processor.py реалізовано логіку

формування кінцевих повідомлень для користувача з урахуванням мови новини та результатів аналізу. Файл search_engine.py відповідає за інтеграцію з Google Search API, здійснює пошук подібних новин у відкритих джерелах, а також виконує фільтрацію за датою, регіоном та релевантністю. Модуль text_processor надає функції для обробки тексту — лемматизацію, видалення стоп-слів, парсинг HTML-сторінок та витяг значущих елементів тексту (наприклад, дат або ключових слів). Ці функції реалізовані з використанням багатомовної бібліотеки Stanza, яка підтримує понад 60 мов, включаючи українську й англійську, що дозволяє масштабувати систему на різні регіони.

Також до структури проєкту входять допоміжні файли: README.md містить загальну документацію щодо архітектури системи та інструкції користувача; requirements.txt містить перелік необхідних бібліотек для встановлення середовища; schema.sql описує структуру таблиць бази даних, що використовується в системі.

Архітектура Telegram-бота побудована відповідно до принципів модульності та розширюваності. Кожен модуль є ізольованою одиницею з чітко визначеною відповідальністю, що дозволяє легко змінювати, тестувати або повторно використовувати окремі частини проєкту. Завдяки цьому систему можна адаптувати для використання не лише в Telegram, а й в інших середовищах, таких як веб-інтерфейси або десктопні додатки. Зокрема, модуль Analyzer, що повертає уніфікований набір числових результатів аналізу, може бути повторно використаний без змін у різних прикладних сценаріях.

Подібна організація сприяє масштабованості, підтримуваності та прозорості логіки системи. Це особливо важливо у контексті розробки застосунків, пов’язаних із критичною інформаційною аналітикою, де є потреба у гнучкому налаштуванні, локалізації, а також швидкому розгортанні оновлень або нових функцій.

## 3.2 Архітектура моделі аналізу провокативності тексту

У системі аналізу новин використовується модель машинного навчання на основі логістичної регресії, реалізована за допомогою бібліотеки Scikit-learn. Цей метод дозволяє класифікувати текстові повідомлення за ознакою провокативності, тобто визначати, чи має текст потенційно маніпулятивний або емоційно навантажений характер.

Під час підготовки моделі використовується розмічений набір даних з відкритих джерел, де кожне повідомлення віднесено до одного з двох класів: провокативне або нейтральне. Перед навчанням дані проходять стандартну обробку: видалення стоп-слів, лемматизацію та перетворення у векторну форму за допомогою TF-IDF. Цей підхід дозволяє зберегти важливу інформацію про зміст тексту та зменшити вплив часто вживаних, але неінформативних слів.

Логістична регресія обчислює ймовірність того, що текст належить до класу «провокативний», за формулою:

$$P(y = 1|x) = \frac{1}{1+ e^{-(wx+b)}}, \quad (3.1)$$

де x – вектор ознак тексту (TF-IDF);

w – ваговий вектор моделі;

b – зміщення;

$P(y = 1|x)$ – ймовірність того, що текст є провокативним**.**

Модель навчається шляхом мінімізації логарифмічної функції втрат (log-loss), яка порівнює прогнозовані ймовірності з реальними мітками. Під час навчання Scikit-learn автоматично виконує оптимізацію вагових коефіцієнтів, а також надає засоби для перевірки якості моделі. Приклад навчання моделі зображено на лістингу 3.1.

Лістинг 3.1 – Розділення даних та навчання моделі

```
X_train,X_test,y_train,y_test=train_test_split(df["text"],
df["label"], test_size=0.2, random_state=42)
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
model = LogisticRegression()
model.fit(X_train_tfidf, y_train)
```

Після навчання модель зберігається за допомогою joblib, що дозволяє швидко її завантажувати у робочому середовищі. Приклад збереження моделі для подальшого використання зображно на лістингу 3.2.

Лістинг 3.2 – Збереження моделі для подальшого використання

```
joblib.dump(model, "models/fake_news_model_lang.pkl")
joblib.dump(vectorizer,"models/tfidf_vectorizer_labg.pkl")
```

У Telegram-боті модель інтегрована в модуль Analyzer. Для кожної мови, яку підтримує бот, створюється окрема модель, навчена на відповідному наборі даних. Приклад використання моделі для аналізу провокативності тексту наведено на лістингу 3.3.

Лістинг 3.3 – Практичне використання моделі для визначення провокативності тексту.

```
def __get_provoking_rate(self, text, lang):
   model_data=self.language_models.get(lang,
   self.language_models["en"])
   vectorizer = model_data["vectorizer"]
   model = model_data["model"]
```

```
    text_vectorized = vectorizer.transform([text])
    prediction = model.predict_proba(text_vectorized)[0][1]
    return prediction
```

Таким чином, поєднання логістичної регресії та TF-IDF забезпечує простий, стабільний і ефективний підхід до первинного аналізу текстів, що дозволяє швидко оцінити рівень провокативності в новинному контенті.

## 3.3 Архітектура бази даних

У реалізованому проєкті база даних PostgreSQL представлена п’ятьма незалежними таблицями, призначеними для зберігання інформації про джерела різного типу: надійні (reliable_sources), ненадійні (unreliable_sources), сатиричні (satirical_sources), інші (other_sources), а також про всі новини, які вже проходили аналіз (news_analysis).

Детальній опис таблиць бази даних наведено в таблиці 3.1.

Таблиця 3.1 – Детальний опис таблиць бази даних

| Найменування таблиці | Опис | Найменування атрибуту | Тип даних |
|---|---|---|---|
| Надійні джерела | Зберігає інформацію про надійні джерела | ID-джерела | INTEGER |
| | | Ім’я джерела | STRING |
| Ненадійні джерела | Зберігає інформацію про ненадійні джерела | ID-джерела | INTEGER |
| | | Ім’я джерела | STRING |

Продовження таблиці 3.1 - Детальний опис таблиць бази даних

| Найменування таблиці | Опис | Найменування атрибуту | Тип даних |
|---|---|---|---|
| Сатиричні джерела | Зберігає інформаціє про сатиричні джерела | ID-джерела | INTEGER |
| | | Ім’я джерела | STRING |
| Інші джерела | Зберігає інформацію про всі інші джерела з відносною надійністю | ID-джерела | INTEGER |
| | | Ім’я джерела | STRING |
| | | Лічильник правдивих новин | INTEGER |
| | | Лічильник фейкових новин | INTEGER |
| | | Показник надійності | FLOAT |
| Оброблені новини | Зберігає інформацію про оброблені новини | Посилання на новину | STRING |
| | | Показник надійності новини | FLOAT |
| | | Показник надійності джерела | FLOAT |
| | | Показник провокативності інформації | FLOAT |

Таблиці джерел зберігають базову інформацію, таку як унікальний ідентифікатор та назву ресурсу. У випадку з таблицею інших джерел додатково зберігаються лічильники правдивих та фейкових новин, а також обчислений показник надійності.

Таблиця оброблених новин містить інформацію про новини, що вже були проаналізовані системою, включаючи посилання на джерело, оцінку достовірності новини, рівень провокативності та оцінку надійності

ресурсу, з якого вона походить. Це дозволяє уникати повторного аналізу вже відомих новин, що підвищує продуктивність системи.

Модуль Database, який відповідає за взаємодію з базою даних, реалізує низку функцій для зчитування, запису та перевірки даних. Основні функції включають підключення до бази (connect_db), перевірку джерела в різних таблицях, збереження результатів аналізу новин (add_news_analysis) та оновлення таблиці інших джерел (add_to_other_sources). Більшість функцій перевірки джерел побудовані на базі універсальної функції check_source_in_table. Опис функцій модуля Database наведено в таблиці 3.2

Таблиця 3.2 – Опис та перелік функцій модуля Database

| Найменування функції | Призначення |
|---|---|
| connect_db | Підключення до бази даних |
| check_in_relialbe_sources | Пошук джерела у таблиці надійних джерел |
| check_in_unrelialbe_sources | Пошук джерела у таблиці ненадійних джерел |
| check_in_satirical_sources | Пошук джерела у таблиці сатиричних джерел |
| check_in_other_sources | Пошук джерела у таблиці інших джерел |
| get_news_analysis | Пошук новини у таблиці оброблених новин |
| get_reliability_score | Отримання показника надійності новини |
| add_to_other_sources | Внесення нового джерела до таблиці інших джерел |
| add_news_analysis | Внесення аналізу новини до таблиці оброблених новин |

Така структура є масштабованою та придатною до подальшого розширення. Водночас вона має певні обмеження. Одним з недоліків є відсутність таблиць для зберігання інформації про користувачів і чати. Це ускладнює контроль над активністю в системі, що потенційно може

призвести до зловживання функціоналом, наприклад, шляхом надмірного навантаження на API або надсилання великої кількості запитів з одного облікового запису. Запровадження механізмів збереження даних про користувачів і групи дало б змогу відслідковувати подібну активність та реагувати на неї — наприклад, шляхом обмеження функціоналу для порушників.

У перспективі архітектура бази даних може бути доповнена додатковими сутностями, що дозволить покращити гнучкість системи, розширити функції аналітики та посилити контроль за поведінкою користувачів.

## 3.4 Аналіз отриманіх результатів

Одним із ключових параметрів функціонування чат-бота є час, необхідний для аналізу новини. Цей показник безпосередньо впливає на задоволеність користувачів системою. Для оцінювання цього параметра в системі використовується функція timer, яка дозволяє вимірювати час виконання окремих функцій у коді. Приклад реалізації функції timer наведено в лістингу 3.4.

Лістинг 3.4 – Реалізація функції timer для визначення продуктивності системи.

```
def timer(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f"Функціяя   {func.__name__}   виконувалася
{end_time - start_time:.6f} секунд")
        return result
```

```
    return wrapper
```

Під час проєктування системи функція timer використовувалася для порівняння методів отримання новин з мережі та оцінки швидкості роботи моделей семантичного аналізу тексту.

У середньому аналіз однієї новини займає близько 25 секунд, що є задовільним показником продуктивності системи. Якщо новина вже була проаналізована раніше, результат виводиться за частки секунди.

Точність визначення провокативності новини є суб’єктивним параметром, оскільки неможливо надати йому чітке числове значення.

Проте цей показник можна покращити шляхом збільшення обсягу вибірки новин, використаних для навчання моделі, а також шляхом подальшого навчання на основі новин, що надсилають користувачі.

Точність аналізу надійності інформації може бути підвищена завдяки застосуванню більш ефективних моделей семантичного аналізу та вдосконаленню процесу попередньої обробки тексту.

# ВИСНОВКИ

У межах кваліфікаційної роботи було реалізовано інтелектуальну систему у вигляді Telegram-бота, що виконує автоматизовану перевірку достовірності новин з використанням методів обробки природної мови, машинного навчання та семантичного аналізу. Розроблена система забезпечує можливість аналізу тексту новини, як введеного користувачем, так і отриманого через інтеграцію з Google Search Console, що дозволяє виявляти інформаційні збіги з перевіреними джерелами.

У процесі реалізації було поєднано логістичну регресію для оцінки провокативності тексту та модель MPNet для векторного представлення речень і обчислення їх семантичної подібності. Попередня обробка текстів здійснювалася за допомогою бібліотеки Stanza, що дозволило забезпечити підтримку декількох мов і адаптувати систему до використання в різних інформаційних середовищах. Зберігання результатів та технічної інформації реалізовано через базу даних PostgreSQL, що забезпечує структуровану й розширювану модель даних.

Система має модульну архітектуру, яка дозволяє легко адаптувати її до нових завдань, підключати інші джерела або реалізовувати додаткову логіку без зміни основного функціоналу. Telegram-бот забезпечує швидкий зворотний зв’язок, що дозволяє інтегрувати систему в повсякденне користування без потреби у складних інтерфейсах.

У результаті було створено ефективне рішення, що об’єднує сучасні технології комп’ютерної лінгвістики та машинного навчання для вирішення актуальної задачі — перевірки достовірності інформації в цифровому середовищі. Перспективи розвитку проєкту передбачають розширення мовної підтримки, удосконалення алгоритмів класифікації на основі більших датасетів, а також інтеграцію з іншими платформами задля розширення функціональності та підвищення точності аналізу.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. StopFake. Проєкт із перевірки фактів і викриття російської дезінформації.
URL: https://www.stopfake.org/uk/ (дата звернення: 10.06.2025).

2. По той бік новин. Платформа з аналізу фейків, що поширюються в українському інформаційному просторі.
URL: https://behindthenews.org/ (дата звернення: 10.06.2025).

3. МедіаЧек — ініціатива Інституту масової інформації з моніторингу етичності та достовірності журналістських матеріалів.
URL: https://imi.org.ua/mediacheck (дата звернення: 10.06.2025).

4. Texty.org.ua. Лабораторія фейків: аналітика про маніпуляції в медіа та використання алгоритмів для їх виявлення.
URL: https://texty.org.ua/tag/laboratoriya-fejkiv/ (дата звернення: 10.06.2025).

5. Disinformation attack. Wikipedia. URL: https://en.wikipedia.org/wiki/Disinformation_attack (дата звернення: 30.04.2025).

6. Інститут масової інформації. Telegram: джерело новин чи майданчик для маніпуляцій? URL: https://imi.org.ua/monitorings/telegram-dzherelo-novyn-chy-majdanchyk-dlya-manipulyatsij-i52898 (дата звернення: 10.06.2025).

7. Texty.org.ua. Перевірка фактів і штучний інтелект: як автоматизовані системи допомагають боротися з фейками.
URL: https://texty.org.ua/articles/110051/yak-shtucnyj-intelekt-dopomagaye-borotysya-z-dezinformaciyeyu/ (дата звернення: 10.06.2025).

8. Telegram is the main social network through which Ukrainians receive news.

URL:https://censor.net/en/news/3510806/telegram_is_the_main_social_network_through_which_ukrainians_receive_news_study (дата звернення:

03.05.2025).

9. Fact-checking automation: NLP techniques for detecting misinformation. URL: https://towardsdatascience.com/fact-checking-automation-nlp-techniques-for-detecting-misinformation-d37f8b850203 (дата звернення: 03.05.2025).

10. Emotion detection for misinformation: A review. URL: https://www.sciencedirect.com/science/article/pii/S1566253524000782 (дата звернення: 03.05.2025).

11. Bag of Words Model in NLP. GeeksforGeeks. URL: https://www.geeksforgeeks.org/bag-of-words-bow-model-in-nlp/ (дата звернення: 03.05.2025).

12. TF-IDF Explained. Towards Data Science. URL: https://towardsdatascience.com/understanding-tf-idf-5aaf0c6383f3 (дата звернення: 03.05.2025).

13. Text Classification Using TF-IDF and Logistic Regression. Analytics Vidhya. URL: https://www.analyticsvidhya.com/blog/2021/06/text-classification-using-tf-idf-in-python/ (дата звернення: 10.05.2025).

14. Text Classification using Logistic Regression. GeeksforGeeks. URL: https://www.geeksforgeeks.org/text-classification-using-logistic-regression/ (дата звернення: 10.05.2025).

15. Comparative Study of Text Classification Algorithms. Journal of Artificial Intelligence Research & Advances. URL: https://www.jairajournal.com/volume6-issue2/text-classification (дата звернення: 10.05.2025).

16. Understanding LSTM Networks. Colah’s Blog. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата звернення: 10.05.2025).

17. Convolutional Neural Networks for Sentence Classification. EMNLP. URL: https://www.aclweb.org/anthology/D14-1181/ (дата звернення: 11.05.2025).

18. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv. URL: https://arxiv.org/abs/1810.04805 (дата звернення: 11.05.2025).

19. Usage Limits | Search Console API - Google for Developers. URL: https://developers.google.com/webmaster-tools/limits (дата звернення: 11.05.2025).

20. TF-IDF (Term Frequency — Inverse Document Frequency) — статистичний показник, що використовується для оцінки важливості слів у контексті документа. Вікіпедія. URL: https://uk.wikipedia.org/wiki/TF-IDF (дата звернення: 12.05.2025).

21. Text Classification using Logistic Regression. GeeksforGeeks. URL: https://www.geeksforgeeks.org/text-classification-using-logistic-regression/ (дата звернення: 12.05.2025).

22. Guido van Rossum: The creator of the Python programming language. URL: https://medium.com/thedeephub/guido-van-rossum-the-creator-of-the-python-programming-language-b85a7db3e8a0 (дата звернення: 12.05.2025).

23. Indentation in Python. URL: https://www.geeksforgeeks.org/indentation-in-python/ (дата звернення: 14.05.2025).

24. Programming Paradigms in Python. URL: https://www.geeksforgeeks.org/programming-paradigms-in-python/ (дата звернення: 14.05.2025).

25. The Python Standard Library — Python 3.13.3 documentation. URL: https://docs.python.org/3/library/index.html (дата звернення: 14.05.2025).

26. Octoverse: AI leads Python to top language as the number of global ... URL: https://github.blog/news-insights/octoverse/octoverse-2024/ (дата звернення: 14.05.2025).

27. Visual Studio Code — безкоштовне середовище розробки від Microsoft. URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 14.05.2025).

28. Python OOPs Concepts. URL: https://www.geeksforgeeks.org/python-oops-concepts/ (дата звернення: 20.05.2025).

29. MPNet: Masked and Permuted Pre-training for Language Understanding. URL: https://arxiv.org/abs/2004.09297 (дата звернення: 28.04.2025).

30. Cosine similarity. URL: https://en.wikipedia.org/wiki/Cosine_similarity (дата звернення: 28.04.2025).

31. PostgreSQL — об'єктно-реляційна система керування базами даних з відкритим кодом. URL: https://uk.wikipedia.org/wiki/PostgreSQL (дата звернення: 20.05.2025).

32. Custom Search JSON API | Programmable Search Engine. URL: https://developers.google.com/custom-search/v1/overview (дата звернення: 20.05.2025).

33. Pandas — Python Data Analysis Library. URL: https://pandas.pydata.org/ (дата звернення: 20.05.2025).

34. NumPy — The fundamental package for scientific computing with Python. URL: https://numpy.org/ (дата звернення: 20.05.2025).

35. NLTK — Natural Language Toolkit. URL: https://www.nltk.org/ (дата звернення: 20.05.2025).

36. Joblib: running Python functions as pipeline jobs. URL: https://joblib.readthedocs.io/ (дата звернення: 20.05.2025).

37. Scikit-learn: Machine Learning in Python. URL: https://scikit-learn.org/ (дата звернення: 20.05.2025).

38. SentenceTransformers Documentation. URL: https://www.sbert.net/ (дата звернення: 20.05.2025).

39. langdetect — PyPI. URL: https://pypi.org/project/langdetect/ (дата звернення: 20.05.2025).

40. psycopg2 — PostgreSQL adapter for Python. URL: https://pypi.org/project/psycopg2/ (дата звернення: 20.05.2025).

41. pyTelegramBotAPI — Python implementation for the Telegram Bot

API. URL: https://pypi.org/project/pyTelegramBotAPI/ (дата звернення: 20.05.2025).

42. python-dotenv — Reads key-value pairs from a .env file and can set them as environment variables. URL: https://pypi.org/project/python-dotenv/ (дата звернення: 20.05.2025).

43. json — JSON encoder and decoder — Python documentation. URL: https://docs.python.org/3/library/json.html (дата звернення: 20.05.2025).

44. asyncio — Asynchronous I/O — Python documentation. URL: https://docs.python.org/3/library/asyncio.html (дата звернення: 20.05.2025).

45. Stanza: A Python NLP Library for Many Human Languages. URL: https://github.com/stanfordnlp/stanza (дата звернення: 20.05.2025).