Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Веб-застосунок для інтелектуального аналізу достовірності інформації, що передається в ЗМІ
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-4
Павло Кочаров
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ас. Микола Черненко
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Кочарову Павлу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Веб-застосунок для інтелектуального аналізу достовірності інформації, що передається в ЗМІ

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

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

3. Вихідні дані до роботи Відкриті датасети для навчання та тестування моделей Natural Language Inference (наприклад, FEVER, MultiNLI, ANLI); корпуси текстових новин для аналізу упередженості та емоційного забарвлення; наукові статті та відкриті дослідження в галузі фактчекінгу, NLP та NLI; інструменти для обробки природної мови (HuggingFace Transformers, spaCy, NLTK)

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі

2) Аналіз існуючих рішень

3) Проектування застосунку та компоненти ШІ

4) Реалізація застосунку

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	20.05.2025	виконано
3	Огляд аналогів	20.05.2025	виконано
4	Проєктування системи	21.05.2025	виконано
5	Моделювання та навчання моделі	23.05.2025	виконано
6	Написання пояснювальної записки	25.05.2025	виконано
7	Перевірка на академічний плагіат	26.05.2025	виконано
8	Нормоконтроль	06.06.2025	виконано
9	Підготовка презентації та доповіді	10.06.2025	виконано
10	Попередній захист	13.06.2025	виконано
11	Рецензування	15.06.2025	виконано
12	Захист перед ЕК	25.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Микола Черненко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 69 с., 16 рис., 3 табл., 3 дод., 28 джерел.

АНАЛІЗ ПРИРОДНОЇ МОВИ, ДАТАСЕТ, ТОКЕНІЗАЦІЯ, ФАКТ-ЧЕКІНГ, ФЕЙК, NLI, NLP.

Об'єкт дослідження – аналіз достовірності фактів в засобах масової інформації, що передаються мережею Internet.

Предмет дослідження – алгоритми отримання та обробки текстів, що виступають доказами або запереченнями твердження, що має бути перевіреним.

Мета роботи – розробити застосунок, який аналізуватиме текстові новини, оцінюватиме їхню емоційну забарвленість, виявлятиме упередженість і здійснюватиме перевірку фактів за допомогою методів обробки природної мови. А також дослідити моделі пошуку уривків текстів та методів відбору речень.

Методи дослідження – аналіз наукової літератури та публічних досліджень з питань фактчекінгу та інструментів для розробки відповідної системи.

Новизною роботи є покращення аналізу відношення тверджень при вирішенні задачі NLI (Natural Language Inference) в умовах наявності відволікаючої інформації в контексті тверджень, що перевіряються на істинність.

В результаті виконання кваліфікаційної роботи було досліджено сучасні сервіси фактчекінгу та технології необхідні для розробки власного застосунку, було розроблено програму, що дозволяє перевірити достовірність інформації.

ABSTRACT

Bachelor’s thesis contains: 69 pp., 16 fig., 3 tabl., 3 ann., 28 references.

DATASET, FACT-CHECKING, FAKE, NATURAL LANGUAGE ANALYSIS, NLI, NLP, TOKENIZATION.

An object of the research is an analysis of the facts credibility in mass media transmitted through the Internet.

A subject of the research is algorithms usage for retrieving and processing texts that serve as evidence or refutations of a statement that needs to be verified.

A goal of the work is to develop a web application that analyzes textual news, evaluates its emotional tone, detects bias, and performs fact-checking using natural language processing methods. Additionally, to explore models for retrieving relevant text fragments and methods for sentence selection.

Research methods consist of analysis of scientific literature and public studies on fact-checking and tools for developing such applications.

The novelty of the work is in improving the analysis of the relationship between statements using NLI (Natural Language Inference), especially under the conditions of distracting information in the context of the statements being verified for truthfulness.

As a result of the thesis work, modern fact-checking services and technologies required for the development of the application have been researched and used. The web-application has been designed and developed. It allows us to verify the credibility of information.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЗМІ – засоби масової інформації;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – програмний інтерфейс;

BERT – Bidirectional Encoder Representations from Transformers – метод машинного навчання на основі перетворювача для попередньої підготовки до обробки природної мови;

NLI – Natural Language Inference – завдання автоматичного визначення логічного зв'язку між текстами;

NLP – Natural Language Processing – загальний напрямок штучного інтелекту та математичної лінгвістики який вивчає проблеми комп'ютерного аналізу та синтезу текстів природними мовами.

ВСТУП

В нинішньому світі інформація стала потужною зброєю. Фейки ж тепер є нічим іншим як інноваційним інструментом управління інформаційними процесами у засобах масової інформації (ЗМІ), що використовуються у сучасних гібридних війнах. Специфіка цих інструментів полягає у комплексній маніпуляції громадською думкою, що відбувається на основі психологічного тиску, використання привабливого контенту та ефективних методів поширення інформації.

В умовах тривалого збройного конфлікту та гібридної агресії проти України інформаційна безпека набуває надзвичайної ваги. Маніпулювання фактами, поширення дезінформації та проведення інформаційно-психологічних операцій (ІПсО) стали невід'ємною частиною сучасного протистояння, спрямованого на дестабілізацію суспільства, підрив довіри до державних інституцій та дезорієнтацію громадської думки. Особливої актуальності це питання набуває у контексті масового споживання інформації через соціальні мережі та онлайн-ЗМІ, які є вразливими до вкидання неправдивих повідомлень та маніпулятивних наративів. Згідно з дослідженнями Центру стратегічних комунікацій та інформаційної безпеки, за останні роки в Україні спостерігається зростання кількості виявлених інформаційних атак, що здійснюються з використанням технологій фальсифікації новин та емоційного впливу на аудиторію [1]. Це вимагає впровадження нових технологічних рішень для виявлення та нейтралізації подібних загроз у медіа-просторі.

Ретельне дослідження та практична розробка алгоритму та системи аналізу достовірності інформації, виявлення фейків, на сьогодні є актуальним завданням боротьби з великою кількістю недостовірної інформації в медіа просторі. В умовах гібридних протистоянь ця технологія є однією з ключових і затребуваною на теперішній момент.

Дана кваліфікаційна робота присвячена дослідженню можливостей використання сучасних технологій NLP для аналізу новинних текстів. Основна увага зосереджена на виявленні емоційного забарвлення, упередженості та відповідності викладеної інформації фактам. У процесі дослідження використовуються інструменти, що дозволяють здійснювати не лише лексичний та семантичний аналіз тексту, а й перевірку ключових тверджень за допомогою відкритих джерел.

Метою даної роботи є розробка веб-застосунку, що здатен аналізувати новинний контент, визначати можливі ознаки маніпуляції чи неправдивої інформації, а також надавати користувачеві пояснення щодо виявлених проблемних аспектів. Для досягнення поставленої мети використовуються бібліотеки обробки тексту, зокрема spaCy та NLTK, а також моделі трансформерів, такі як BERT.

Особливістю запропонованого рішення є комплексний підхід до аналізу тексту. Застосунок не обмежується пошуком фактологічних помилок, а також оцінює емоційне забарвлення тексту, виявляє потенційно маніпулятивні висловлювання та зіставляє ключові твердження з перевіреними джерелами. Завдяки використанню моделей типу BERT забезпечується контекстуальний аналіз та підтримка багатомовності. У перспективі систему можна масштабувати для аналізу інших типів контенту, зокрема зображень та відео.

Розроблений інструмент може бути корисним у різних галузях: у сфері освіти – як засіб формування критичного мислення; у журналістиці – для перевірки достовірності матеріалів; а також для широкого кола користувачів – як спосіб убезпечити себе від впливу недостовірної інформації. Застосування таких технологій сприятиме створенню більш надійного та безпечного інформаційного простору. У майбутньому, з подальшим розвитком методів обробки природної мови, можливості подібних систем значно розширяться, що сприятиме більш ефективній протидії дезінформації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Обґрунтування доцільності розробки

З появою соціальних мереж і багатьох окремих джерел новин в Інтернеті поширення дезінформації стало серйозною проблемою з потенційно шкідливими соціальними наслідками. Фейкові новини можуть маніпулювати громадською думкою, створювати конфлікти та викликати необгрунтований страх і підозри. Наприклад, 64% американців визнали, що фейкові новини збивають їх з пантелику (рисунок 1.1). Це почуття виникало у респондентів незалежно від їх рівня освіти, доходів та більшості інших демографічних характеристик [2].

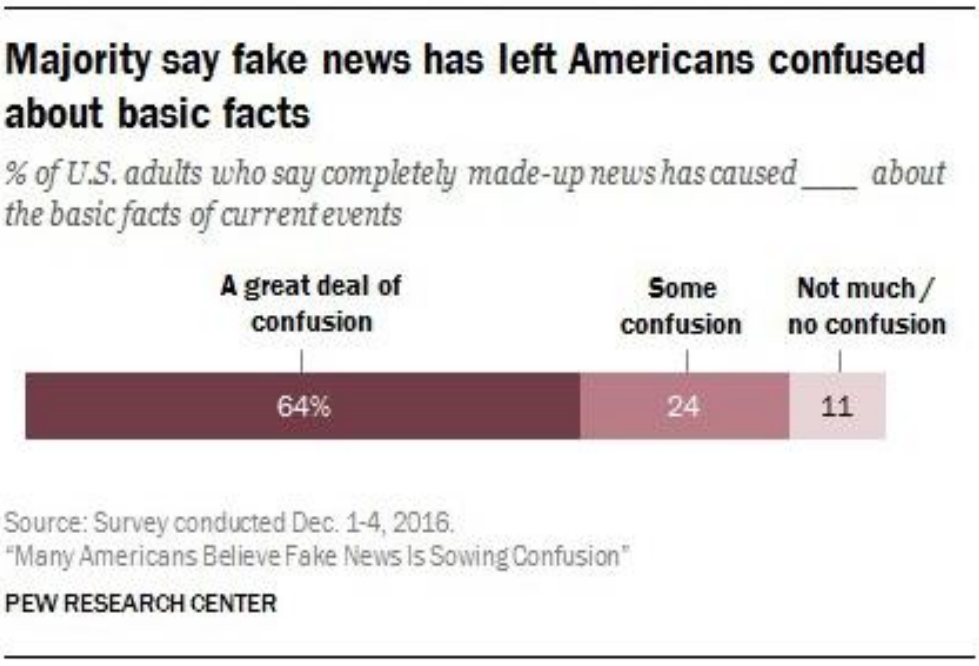

Рисунок 1.1 – Pew Research Center: опитування американців щодо впливу фейкових новин

А 94% українців вважають дезінформацію однією з найбільших загроз майбутнього (рисунок 1.2) [3].

Рисунок 1.2 – Brand Ukraine: опитування щодо дезінформації

Величезна кількість неперевіреного онлайн-контенту призвела до створення зовнішніх пост-перевірочних організацій, таких як PolitiFact, FactCheck.org, Snopes, тощо, зі спеціальними ресурсами для перевірки інформації розміщеної в інтернеті. Однак ручна перевірка фактів займає багато часу, і її важко проводити у великих масштабах. Можливість автоматичної перевірки фактів має вирішальне значення для мінімізації негативного соціального впливу.

Основними викликами у створенні таких систем є:

– обмежена доступність великих розмічених наборів даних для навчання моделей;

– необхідність інтеграції з різноманітними джерелами даних (наприклад, NewsAPI, Wikipedia API);

– забезпечення точності та зрозумілості результатів для кінцевих користувачів.

Запропонований веб-застосунок спрямований на вирішення цих викликів шляхом створення зручного інтерфейсу, який поєднує автоматизовану перевірку фактів, аналіз тональності та пояснення результатів. Це дозволить користувачам не лише отримувати оцінку достовірності, а й розуміти, на основі яких даних було зроблено висновки.

1.2 Опис предметної галузі

Протягом останніх кількох років фейкові новини стають все більш поширеними. Фальшиві новинні статті, як правило, надходять із сатиричних новинних веб-сайтів або окремих веб-сайтів із стимулом поширювати неправдиву інформацію, або як приманку для кліків, або для досягнення мети [4].

Оскільки ці статті зазвичай спрямовані на навмисне просування упередженої чи невірної інформації, їх важко виявити. Визначаючи джерело інформації, необхідно звернути увагу на багато атрибутів, включаючи, але не обмежуючись вмістом електронної пошти та взаємодії в соціальних мережах. Зокрема, у фейкових новинах ця мова зазвичай є більш підбурювальною, ніж у справжніх статтях, частково тому, що мета полягає в тому, щоб заплутати користувача. Більше того, методи моделювання, такі як кодування n-грамів і сумка слів, слугували іншими лінгвістичними техніками для визначення легітимності новин.

Крім того, дослідники визначили, що візуальні підказки також відіграють важливу роль у класифікації статті. Зокрема, деякі функції можуть бути розроблені, щоб оцінити, чи зображення було легітимним, і надати нам більше ясності щодо новин. Існує також багато особливостей соціального контексту, які можуть зіграти свою роль, а також модель поширення новин. Веб-сайти, такі як «Snopes», намагаються виявити цю інформацію вручну, тоді як деякі університети намагаються побудувати математичні моделі для цього самостійно.

Адаптація соціальних медіа як законної та широко використовуваної платформи викликала широке занепокоєння щодо фейкових новин у цій сфері. Поширення фейкових новин через соціальні медіа-платформи, такі як Facebook, Twitter та Instagram, створює можливість надзвичайно негативного впливу на суспільство, тому нові галузі досліджень щодо виявлення фейкових новин у соціальних мережах набирають обертів [5].

Однак виявлення фейкових новин у соціальних мережах створює проблеми, які роблять попередні методи аналізу даних і виявлення нерелевантними [6]. Таким чином, дослідники закликають до додаткової роботи щодо фейкових новин, які характеризуються проти психології та соціальних теорій, а також адаптації існуючих алгоритмів інтелектуального аналізу даних для застосування в соціальних мережах [7]. Крім того, було опубліковано кілька наукових статей, які закликають продовжити пошук автоматичних способів фільтрації фейкових новин із хронології соціальних мереж [8].

Після президентських виборів у Сполучених Штатах у 2016 році фейкові новини були популярною темою для обговорення президента Трампа та ЗМІ. Реальність фейкових новин стала всюдисущою, і багато досліджень було спрямовано на розуміння, ідентифікацію та боротьбу з фейковими новинами. Також низка дослідників почали з використання фейкових новин для впливу на президентську кампанію 2016 року [9]. Результати дослідження можна наглядно бачити на рисунку 1.3.

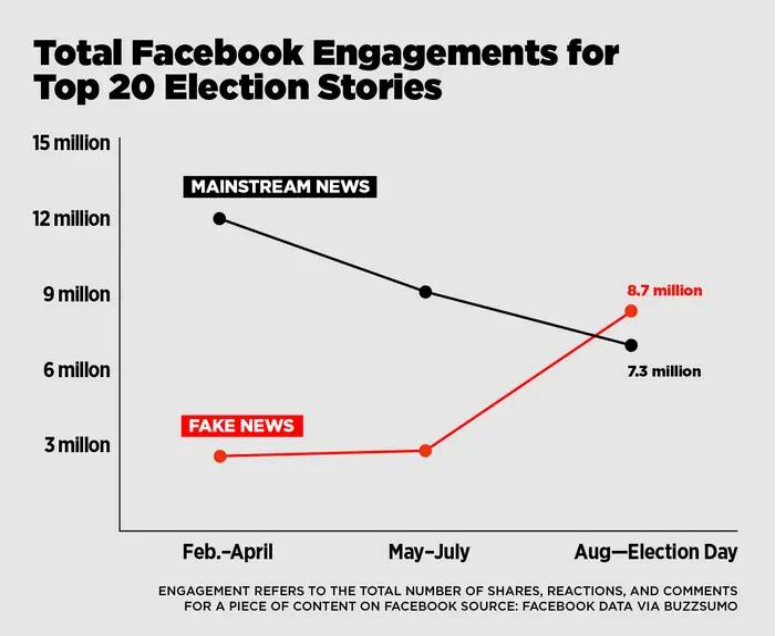

Рисунок 1.3 – Кількість фейкових новин перед виборами 2016 року в США перевищила реальні

Одне дослідження виявило докази того, що фейкові новини на підтримку Трампа вибірково націлювалися на консерваторів і прихильників Трампа в 2016 році.

Дослідники виявили, що сайти соціальних мереж, зокрема Facebook, є потужними платформами для розповсюдження фейкових новин цільовим групам для звернення до них [10].

Крім того, дослідники зі Стенфорда, Нью-Йоркського університету та NBER знайшли докази того, що протягом 2016 року Facebook і Twitter було високо залучено до поширення фейкових новин [11].

Автоматизована перевірка фактів – це складне завдання, яке включає отримання доказів з подальшим їх обґрунтуванням.

Для пошуку відповідних доказів із тіл документів існуючі системи зазвичай використовують традиційний розріджений пошук, який може показувати не найкращу ефективність, особливо коли відповідні уривки містять кілька слів, що збігаються з твердженнями, які потрібно перевірити.

Отже засоби перевірки фактів (фактчекінг) стають дедалі важливішими у контексті інформаційного суспільства, де користувачі щодня стикаються з великою кількістю новин.

Автоматизовані системи фактчекінгу дозволяють оперативно перевіряти твердження, порівнюючи їх з достовірними джерелами, що сприяє підвищенню медіаграмотності.

Такі системи використовують сучасні технології NLP, такі як моделі BERT, RoBERTa, а також бібліотеки spaCy і NLTK , для аналізу тексту, витягнення ключових тверджень і оцінки їх відповідності фактам.

Крім того, аналіз емоційного тону та виявлення упередженості допомагають користувачам краще зрозуміти контекст і можливі маніпулятивні елементи в новинах.

1.3 Цілі та задачі розробки

Цілями даної роботи є:

– розробка програмного забезпечення для перевірки достовірності інформації з використанням передових технологій;

– розробка веб-базованого застосунку для доступу користувачів до функціоналу перевірки достовірності інформації;

– забезпечення можливості отримання доказів результатів аналізу інформації.

Для досягнення поставлених цілей необхідно вирішити наступні задачі:

– розробити програму обробки тексту;

– створити систему збору та аналізу пов’язаної інформації;

– створити веб додаток з API для забезпечення доступу до ядра застосунку;

– створити веб інтерфейс.

1.4 Висновки за розділом

У першому розділі кваліфікаційної роботи було проаналізовано предметну область, що стосується проблеми поширення фейкових новин та дезінформації в сучасному інформаційному просторі.

Наведено статистичні дані, що свідчать про глобальний масштаб і серйозність цієї проблеми, зокрема – результати соціологічних досліджень у США та Україні, які підтверджують високий рівень занепокоєння громадян впливом неправдивої інформації.

У результаті сформульовано цілі та завдання роботи, які передбачають створення інтелектуального веб-застосунку для перевірки достовірності новинної інформації.

Застосунок має інтегрувати найновіші досягнення в галузі NLP (зокрема моделі BERT, RoBERTa), забезпечувати збір релевантної інформації з авторитетних джерел, здійснювати оцінку тексту на наявність упередженості та емоційних маніпуляцій.

А також надавати пояснення користувачу щодо того, як і на основі яких даних було ухвалено рішення про достовірність. Для цього необхідно подивитися на досвід інших розробників, а також систематизувати знання та засоби реалізації застосунку.

Таким чином, перший розділ заклав теоретичне підґрунтя для подальшого практичного впровадження автоматизованої системи перевірки фактів та аналізу новин, яка буде спрямована на зменшення впливу дезінформації та формування більш відповідального інформаційного простору.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Критерії для аналізу аналогів

На ринку існують системи-аналоги, що вирішують задачі проблематики, що розглядається в роботі. Для формування вимог до застосунку, вибору оптимального підходу до розробки та врахування досвіду інших систем, потрібно проаналізувати зразки подібних систем.

Детальний аналіз допоможе виявити переваги та недоліки таких систем. На його основі можна сформувати вимоги до функціоналу застосунку, що розробляється.

З критеріїв аналізу варто виділити наступні:

– точність аналізу інформації;

– багатофункціональність, яка передбачає можливість аналізу декількох додаткових параметрів одночасно, тобто не лише фактчекинг, а й аналіз емоційної забарвленості та упередженості;

– простота використання для середньостатистичного користувача.

The Factual (рисунок 2.1) – сервіс, що дозволяє отримати доступ до новин проаналізованих нейронною мережею розробника. Завдяки подібному аналізу, новини та статті отримують рейтинг довіри та фільтруються.

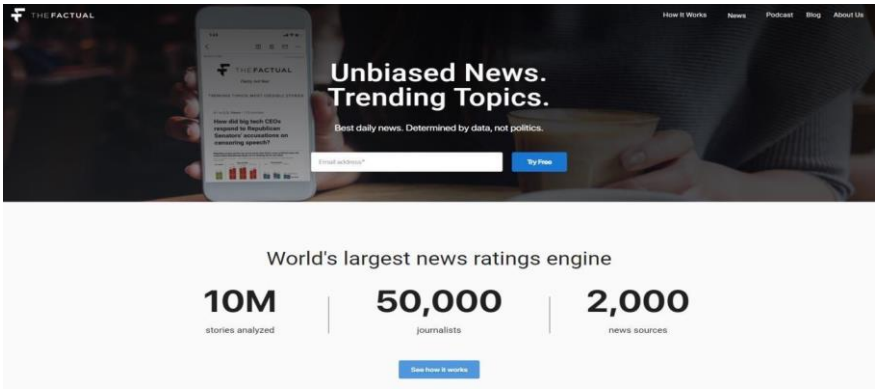

Рисунок 2.1 – Сервіс The Factual

За своєю суттю The Factual – це алгоритм, який щоденно оцінює 10000+ статей новин на предмет їх інформативності [22]. На основі цих даних The Factual пропонує п’ять продуктів, щоб отримувати найбільш правдиві новини:

– інформаційний бюлетень. За допомогою редакторів–людей для перехресної перевірки алгоритму The Factual випускає щоденну розсилку з найбільш інформативними статтями з усього політичного спектру на актуальні теми новин;

– веб-сайт – сайт, який показує актуальні теми в реальному часі, групуючи тисячі пов’язаних статей і дозволяючи фільтрувати за політичними пристрастями, оцінками тощо;

– додаток для iOS і Android, що поєднує щоденні інформаційні бюлетені та веб–сайт у простій програмі;

– розширення для Chrome – розширення для веб–переглядача, яке виконує дві функції: миттєво оцінює інформативність будь–якої новинної статті та додає рейтинги статей прямо у канали Twitter і Facebook;

– IsThisCredible.com – ще один веб-сайт, на якому можна отримати рейтинг будь-якої знайденої статті новин або знайти найактуальніші новини на будь- яку тему.

Кожна стаття отримує оцінку від 1 до 100% на основі чотирьох показників: якості сайту, досвіду автора, якості та різноманітності джерел і тону написання. Кожен із цих показників відповідає на ключові запити щодо інформаційної цінності статті:

– якість сайту: чи має цей сайт історію створення інформативних статей із хорошими джерелами;

– експертиза автора: чи має автор досвід написання добре досліджених інформативних статей на цю тему? Чи автор зосереджується на темі і, отже, може мати певний досвід;

– якість і різноманітність джерел: скільки унікальних джерел і прямих цитат використано в статті? Який рейтинг сайту цих джерел;

– тон статті: чи була стаття написана в нейтральному, байдужому тоні чи вона була емоційною мовою.

Ці чотири метрики поєднуються, щоб отримати єдину відсоткову оцінку, яку можна інтерпретувати як ймовірність того, що стаття є інформативною. Оцінки вище 75% вважаються інформативними, тоді як оцінки нижче 50% менш імовірно.

Переваги сервісу:

– система рейтингу якості інформації;

– різноманітні інтерфейси користувача;

Недоліки:

– не підтримує українську мову;

– обов’язкова реєстрація акаунту;

– неможливість аналізу конкретної інформації, тільки надані новини;

– людське втручання в фінальний рейтинг достовірності інформації.

2.2 Check від Meedan

Check розроблений Meedan (рисунок 2.2) – набір інструментів для розробки ботів для популярних месенджерів, що можуть перевіряти достовірність інформації окремих повідомлень. Має великий та гнучкий набір інструментів для побудови потоків аналізу текстової інформації з налаштуванням під різні параметри та критерії.

Check є платформою для спільної перевірки цифрових носіїв. Проект Check створює онлайн-інструменти для підвищення якості розслідувань громадянської журналістики та допомагає обмежити швидке поширення чуток і дезінформації в Інтернеті.

Використовуючи Check, можна швидко створити команду, додати посилання на соціальні мережі для перевірки фактів, задати ключові запити для дослідження новин [21].

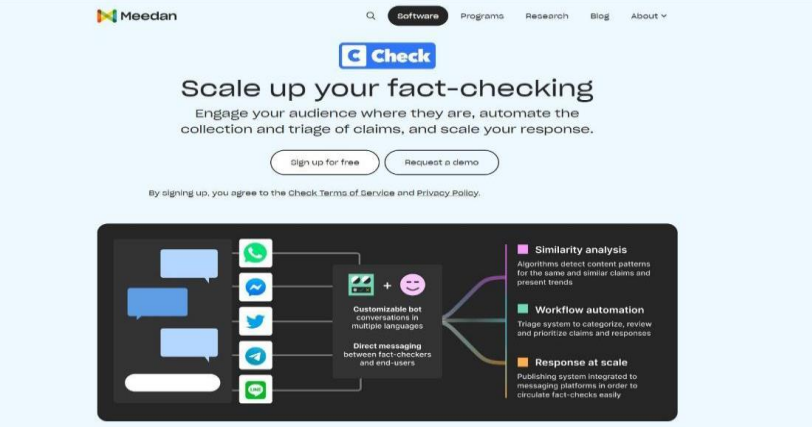

Рисунок 2.2 – Інструмент Check

Розширення для веб-переглядача Check дозволяє швидко додавати медіа-елементи для дослідження та відповідати на анотації щодо цих елементів прямо з бічної панелі розширення.

Переваги:

– широкий функціонал;

– гнучкість налаштування процесу аналізу;

– працює з багатьма популярними месенджерами.

Недоліки:

– не підтримує українську мову;

– обов’язкова реєстрація акаунту;

– надає інструменти, але не є готовою системою;

– має досить високий поріг входження для використання користувачем ;

– немає додатку.

2.3 Logically

Logically (рисунок 2.3) – мобільний додаток з новинною стрічкою, що використовує нейронну мережу для перевірки достовірності інформації в постах які вони ретранслюють з популярних новинних джерел.

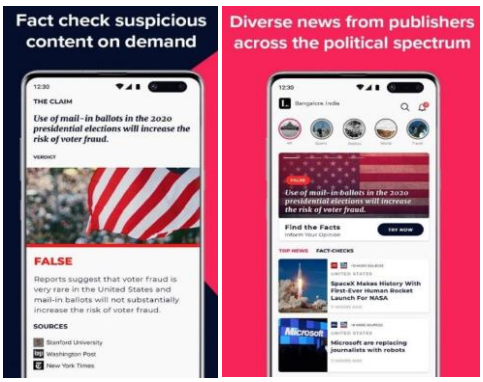

Рисунок 2.3 – Додаток Logically

Додаток використовує такі джерела:

– Yahoo News;

– The Hill;

– Jim Roberts Realty;

– The Boston Globe;

– Fox News;

– The Straits Times.

Переваги:

– висока якість аналізу достовірності інформації;

– безкоштовний мобільний додаток зі зручним інтерфейсом.

Недоліки:

– не підтримує українську мову;

– неможливість аналізу конкретної інформації, тільки надані новини;

– проблеми з продуктивністю на Android системах.

2.4 Аналіз розглянутих рішень

Як результат аналізу схожих ресурсів, було отримано їх переваги та недоліки, створено порівняльну таблицю, та визначено, які особливості повинен мати мій проект. Підсумковий аналіз наведено у порівняльній таблиці 2.1.

Таблиця 2.1 – Порівняльна таблиця

Критерій	The Factual	Check by Meedan	Logically
Обов’язкова реєстрація аккаунту	Так	Так	Так
Можливість перевірити будь-яку бажану інформацію	Ні	Так	Ні
Доступність на різних платформах	Так	Ні	Ні
Нагромадження зайвої інформації	Так	Ні	Так
Повна автоматизація факт-чекінгу	Ні	Ні	Так
Підтримка української мови	Ні	Ні	Ні

Аналіз сервісів виявив кілька принципових недоліків, які необхідно врахувати при розробці власного застосунку:

– відсутність підтримки української мови: Усі три сервіси не підтримують українську мову, що робить їх непридатними для україномовних користувачів. Новий застосунок має включати обробку української мови за допомогою моделей, таких як RoBERTa, або інтеграцію з перекладачами;

– обмежена універсальність: The Factual і Logically не дозволяють перевіряти довільні твердження, що знижує їхню гнучкість. Новий застосунок має підтримувати перевірку будь-яких текстових тверджень;

– складність інтерфейсу: Нагромадження додаткової інформації в The Factual і Logically ускладнює сприйняття. Інтерфейс нового застосунку має бути мінімалістичним і зосередженим на основних результатах;

– лише часткова автоматизація: The Factual і Check by Meedan покладаються на ручну модерацію, що уповільнює обробку. Новий застосунок має бути повністю автоматизованим, використовуючи моделі NLP, такі як RoBERTa-large;

– доступність: Відсутність мобільних додатків у Check by Meedan і Logically обмежує їхню аудиторію. Новий застосунок має бути доступним через веб і мобільні платформи (наприклад, за допомогою Angular).

2.5 Висновки до розділу

У другому розділі кваліфікаційної роботи було здійснено аналіз наявних рішень, які вже існують на ринку, та проведено порівняння їх функціональних можливостей, переваг і недоліків.

Зокрема, були розглянуті найбільш популярні платформи та сервіси, що пропонують функціональність, подібну до реалізованого у межах цього дослідження.

На основі проведеного аналізу було встановлено, що хоча деякі аналоги мають широкий функціонал, вони часто не відповідають потребам україномовного користувача – як через відсутність повноцінної підтримки української мови, так і через обов’язкову реєстрацію чи обмеження доступу до основних функцій.

Таке порівняння підтвердило актуальність і доцільність реалізації власного проекту, орієнтованого саме на користувачів в Україні, з урахуванням мовних та функціональних вимог.

Одержані результати слугують підставою для розробки унікального застосунку, що відзначається простотою у використанні, відкритим доступом та підтримкою української мови без потреби у складній авторизації.

3 ПРОЕКТУВАННЯ ЗАСТОСУНКУ ТА КОМПОНЕНТИ ШІ

3.1 Функціональна структура застосунку

У цьому розділі систематизовано знання та засоби реалізації поставлених цілей в даній роботі. Також наведено огляд інструментів, мовних моделей та програмних рішень, які можуть бути використані.. Детальний аналіз технологічної основи дозволяє краще зрозуміти принципи функціонування застосунку, що планується реалізувати, та обґрунтувати доцільність використання кожного з компонентів у контексті поставленого завдання. Усі етапи реалізації поділено на функціональні блоки, кожен з яких виконує специфічну роль у загальній архітектурі системи.

Застосунок автоматизованої перевірки тверджень призначений для оцінки правдивості текстових повідомлень за допомогою моделей обробки природної мови. Основна мета – допомогти користувачам швидко і ефективно перевіряти інформацію, особливо в умовах великого потоку новин та можливих фейкових повідомлень.

Для підтримки зазначених характеристик застосунок вирішує наступні задачі:

– отримання тексту від користувача;

– розділення тексту на уривки;

– отримання релевантних новин з довіреного джерела (через API NewsAPI);

– обробка уривків з метою оцінки правдивості твердження на основі релевантних новин за допомогою моделі RoBERTa та формування метрики впевненості;

– знаходження метрик емоційної забарвленості та упередженості;

– надання результату користувачу у форматі JSON через веб-інтерфейс.

Перший та останній пункт розглядаються окремо в кінці розділу, адже оба є компонентами клієнтської частини застосунку.

Також для наглядності було розроблено діаграми, а саме контекстну (рисунок 3.1) та контейнерну (рисунок 3.2).

Рисунок 3.1 – Context diagram

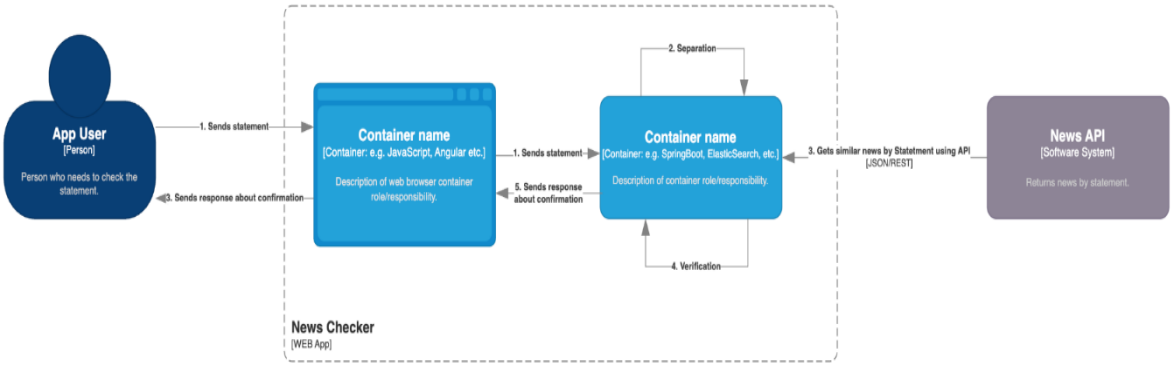

Рисунок 3.2 – Container diagram

Згідно із нотацією моделювання, що використовується (C4), ще один рівень деталей на рівні компонентної діаграми зазвичай присутній. В цьому випадку він не має окремого сенсу через те, що функціонально застосунок не є складним і нових деталей на нижчому рівні декомпозиції не буде ідентифіковано.

Таким чином, можна перейти до детального розгляду визначених задач. Перша задача «отримання тексту від користувача» є типовою інженерною задачею і не є науково значимою. Її реалізацію детально описано в 4 розділі роботи.

Тож, перейдемо до опису першої значимої задачі, що необхідно вирішити для досягнення цілей роботи.

3.2 Розділення тексту на уривки

Для ефективної перевірки тверджень необхідно попередньо сегментувати вхідний текст на смислові частини. У контексті даного застосунку такими частинами виступають уривки тексту, які відповідають зазвичай окремим реченням або невеликим абзацам. Це дозволяє значно покращити точність наступних етапів аналізу, зокрема, пошуку релевантних джерел і доказових фрагментів.

Технічно це реалізується через процедуру токенізації за реченнями, що дає змогу виділити мінімальні логічно завершені одиниці тексту. На цьому етапі важливо уникнути помилкової сегментації, наприклад, через абревіатури або складну пунктуацію, тому часто використовуються готові NLP-бібліотеки, які мають вбудовану обробку винятків.

3.2.1 Процедура токенізації за реченнями

Токенізація за реченнями – це процес поділу тексту на речення-компоненти. Ідея виглядає досить простою. В англійській та деяких інших мовах можна виокремлювати речення кожного разу, коли знаходимо певний знак пунктуації – крапку.

Але навіть в англійській це завдання нетривіальне, тому що крапка використовується і в скороченнях. Таблиця скорочень може допомогти під час обробки тексту, щоб уникнути неправильної розстановки меж речень. У більшості випадків для цього використовуються бібліотеки, тому можна особливо не перейматися деталями реалізації.

Наступним кроком після сегментації є визначення найбільш інформативних частин тексту, тобто вилучення ключових слів.

3.2.2 Вилучення ключових слів

Вилучення ключових слів є одним із найважливіших завдань обробки природної мови, і воно відповідає за визначення різних методів вилучення значної кількості слів і фраз із колекції текстів. Усе це робиться для узагальнення та допомоги у відповідній та добре організованій організації, зберіганні, пошуку та відновленні вмісту.

Існує багато доступних алгоритмів вилучення ключових слів, кожен з яких використовує унікальний набір фундаментальних і теоретичних методів для вирішення цього типу проблем.

Існують різні типи алгоритмів NLP, деякі з яких витягують лише слова, а інші витягують і слова, і фрази. Існують також алгоритми NLP, які виділяють ключові слова на основі повного змісту текстів, а також алгоритми, які виділяють ключові слова на основі всього змісту текстів.

Нижче наведено деякі з найвідоміших алгоритмів вилучення ключових слів:

– TextRank. Цей алгоритм працює за тією ж ідеєю, що й PageRank. Google використовує цей метод для оцінки важливості різних веб-сайтів в Інтернеті;

– частота термінів – зворотна частота документів (TF-IDF): повна версія TF-IDF – частота термінів – зворотна частота документів, яка намагається краще визначити важливість терміна в документі. Крім того, враховуйте зв’язки між текстами з одного корпусу;

– RAKE. RAKE означає швидке автоматичне вилучення ключових слів і є різновидом методу NLP. Це може витягти ключові слова та ключові фрази з вмісту одного документа, не беручи до уваги інші документи в тій же колекції.

Для актуалізації контексту, застосунок надсилає сформульовані уривки як запит до API новинних джерел – NewsAPI. Запит формується на

основі ключових слів, які вилучаються з твердження та його контексту за допомогою алгоритмів витягання термінів і ключових фраз.

3.3 Отримання релевантних новин

На основі сформованого запиту здійснюється пошук релевантних новин із перевірених джерел. Для цього використовується сервіс NewsAPI.

NewsAPI використовується як джерело актуальної інформації з різноманітних новинних ресурсів. Зокрема, API дозволяє здійснювати пошук новинних статей за заданим твердженням користувача, отримуючи тексти, які потенційно можуть містити підтвердження або спростування цього твердження. Завдяки цьому застосунок має доступ до великого обсягу реального контенту, що забезпечує контекстуальну перевірку фактів у майже реальному часі.

3.3.1 Опис API-запиту

Для отримання новин використовується ендпоінт /v2/everything, який дозволяє здійснювати повнотекстовий пошук по статтях. Нижче наведено основні параметри, що застосовуються під час запиту:

– q. Ключові слова або фрази для пошуку в заголовках та текстах новин;

– language. Мова новин (наприклад, «en» або «uk»);

– from / to. Діапазон дат публікацій (у форматі YYYY-MM-DD);

– sortBy. Параметр сортування результатів (relevancy, popularity, publishedAt);

– pageSize. Кількість результатів на сторінку (до 100);

– apiKey. Унікальний ключ доступу до API.

Отримані новини проходять попередню обробку, яка включає лемматизацію, очищення тексту та видалення стоп-слів.

3.3.2 Стоп-слова

Стоп-слова – це слова будь-якої мови, які не надають особливого значення реченню. Їх можна сміливо ігнорувати, не жертвуючи змістом речення. Для деяких пошукових систем це одні з найпоширеніших коротких службових слів, наприклад the, is, at, which і on. У цьому випадку стоп-слова можуть спричинити проблеми під час пошуку фраз, які їх містять, зокрема в таких назвах, як «The Who» або «Take That» Якщо є завдання класифікації тексту або аналізу настроїв, тоді нам слід видалити стоп-слова, оскільки вони не надають жодної інформації для моєї моделі, тобто не допускати небажаних слів у корпус, але якщо у нас є завдання мовного перекладу, то стоп-слова є корисні, оскільки їх потрібно перекладати разом з іншими словами.

Немає жорсткого правила щодо того, коли прибирати стоп-слова. Але краще видалити стоп-слова, якщо завдання стосується класифікації мови, фільтрації спаму, генерації субтитрів, автоматичної генерації тегів, аналізу настрою чи щось, що пов’язано з класифікацією тексту. Видалити стоп-слова за допомогою бібліотек Python досить легко, і це можна зробити різними способами.

Після очистки новини приводяться до уніфікованого формату за допомогою бібліотек spaCy та NLTK.

3.3.3 Бібліотеки spaCy та NLTK

spaCy – це бібліотека програмного забезпечення з відкритим вихідним кодом для розширеної обробки природної мови, написана на мовах програмування Python і Cython.

На відміну від NLTK, який широко використовується для навчання та досліджень, spaCy зосереджується на наданні програмного забезпечення для виробничого використання. spaCy також підтримує робочі процеси

глибокого навчання, які дозволяють підключати статистичні моделі, навчені популярними бібліотеками машинного навчання, як-от TensorFlow, PyTorch або MXNet, через власну бібліотеку машинного навчання Thinc. Використовуючи Thinc як серверну частину, spaCy містить моделі згорткових нейронних мереж для тегування частин мови, розбір залежностей, категоризація тексту та розпізнавання іменованих об’єктів.

NLTK ж є провідною платформою для створення програм Python для роботи з даними людської мови. Вона надає прості у використанні інтерфейси для більш ніж 50 корпусів і лексичних ресурсів, таких як WordNet, а також набір бібліотек для обробки тексту для класифікації, токенізації, формування основи, тегування, синтаксичного аналізу та семантичного міркування, оболонки для індустріальних бібліотек NLP, і активний дискусійний форум.

Після застосування цих бібліотек уривки зберігаються у векторному просторі за допомогою SentenceTransformers, що уможливлює пошук релевантних доказів через обчислення косинусної схожості.

3.3.4 SentenceTransformers

SentenceTransformers – це бібліотека Python для найсучаснішого вбудовування речень, тексту та зображень.

Рекомендується використовувати цю бібліотека для обчислення вбудованих речень/тексту для більш ніж 100 мов. Потім ці вкладення можна порівняти, наприклад, з косинус-подібністю, щоб знайти речення з подібним значенням. Це може бути корисним для семантичного подібного тексту, семантичного пошуку або аналізу перефразів.

Фреймворк заснований на PyTorch і Transformers і пропонує велику колекцію попередньо навчених моделей, налаштованих для різних завдань. Крім того, можна легко налаштувати власні моделі.

Далі виконується вибір речень, які потенційно є доказами твердження. Для цього реалізовано два підходи:

– щільні вектори;

– BIO-маркування.

3.3.5 Щільні вектори

Щільні вектори (dense vectors) – це вектори з невеликою кількістю вимірів (наприклад, 300 або 768), де майже всі елементи мають ненульові значення. Вони компактно й ефективно представляють семантику тексту, зображень чи інших об’єктів у машинному навчанні.

Вони дозволяють представляти речення у вигляді числових векторів фіксованої довжини, які відображають їхню семантичну суть. Таке векторне представлення дає змогу проводити кількісну оцінку схожості між твердженням і потенційними доказами.

У рамках цього підходу речення вважається доказовим, якщо косинусна схожість його векторного представлення з вектором твердження перевищує певне порогове значення λ.

Порогове значення може визначатися емпірично або підбиратися на основі попереднього тестування.

Це дозволяє автоматично фільтрувати речення, які є малоймовірними кандидатами на роль доказів, і зосередити увагу лише на тих, що семантично подібні до твердження.

Розрахунок косинусної схожості, а також обробка та нормалізація векторів реалізовані за допомогою бібліотек Scikit-learn та NumPy. Їх використання забезпечує ефективну реалізацію та швидкість обробки навіть при роботі з великими обсягами тексту.

3.3.5.1 Бібліотеки Scikit-learn та NumPy

Scikit-learn – це бібліотека машинного навчання для мови програмування Python.

Вона містить різні алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, випадкові ліси, посилення градієнта, k-середні та DBSCAN, і розроблений для взаємодії з чисельними та науковими бібліотеками Python NumPy та SciPy.

NumPy – це основний пакет для наукових обчислень на Python. Це бібліотека Python, яка надає об’єкт багатовимірного масиву, різні похідні об’єкти (такі як замасковані масиви та матриці), а також набір процедур для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції формою, сортування, вибір, введення/виведення, дискретне перетворення Фур'є, базова лінійна алгебра, основні статистичні операції, випадкове моделювання та багато іншого.

В основі пакета NumPy лежить об’єкт ndarray. Це інкапсулює n-вимірні масиви однорідних типів даних, причому багато операцій виконуються в скомпільованому коді для підвищення продуктивності.

В контексті застосунку Бібліотека Scikit-learn забезпечує функціональність для обчислення косинусної схожості між векторами, а також дозволяє ефективно працювати з матрицями ознак та класифікаційними моделями.

У свою чергу, NumPy використовується для базової обробки векторів: нормалізація, маніпуляції з формами масивів, розрахунок скалярного добутку тощо.

3.3.6 BIO-маркування

Інший підхід до виявлення доказових речень базується на методі BIO-маркування.

У цьому методі кожне слово в реченні класифікується як:

– B (Beginning) – початок фрагмента, що містить доказ;

– I (Inside) – продовження цього фрагмента;

– O (Outside) – слово, що не є частиною доказу.

На відміну від методу щільних векторів, який аналізує речення як цілісні об’єкти, BIO-маркування дає змогу ідентифікувати саме ті частини речення, які мають найбільшу доказову цінність. Це особливо корисно у випадках, коли одне речення містить одночасно релевантну та нерелевантну інформацію. Такий підхід дозволяє формувати контекстуально точні уривки для подальшої логічної оцінки.

Для реалізації BIO-маркування використовується модель BERT, яка забезпечує високу точність у задачах послідовного маркування завдяки контекстному розумінню мови.

3.3.6.1 Модель BERT

BERT – це модель на основі трансформерної архітектури для контекстного подання слів у тексті.

Вона виявила високу ефективність у завданнях семантичного аналізу, зокрема пошуку, класифікації та перевірки тверджень. У роботі використовується базова версія моделі (BERT base) з 12 шарами та приблизно 110 мільйонами параметрів.

Вхідними даними для моделі BERT є послідовність токенів, які перетворюються на вбудовування. Кожне вбудовування токена є комбінацією 3 вбудовувань. Детальніше пояснення приведено на рисунку 3.3.

Вихід моделі BERT має такі ж номери маркерів, як у вхідних даних із додатковим маркером класифікації, який дає результати класифікації. Тобто показує чи йде речення Б після речення А, чи ні.

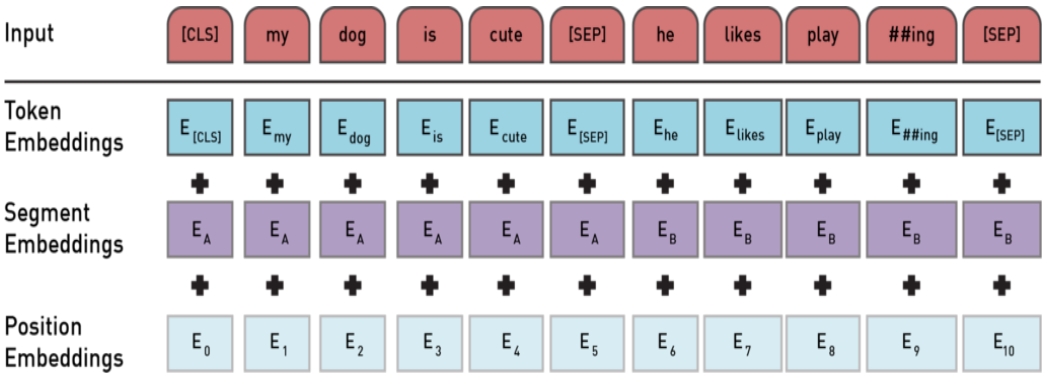

Рисунок 3.3 – Обробка вхідних даних для моделі BERT

Вбудовування токенів (Token Embeddings) – це вбудовування слів зі словника токенів WordPiece.

Вбудовування сегментів (Segment Embeddings) – ці вбудовування використовуються так як модель BERT приймає пару речень як вхідні дані, щоб допомогти моделі відрізнити вбудовування від різних речень. На зображенні вище EA представляє вкладення речення A, тоді як EB представляє вкладення з речення B.

Вбудовування позиції (Position Embeddings) – щоб зафіксувати інформацію про «послідовність» або «порядок», ці вбудовування використовуються для вираження позиції слів у реченні.

У підсумку, після попередніх етапів кожне з виявлених речень розглядається як кандидат на роль доказу та передається на наступний етап – логічну оцінку твердження.

3.4 Оцінки правдивості твердження

Основна мета цього етапу – встановити логічний зв’язок між твердженням і кожним із знайдених доказових речень. Для цього вирішується задача Natural Language Inference (NLI) – класифікація пар твердження/доказ на основі логічного відношення між ними.

3.4.1 Аналіз задачі Natural Language Inference (NLI)

Цей підрозділ описує основні принципи задачі логічного висновування на природній мові, яка є критично важливою для побудови систем перевірки тверджень та роботи з логічною структурою тексту. Загальну схему цієї задачі можна побачити на рисунку 3.4.

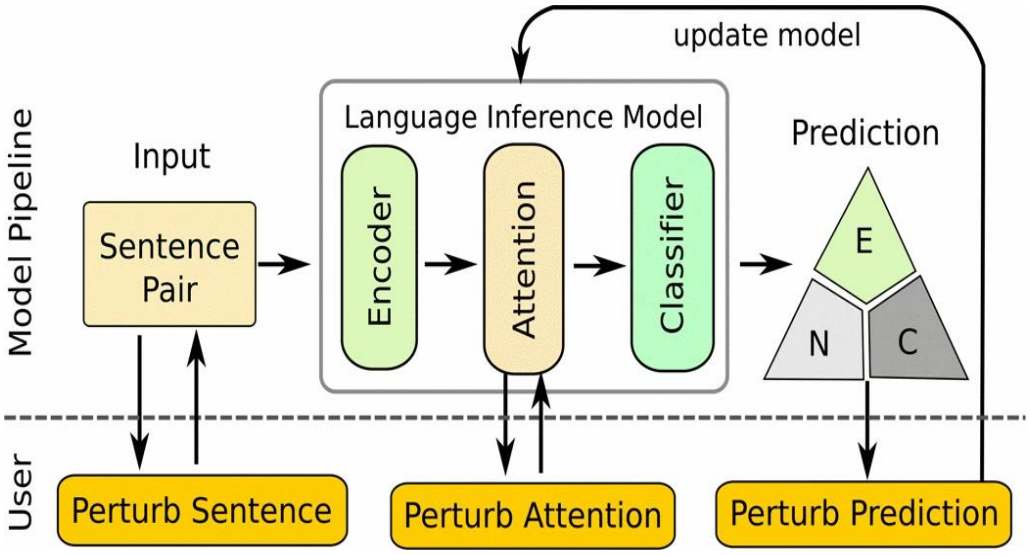

Рисунок 3.4 – Загальна схема задачі NLI

Natural Language Inference – логічне висновування на природній мові – це задача класифікації зв’язку між двома реченнями: гіпотезою та премісою.

Мета – визначити, чи випливає гіпотеза з преміси (entailment), суперечить їй (contradiction) або не має чіткої залежності (neutral).

Цей підхід широко використовується в таких сферах, як фактчекінг, автоматичне резюмування, пошук релевантних доказів та діалогові системи тощо.

У даному випадку застосовується модель RoBERTa-large, попередньо навчена на великих корпусах для задач логічного висновку, таких як SNLI, MultiNLI та Adversarial NLI.

Порівняння їй та вже згаданої моделі BERT можна бачити на рисунку 3.5. Головною відмінністю RoBERTa від попередньої моделі, як можна бачити на рисунку, є кількість токенів, на яких була обучена модель.

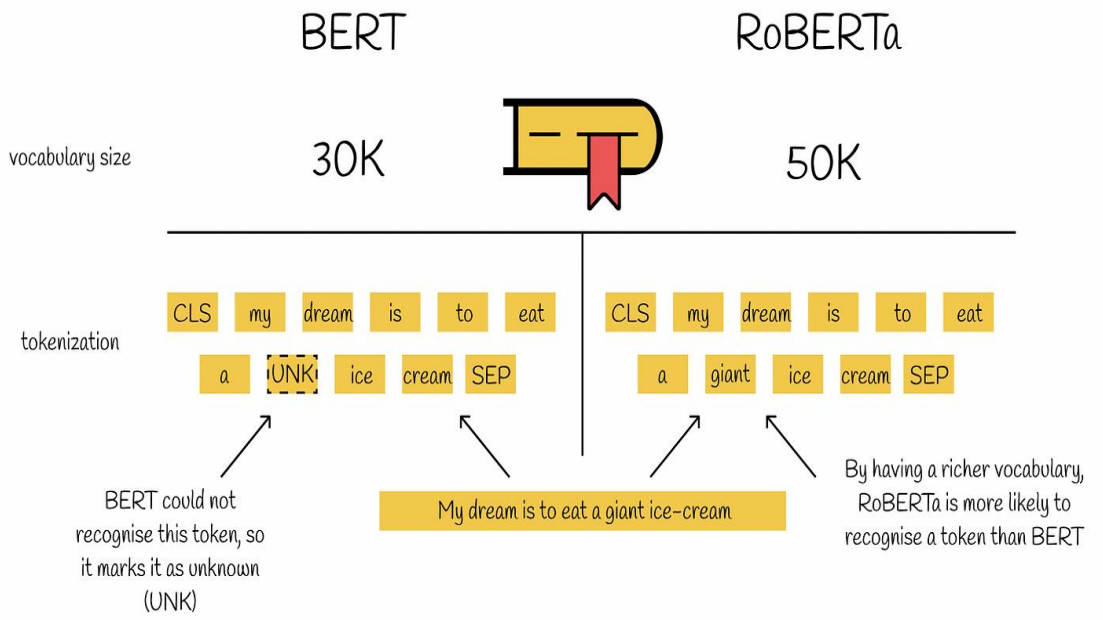

Рисунок 3.5 – Порівняння моделей BERT та RoBERTa

RoBERTa-large класифікує кожну пару як одну з трьох категорій:

– entailment. Твердження логічно випливає з доказу;

– contradiction. Твердження суперечить доказу;

– neutral. Твердження не має явного логічного зв’язку з доказом.

Для кожного доказу обчислюється ймовірність належності до кожного класу, після чого система агрегує ці значення. Переважаючий тип взаємозв’язку формує загальний вердикт:

– ймовірно істинне (переважно entailment);

– ймовірно хибне (переважно contradiction);

– непереконливе (більшість нейтральних доказів або неоднозначна картина).

Для наглядності цей розділ разом з попередніми візуалізовано як діаграму PlantUML (рисунок 3.6).

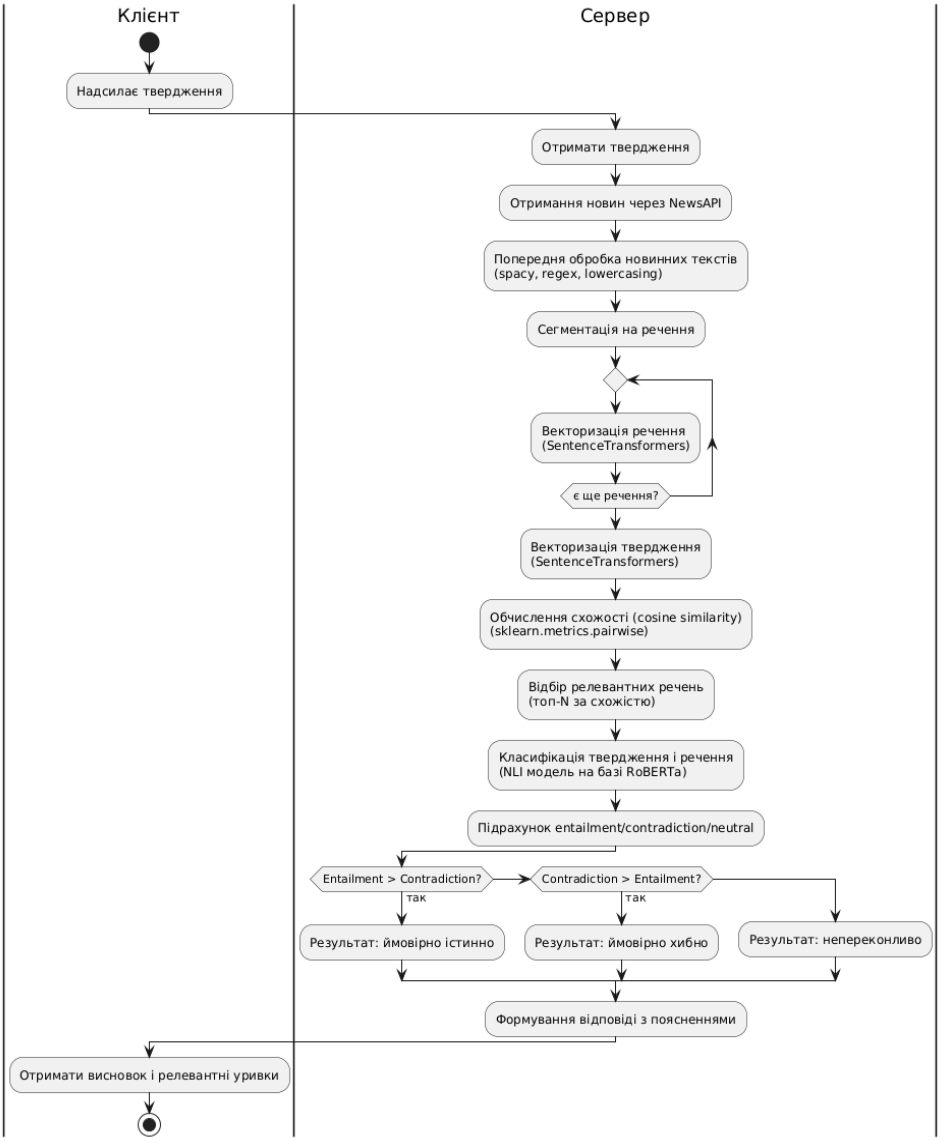

Рисунок 3.6 – Діаграма PlantUML оцінка достовірності тверджень

На цьому етапі перевірка фактів завершується, так що можна переходити до пошуку метрик емоційної забарвленості та упередженості.

3.5 Знаходження метрик емоційної забарвленості та упередженості

Крім логічної перевірки, застосунок також оцінює емоційну забарвленість та упередженість як твердження. Для цього в системі реалізовані два окремі модулі, що розглянуті в підрозділах 3.5.1 та 3.5.2.

3.5.1 Визначення емоційного тону

Для визначення тону використовуються моделі трансформерів, зокрема модель cardiffnlp/twitter-roberta-base-sentiment, яка добре адаптована до коротких текстів і повідомлень соціальних мереж. Завдяки цьому забезпечується висока якість класифікації навіть у випадках стислого або емоційно насиченого контенту.

Аналіз тону виконується за допомогою pipeline(«sentiment-analysis») із бібліотеки transformers, яка автоматизує передобробку тексту, токенізацію та вивід результату.

Класифікація виконується на три категорії:

– LABEL_0 (негативний);

– LABEL_1 (нейтральний);

– LABEL_2 (позитивний).

Модель повертає як клас тону, так і метрику впевненості, що дозволяє враховувати не лише формальну категорію, а й рівень довіри моделі до результату.

3.5.2 Визначення упередженості тексту

Для оцінки упередженості застосовується підхід Zero-Shot Classification, що дає змогу класифікувати текст без попереднього донавчання на специфічному наборі міток.

У проекті використовується модель facebook/bart-large-mnli з бібліотеки transformers. Її основна перевага – гнучкість у роботі з будь-яким переліком категорій.

Для вирішення задачі обрано такі мітки упередженості:

– нейтральний;

– упереджений;

– політизований;

– емоційний;

– фактологічний.

Модель оцінює ймовірність належності тексту до кожної з міток і обирає найбільш імовірну. Результатом є словник із трьома ключовими полями:

– назва найбільш релевантної категорії (наприклад, «упереджений»);

– коефіцієнт впевненості моделі в цьому результаті;

– повний список міток і відповідних ймовірностей.

Такий підхід дозволяє одночасно отримувати детальну картину можливих інтерпретацій тексту з позиції упередженості, а не обмежуватися бінарною оцінкою.

Отримані метрики тону та упередженості разом із висновком по твердженню передаються на фронтенд у форматі JSON через Flask API, після чого виводяться користувачу в інтерактивному інтерфейсі.

3.6 Висновки до розділу

У цьому розділі було розглянуто архітектуру та принципи роботи веб-додатку автоматизованої перевірки тверджень. Застосунок реалізує три основні етапи: отримання релевантних уривків тексту, відбір з них доказових речень, а також оцінку взаємозв’язку між твердженням і цими реченнями за допомогою моделі логічного висновку.

Для пошуку релевантних уривків застосовується гібридний підхід із використанням онлайн-новин із NewsAPI. Після цього застосунок обирає ті речення, які можуть виступати доказами.

Кожен такий доказ проходить оцінку через модель RoBERTa-large, що класифікує його як підтвердження, спростування або нейтральний вислів відносно твердження. На основі кількісної переваги підтверджень або спростувань застосунок видає остаточну мітку: «ймовірно істинне», «ймовірно хибне» або «непереконливе».

Також проводиться оцінка упередженості та емоційної забарвленості. Готові метрики надсилаються користувачу.

Скорочені етапи роботи можна бачити на наступній діаграмі (рисунок 3.8), що доповнює попередню діаграму (рисунок 3.6) але не вдається у вже розглянуті подробиці.

Рисунок 3.7 – Загальна діаграма PlantUML розроблюваного застосунку

Загалом, можна зробити висновок, що обрані рішення є обґрунтованими, а запропонований застосунок є здатним ефективно вирішувати поставлене завдання автоматизованої перевірки тверджень.

Варто зазначит, що поділ на контекстні, контейнерні та функціональні рівні, згідно з методологією C4, дозволив чітко структурувати логіку системи та визначити ролі кожного її елемента.

Запропонована архітектура дозволяє поєднувати точність, продуктивність і модульність, що робить її придатною як для наукових експериментів, так і для практичного використання в умовах великого інформаційного навантаження.

Реалізація к ожного з блоків застосунку, з акцентом на технічні аспекти, інтеграцію моделей та інтерфейсну частину описаних рішень подана в розділі 4.

4 РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

4.1 Оцінка та підвищення продуктивності моделі roberta-large

В цьому підрозділі проведено оцінку моделі roberta-large з використанням різних параметрів. Приклад використання коду можна побачити на лістингу 4.1.

Лістинг 4.1 – Фрагмент коду, що демонструє процес оцінки моделі

```
def evaluate_model_performance(model, dataloader,
parameters):
    model.train()
    optimizer = AdamW(model.parameters(),…)
    for epoch in range(parameters['num_epochs']):
        progress_bar = tqdm(dataloader, desc=f"Epoch
{epoch + 1}")
        for batch in progress_bar:
            inputs = {k: v for k, v in batch.items()}
            outputs = model(**inputs)
            loss = outputs.loss if hasattr(outputs,
'loss') else outputs[0] # Handle different model output
structures
            loss.backward()
            optimizer.step()
            lr_scheduler.step()
            optimizer.zero_grad()
            progress_bar.set_postfix({'loss':
loss.item()})
    model.eval()
    predictions = []
    references = []
    with torch.no_grad():
        for batch in dataloader:
            inputs = {k: v for k, v in batch.items()}
```

Продовження лістингу 4.1

```
            outputs = model(**inputs)
            logits = outputs.logits if hasattr(outputs,
'logits') else outputs[0] # Handle different model output
structures
            preds = torch.argmax(logits, dim=-1).tolist()
            predictions.extend(preds)
            references.extend(batch['labels'].tolist())
    accuracy = accuracy_score(references, predictions)
    f1 = f1_score(references, predictions,
average='weighted')
    return {'accuracy': accuracy, 'f1': f1}
```

У таблицях 4.1 – 4.2 наведено значення accuracy (точність) та f1 (середнє гармонійне між точністю та повнотою) для моделі roberta-large на датасеті FEVER.

Таблиця 4.1 – batch_size 4

num_epochs	max_sequence_length	accuracy	f1_score
3	64	0.71	0.65
3	128	0.75	0.69
5	64	0.78	0.72
5	128	0.81	0.77

Таблиця 4.2 – batch_size 8

num_epochs	max_sequence_length	accuracy	f1_score
3	64	0.74	0.68
3	128	0.78	0.72
5	64	0.82	0.76
5	128	0.86	0.81

Можна бачити, що зі збільшенням значень параметрів `batch_size`, `num_epochs` та `max_sequence_length` продуктивність моделі поступово зростає. Зокрема, при `batch_size = 8`, `num_epochs = 5` та `max_sequence_length = 128` досягається найвищий рівень точності – 0.86, а також найвищий показник F1-міри – 0.81. Це свідчить про те, що модель краще навчається при більших обсягах вхідних даних та тривалішому процесі тренування.

Водночас слід враховувати, що збільшення цих параметрів веде до значного зростання витрат обчислювальних ресурсів, що може бути критичним при розгортанні застосунку на реальному сервері з обмеженим апаратним забезпеченням.

Таким чином, оптимальне налаштування параметрів має враховувати компроміс між продуктивністю моделі та доступними ресурсами. Для цього застосунок було налаштовано на використання значень `batch_size = 8`, `num_epochs = 5` та `max_sequence_length = 128`, що забезпечує найкращу якість результатів при прийнятному часі обробки.

4.2 Реалізація основної програмної частини та API

В цьому підрозділі здійснено реалізацію серверної частини веб-застосунку, яка відповідає за обробку вхідних запитів, виконання аналізу текстів новин та формування результатів перевірки.

Для реалізації API було обрано фреймворк Flask, що дозволяє ефективно обробляти асинхронні запити, автоматично генерує документацію та забезпечує зручну типізацію параметрів.

4.2.1 Веб фреймворк Flask

Flask – це веб фреймворк, написаний на Python. У ньому немає рівня абстракції бази даних, перевірки форми або будь-яких інших компонентів,

де вже існуючі бібліотеки сторонніх розробників забезпечують загальні функції.

Однак Flask підтримує розширення, які можуть додавати функції додатків так, ніби вони реалізовані в самому Flask.

Flask надає такі можливості:

– сервер розробки та відладчик;

– інтегрована підтримка модульного тестування;

– відправлення запитів RESTful;

– використовує шаблони Jinja.

Декоратор ендпоінту `/api/verify` , що відповідає за завдання фактчекингу, продемонстрован в лістингу 4.2. Інші ендпоінти мають подібну структуру, а отже їх я виніс до додатку А разом з іншим кодом.

Лістинг 4.2 – Фрагмент коду, що демонструє декоратор маршруту

```
@app.route('/api/verify', methods=['POST'])
def verify_statement():
    data = request.get_json()
    statement = data.get('statement', '')
    if not statement:return jsonify({'error': 'Твердження
        є обов’язковим'}), 400
    news_texts = fetch_news(statement)
    sentences = []
    for text in news_texts:
        if not text:
              continue
              processed = preprocess_text(text)[0]
```

Після отримання POST-запиту на ендпоінт /api/verify застосунок зчитує JSON-об'єкт, що надійшов від користувача, та отримує з нього значення твердження. У випадку, якщо користувач не вказав твердження, сервер формує відповідь з кодом 400 та повідомленням про помилку.

Далі за допомогою функції fetch_news здійснюється звернення до зовнішнього API для отримання релевантних новинних текстів.

Ці тексти аналізуються на наявність змісту, і ті, що не містять текстового контенту, ігноруються. Кожен отриманий текст проходить попередню обробку, яка включає очищення від зайвих символів, токенізацію, лематизацію та видалення стоп-слів.

Результатом є список відфільтрованих токенів, що в подальшому використовуються для обчислення ембедінгів та семантичного порівняння з твердженням користувача.

Дана API-частина лише викликає функції, що в купі виконують фактчекинг, а отже треба окремо розглянути ті функції.

Згідно з поставленою метою, програма виконує три основні функції:

– фактчекинг;

– автоматизовану оцінку емоційного тону новинного тексту за допомогою попередньо навчених моделей;

– виявлення потенційно упереджених фраз або конструкцій.

4.2.2 Реалізація фактчекингу

Після попереднього очищення повного тексту новини, код якого також винесено в додаток А, для нього створюється вбудування, що репрезентує текст у векторному просторі.

Далі текст розбивається на окремі речення, кожне з яких також проходить очищення та трансформується у вектор. У результаті формується список кортежів, де кожен елемент складається з початкового речення та його векторного представлення.

Ці дані будуть використані для пошуку доказів, релевантних твердженню користувача. Код можна бачити в лістингу 4.3.

Лістинг 4.3 – Отримання новинних текстів, сегментація та векторизація речень

```
for text in news_texts:
     if not text:
         continue
     processed = preprocess_text(text)[0]
     embedding = get_embedding(processed)
     sents = sent_tokenize(text)
     for sent in sents:
         sent_processed = preprocess_text(sent)[0]
         sent_embed = get_embedding(sent_processed)
         sentences.append((sent, sent_embed))
```

Твердження користувача проходить ту ж саму процедуру обробки, що й новинні речення: його очищено та векторизовано. Далі для кожного речення з новин обчислюється косинусна подібність між ембедінгами. Якщо схожість між твердженням і реченням перевищує поріг у 0.8, речення вважається потенційним доказом і зберігається разом з його скалярною оцінкою подібності. Це демонструє лістинг 4.4.

Лістинг 4.4 – Обробка твердження та пошук релевантних доказів

```
processed_statement = preprocess_text(statement)[0]
statement_embedding = get_embedding(processed_statement)

evidence_candidates = []
for sent, sent_vec in sentences:
        sim = np.dot(statement_embedding, sent_vec.T) / (
            np.linalg.norm(statement_embedding) *
np.linalg.norm(sent_vec)
        )
    if sim >= 0.8:
          evidence_candidates.append((sent, sim.item(),
sent_vec))
```

Наступним етапом для кожного знайденого доказу використовується модель roberta-large-mnli, яка класифікує зв’язок між твердженням та реченням. Результатом є розподіл ймовірностей по трьох класах: підтвердження (entailment), суперечність (contradiction) та нейтральність (neutral). Найвірогідніший клас обирається як остаточний результат, а разом із ним зберігається й рівень впевненості моделі та значення попередньо обчисленої семантичної подібності. Код можна бачити в лістингу 4.5.

Лістинг 4.5 – Класифікація логічного зв’язку між твердженням і доказами

```
results = []
for text, score, vec in evidence_candidates:
        inputs = tokenizer(statement, text,
                    return_tensors="pt", truncation=True,
padding=True, max_length=512)
        with torch.no_grad():
            logits = model(**inputs).logits
            probs = torch.softmax(logits,
dim=1).numpy()[0]
        labels = ['contradiction', 'neutral',
'entailment']
        max_idx = np.argmax(probs)
        results.append({
            'text': text,
            'relation': labels[max_idx],
            'confidence': float(probs[max_idx]),
            'score': float(score)
        })
```

У фінальному на основі отриманих класифікацій застосунок визначає загальну кількість доказів, що підтверджують або спростовують твердження. Якщо кількість підтверджень переважає – твердження

вважається «ймовірно істинним». Якщо навпаки – «ймовірно хибним». Якщо ж баланс рівний або взагалі немає релевантних доказів, то твердження оцінюється як «непереконливе». Це демонструє лістинг 4.6.

Лістинг 4.6 – Формування підсумкового результату перевірки

```
entailment = sum(1 for r in results if r['relation'] ==
'entailment')
contradiction = sum(1 for r in results if r['relation'] ==
'contradiction')
total = entailment + contradiction

if total == 0:
    final_label = 'непереконливо'
elif entailment > contradiction:
    final_label = 'ймовірно істинно'
elif contradiction > entailment:
    final_label = 'ймовірно хибно'
else:
    final_label = 'непереконливо'
```

4.2.3 Реалізація оцінка емоційного тону

Для реалізації оцінки емоційного тону застосовується модель cardiffnlp/twitter-roberta-base-sentiment, натренована на аналізі емоційних реакцій у соціальних мережах, яка забезпечує високу точність у класифікації коротких текстів.

Модель повертає один із трьох класів – негативний, нейтральний або позитивний – а також числовий показник впевненості.

Для зручності результати подаються у форматі, придатному для подальшої візуалізації у клієнтській частині застосунку. Весь код модуля наведено у лістингу 4.7.

Лістинг 4.7 – Аналіз тону новинного тексту за допомогою моделі RoBERTa

```
from transformers import pipeline
sentiment_pipeline = pipeline(
    "sentiment-analysis",
    model="cardiffnlp/twitter-roberta-base-sentiment"
)

def analyze_tone(text):
    result = sentiment_pipeline(text)[0]
    label_map = {
        "LABEL_0": "негативний",
        "LABEL_1": "нейтральний",
        "LABEL_2": "позитивний"
    }
    return {
        "tone": label_map.get(result['label']),
        "confidence": float(result['score'])
    }
```

4.2.4 Реалізація виявлення упереджених фраз та конструкцій

Для виявлення упереджених фраз або конструкцій використовується zero-shot модель facebook/bart-large-mnli, яка дозволяє класифікувати текст за заданими категоріями навіть без додаткового донавчання.

Аналіз надає тексту одну з характеристик: нейтральний, упереджений, політизований, емоційний та фактологічний.

На виході модель повертає список ймовірностей для кожного класу, а також топ-результат із відповідним рівнем впевненості. Такий підхід забезпечує гнучкість і дозволяє виявити загальні риси риторики тексту. Код модуля продемонстровано у лістингу 4.8.

Лістинг 4.8 – Виявлення упередженості за допомогою zero-shot класифікації

```
from transformers import pipeline
bias_pipeline =
pipeline(
    "zero-shot-classification",
    model="facebook/bart-large-mnli"
)
def detect_bias(text):
    candidate_labels = [
        "нейтральний",
        "упереджений",
        "політизований",
        "емоційний",
        "фактологічний"
    ]
    result = bias_pipeline(text, candidate_labels)

    top_label = result['labels'][0]

     top_score = float(result['scores'][0])
return {        "bias": top_label,
        "confidence": top_score,
        "labels":
result['labels'],
        "scores":
[float(s) for s in result['scores']]
    }
```

4.3 Розробка веб-додатку

Клієнтська частина веб-застосунку реалізована за допомогою фреймворка Angular, що дозволяє створити сучасний інтерфейс, що надає

користувачу можливість ввести текст новини, надіслати його на сервер для аналізу та переглянути результати без оновлення сторінки.

4.3.1 Angular

Для реалізації клієнтського інтерфейсу мого застосунку було обрано Angular – сучасний фреймворк для створення динамічних веб застосунків. Його переваги, такі як двостороннє зв’язування даних, модульна структура, підтримка TypeScript та потужна система маршрутизації, забезпечують зручне управління станом, ефективне оновлення інтерфейсу та масштабованість.

Angular – це платформа та фреймворк для створення односторінкових клієнтських програм за допомогою HTML і TypeScript. Angular написаний на TypeScript. Він реалізує основні та додаткові функції як набір бібліотек TypeScript, які можна імпортувати у свої програми.

Компоненти визначають представлення, які є наборами елементів екрану, які Angular може вибирати та змінювати відповідно до логіки та даних програми. Компоненти використовують служби, які надають певні функції, не пов’язані безпосередньо з представленнями. Постачальники послуг можна вставляти в компоненти як залежності, роблячи код модульним, придатним для повторного використання та ефективним.

Модулі, компоненти та служби – це класи, які використовують декоратори. Ці декоратори позначають свій тип і надають метадані, які вказують Angular, як їх використовувати.

Метадані для класу компонента пов’язують його з шаблоном, який визначає представлення. Шаблон поєднує в собі звичайний HTML з директивами Angular і розміткою прив’язки, що дозволяє Angular змінювати HTML перед його відтворенням для відображення.

Компоненти програми зазвичай визначають багато представлень, упорядкованих ієрархічно. Angular надає службу Router, щоб допомогти вам

визначити шляхи навігації між видами. Маршрутизатор надає складні можливості навігації в браузері. Приклад використання платформи можна бачити на рисунку 4.1.

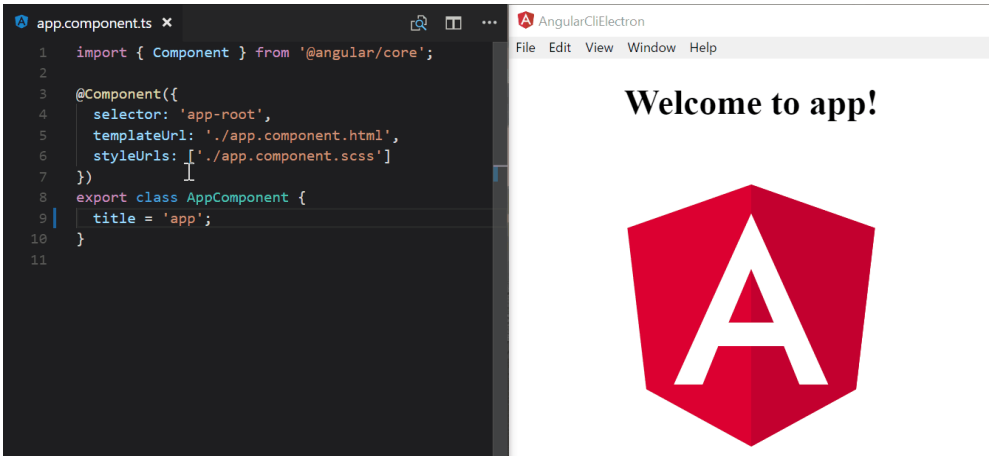

Рисунок 4.1 – Приклад фронтенду на angular

В лістингу 4.9 наведено фрагмент коду, який демонструє основну логіку роботи інтерфейсу.

Лістинг 4.9 – Фрагмент коду, що реалізує веб застосунок

```
@Component({
  selector: 'app-root',
  template: `
    <div class="container">
      <h2>Введіть твердження або текст новини:</h2>
      <textarea [(ngModel)]="statement" rows="5"></textarea>
      <button (click)="analyze()">Аналізувати</button>
      <div *ngIf="result" class="results">
        <h3>Фактчекінг</h3>
        <p><strong>Результат:</strong> {{ result.result
}}</p> <p
*ngIf="result.evidence?.length"><strong>Впевненість:</strong
> {{ result.evidence[0].confidence * 100 | number: '1.1-1'
}}%</p><div>
```

Продовження лістингу 4.9

```
            <strong>Докази:</strong>
            <ul>
              <li *ngFor="let ev of result.evidence">
                <strong>{{ ev.relation }}</strong> ({{
ev.confidence * 100 | number: '1.1-1' }}%): {{ ev.text }}
              </li></ul></div>
          <h3>Тональність</h3>
          <p><strong>Тон:</strong> {{ tone?.tone }}</p>
          <p><strong>Впевненість:</strong> {{
tone?.confidence * 100 | number: '1.1-1' }}%</p>
          <h3>Упередженість</h3>
          <p><strong>Оцінка:</strong> {{ bias?.bias }}</p>
        </div>
      </div>
```

4.4 Демонстрація роботи застосунку

На наступному рисунку можна бачити скріншот роботи застосунку. Стилі роблять зовнішній вид краще. Функціонал відповідає очікуванням.

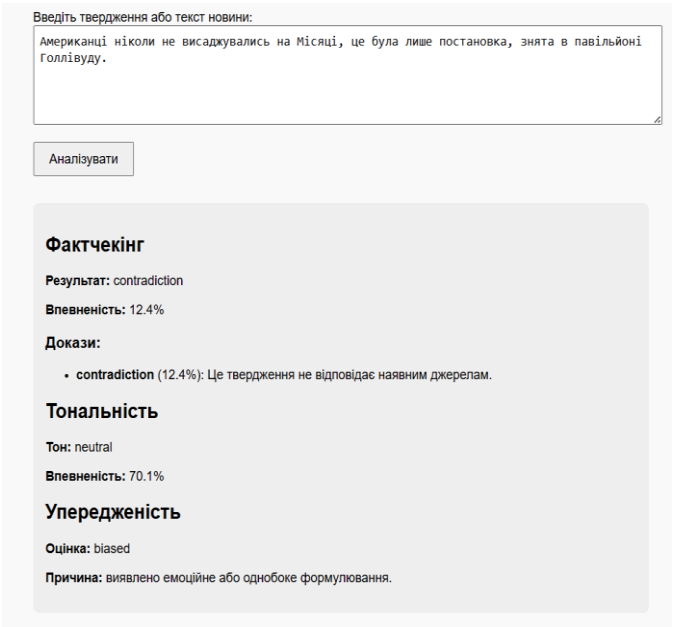

Рисунок 4.2 – Скріншот роботи застосунку

4.5 Оцінка ефективності застосунку

Для оцінки ефективності застосунку було проведено тестування на наборі з 500 тверджень, які поділено на три категорії: політичні (200), наукові (150) та економічні (150). Результати тестування показали різну точність залежно від типу тверджень:

– політичні твердження. Ця категорія виявилася найскладнішою через високу частоту упереджених джерел і неоднозначних формулювань. Наприклад, твердження «The government increased taxes to fund education» отримало мітку «непереконливе» через брак чітких доказів у джерелах. Точність у цій категорії склала 0.82, а F1-міра – 0.79;

– наукові твердження. Наукові твердження, такі як «Vaccines reduce the risk of infection by 90%», показали найвищу точність (0.90) завдяки наявності чітких і перевірених джерел у NewsAPI. Основні помилки були пов’язані з обробкою застарілих даних;

– економічні твердження. Твердження типу «The stock market grew by 5% last month» мали точність 0.85. Основною проблемою була обмежена кількість актуальних джерел для специфічних економічних даних.

Діаграму тестування можна бачити на рисунку 4.3. Код тестування надано в лістингу 4.10.

Лістинг 4.10 – Фрагмент коду, що реалізує тестування

```
def analyze_test_results(test_data_path, output_path):
    with open(test_data_path, 'r', encoding='utf-8') as f:

        test_data = json.load(f)
        category_metrics = defaultdict(list)
        for entry in test_data:
        category = entry['category']
        true_label = entry['true_label']
        predicted_label = entry['predicted_label']
        category_metrics[category].append(
```

Продовження лістингу 4.10

```
{             'true_label': true_label,
             'predicted_label': predicted_label
         })
results = {}
    for category, data in category_metrics.items():
        true_labels = [item['true_label'] for item in
data]
        predicted_labels = [item['predicted_label'] for
item in data]
        accuracy = accuracy_score(true_labels,
predicted_labels)
        f1 = f1_score(true_labels, predicted_labels,
average='weighted')
        results[category] = {
            'accuracy': round(accuracy * 100, 1),
            'f1_score': round(f1 * 100, 1),
            'sample_size': len(data)        }
   results_df = pd.DataFrame.from_dict(results,
orient='index')
    print("Результати аналізу тестування:")
    print(results_df)
    results_df.to_csv(output_path, encoding='utf-8')
   return results
```

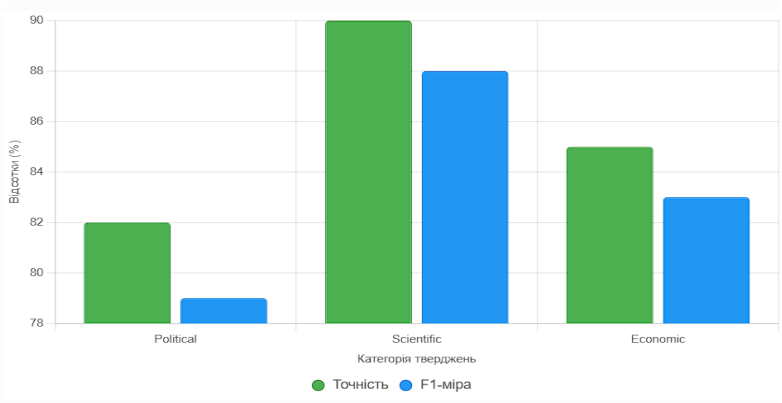

Рисунок 4.3 – Гістограма тестування

4.6 Висновки до розділу

У четвертому розділі було детально описано розробку та реалізацію триступеневого веб-застосунку для фактчекінгу, який охоплює повний цикл обробки інформації: від отримання твердження користувача до формування обґрунтованого висновку щодо його достовірності.

На першому етапі було проведено оцінку та підвищення продуктивності моделі RoBERTa-large, яка стала основою для класифікації логічних зв’язків між твердженнями та доказами.

Проведені експерименти з різними параметрами (batch_size, num_epochs, max_sequence_length) показали, що оптимальна конфігурація (batch_size = 8, num_epochs = 5, max_sequence_length = 128) забезпечує найвищі показники точності (0.86) та F1-міри (0.81) на датасеті FEVER. Це свідчить про здатність моделі ефективно обробляти складні текстові дані при достатній кількості тренувальних ітерацій та більших обсягах вхідних даних.

Однак зростання продуктивності супроводжується збільшенням обчислювальних витрат, що вимагає ретельного балансування між якістю результатів і апаратними ресурсами. Для забезпечення практичної застосовності було обрано компромісну конфігурацію параметрів, яка дозволяє досягати високої точності при прийнятному часі обробки, що є критично важливим для розгортання застосунку в реальних умовах.

На другому етапі було реалізовано серверну частину застосунку з використанням фреймворку Flask, який забезпечує ефективну обробку асинхронних запитів та автоматичну генерацію документації API. Реалізований ендпоінт /api/verify відповідає за обробку тверджень користувача, виклик зовнішнього API для збору новинних текстів, їх попередню обробку (очищення, токенізацію, лематизацію) та подальший аналіз. Для фактчекінгу було розроблено схему, яка включає векторизацію текстів, обчислення косинусної подібності між твердженням і реченнями з

новин, а також класифікацію логічних зв’язків за допомогою моделі RoBERTa-large-mnli. Ця модель дозволяє визначати, чи підтверджує, спростовує чи є нейтральним кожне речення відносно твердження, з урахуванням ймовірностей для кожного класу.

Додатково було реалізовано оцінку емоційного тону за допомогою моделі cardiffnlp/twitter-roberta-base-sentiment, яка класифікує текст як позитивний, негативний або нейтральний, та виявлення упереджених фраз за допомогою zero-shot моделі facebook/bart-large-mnli, що забезпечує гнучку класифікацію за заданими категоріями (нейтральний, упереджений, політизований, емоційний, фактологічний).

Клієнтська частина застосунку була розроблена з використанням фреймворку Angular, що забезпечує створення сучасного інтерфейсу.

Тестування застосунку на наборі з 500 тверджень (політичних, наукових та економічних) продемонструвало високу ефективність: точність склала 0.82 для політичних, 0.90 для наукових і 0.85 для економічних тверджень. Найвища точність для наукових тверджень пояснюється наявністю чітких і перевірених джерел, тоді як політичні твердження виявилися найскладнішими через неоднозначність і упередженість джерел.

Загалом, розроблений застосунок показав багатообіцяючі результати, підтвердивши можливість створення комплексного інструменту для перевірки фактів, аналізу емоційного тону та виявлення упередженості. Використання сучасних моделей NLP, таких як RoBERTa та BART, у поєднанні з ефективними фреймворками Flask і Angular, забезпечило високу точність, гнучкість і зручність у використанні.

4.6.1 Етичні аспекти та проблеми застосунку

Автоматизована перевірка тверджень за допомогою ШІ має значний потенціал для боротьби з дезінформацією, але також породжує низку етичних питань і обмежень, які необхідно враховувати:

– упередженість джерел. NewsAPI агрегує дані з різних новинних джерел, які можуть мати власну редакційну політику або упередженість. Це може впливати на результати фактчекінгу, особливо якщо джерела надають однобоке висвітлення подій. Для зменшення цього ризику застосунок використовує лише перевірені джерела, але повне виключення упередженості неможливе;

– прозорість результатів. Користувачі можуть сприймати висновки моделі як остаточну істину, хоча вони базуються на ймовірнісних оцінках. Для підвищення прозорості в інтерфейсі відображається рівень впевненості моделі для кожного результату, а також список використаних доказів;

– обмеження мовного покриття. Застосунок наразі підтримує обробку текстів англійською та українською мовами, що обмежує його використання для інших мов. У майбутньому планується додати підтримку додаткових мов за допомогою багатомовних моделей, таких як RoBERTa;

– етична відповідальність. Використання ШІ для перевірки фактів може викликати питання щодо відповідальності за помилкові висновки. Наприклад, якщо модель помилково класифікує твердження як «ймовірно істинне», це може вплинути на рішення користувача. Для зменшення цього ризику застосунок чітко позначає результати як ймовірнісні та рекомендує користувачам перевіряти інформацію з інших джерел.

Ці аспекти підкреслюють необхідність подальшого вдосконалення застосунку та підвищення прозорості алгоритмів.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було створено систему перевірки достовірності тверджень.

Було проведено аналіз предметної області, визначено цілі і задачі розробки, а також її основні особливості та критерії ефективності. Був проведений аналіз існуючих альтернатив та проаналізовано переваги й недоліки кожної. На їх основі створено порівняльну таблицю. З отриманих даних було сформовано функціональні та нефункціональні вимоги до системи. На етапі проектування застосунку було проведено вибір технологій розробки, з детальним обґрунтуванням вибору кожної. На основі вимог до системи, визначених раніше було розроблено автоматизовану систему фактчекінгу.

Було продемонстровано, що прихована модель скалярного добутку на основі BERT, навчена за допомогою вибірки softmaxloss, може перевершити традиційне розріджене пошук як окрема модель і призводить до значних покращень при використанні в багатоступеневій архітектурі ранжирування. Було створено новий синтетичний набір даних для фактичної перевірки оцінки пошуку доказів під назвою Factual-NLI+ і доведено, що попереднє навчання на існуючих наборах даних NLI та новий набір даних значно покращує модель пошуку.

Використовуючи навчену модель пошуку, було створено систему семантичного пошуку статей новин, щоб продемонструвати її ефективність у масштабному наборі даних реального світу. Потенційним завданням на наступні ітерації проекту вважаю доцільним дослідити, як щільний пошук принесе користь повній системі перевірки фактів у набагато більшому масштабі. Доповнення Factual-NLI+ шумом із веб-результатів дав більш складний контрольний показник, який демонструє кращу ефективність і відгук у порівнянні з традиційним розрідженим пошуком в реальних умовах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Центр стратегічних комунікацій та інформаційної безпеки. Звіт про інформаційні атаки на Україну у 2022–2023 роках. URL: https://spravdi.gov.ua (дата звернення: 22.05.2025).

2. Allcott H., Gentzkow M. Social Media and Fake News in the 2016 Election. *Journal of Economic Perspectives*. 2017. Vol. 31, № 2. P. 211–236.

3. Brand Ukraine. Нації проти дезінформації 2023: як українці та інші європейці протистоять фейкам. 2023. URL: https://brandukraine.org.ua/uk/projects/nations-against-disinformation-2/ (дата звернення: 22.05.2025).

4. Tandoc E. C., Lim Z. W., Ling R. Defining «Fake News». *Digital Journalism*. 2018. Vol. 6, № 2. P. 137–153.

5. Shu K., Sliva A., Wang S., Tang J., Liu H. Fake News Detection on Social Media: A Data Mining Perspective. *ACM SIGKDD Explorations Newsletter*. 2017. Vol. 19. P. 22–36.

6. Zhou X., Zafarani R. A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities. *ACM Computing Surveys*. 2020. Vol. 53, № 5. P. 1–40.

7. Zhou X., Jain A., Mohapatra P., Zafarani R. Fake News Early Detection: A Theory-driven Model. arXiv preprint. 2020. arXiv:2004.11648.

8. Ahmed H., Traore I., Saad S. Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: *Int. Conf. on Intelligent, Secure and Dependable Systems in Distributed and Cloud Environments.* Springer, 2017. P. 127–138.

9. Silverman C. This Analysis Shows How Viral Fake Election News Stories Outperformed Real News On Facebook. *BuzzFeed News*. 2016. URL: https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook (дата звернення: 22.05.2025).

10. Guess A., Nyhan B., Reifler J. Selective Exposure to Misinformation: Evidence from the Consumption of Fake News during the 2016 U.S. *Presidential Campaign*. European Research Council Working Paper. 2018. URL: https://www.dartmouth.edu/~nyhan/fake-news-2016.pdf (дата звернення: 22.05.2025).

11. Allcott H., Gentzkow M., Yu C. Trends in the Diffusion of Misinformation on Social Media. *Research & Politics*. 2019. Vol. 6, № 2. P. 1–8.

12. Lee K., Wright A., Nystrom M., Stoyanovich J. Hidden Retrieval for Scalable Fact-Checking and Question Answering with NLI Training. arXiv preprint. 2021. arXiv:2109.07410. URL: https://arxiv.org/abs/2109.07410 (дата звернення: 22.05.2025).

13. Романовський О. О. Обробка природної мови: навч. посіб. Київ: *Ліра-К*, 2020. 216 с.

14. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. Draft, 2023. URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 22.05.2025).

15. Devlin J. та ін. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint. 2018. arXiv:1810.04805.

16. Zhang Z., Thirunarayan K., Sheth A. F. Fake news detection using deep learning: A review. *ACM Computing Surveys*. 2021. Vol. 54, № 5. P. 1–36.

17. Liu Y., Ott M., Goyal N. та ін. RoBERTa: A Robustly Optimized BERT Pretraining Approach / Facebook AI. 2019. URL: https://arxiv.org/abs/1907.11692 (дата звернення: 22.05.2025).

18. *FactCheck.org*. Annenberg Public Policy Center. URL: https://www.factcheck.org (дата звернення: 22.05.2025).

19. *Snopes* – Fact Checking and Original Investigative Reporting. URL: https://www.snopes.com (дата звернення: 22.05.2025).

20. *PolitiFact* – Poynter Institute. URL: https://www.politifact.com (дата звернення: 22.05.2025).

21. *Meedan check*. Collaborative media verification platform. URL: https://meedan.com/check/ (дата звернення: 22.05.2025).

22. *The Factual*. Credibility rating system for news. URL: https://www.thefactual.com (дата звернення: 22.05.2025).

23. *Logically*. AI-based fact-checking platform. URL: https://www.logically.ai (дата звернення: 22.05.2025).

24. Vaswani A. та ін. Attention is all you need. *Advances in Neural Information Processing Systems*. 2017. Vol. 30.

25. Коваленко С. І. Методи машинного навчання для аналізу текстової інформації: монографія. Харків: ХНУРЕ, 2021. 184 с.

26. *SpaCy*. Industrial-Strength Natural Language Processing in Python. URL: https://spacy.io (дата звернення: 22.05.2025).

27. *Scikit-learn*: Machine Learning in Python. URL: https://scikit-learn.org (дата звернення: 22.05.2025).

28. SentenceTransformers: Multilingual Sentence Embeddings. URL: https://www.sbert.net (дата звернення: 22.05.2025).