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)

Додаток для виявлення фейкових новин за допомогою трансформерної моделі
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-2

Олексій Тімонін
(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 доц. Євген Павленко
(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 Тімоніну Олексію Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Додаток для виявлення фейкових новин за допомогою трансформерної моделі

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 Python, HTML, CSS, JavaScript, Fast API, Visual Studio Code, наукові статті, інформація з відкритих джерел

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі

2) Теоретичні дослідження

3) Розробка додатку

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 – 24.05	виконано
3	Постановка задачі	25.05 – 26.05	виконано
4	Теоретичні дослідження	27.05 – 30.05	виконано
5	Обґрунтування вибору використаних технологій	31.05 – 02.06	виконано
6	Реалізація проекту	03.06 – 10.06	виконано
7	Оформлення пояснювальної записки	11.06 – 17.06	виконано
8	Підготовка до захисту	18.06 – 24.06	виконано
9	Захист перед ЕК	25.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач  _____
(підпис)

Керівник роботи _____ доц. Євген Павленко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 68 с., 20 рис., 2 табл., 2 дод., 24 джерела.

ВЕБДОДАТОК, МАШИННЕ НАВЧАННЯ, ТРАНСФОРМЕРНІ МОДЕЛІ, ФЕЙКОВІ НОВИНИ, ШТУЧНИЙ ІНТЕЛЕКСТ, API, CSS, HTML, JAVASCRIPT, NLP.

Об’єкт дослідження – процес виявлення неправдивої інформації у новинах за допомогою методів штучного інтелекту.

Предмет дослідження – особливості використання трансформерних моделей для аналізу тексту з метою ідентифікації фейкової інформації.

Мета роботи – дослідження та розробка програмного додатку для класифікації текстових новин на правдиві та фейкові з використанням трансформерних моделей

Методи дослідження – аналіз наукової літератури з обробки природної мови та виявлення фейкових новин, огляд та порівняння сучасних трансформерних моделей, аналіз існуючих програмних рішень, аналіз існуючих підходів до оцінки достовірності новин.

У результаті дослідження було створено вебдодаток для виявлення фейкових новин за допомогою трансформерної моделі.

ABSTRACT

Bachelor’s thesis contains: 68 pp., 20 fig., 2 tabl., 2 ann., 24 references.

API, ARTIFICIAL INTELLIGENCE, CSS, FAKE NEWS, HTML, JAVASCRIPT, MACHINE LEARNING, TRANSFORMATIONAL MODELS, WEB APPLICATION.

The object of research is the process of identifying false information in the news using artificial intelligence methods.

The subject of the study is the peculiarities of using transformational models for text analysis to identify fake information.

The purpose – to research and develop a software application for classifying text news into true and fake news using transformational models.

Research methods – analysis of scientific literature on natural language processing and fake news detection, review and comparison of modern transformational models, analysis of existing software solutions, analysis of existing approaches to assessing the reliability of news.

The study resulted in the creation of a web application for detecting fake news using a transformational model.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект;

API – Application Programming Interface – програмний інтерфейс;

CNN – Convolutional Neural Networks – згорткові нейронні мережі;

CSS – Cascading Style Sheets – каскадні таблиці стилів;

GRU – Gated Recurrent Unit – вентильний рекурентний блок;

HTML – HyperText Markup Language – мова розмітки гіпертексту;

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту;

LSTM – Long Short-Term Memory – довга короткочасна пам'ять;

MLP – Multilayer Perceptron – багатошаровий перцептрон;

NLP – Natural Language Processing – обробка природної мови;

RNN – Recurrent Neural Networks – рекурентні нейронні мережі;

SVM – Support Vector Machine – метод опорних векторів.

ВСТУП

З розвитком цифрових технологій проблема фейкових новин з кожним днем стає все більш актуальною. Соціальні мережи та онлайн-медіа стали основними джерелами інформації для мільйонів користувачів по всьому світу. Доступність різноманітних засобів комунікації дає можливість розповсюджувати будь-яку інформацію, в тому числі ту, яка може бути сумнівною або недостовірною. Усе це дає можливість поширювати недостовірну інформацію та новини, які можуть мати серйозні наслідки для деяких осіб, суспільства та держави в цілому. Такі новини створюються з метою дезінформації, маніпуляцій громадською свідомістю або отримання фінансової чи політичної вигоди.

Фейкові новини можуть мати суттєвий вплив на демократію, громадську безпеку, медичну обізнаність і навіть міжнародні відносини [1]. Класичні методи виявлення такої інформації, такі як фактчекінг вручну, вимагають великих ресурсів, тому не можуть масштабуватись до обсягів сучасного інформаційного потоку. З огляду на це особливого значення набуває автоматизація процесу виявлення фейкових новин. Одним із найбільш перспективних напрямів у цій сфері є використання сучасних методів штучного інтелекту, зокрема трансформерних моделей глибокого навчання.

Завдяки появі трансформерних моделей, таких як BERT, RoBERTa, XLNet, стало можливим вирішувати задачі класифікації текстів з високою точністю. Ці моделі показують високу ефективність у задачах тематичної класифікації, визначення сентименту та виявлення фейкових новин. Використання таких моделей дозволяє розробити системи, для автоматичного визначення достовірності новин.

Однак застосування трансформерів пов’язане з однією суттєвою проблемою – відсутністю прозорості у прийнятті рішень. Тобто, модель може класифікувати новину як фейкову, але користувач не отримає

пояснення чому саме вона дійшла до такого висновку. Через це виникають труднощі із застосуванням таких систем у умовах, де необхідно забезпечити обґрунтування прийнятих рішень. Для подолання цієї проблеми виник окремий напрям досліджень – пояснюваний штучний інтелект.

Пояснюваний ШІ дозволяє візуалізувати або інтерпретувати рішення моделі: наприклад, підсвічувати слова, які вплинули на класифікацію, або будувати графіки впливу окремих характеристик тексту. Це забезпечує більшу довіру користувачів до системи та дає можливість перевірити коректність її роботи [3].

Мета цієї кваліфікаційної роботи це розробка застосунку для виявлення фейкових новин за допомогою трансформерної моделі. Також, буде досліджено існуючі методи виявлення фейкових новин та розроблено нові підходи, щоб забезпечити максимально точне визначення достовірності інформації для користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Опис предметної галузі

У сучасному інформаційному середовищі, для якого характерне стрімке зростання обсягів даних та швидкості їх поширення, проблема дезінформації та фейкових новин набуває особливої актуальності.

Фейкові новини – це навмисно створені повідомлення, які містять недостовірну інформацію, розроблені з метою введення в оману аудиторії, маніпулювання громадською думкою, отримання фінансової або політичної вигоди. На відміну від журналістських помилок, які можуть бути виправлені, фейкові новини цілеспрямовано поширюються для досягнення конкретних мети їх авторів.

Типи фейкових новин включають в себе різноманітні категорії, такі як:

– сатиричні або пародійні– створені з розважальною метою, але можуть сприйматися як реальні;

– сфабриковані – повністю неправдива інформація, створена з метою дезінформації;

– маніпулятивні – поєднання правдивої інформації з недостовірними елементами;

– псевдонаукові – контент, що використовує наукоподібну термінологію для надання хибним твердженням видимості достовірності;

– пропагандистські – інформація, спрямована на формування певних політичних або соціальних поглядів.

Механізми поширення фейкових новин охоплюють різноманітні платформи: соціальні мережі, месенджери, новинні агрегатори, традиційні ЗМІ та спеціалізовані вебсайти. Особливої уваги заслуговують соціальні мережі, де швидкість поширення інформації максимальна, а механізми верифікації мінімальні. Дослідження показують, що фейкові новини в

соціальних мережах поширюються в середньому на 70% швидше, ніж правдиві повідомлення, досягаючи в 6 разів більшої аудиторії [1].

Технічні аспекти ідентифікації фейкових новин базуються на застосуванні методів обробки природної мови для аналізу текстових даних. Якщо традиційні підходи спиралися на статистичні методи та класичні алгоритми машинного навчання, то фундаментальним проривом у цій галузі стало впровадження нейромережевих архітектур, зокрема трансформерів.

Архітектура трансформерів змінила підходи до обробки послідовних даних. Основна перевага трансформерів полягає в механізмі уваги, який дозволяє моделі фокусуватися на різних частинах вхідної послідовності при формуванні вихідних даних, що особливо важливо для розуміння контексту та семантичних зв'язків у тексті [2].

На відміну від рекурентних нейронних мереж, трансформери можуть обробляти весь текст паралельно, що значно підвищує швидкість навчання та надання результатів. Крім того, вони ефективніше зберігають інформацію про довгострокові залежності в тексті, що важливо для аналізу складних текстів у новинах.

Сучасні трансформерні моделі, такі як BERT, RoBERTa, XLNet та GPT, демонструють високу ефективність у вирішенні широкого спектра NLP-задач. Завдяки попередньому навчанню на масивних текстових даних, ці моделі набувають глибокого «розуміння» мови, що дозволяє успішно донавчати їх для вузькоспеціалізованих завдань, зокрема для класифікації новинних текстів.

Застосування трансформерних моделей для виявлення фейкових новин дозволяє автоматизувати аналіз великих обсягів інформації, підвищити точність класифікації та адаптуватися до нових методів створення дезінформації. Водночас, ефективність таких систем залежить від якості навчальних даних, оптимального вибору архітектури моделі та її гіперпараметрів.

1.2 Актуальність дослідження

Проблема поширення фейкових новин набула глобального масштабу, становлячи серйозну загрозу для інформаційної безпеки суспільства. За даними дослідження Європейської комісії, понад 83% громадян ЄС вважають фейкові новини загрозою для демократії [4]. Аналітичний центр Pew Research Center повідомляє, що близько 64% дорослих американців вважають, що фейкові новини спричиняють значну плутанину щодо базових фактів поточних подій [5].

Актуальність розробки методів автоматизованого виявлення фейкових новин зумовлена низкою факторів політичного, економічного та технологічного характеру.

У політичній сфері фейкові новини є інструментом інформаційної війни та гібридної агресії. Їх використання здатне впливати на результати виборів, підривати довіру до державних інститутів та дестабілізувати суспільно-політичну ситуацію

Економічні наслідки поширення фейкових новин також значні. Дезінформація може призводити до:

– фінансових втрат через маніпуляції на фондових ринках;

– репутаційних збитків для компаній, що стають об'єктами недостовірних новин або повідомлень;

– нераціонального використання ресурсів через прийняття рішень на основі хибної інформації;

– витрат на боротьбу з наслідками дезінформації.

Технологічний контекст визначається двома протилежними тенденціями. З одного боку, розвиток генеративних моделей ШІ, таких як GPT-4, Midjourney, та технологій deepfake значно спрощує створення високоякісного фейкового контенту. З іншого боку, прогрес у галузі машинного навчання, зокрема розвиток трансформерних архітектур, відкриває нові можливості для їх ефективного виявлення.

Одночасно розвиток методів машинного навчання, особливо трансформерних моделей, надає нові можливості для автоматизованого виявлення фейкових новин. Трансформери демонструють значний прогрес у розумінні контексту та семантичних нюансів тексту, що дозволяє ефективніше ідентифікувати ознаки маніпулятивних повідомлень.

Отже, дослідження методів автоматизованого виявлення фейкових новин із застосуванням трансформерних моделей є своєчасною відповіддю на виклики сучасного інформаційного суспільства. Розробка ефективних інструментів для боротьби з дезінформацією має значний потенціал для зміцнення інформаційної безпеки, підвищення медіаграмотності населення та захисту демократичних цінностей.

1.3 Аналіз і порівняння існуючих методів виявлення фейкових новин

Методи виявлення фейкових новин можна класифікувати за різними критеріями, наприклад за алгоритмічними методами та архітектурою систем. Серед всіх існуючих методів, можна виділити чотири категорії: лінгвістичні методи, методи машинного навчання традиційного типу, методи глибокого навчання та мультимодальні підходи.

Лінгвістичні методи.

Лінгвістичні методи спрямовані на виявлення ознак дезінформації в текстах, використовуючи інструменти та ідеї лінгвістики. Цей підхід до розпізнавання фейкових новин передував сучасним алгоритмам машинного навчання. Вони включають в себе:

– стилометричний аналіз – дослідження авторського стилю на основі статистичних характеристик тексту;

– аналіз тональності – визначення емоційного забарвлення тексту;

– семантичний аналіз – дослідження смислових зв'язків у тексті;

– аналіз дискурсу – вивчення структури тексту та зв'язків між його частинами.

Основні переваги лінгвістичних методів включають можливість інтерпретувати результати, можливість виявлення конкретних маніпулятивних прийомів та низькі обчислювальні вимоги. Однак вони стикаються з проблемами, як масштабованість, залежність від мови контенту та обмежена здатність враховувати широкий контекст.

Методи машинного навчання традиційного типу.

Традиційні методи машинного навчання базуються на математичних моделях і статистичних підходах, які не вимагають глибокої нейронної архітектури, але залежать від чітко визначених ознак, таких як метадані, граматичні структури та частота слів. До цих методів належать:

– наївний баєсівський класифікатор – імовірнісний метод, що базується на теоремі Баєса;

– метод опорних векторів – метод, що знаходить оптимальну гіперплощину для розділення класів;

– дерева рішень та ансамблеві методи – алгоритми, що будують ієрархічні моделі прийняття рішень;

– логістична регресія – лінійний класифікатор, що моделює ймовірність належності до класу. У комбінації з якісними ознаками цей метод залишається конкурентоспроможним.

Традиційні методи машинного навчання виділяються відносною простотою реалізації, інтерпретованістю моделей та ефективністю при обмежених обчислювальних ресурсах. Однак, вони мають суттєві обмеження, такі як залежність від якості обраних ознак, складність врахування контексту та послідовності слів, а також недостатню гнучкість при роботі з різними типами текстів.

Методи глибокого навчання.

Методи глибокого навчання є одним із напрямів машинного навчання, які використовують штучні нейронні мережі з великою кількістю шарів. Глибоке навчання здійснило революцію в сфері обробки природної мови, включаючи виявлення фейкових новин. Основні архітектури включають:

– рекурентні нейронні мережі – архітектури, спеціалізовані на обробці послідовностей, які ефективно враховують довгострокові залежності в тексті;

– згорткові нейронні мережі – архітектури, що ефективно виявляють локальні патерни в даних;

– трансформерні моделі – архітектури, засновані на механізмі самоуваги, який дозволяє їм зважувати важливість кожного слова в контексті речення або тексту.

Мультимодальні підходи.

Мультимодальні підходи використовують не лише текст, але й інші типи даних для підвищення точності виявлення фейкових новин:

– аналіз зображень – дослідження візуального контенту новин;

– аналіз соціального графа – дослідження структури поширення новин у соціальних мережах;

– часові характеристики поширення – аналіз патернів розповсюдження новин з часом.

Для систематизованого порівняння методів виявлення фейкових новин необхідно проаналізувати їх ефективність за кількома основними критеріями. Трансформерні моделі, такі як BERT, RoBERTa та XLNet, демонструють найвищу точність класифікації, досягаючи 95+% на стандартних датасетах. Вони значно перевершують традиційні алгоритми машинного навчання, які мають точність від 80 до 90 відсотків, і лінгвістичні підходи, які мають точність від 70 до 80 відсотків.

Класичні алгоритми та лінгвістичні методи, такі як логістична регресія або SVM, потребують менше ресурсів, коли йдеться про обчислювальну складність. Наприклад, навчання SVM на звичайному процесорі займає кілька годин, тоді як навчання моделі BERT потребує використання кількох графічних процесорів протягом кількох діб. Отже, важливо знайти баланс між необхідною точністю, яку часто забезпечують

складніші моделі, та доступними обчислювальними ресурсами, що вимагає ретельного аналізу співвідношення потенційних переваг і ресурсних витрат.

1.4 Постановка задачі

Стрімке поширення фейкових новин та дезінформації становить значну загрозу, тому розробка ефективних інструментів для їх автоматичного виявлення є надзвичайно актуальною. Метою роботи є дослідження та розробка програмного додатку для класифікації текстових новин на правдиві та фейкові з використанням трансформерних моделей.

Для досягнення поставленої мети головним завданням є вибір та адаптація трансформерної моделі для задачі класифікації новин. Трансформерні моделі розглядаються як основний інструмент завдяки їхній здатності ефективно аналізувати семантичні та контекстуальні особливості текстових даних, що є важливим для фейкових новин. Процес розробки включатиме підбір або створення датасету з маркованими новинами для навчання моделі.

Технічна реалізація проекту передбачає етап підготовки даних, що включатиме їх очищення та структурування. Наступним кроком буде навчання обраної моделі на підготовлених текстових даних новин. Результатом роботи моє стати програмний додаток з зручним інтерфейсом користувача, який дозволить вводити текст новин та отримувати оцінку з обґрунтуванням результатів. Завершальним етапом буде тестування розробленого додатку для перевірки його коректної роботи та оцінки точності класифікації на нових, раніше не бачених даних.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Основи обробки природної мови

Обробка природної мови (Natural Language Processing, NLP) являє собою галузь, яка поєднує в собі методи з лінгвістики, комп'ютерних наук, когнітивної психології та штучного інтелекту. Її метою є надання комп'ютерним системам здатності не просто обробляти, а й «розуміти», інтерпретувати, генерувати та усвідомлено взаємодіяти з людською мовою, як у її текстовій, так і в усній формі, на рівні, що наближається до людських когнітивних здібностей. NLP є важливою частиною в структурі сучасного штучного інтелекту, оскільки мова слугує основним інструментом передачі знань, вираження думок та соціальної взаємодії. Без ефективної обробки мови можливості інтелектуальних систем для аналізу великих масивів неструктурованої інформації, що створюється людством, були б значно обмежені. За останнє десятиліття обробка природної мови стала невід'ємною частиною нашого повсякденного життя: автоматичний машинний переклад використовується в соціальних мережах, класифікація текстів фільтрує електронну пошту від спаму, пошукові системи, інтегруючи складні лінгвістичні моделі NLP, демонструють значно вищий рівень лінгвістичного аналізу діалогові системи стають дедалі ефективнішими, що стало можливим завдяки здатності розуміти та генерувати людську мову [6]. Основними проблемами в NLP є властива природній мові неоднозначність, варіативність вираження однієї й тієї ж думки, наявність ідіоматичних виразів, сарказму, іронії та залежність значення від контексту.

У процесі розробки системи виявлення фейкових новин особливо важливі завдання NLP, такі як класифікація текстів, аналіз тональності, вилучення інформації та розуміння природної мови, потребують детального розгляду.

Основним завданням NLP є класифікація текстів, яке класифікує новини до однієї або декількох визначених категорій, наприклад, «правдива новина» або «фейкова новина». Це може бути бінарним або багатокласовим завданням. Такі метрики, як точність, повнота, F-міра та прецизійність, використовуються для оцінки якості класифікаторів тексту. Ці метрики особливо важливі при роботі з незбалансованими наборами даних, де один із класів, наприклад, фейкові новини, може бути менш представленим.

Аналіз тональності – процес автоматичного визначення емоційного забарвлення або суб'єктивної оцінки, вираженої в тексті позитивно, негативно чи нейтрально. Його можна розглядати як пряме застосування класифікації за умови, що можна отримати надійні мітки [6]. У випадку фейкових новин, аналіз тональності може допомогти виявити надмірно емоційну, поляризуючу або маніпулятивну лексику, яка часто використовується для впливу на емоційний стан читача та формування упередженої думки. Аналіз тональності може проводитися на різних на рівні документа, речення або навіть окремих аспектів тексту, що дозволяє отримувати більш детальне або, навпаки, узагальнене уявлення про емоційне забарвлення тексту.

Вилучення інформації – це процес автоматичного виявлення та подальшого перетворення фрагментів інформації з великих обсягах неструктурованих або напівструктурованих текстових даних у структурований формат для машинного аналізу. Основними підзадачами у цьому процесі є розпізнавання іменованих сутностей, що ідентифікує згадки людей, організацій, географічних об'єктів, дат тощо, та вилучення відношень, що визначає семантичні зв'язки між цими сутностями. У контексті виявлення фейків, вилучення інформації може застосовуватись для вилучення важливих тверджень з новини, які потім можа перевірити, порівнюючи їх з достовірними джерелами або базами знань.

Розуміння природної мови – це глибша і складніша підгалузь NLP, що фокусується на здатності машини не просто обробляти символи, а

інтерпретувати значення тексту, враховуючи його нюанси, неоднозначності, прихований зміст, прагматику та імплікації. Системи розуміння природної мови значною мірою покладаються на нейронні мережі, які навчаються створювати внутрішні представлення мовних одиниць, які відображають їхнє значення та сприяють інтерпретації комунікативних намірів [7]. Розуміння природньої мови виконує комплекс взаємопов'язаних завдань, таких як вирішення анафори, що дозволяє правильно встановлювати зв'язки між різними згадками одних і тих самих сутностей у тексті, аналіз дискурсу, який досліджує логічну структуру та зв'язність тексту на рівні вищому за окремі речення, та розпізнавання намірів автора, спрямоване на визначення основної мети або ставлення мовця. Крім того, важливу роль відіграють визначення семантичних ролей, вилучення відношень між сутностями, а також вирішення лексичної неоднозначності, коли одне слово може мати кілька значень залежно від контексту. Це надає системі більш та точне уявлення про зміст тексту, що необхідно для ефективного виявлення дезінформації та маніпуляцій, прихованих у фейкових новинах.

Щоб алгоритми машинного навчання могли ефективно аналізувати текстові дані, вони повинні пройти низку етапів попередньої обробки, які спрямовані на очищення, нормалізацію та структурування тексту, перетворюючи його у формат, придатний для подальшого застосування в алгоритмах машинного навчання. Основні з цих етапів включають:

– токенізація;

– лемматизація та стемінг;

– видалення стоп-слів;

– частиномовна розмітка;

– синтаксичний аналіз.

Токенізація.

Токенізація є важливим етапом у попередній обробці текстових даних. Цей етап полягає в тому, щоб розділити загальний потік символів у тексті

на послідовність окремих значущих одиниць, відомих як токени. Ці токени слугують базовими будівельними блоками для подальшого аналізу. Вибір типу токенів значною мірою залежить від специфіки завдання NLP та архітектури обраної моделі. Найчастіше токенами виступають окремі слова, розділені пробілами або пунктуацією. Однак, токенізація може відбуватися на рівні речень, де у деяких випадках, наприклад, при аналізі структури документа, кожне речення може розглядатися як окремий токен, на рівні символів, коли текст може бути розбитий на окремі символи, на рівні частин слів, розбиваючі рідкісні або великі слова на менші частини.

Лемматизація та стемінг.

Ці методи використовуються для зменшення форми слів до їх базової або канонічної форми, що допомагає уніфікувати представлення слів і зменшити розмірність простору ознак. Стемінг є більш грубим процесом, який зазвичай відсікає закінчення слів за допомогою набору правил, не завжди враховуючи морфологічну структуру слова, наприклад, «бібліотека», «бібліотекар», «бібліотечний» можуть бути зведені до основи «бібліотек». Лемматизація, навпаки, є більш складним і лінгвістично обґрунтованим процесом, який використовує словники та морфологічний аналіз для приведення слова до його канонічної форми, наприклад, «йшов», «йду», «піде» будуть зведені до леми «йти». Лемматизація зазвичай дає більш точні результати, але збільшує навантаження.

Видалення стоп-слів.

У NLP стоп-слова – це слова з низьким смисловим навантаженням які відфільтровуються під час попередньої обробки тексту, наприклад, прийменники «в», «на», сполучники «і», «але», частки «не», «б» [8]. Їх усунення зменшує розмірність даних, що може прискорити навчання моделей та потенційно покращити результати класифікації. Проте, слід враховувати, що в деяких ситуаціях, наприклад при аналізі тональності, де заперечення «не» є важливим для контексту, видалення стоп-слів є недоцільним.

Частиномовна розмітка.

Це процес класифікації та маркування слів у реченні відповідно до їх граматичних категорій, тобто дієслів, іменників, часток тощо [9]. Теги частиномовної розмітки можуть бути корисними для вирішення лексичної неоднозначності та слугувати важливими ознаками для більш глибокого синтаксичного та семантичного аналізу.

Синтаксичний аналіз.

На цьому етапі визначається граматична структура речення, тобто синтаксичні зв’язки між словами та фразами. Це зазвичай призводить до створення дерева залежностей, яке ієрархічно відображає структуру речення. Для завдань, які вимагають глибокого розуміння тексту, синтаксичний аналіз є важливим, оскільки він дозволяє зрозуміти, як слова поєднуються, щоб створити зміст. Основні підходи включають парсинг залежностей, що фокусується на бінарних зв'язках між словами, та парсинг на основі складових частин, який розглядає речення як структуру вкладених компонентів.

Після попередньої обробки текст необхідно представити у числовому форматі, оскільки алгоритми машинного навчання працюють саме з числовими даними. Для цього перетворення існують різні методи, які відрізняються за складністю та здатністю охоплювати семантичні елементи тексту.

Модель «мішок слів».

Модель «мішок слів» (Bag of Words) один з найпростіших і найстаріших методів представлення тексту. Текст розглядається як невпорядкований набір слів, де ігнорується граматика, синтаксис та порядок слів, але зберігається інформація про частоту появи кожного слова зі словника, усіх унікальних слів у навчальному датасеті. Кожен документ представляється вектором, розмірність якого дорівнює розміру словника, а значення і-ї компоненти вектора відповідає кількості входжень і-го слова словника в документі. Недоліком є велика розмірність та розрідженість

векторів, а також втрата контекстуальної інформації. Отже, хоча «мішок слів» є основним підходом для кількісного представлення текстових даних, його нездатність враховувати порядок слів і їхні семантичні зв’язки значно обмежує ефективність у складних задачах NLP.

Інтуїтивний вигляд моделі «мішок слів» показано на рисисунку 2.1. У прикладі на рисунку замість того, щоб представляти порядок слів у всіх фразах на кшталт «Я люблю цей фільм» і «Я б його рекомендував», текст трансформується у набір частот слів, де для всього уривку буде вказано, що слово я зустрічається 5 разів у всьому уривку, слово він – 6 разів, слова люблю, рекомендую і фільм – по одному разу, і так далі [8].

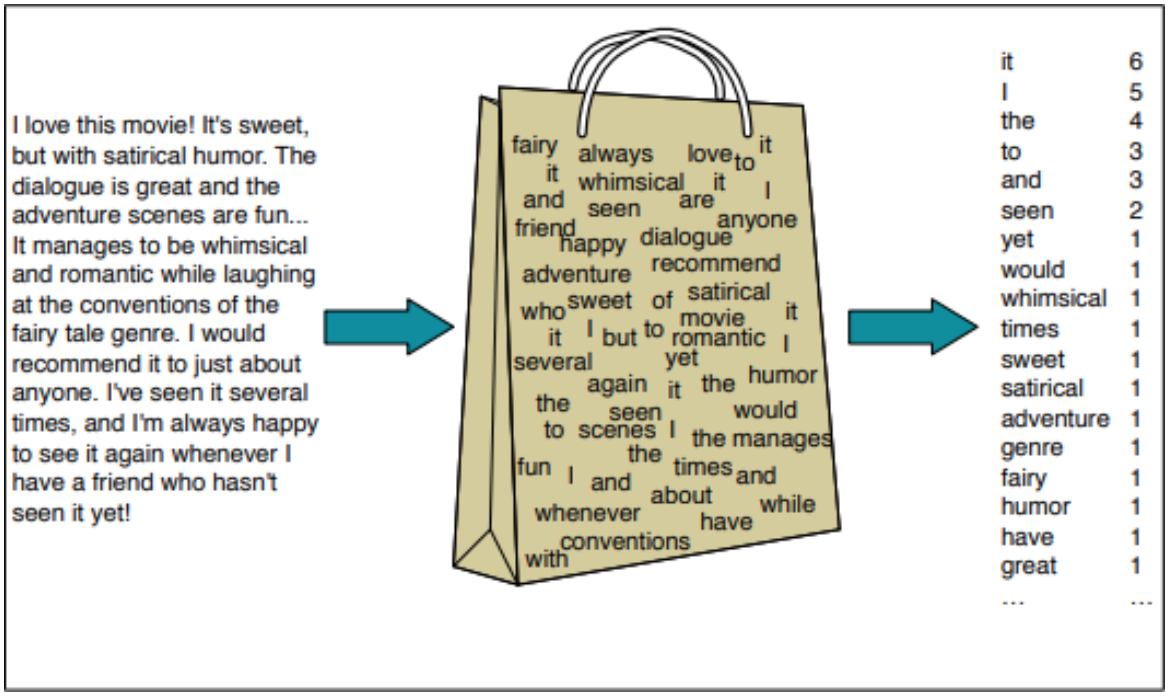

Рисунок 2.1 – Інтуїтивний вигляд моделі «мішок слів» застосованого до рецензії на фільм

TF-IDF.

TF-IDF є поширеним вдосконаленням моделі «мішок слів», що дозволяє оцінити важливість слова не лише в межах одного документа, а й у контексті всього набору даних. Term Frequency – це частота появи слова в

конкретному документі. Inverse Document Frequency – це міра, яка зменшує вагу слів, що часто зустрічаються у багатьох документах і, відповідно, є менш інформативними, та збільшує вагу слів, що є специфічними для певних документів. Значення TF-IDF для слова обчислюється як добуток його TF та IDF. Це дозволяє виділити ключові слова, але, як і Bag of Words, TF-IDF не враховує порядок слів та семантичні зв'язки між ними.

TF-IDF це добуток двох термінів: частоти tf та зворотної частоти документа idf [8]. Це визначається формулою (2.1):

$$\text{tf-idf(t,d)} = \text{tf}_{t,d} * \text{idf}_t, \tag{2.1}$$

де $\text{tf}_{t,d}$ – частота слова t у документі d;

idf_t – значення оберненої частоти документа для слова t.

З формули (2.1) розглянемо формулу (2.2) для TF та формулу (2.3) для IDF:

$$\text{tf}_{t,d} = \log_{10}(\text{count}(t,d) + 1), \tag{2.2}$$

де count(t,d) – кількість входжень слова t в документ d.

$$\text{idf}_t = \log_{10} \frac{N}{\text{df}_t}, \tag{2.3}$$

де N – загальна кількість документів у датасеті;

df_t – кількість документів у яких зустрічається слово t.

Векторні представлення слів.

Векторні представлення слів на відміну від розріджених та високорозмірних векторів моделі «мішок слів» та TF-IDF, є щільними векторами фіксованої, відносно невеликої розмірності. Вони навчаються на великих текстових датасетах таким чином, щоб слова зі схожими

семантичними або синтаксичними властивостями мали близькі векторні представлення у багатовимірному векторному просторі. Це дозволяє моделям узагальнювати знання про слова та їх взаємозв'язки. Наприклад, вони можуть захоплювати семантичні аналогії, як показано на рисунку 2.2, де відношення «король» до «королева» подібне до відношення «чоловік» до «жінка» у векторному просторі.

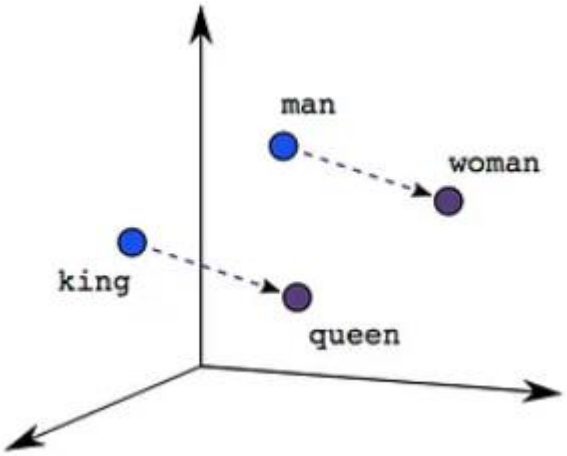

Рисунок 2.2 – Представлення семантичних зв’язків між словами у векторному просторі [10]

Моделі «мішок слів» та TF-IDF легко використовувати та інтерпретувати, але їхні недоліки, такі як ігнорування порядку слів, семантична близькость та генерація розріджених векторів високої розмірності, обмежують їхню ефективність у складних задачах NLP. Статичні векторні представлення слів значно перевершують їх, оскільки здатні захоплювати тонкі семантичні відношення між словами. Це дозволяє моделям машинного навчання краще узагальнювати та «розуміти» зміст тексту. Тим не менш, основним обмеженням цих традиційних векторних методів представлення слів є те, що вони надають кожному слову один глобальний статичний вектор, незалежно від контексту, в якому воно використовується. Це велика проблема для омонімів і слів, значення яких сильно залежить від контексту. Більш сучасні та потужні трансформерні

моделі створюють числові представлення слів, що ефективно вирішує цей основний недолік. Трансформери здатні створювати динамічні векторні представлення для одного й того ж слова, що змінюються залежно від його конкретного оточення в реченні, дозволяючи набагато точніше відображати нюанси значення слова та вирішувати проблему багатозначності слів. Це робить їх придатними для завдань, що вимагають глибокого семантичного аналізу тексту, яким, безперечно, є виявлення фейкових новин. Таким чином, розуміння як класичних, так і передових методів NLP є необхідним для побудови ефективної системи виявлення фейкових новин.

2.2 Глибоке навчання та його роль в обробці природної мови

Глибоке навчання є підгалуззю машинного навчання, що базується на штучних нейронних мережах зі складною багатошаровою архітектурою. Воно дозволяє обчислювальним моделям, що складаються з декількох обробних шарів, вивчати представлення даних з кількома рівнями абстракції, таким чином виявляючи складну структуру у великих наборах даних [11]. За останнє десятиліття глибоке навчання швидко розвивалося в багатьох галузях, зокрема в обробці природної мови, завдяки своїй здатності автоматично вивчати ієрархічні представлення даних та моделювати складні залежності.

Ця перевага глибокого навчання зумовлена низкою фундаментальних відмінностей від традиційних методів машинного навчання, які роблять його ефективним для вирішення складних завдань в області NLP. Автоматичне вилучення ознак є однією з найважливіших таких відмінностей. Моделі глибокого навчання самостійно вивчають відповідні ознаки безпосередньо з сирих даних, таких як послідовності слів або символів. Це відрізняється від класичного машинного навчання, де значна частина зусиль дослідника витрачається на ручне конструювання ознак, що вимагає глибокого розуміння предметної області та часто є трудомістким

процесом. Наприклад, для аналізу тексту це може бути частота слів, TF-IDF, наявність певних n-грам. Такі нейронні мережі можуть формувати ієрархію ознак, починаючи від простих патернів на нижніх рівнях і закінчуючи складними семантичними концепціями на вищих рівнях абстракції завдяки своїй багатошаровай структурі.

Іншою перевагою глибокого навчання є його здатність моделювати складні нелінійні залежності. Текстові дані за своєю природою є високорозмірними та характеризуються наявністю складних, часто нелінійних взаємозв'язків між лексичними одиницями та їх послідовностями. Класичні моделі, такі як логістична регресія, часто мають обмежену здатність ефективно фіксувати такі залежності. Глибокі нейронні мережі, завдяки використанню численних шарів та нелінійних функцій активації спроможні апроксимувати надзвичайно складні функції. Це дає їм можливість розпізнавати тонкі семантичні нюанси та контекстуальні зв’язки в даних, що є критично важливим для отримання глибшого розуміння природної мови. Крім того, збільшення кількості доступних даних значно підвищує ефективність моделей глибокого навчання. Це особливо актуально для NPL, оскільки там використовуються великі обсяги текстової інформації. Глибокі моделі, на відміну від багатьох класичних алгоритмів, продовжують покращувати свої результати при збільшенні навчальної вибірки, що дозволяє їм краще узагальнювати та виявляти складні патерни в текстових даних.

Інші типи нейронних мереж активно використовувалися до появи трансформерних архітектур, які де-факто стали стандартом для багатьох завдань NLP. Серед них варто відзначити багатошаровий перцептрон (Multilayer Perceptron, MLP), який є однією з найпростіших архітектур глибокого навчання. Він складається з вхідного шару, одного або декількох прихованих шарів та вихідного шару, де кожен нейрон у шарі з'єднаний з усіма нейронами попереднього шару. MLP може використовуватися для завдань класифікації тексту, якщо текст попередньо

перетворено у векторне представлення фіксованої довжини, наприклад, за допомогою моделі «мішок слів» або TF-IDF. Однак MLP не враховує послідовну природу тексту та локальний контекст слів, що обмежує його ефективність у складних завданнях NLP.

Для кращого врахування локальних текстових патернів, які часто несуть важливу семантичну інформацію, були адаптовані згорткові нейронні мережі (Convolutional Neural Networks, CNN),спочатку розроблені для обробки зображень. При роботі з текстом, ці мережі використовують спеціальні фільтри, які послідовно аналізують векторні представлення слів. Кожен такий фільтр, по суті, налаштовується на виявлення певних локальних патернів або сполучень слів, але вже на рівні глибинних семантичних ознак, а не просто їх поверхневих послідовностей. Наприклад, один фільтр може навчитися розпізнавати фрази, що виражають позитивне чи негативне емоційне забарвлення, тоді як інший може спеціалізуватися на виявленні певних стилістичних конструкцій. Здатність ідентифікувати такі локальні смислові ознаки, незважаючи на їхнє конкретне розташування в тексті, є основною перевагою CNN в задачах NLP. Оскільки обчислення, пов'язані з різними фільтрами, можуть виконуватися незалежно і паралельно, вони є обчислювально ефективнішими за деякі інші архітектури. Проте, важливо розуміти, що CNN мають обмежену здатність моделювати довгострокові залежності в тексті, оскільки зона тексту, яку одночасно аналізує один фільтр, зазвичай охоплює лише невелику кількість сусідніх слів. Хоча використання декількох шарів таких фільтрів, де вихід одного шару стає входом для наступного, та застосування спеціальних операцій для узагальнення та скорочення виявлених ознак може дещо розширити загальну ділянку тексту, яку охоплює модель за один раз, фіксування справді глобального контексту всього тексту, особливо якщо він великий, залишається для них проблемою. Архітектуру моделі CNN можна побачити на рисунку 2.3.

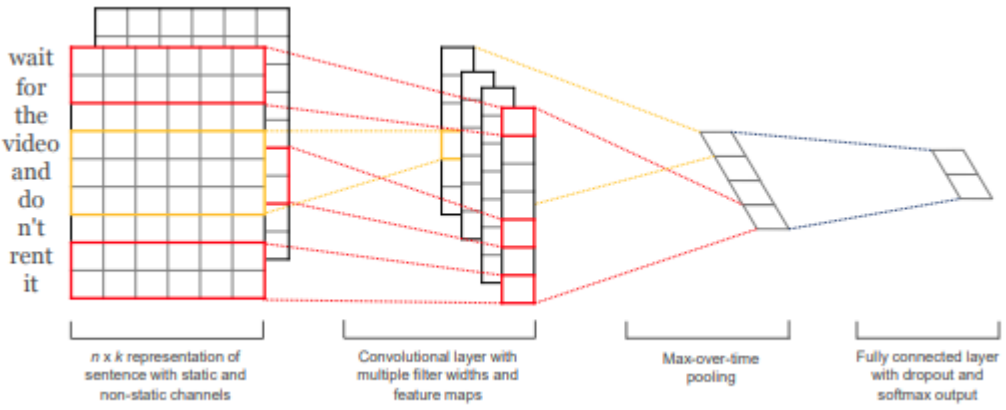

Рисунок 2.3 – Архітектура CNN для класифікації речень [12]

Для обробки послідовних даних були розроблені рекурентні нейронні мережі (Recurrent Neural Networks, RNN). Головною особливістю RNN є наявність «пам'яті» у вигляді прихованого стану, який оновлюється на кожному кроці обробки послідовності. Цей процес, де мережа фактично застосовує ту саму функцію до кожного елемента послідовності, враховуючи попередній стан, наочно ілюструється розгортанням обчислювального графу в часі, як показано на рисунку 2.4. Це дозволяє мережі враховувати попередній контекст при обробці поточного елемента.

Однак, у простих RNN виникає серйозна складність, відома як проблема зникаючих або вибухаючих градієнтів. Коли мережа навчається, вона аналізує свої помилки і намагається їх виправити, посилаючи сигнал коригування назад по всіх кроках обробки послідовності. Якщо внутрішні налаштування мережі на кожному кроці є малими, цей коригувальний сигнал слабшає і може майже повністю зникнути до того, як дійде до початку послідовності. Через це мережі важко навчитися враховувати інформацію, яка була на самому початку довгої послідовності.

Крім того, великі налаштування можуть призвести до надмірного посилення сигналу, що зробить процес навчання нестабільним і непередбачуваним.

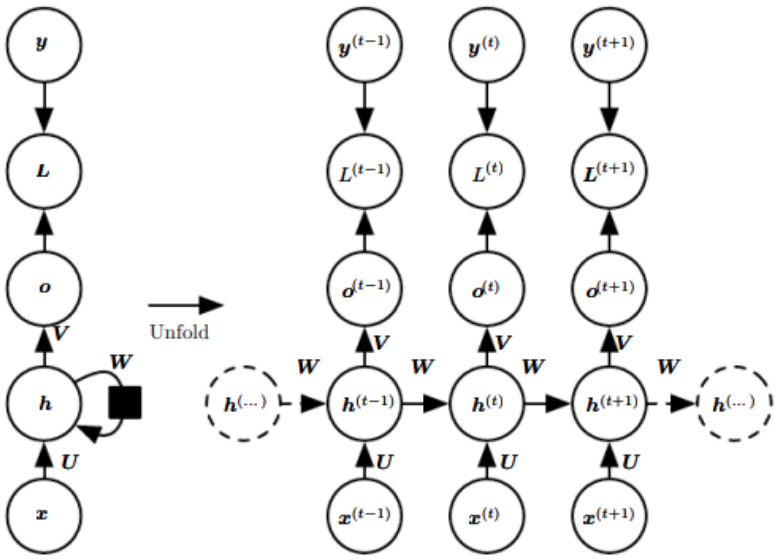

Рисунок 2.4 – Схема рекурентної нейронної мережі та її розгортання в часі [13]

Для подолання цих проблем RNN були розроблені складніші рекурентні архітектури, зокрема LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit). Детальна структура типової LSTM-комірки представлена на рисунку 2.5.

Як видно зі схеми, архітектура LSTM включає спеціальні комірки пам'яті та три типи вентилів (вхідний, забування та вихідний), які регулюють потік інформації, дозволяючи вибірково зберігати та отримувати доступ до інформації протягом тривалих періодів. GRU є спрощеною версією LSTM з двома вентилями (оновлення та скидання), що часто демонструє схожу продуктивність при меншій кількості параметрів і потенційно швидшому навчанні. Завдяки своїй здатності ефективно моделювати послідовності та долати обмеження простих RNN, LSTM та GRU стали стандартом для багатьох NLP завдань до появи трансформерів, таких як машинний переклад, аналіз тональності та генерація тексту.

Незважаючи на значні переваги, які мають LSTM і GRU, ці рекурентні архітектури мають деякі обмеження. Через свою послідовну природу RNN важко паралелізувати на рівні елементів послідовності. Це робить їх

навчання повільним, особливо на довгих послідовностях. Хоча LSTM і GRU значно покращили здатність моделювати довгострокові залежності, вони все ще можуть мати проблеми з надзвичайно довгими послідовностями, у яких інформація з дуже віддалених кроків може поступово втрачатися або спотворюватися, а обчислювальна складність зростає пропорційно довжині послідовності.

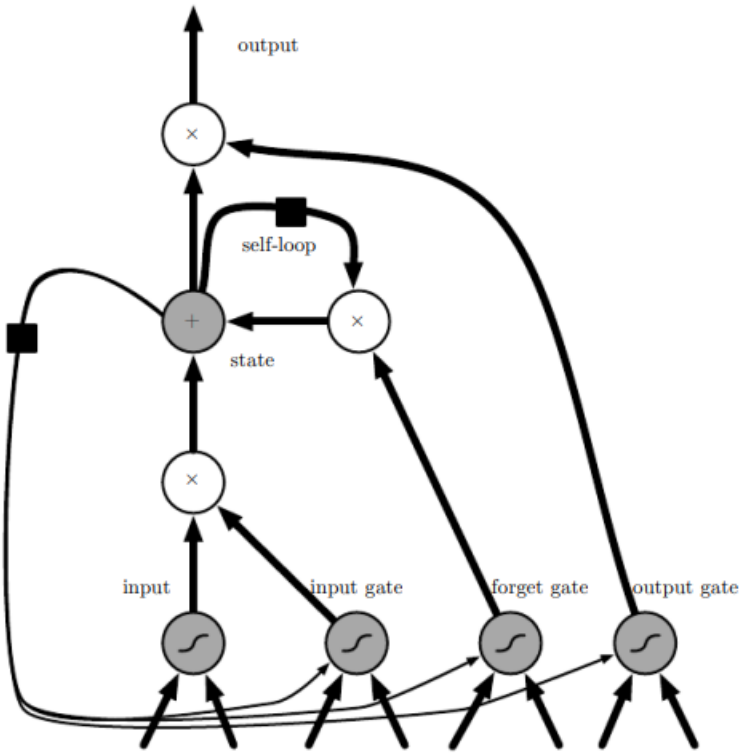

Рисунок 2.5 – Блок-схема комірки LSTM [13]

2.3 Загальні принципи машинного навчання для класифікації текстів

Машинне навчання – це галузь штучного інтелекту, яка зосереджена на розробці алгоритмів, здатних навчатися на основі даних без явних інструкцій для конкретних задач. У контексті аналізу текстової інформації, зокрема для виявлення фейкових новин, методи машинного навчання надають інструменти для автоматичної класифікації текстів.

Існує кілька основних парадигм машинного навчання, серед яких виділяють:

– навчання з учителем;

– навчання без учителя;

– навчання з підкріпленням.

Навчання з учителем.

В основі навчання з учителем лежить використання набору даних, де для кожного вхідного прикладу відома правильна відповідь або мітка класу. Алгоритм навчається на цих даних, намагаючись знайти закономірності, що пов'язують вхідні характеристики об'єктів з їхніми мітками. Метою є побудова моделі, здатної прогнозувати мітки для нових, раніше не бачених об'єктів.

Навчання без учителя.

На відміну від навчання з учителем, у цьому випадку алгоритм працює з даними, що не мають попередньо визначених міток. Метою є виявлення прихованих структур або закономірностей. Прикладами задач є кластеризація, зниження розмірності або пошук асоціативних правил.

Навчання з підкріпленням.

Ця парадигма передбачає взаємодію «агента» з «середовищем». Агент виконує дії в середовищі, а у відповідь отримує винагороди або покарання. Метою агента є навчання такої стратегії поведінки, яка максимізує сукупну винагороду з часом. Навчання з підкріпленням застосування в таких сферах, як створення штучного інтелекту для комп'ютерних ігор, розробці робототехнічних систем та для побудови різноманітних систем управління.

Серед різноманітних задач, які вирішує машинне навчання, одним із основних та найчастіше використовуваних є задача класифікації. Класифікація в машинному навчанні полягає у віднесенні об'єкта до одного з попередньо визначених класів на основі його ознак, отже, маючи простір об'єктів X та скінченну множину міток класів Y, задача полягає у побудові функції, яка для кожного об'єкта X визначає його клас Y. В залежності від

кількості класів, класифікація може бути бінарною, коли існує лише два можливі класи, або багатокласовою, коли класів три або більше.

Процес побудови ефективної моделі класифікації зазвичай включає такі етапи, як збір та підготовка даних, вибір ознак, вибір моделі, навчання моделі, оцінка якості, налаштування гіперпараметрів.

Збір та підготовка даних.

На цьому етапі виконується агрегація відповідних текстових матеріалів з відповідними мітками, очищення їх від шумів, обробка пропущених значень і попередня текстова обробка. Це включає в себе такі етапи обробки, як токенізація, лемматизація або стемінг, видалення стоп слів, частиномовну розмітку та синтаксичний аналіз.

Вибір ознак.

Історично цей етап, особливо до широкого розповсюдження методів глибокого навчання, полягав у ручному або автоматизованому відборі найбільш інформативних характеристик або конструюванні нових ознак на основі наявних. Для текстових даних це могло включати, наприклад, створення «мішка слів», розрахунок TF-IDF або аналіз тональності. Сучасні трансформерні моделі здатні автоматично вивчати релевантні ознаки, проте розуміння цього етапу залишається важливим, оскільки це сприяє глибшій інтерпретації поведінки моделі, дозволяє обґрунтовано підходити до комбінування текстових ознак, вивчених трансформером, з іншими типами даних, а також є корисним при адаптації та оптимізації моделей для специфічних завдань, діагностики проблем або в умовах обмежених обчислювальних ресурсів чи невеликих наборів даних, де явне конструювання ознак може надавати переваги.

Вибір моделі.

На цьому етапі обирається конкретний алгоритм машинного навчання на основі якого буде побудовано класифікатор. У рамках даної кваліфікаційної роботи, враховуючи високу ефективність сучасних

трансформерних архітектур, вибір буде проводитися серед: BERT, RoBERTa та XLNet.

Навчання моделі.

Навчання моделі необхідно для того, щоб алгоритм налаштовував свої внутрішні параметри для мінімізації помилки передбачення.

Оцінка якості.

Після навчання модель тестується на даних, які не були використані при навчанні. Для цього використовуються окремі набори даних: валідаційний для проміжної оцінки та налаштування, та тестовий для фінальної оцінки.

Налаштування гіперпараметрів.

Для оптимізації продуктивності моделі проводиться налаштування гіперпараметрів – параметрів, що встановлюються до початку навчання, наприклад, швидкість навчання, кількість шарів у нейронній мережі, часто за допомогою методів пошуку по сітці або випадкового пошуку на валідаційній вибірці.

Для об'єктивної оцінки продуктивності моделі класифікації використовуються різноманітні метрики, такі як Confusion Matrix, Accuracy, Precision, Recall, F-measure, графік ROC (Receiver Operating Characteristic), ROC-крива та AUC (Area Under the Curve).

Confusion Matrix візуалізує ефективність алгоритму класифікації. Для бінарної класифікації вона має вигляд:

Таблиця 2.1 – Вигляд Confusion Matrix

	Прогнозований: позитивний	Прогнозований: негативний
Справжній: позитивний (P)	True Positive (TP)	False Negative (FN)
Справжній: негативний (N)	False Positive (FP)	True Negative (TN)

Значення цієї матриці можна інтерпретувати наступним чином: True Positive (TP) – це кількість об'єктів, правильно класифікованих як позитивні; True Negative (TN) – кількість об'єктів, правильно класифікованих як негативні; False Positive (FP), також відомі як Помилка I роду, – це кількість об'єктів, помилково класифікованих як позитивні; False Negative (FN), або Помилка II роду, – це кількість об'єктів, помилково класифікованих як негативні.

Accuracy – це частка правильно класифікованих об'єктів від загальної кількості об'єктів. Визначається за формулою (2.4) [14]:

$$\text{accuracy} = \frac{TP+TN}{P+N}, \tag{2.4}$$

де *TP* – це кількість об'єктів, правильно класифікованих як позитивні;

TN – кількість об'єктів, правильно класифікованих як негативні;

P – загальна кількість позитивних екземплярів;

N – загальна кількість негативних екземплярів.

Precision – Частка об'єктів, які дійсно є позитивними, серед усіх об'єктів, які модель класифікувала як позитивні. Визначається за формулою (2.5) [14]:

$$\text{precision} = \frac{TP}{TP+FP}, \tag{2.5}$$

де *TP* – це кількість об'єктів, правильно класифікованих як позитивні;

FP – кількість об'єктів, помилково класифікованих як позитивні.

Recall – Частка об'єктів, які дійсно є позитивними, серед усіх об'єктів, які модель класифікувала як позитивні.

Визначається за формулою (2.6) [14]:

$$\text{recall} = \frac{TP}{P}, \tag{2.6}$$

де *TP* – це кількість об'єктів, правильно класифікованих як позитивні;

P – загальна кількість позитивних екземплярів.

F-measure – середнє між влучністю та повнотою. Визначається за формулою (2.7) [14]:

$$\text{F-measure} = \frac{2}{1/\text{precition} + 1/\text{recall}}, \tag{2.7}$$

Графік ROC, візуалізований на рисунку 2.6, є інструментом для графічного представлення та аналізу діагностичної здатності бінарних класифікаторів. Даний графік будується у двовимірному просторі, де по осі Y відкладається частка істинно позитивних спрацювань (TPR), а по осі X – частка хибно позитивних спрацювань (FPR) [14].

При цьому, показник TPR розраховується за тією ж формулою, що й повнота. Натомість, FPR визначається як відношення кількості хибно позитивних результатів (FP) до загальної кількості фактично негативних екземплярів (N). Оскільки кожна точка на графіку відповідає конкретному значенню порогу класифікації моделі, вона візуалізує компроміс між прагненням до високого TPR та низького FPR.

Крива ROC – це двовимірне зображення ефективності класифікатора. Для порівняння класифікаторів їхню ефективність часто зводять до єдиного скалярного значення – площі під ROC-кривою, скорочено AUC, що зображено на рисунку 2.7.

Оскільки AUC є частиною площі одиничного квадрата, його значення завжди буде між 0 та 1.0. Випадкове вгадування дає діагональну лінію між (0,0) та (1,1), яка має площу 0.5, тому жоден реалістичний класифікатор не повинен мати AUC менше 0.5 [14].

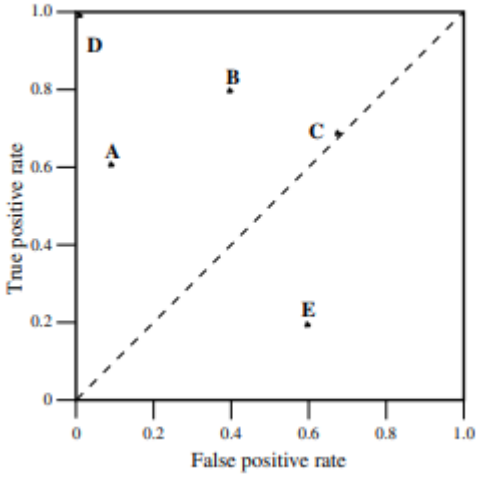

Рисунок 2.6 – Базовий графік ROC, що показує п'ять дискретних класифікаторів

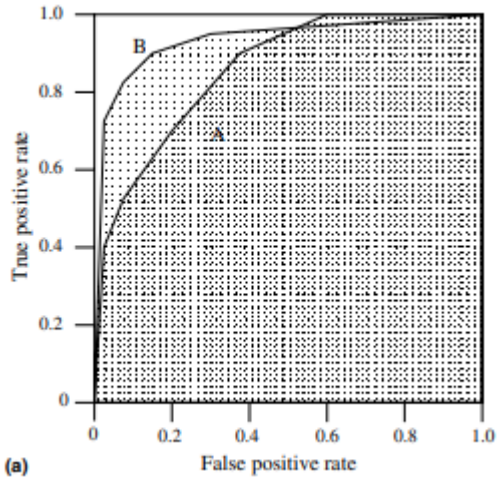

Рисунок 2.7 – Площа під двома ROC-кривими

Вибір адекватної метрики для оцінки є критично важливим, особливо при роботі з незбалансованими наборами даних. Незбалансований датасет – це датасет, у якому кількість об'єктів одного класу значно переважає кількість об'єктів іншого класу.

У таких випадках метрика Accuracy може бути оманливою. Наприклад, якщо 95% новин у датасеті є правдивими, модель, яка просто класифікує всі новини як правдиві, матиме точність 95%, хоча вона

абсолютно нездатна виявляти фейки. Тому для незбалансованих датасетів більш інформативними є Precision, Recall, F1-score та AUC. Наприклад, у задачі виявлення фейкових новин може бути критично важливим мінімізувати кількість пропущених фейків (FN), що вимагає високого Recall, навіть якщо це призведе до деякого зниження Precision (збільшення FP – правдиві новини, помилково марковані як фейки). Або ж, навпаки, пріоритетом може бути уникнення помилкового маркування правдивих новин як фейкових, що вимагає високої Precision. F1-score допомагає знайти компроміс, а AUC дає загальну оцінку продуктивності моделі незалежно від обраного порогу класифікації.

2.4 Архітектура трансформер та механізм уваги

Поява архітектури тансформер вважається революційним кроком у галузі обробки природної мови та багатьох інших сферах штучного інтелекту. Основною інновацією стало те, що ця архітектура використовує механізм уваги замість рекурентних і згорткових шарів, які раныше використовувалися для обробки послідовностей. Це дозволило досягти значного прогресу у вирішенні різноманітних завдань NLP, таких як машинний переклад, генерація тексту та, зокрема, виявлення фейкових новин.

Передумовою для такого прориву стало вдосконалення та переосмислення механізму уваги. До появи трансформерів основними архітектурами для обробки послідовностей були рекурентні нейронні мережі, зокрема їх більш просунуті варіанти, такі як Long Short-Term Memory та Gated Recurrent Unit. Ці моделі обробляють вхідну послідовність покроково, зберігаючи прихований стан, який передається на наступний крок. Однак, RNN мають суттєві обмеження, пов'язані з труднощами у моделюванні довгострокових залежностей та обмеженими можливостями паралелізації обчислень. Механізм уваги, вперше запропонований для

моделей «послідовність-до-послідовності», імітує людську здатність фокусуватися на певних частинах інформації під час виконання завдання. Увага допомагає моделі динамічно виявляти важливість різних частин вхідної послідовності при генерації кожного елемента вихідної послідовності, дозволяючи декодеру на кожному кроці генерації звертатися до всіх прихованих станів кодера і визначати найбільш релевантні з них [15]. Це досягається шляхом обчислення ваг уваги для кожного елемента вхідної послідовності.

Існувало декілька різновидів механізму уваги в контексті Sequence-to-Sequence (seq2seq) моделей розроблених для подолання обмежень традиційного підходу з кодуванням всієї вхідної послідовності в один вектор фіксованої довжини. Зокрема увага Баданау, що визначає ваги за допомогою нейромережі, яка на основі прихованих станів кодера і декодера динамічно обирає релевантні частини вхідної послідовності. Іншим поширеним підходом є увага Луонга, яка обчислює ваги на основі скалярного добутку між поточним прихованим станом декодера та станами кодера, що робить цей підхід простішим та обчислювально ефективнішим. Ці механізми уваги значно покращили якість seq2seq моделей, особливо для довгих послідовностей, але їхній потенціал був повністю розкритий та став головним елементом саме в архітектурі трансформер, що ілюструє взаємодію кодера та декодера, зображено на рисунку 2.8.

Основним компонентом моделі трансформера є самоувага. На відміну від уваги в класичних seq2seq моделях, самоувага дозволяє кожному слову в послідовності «дивитися» на інші слова в послідовності для обчислення власного контекстуального представлення.

Основна концепція самоуваги полягає в тому, що кожне слово в реченні взаємодіє з усіма іншими словами, щоб визначити, які слова є найбільш важливими для розуміння його власного значення в даному контексті. Для реалізації цього, для кожного вхідного токена генеруються три вектори: Query, Key та Value. Ці вектори отримуються шляхом

множення вхідного ембедингу на три окремі матриці ваг WQ, WK, WV, які навчаються в процесі тренування моделі. Query представляє поточне слово, для якого обчислюється увага. Key представляє всі слова в послідовності, з якими порівнюється поточне слово. Value представляє інформацію, яку необхідно отримати від слів, на які звернули увагу.

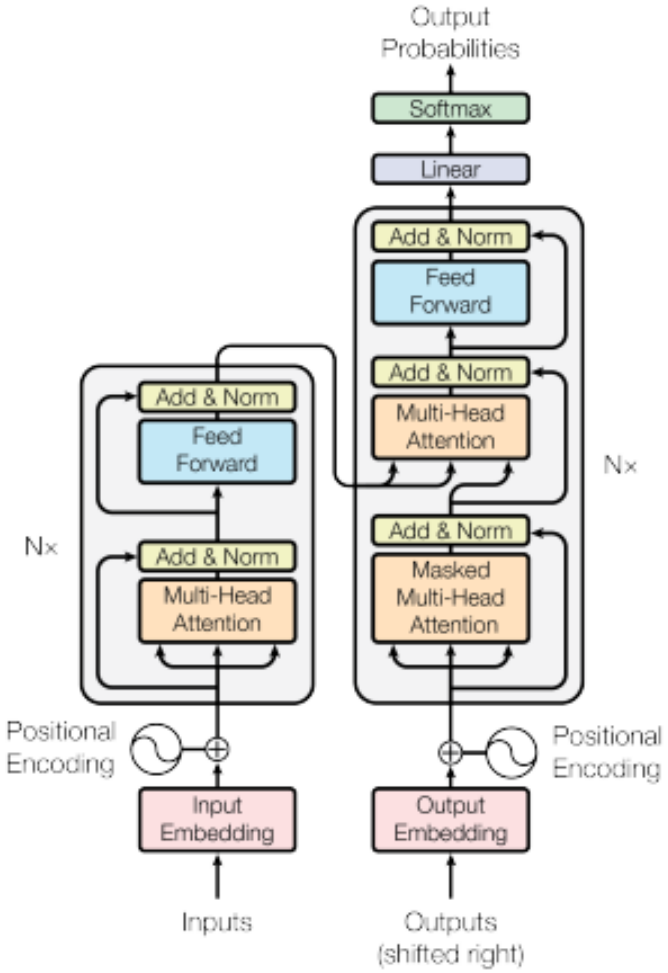

Рисунок 2.8 – Архітектура моделі Трансформер

Обчислення уваги відбувається за допомогою масштабованого скалярного добутку (Scaled Dot-Product Attention) за формулою (2.5) [2]:

$$\text{Attencion}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.5}$$

де Q – це матриця запитів;

K – матриця ключів;

V – матриця значень значення

K^T – транспонована матриця ключів;

d_k – розмірність векторів ключів.

Замість того, щоб застосовувати один механізм самоуваги, трансформер використовує Multi-Head Attention. Це дозволяє моделі одночасно фокусуватися на різних аспектах інформації. Реалізується це шляхом паралельного виконання декількох шарів самоуваги, де для кожного шару Q, K, V лінійно проектуються h разів з різними, навченими матрицями проекцій. Отримані h вихідних векторів об’єднуються, а потім знову лінійно проектуються для отримання фінального вихідного вектора.

Оскільки трансформер обробляє всі токени паралельно і не використовує рекурентні зв'язки, він за своєю природою не враховує порядок слів. Тому необхідним є позиційне кодування, яке додає інформацію про відносну або абсолютну позицію токенів у послідовності.

Структура кодера трансформера, що відповідає за обробку вхідної послідовності та створення її багатовимірного представлення, складається зі стеку N однакових блоків. Кожен блок кодера має два основні підшари: згадану Multi-Head Self-Attention та Position-wise Feed-Forward Network (FFN). FFN – це повнозв'язна мережа прямого поширення, яка застосовується до кожного позиційного вектора окремо та ідентично, і зазвичай складається з двох лінійних шарів з функцією активації Rectified Linear Unit (ReLU) між ними.

Функція ReLU вносить нелінійність у FFN, дозволяючи моделювати складніші перетворення даних, і є обчислювально ефективною, що сприяло її широкому застосуванню в глибоких нейронних мережах. Навколо кожного з двох підшарів використовується залишкове з'єднання (Residual Connection), яке дозволяє градієнтам легше поширюватися через глибокі мережі та полегшує навчання ідентичностних відображень [16]. Після

залишкового з'єднання застосовується шар нормалізації (Layer Normalization), що стабілізує динаміку навчання шляхом нормалізації активацій всередині кожного шару для кожного окремого прикладу даних [17]. Декілька блоків кодера складаються в стек, де вихід одного блоку стає входом для наступного, дозволяючи моделі навчатися все більш складним представленням.

Повна архітектура трансформера також включає структуру декодера, яка відповідає за генерацію вихідної послідовності, хоча для задач класифікації тексту, як у даній роботі, часто використовується лише кодерна частина. Декодер також складається зі стеку N однакових блоків. Кожен блок декодера, окрім двох підшарів, аналогічних тим, що є в кодері, має третій підшар.

Відмінності від кодера полягають у тому, що перший підшар самоуваги в декодері є маскованим для запобігання «підгляданню» в майбутні токени під час генерації. Також додається другий підшар уваги, де Запити (Q) надходять від попереднього шару декодера, а Ключі (K) та Значення (V) – від виходу кодера, дозволяючи декодеру враховувати всю вхідну послідовність. Навколо кожного з трьох підшарів декодера також використовуються залишкові з'єднання та шари нормалізації.

Feed-Forward Networks всередині блоків кодера та декодера додають нелінійність та дозволяють моделі навчатися більш складним перетворенням представлень, тоді як шари нормалізації та залишкові з'єднання відіграють критичну роль у стабілізації навчання глибоких мереж.

Архітектура трансформер продемонструвала значні переваги над традиційними RNN та CNN у вирішенні багатьох NLP задач. По-перше, це краща паралелізація обчислень, оскільки всі токени можуть оброблятися одночасно. По-друге, ефективніше моделювання довгострокових залежностей, оскільки шлях між будь-якими двома токенами в механізмі самоуваги має постійну довжину (O(1)). По-третє, трансформери встановили нові рекорди по показнику якості на широкому спектрі задач,

таких як машинний переклад, аналіз тональності, узагальнення тексту та класифікацію тексту.

Таким чином, архітектура трансформер стала основою для сучасних моделей обробки природної мови завдяки використанню механізму уваги, особливо самоуваги та можливостей паралелізації. Її здатність ефективно моделювати контекстуальні залежності в тексті робить її перспективною для розробки програм, які виявдяють фейкові новини.

2.5 Попередньо навчені трансформерні моделі

Попередньо навчені трансформерні моделі здійснили революцію в галузі NLP, продемонструвавши видатні результати на широкому спектрі завдань. Їх успіх значною мірою зумовлений концепцією трансферного навчання та інноваційними методами попереднього навчання на великих текстових даних. У контексті виявлення фейкових новин, ці моделі пропонують потужні інструменти для аналізу текстового змісту та виявлення ознак дезінформації.

2.5.1 Концепція трансферного навчання в NLP

Трансферне навчання в NLP базується на ідеї використання знань, отриманих під час розв'язання однієї задачі, зазвичай, загальної задачі моделювання мови на великому нерозміченому датасеті текстів, для покращення ефективності розв'язання іншої, більш специфічної задачі [18]. Замість навчання моделі з нуля на відносно невеликому цільовому наборі даних, трансферне навчання дозволяє ініціалізувати модель, що вже інкапсулює знання про структуру мови, синтаксис, семантику та певні загальні знання.

Переваги трансферного навчання в NLP є суттєвими. По-перше, воно забезпечує покращення якості на задачах з обмеженими даними. Для

багатьох специфічних завдань, таких як виявлення фейкових новин на певну тематику або в певному регіоні, доступна кількість розмічених даних може бути невеликою. Попередньо навчені моделі, що бачили мільярди слів, можуть ефективніше працювати з такими обмеженими наборами, оскільки вони вже мають фундаментальне «розуміння» мови [19].

По-друге, трансферне навчання сприяє скороченню часу навчання, оскільки модель вже має початкові знання, і процес її адаптації до цільової задачі потребує значно менше часу та обчислювальних ресурсів порівняно з навчанням складної моделі з випадковою ініціалізацією ваг. По-третє, воно призводить до кращої генералізації: моделі, попередньо навчені на різноманітних даних, мають тенденцію до кращої генералізації, тобто вони краще працюють на нових, раніше не бачених даних.

Головну роль у цьому процесі відіграють два етапи. Перший етап це pre-training – процес навчання моделі на великих, зазвичай нерозмічених, наборах даних з метою вивчення загальних ознак та представлень цих даних. Попереднє навчання допомагає моделі сформувати корисні початкові ваги, що відображають структуру даних, і слугує формою регуляризації, що покращує подальшу оптимізацію [20].

На практиці, на цьому етапі модель навчається на великому загальному текстовому датасеті (наприклад, Wikipedia, BookCorpus) щоб виконувати одну або декілька самоконтрольованих (self-supervised) задач, таких як моделювання мови або передбачення наступного речення, з метою навчити модель глибоким представленням слів та контексту.

Наступним етапом йде fine-tuning. Це процес адаптації знань, отриманих моделлю на попередньому етапі, до конкретної цільової задачі. На цьому етапі попередньо навчена модель донавчається на меншому, специфічному для цільової задачі, розміченому наборі даних. Це зазвичай включає додавання одного або декількох специфічних для задачі шарів поверх попередньо навченої моделі та подальше навчання всієї моделі, або її частини, з використанням цих розмічених даних.

2.5.2 Методи попереднього навчання трансформерів

Ефективність попередньо навчених трансформерів значною мірою залежить від стратегій, використаних під час їх навчання. Основною задачею є моделювання мови (Language Modeling, LM), яке змушує модель вивчати статистичні закономірності та семантичні зв'язки в тексті. Одним з найуспішніших підходів є Masked Language Model (MLM), популяризований моделлю BERT [21].

Принцип роботи MLM полягає в тому, що під час навчання випадковим чином маскується певний відсоток токенів у вхідній послідовності (зазвичай замінюються спеціальним токеном [MASK], рідше – випадковим токеном або залишаються без змін), а завдання моделі – передбачити оригінальні значення цих замаскованих токенів на основі контексту з обох боків. Це дозволяє моделі вивчати глибокі двонапрямкові контекстуальні залежності. MLM використовується в таких відомих моделях, як BERT та RoBERTa.

Іншим важливим методом є Permutation Language Model (PLM), представлений у моделі XLNet [22]. PLM є авторегресійним підходом, що навчає модель передбачати токени в послідовності для всіх можливих перестановок порядку токенів, що дозволяє XLNet захоплювати двонапрямковий контекст, зберігаючи авторегресійну природу навчання та уникаючи деяких недоліків MLM.

Історично, в BERT також використовувалася задача Next Sentence Prediction (NSP), метою якої було навчити модель розуміти зв'язки між реченнями шляхом передбачення, чи є одне речення логічним продовженням іншого [21]. Однак, подальші дослідження, зокрема при розробці RoBERTa, показали, що ця задача може бути недостатньо ефективною, тому в багатьох пізніших моделях від неї відмовилися або модифікували [23].

2.5.3 Огляд ключових трансформерних моделей для задачі класифікації новин

Серед великої кількості трансформерних архітектур, для задачі класифікації новин, зокрема виявлення фейків, особливий інтерес становлять такі моделі:

– BERT;

– RoBERTa;

– XLNet.

BERT.

BERT, представлена командою Google AI Language у 2018 році, стала проривною архітектурою. Вона базується на кодувальній частині трансформера з шарами багатошарової уваги та повнозв'язними мережами. Попереднє навчання BERT здійснюється на двох задачах: Masked Language Model (MLM), що навчає модель передбачати випадково замасковані токени в тексті, та Next Sentence Prediction (NSP), спрямованій на розуміння взаємозв'язків між реченнями [21]. Для задач класифікації тексту, на початок кожного вхідного речення додається спеціальний токен [CLS], фінальне приховане представлення якого використовується як агреговане представлення всієї послідовності. Існують різні версії, такі як BERT-base (близько 110 млн параметрів) та BERT-large (близько 340 млн параметрів), а також варіанти cased (з урахуванням регістру) та uncased (без урахування регістру), вибір яких може впливати на результати.

RoBERTa.

RoBERTa, представлена Facebook AI у 2019 році, є оптимізацією підходу до попереднього навчання BERT [23]. Основні відмінності та покращення RoBERTa від BERT включають декілька аспектів. По-перше, впроваджено динамічне маскування: замість статичного маскування, RoBERTa застосовує маскування динамічно під час кожного навчального кроку. По-друге, було здійснено відмову від задачі Next Sentence

Prediction (NSP). По-третє, RoBERTa була навчена на значно більших обсягах даних (близько 160 ГБ тексту, включаючи CC-News). Нарешті, модель навчалася з більшими розмірами батчів та протягом довшого часу. Завдяки цим оптимізаціям, RoBERTa часто демонструє кращі результати порівняно з BERT.

XLNet.

XLNet, представлена Carnegie Mellon University та Google AI Brain Team у 2019 році, пропонує авторегресійний підхід з використанням Permutation Language Modeling (PLM), що дозволяє моделювати двонапрямковий контекст без артефактів маскування, характерних для BERT [22]. Ця модель також інтегрує ідеї з Transformer-XL для обробки довших послідовностей. Потенційні переваги XLNet, такі як краще моделювання довготривалих залежностей та двонапрямкового контексту, можуть бути корисними для виявлення складних форм дезінформації. Здатність розуміти контекст на глибшому рівні допомогає відрізнити тонкі маніпуляції в тексті.

3 РОЗРОБКА ДОДАТКУ

3.1 Навчання та оцінка моделей

Основним етапом розробки додатку являється процес навчання трансформерних моделей. Цей процес було розділено на кілька послідовних етапів: підготовка та попередня обробка даних, конфігурація та навчання моделей BERT, RoBERTa та XLNet, а також фінальна оцінка їхньої продуктивності на незалежній тестовій вибірці.

Навчання моделей відбувалось на набору даних, з платформи Kaggle, який складається з двох файлів: Fake.csv та True.csv [24]. Перший файл містить набор новинних статей, які класифіковані як фейкові, а другий – правдиві статті. Кожен запис у файлах містить заголовок та основний текст. Для того, щоб надати моделям максимально повний контекст для аналізу, було прийнято рішення об'єднати ці два поля в єдиний текстовий фрагмент full_text. Такий підхід дозволяє моделі одночасно аналізувати як стилістичні особливості заголовка, який часто містить клікбейтні елементи, так і зміст тексту.

Після об'єднання даних було проведено їх маркування: новинам з файлу Fake.csv було присвоєно мітку 1, а новинам з True.csv — мітку 0. Оскільки повний обсяг даних є надзвичайно великим і потребує значних обчислювальних ресурсів для навчання, для експериментів було використано випадкову репрезентативну вибірку.

Фінальним кроком підготовки даних являється їх розподіл на тренувальний та тестовий набори за допомогою функції train_test_split з бібліотеки scikit-learn. Для забезпечення об'єктивної оцінки було виділено 30% даних для тестування, які не використовувалися в процесі навчання. Важливим параметром при розподілі було використання стратифікації (stratify=labels). Цей метод гарантує, що співвідношення класів (фейкових та правдивих новин) у тренувальній та тестовій вибірках

буде ідентичним до співвідношення в початковому наборі даних. Це допомагає уникнути упередженості моделі та отримання надійних метрик її продуктивності.

Для ефективної подачі даних у трансформерні моделі в середовищі PyTorch було реалізовано спеціалізований клас NewsDataset, який успадковує torch.utils.data.Dataset. Цей клас інкапсулює логіку токенізації та перетворення тексту в числові тензори. Основним методом цього класу є __getitem__, який виконується для кожного окремого зразка даних. Його реалізація показана у лістингу 3.1.

Лістінг 3.1 – Реалізація методу __getitem__ у класі NewsDataset

```
def __getitem__(self, idx):
    text = str(self.texts[idx])
    label = self.labels[idx]
    encoding = self.tokenizer(
        text,
        add_special_tokens=True,
        max_length=self.max_length,
        padding='max_length',
        truncation=True,
        return_tensors='pt'
    )
    return {
        'input_ids': encoding['input_ids'].flatten(),
        'attention_mask':
encoding['attention_mask'].flatten(),
        'labels': torch.tensor(label, dtype=torch.long)
    }
```

У цьому методі відбувається процес токенізації. Параметр add_special_tokens=True додає спеціальні токени, такі як [CLS] та [SEP], які необхідні для класифікації. Для уніфікації довжини вхідних послідовностей використовуються параметри max_length, padding='max_length' (доповнення

коротких текстів нульовими токенами) та truncation=True (обрізання занадто довгих текстів). На виході метод повертає словник з тензорами input_ids (індекси токенів), attention_mask (маска уваги, що вказує моделі, які токени ігнорувати) та labels (мітка класу).

Сам процес навчання моделей відбувається як донавчання попередньо навчених моделей bert-base-uncased, roberta-base та xlnet-base-cased. Тренувальний цикл, реалізований у функції train_model, використовує оптимізатор AdamW та функцію втрат CrossEntropyLoss. Після кожної епохи навчання виконувався етап валідації, за результатами якого зберігається версія моделі з найменшим значенням валідаційної втрати, що дозволяє запобігти перенавчанню та обрати оптимальну ітерацію моделі.

Фінальна оцінка навчених моделей проводилась на відкладеній тестовій вибірці. Цей процес було реалізовано у файлі test_all_models.py, який послідовно завантажував кожну навчену модель і розраховував для них набір основних метрик класифікації: загальну точність (Accuracy), точність для класу (Precision), повноту (Recall) та F1-міру (F1-Score). Результати тестування візуалізовано у вигляді матриць помилок, представлені у таблиці 3.1.

Таблиця 3.1 – Результати тестування всіх моделей

	BERT	RoBERTa	XLNet
Accuracy	0.9989	0.9775	0.9975
Precision (True)	0.9986	0.9551	0.955
Recall (True)	0.9991	0.9999	0.9999
F1 (True)	0.9988	0.977	0.9769
Precision (False)	0.9991	0.9999	0.9999
Recall (False)	0.9987	0.9572	0.957
F1 (False)	0.9989	0.978	0.978

Матриця плутанини, яку можна побачити на рисунку 3.1, для моделі BERT демонструє найбільш збалансовану продуктивність серед трьох моделей. Модель правильно ідентифікувала 21397 правдивих новин (TN) та 23450 фейкових новин (TP). Кількість помилок першого роду (FP) є дуже низькою і становить 20 випадків. Це означає, що BERT лише у 20 випадках помилково позначив достовірну новину як фейк, що свідчить про високу надійність її прогнозів щодо правдивих новин. Водночас, кількість помилок другого роду (FN) також є мінімальною – 31 випадок. Це вказує на те, що модель ефективно виявляє дезінформацію і рідко пропускає фейкові статті. Низькі показники обох типів говорить про те, що BERT добре справляється як із завданням ідентифікації фейків, так і з уникненням помилок що до визначення правдивих новин.

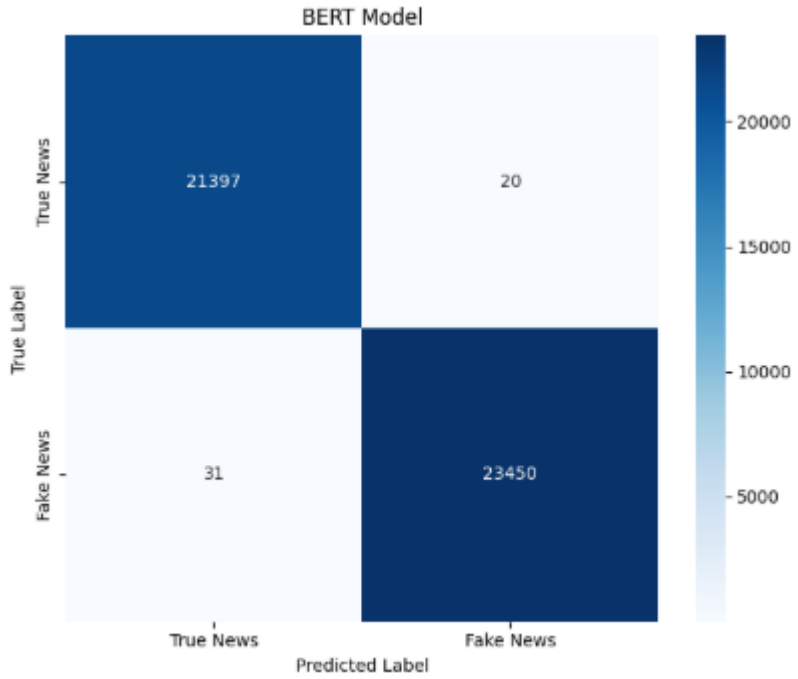

Рисунок 3.1 – Матриця плутанини для моделі BERT

Поведінка моделі RoBERTa, як видно з її матриці помилок, зображеної на рисунку 3.2, кардинально відрізняється. Модель демонструє майже ідеальну здатність розпізнавати правдиві новини: вона правильно

класифікувала 21414 таких новин, допустивши лише 3 помилки першого роду (FP). Це високий показник, який свідчить про те, що якщо RoBERTa ідентифікує новину як правдиву, то цьому прогнозу можна довіряти з дуже високою ймовірністю.

Однак, кількість помилок другого роду (FN) велика та становить 1006 випадків. Це означає, що модель не змогла розпізнати більше тисячі фейкових новин, помилково класифікувавши їх як правдиві. Такий результат свідчить про явну упередженість моделі: вона налаштована на те, щоб за будь-яку ціну уникнути помилкового маркування правдивої новини як фейкової. Її точність (Precision) для класу «фейк» є надзвичайно високою (близько 99.98%), але повнота (Recall) є значно нижчою (близько 95.7%).

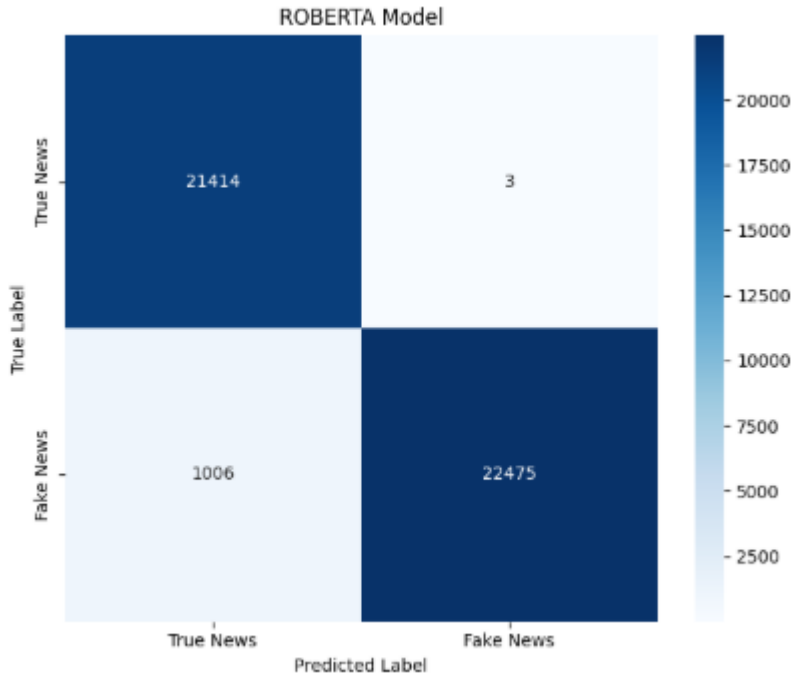

Рисунок 3.2 – Матриця плутанини для моделі BoBERTa

Матриця плутанини XLNet, яка зображена на рисунку 3.3, дуже схожа на матрицю RoBERTa. Кількість помилок першого роду є найнижчою серед усіх моделей – всього 2 випадки. Це робить XLNet найнадійнішою моделлю

для ідентифікації правдивих новин. Якщо ця модель каже, що новина правдива, вона практично ніколи не помиляється. Разом з тим, вона має найвищий показник помилок другого роду – 1010 випадків. Це означає, що XLNet є найменш чутливою до виявлення фейків моделлю з трьох, пропускаючи найбільшу кількість дезінформації. Таким чином, XLNet є моделлю з найвищою точністю (Precision ≈ 99.99%), але найнижчою повнотою (Recall ≈ 95.69%). Вона є максимально "консервативною" і маркує новину як фейк лише у випадку абсолютної впевненості.

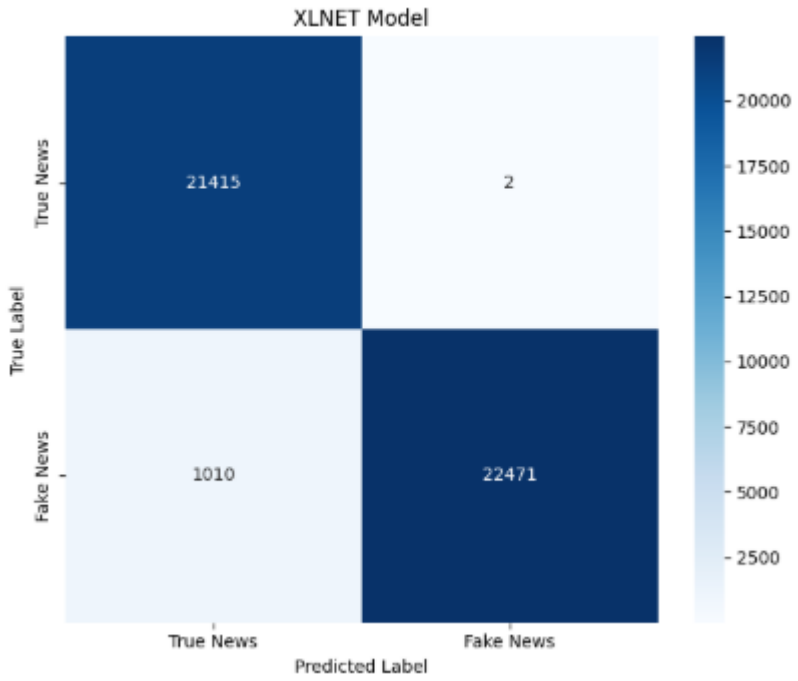

Рисунок 3.3 – Матриця плутанини для моделі XLNet

На основі проведеного порівняльного аналізу, модель BERT була найефективнішою. RoBERTa та XLNet показали дуже схожі результати, головною перевагою яких є надзвичайно висока точність у визначенні правдивих новин. Однак, ця перевага досягається ціною значної кількості пропущених фейків, що свідчить про їхню схильність класифікувати неоднозначні тексти як правдиві.

3.2 Реалізація серверної частини

Серверна частина є архітектурним ядром розробленого додатку. Вона інкапсулює всю обчислювальну логіку, пов'язану із завантаженням трансформерних моделей, обробкою даних та виконанням аналізу. Для її реалізації було обрано сучасний асинхронний веб-фреймворк FastAPI

Архітектура API була спроектована за принципами REST, з логічним групуванням ендпоінтів за їх функціональним призначенням, що забезпечує інтуїтивно зрозумілу та легко розширювану структуру.

Для забезпечення надійності та коректності обробки даних, які надходять від користувачів, було впроваджено строгу валідацію на основі Pydantic-моделей. У файлі api.py визначено спеціальні класи, які успадковують BaseModel, такі як NewsRequest, BatchRequest та FileAnalysisResponse. Ці моделі даних чітко описують очікувану структуру, типи даних, обмеження та значення за замовчуванням для кожного ендпоінту. Приклад такої моделі наведено у лістингу 3.2.

Лістинг 3.2 – Pydantic-модель для валідації вхідного запиту

```
class NewsRequest(BaseModel):
    text: str = Field(..., description="Текст новини для
аналізу", min_length=5)
    model_name: str = Field(default="BERT",
description="Назва моделі для аналізу")
    @validator('model_name')
    def validate_model_name(cls, v):
        allowed_models = ['BERT', 'RoBERTa', 'XLNet']
        if v not in allowed_models:
            raise ValueError(f'Модель має бути одною з:
{", ".join(allowed_models)}')
        return v
```

Це дозволяє FastAPI автоматично перевіряти всі вхідні дані, повертаючи клієнту змістовну помилку у випадку невідповідності.

Основний функціонал системи реалізовано через набір API-методів для аналізу. Для обробки одиночних запитів застосовується маршрут POST /analyze, який приймає один текстовий фрагмент. З метою підвищення ефективності при роботі з великими обсягами інформації було додано можливість пакетної обробки через POST /analyze/batch, який дозволяє передавати список текстів одним запитом, зменшуючи накладні витрати на HTTP-з'єднання. Крім того, система надає діагностичний метод POST /analyze/compare, який виконує аналіз одного тексту всіма доступними моделями одночасно, повертаючи консенсусне рішення для порівняльної оцінки.

Важливим архітектурним рішенням є управління життєвим циклом додатку за допомогою контекстного менеджера lifespan. Цей механізм FastAPI дозволяє виконувати код на етапах запуску та зупинки сервера. У цьому додатку він використовується для виклику функції load_models один раз при старті. Такий підхід гарантує, що навіть у випадку помилки завантаження однієї з моделей, наприклад, через відсутність файлу ваг або пошкодження даних, сервер продовжить свою роботу, надаючи доступ до решти успішно завантажених моделей.

Центральна логіка класифікації тексту інкапсульована у функції predict_news. Процес прогнозування складається з кількох послідовних кроків. Спочатку вхідний текст проходить через відповідний токенізатор, де він перетворюється на числовий тензор з максимальною довжиною 512 токенів. Важливо, що процес отримання прогнозу виконується в контексті with torch.no_grad(), що вимикає обчислення градієнтів і значно прискорює отримання результату. На виході модель генерує логіти, до яких застосовується функція активації softmax для отримання ймовірнісного розподілу між класами «правдива новина» та «фейкова новина». Клас з

найвищою ймовірністю і є результатом прогнозу, а сама ймовірність слугує показником впевненості моделі.

Однією з найважливіших частин розробленого бекенду є розширений механізм пояснення результатів. Після отримання прогнозу система викликає функцію generate_explanation, яка, в свою чергу, використовує спеціалізований модуль ImprovedNewsAnalyzer. Цей модуль проводить аналіз ваг уваги з кількох останніх шарів трансформера. На відміну від базового підходу, який аналізує лише останній шар, даний метод усереднює ваги з трьох останніх шарів, що дозволяє отримати більш стабільну та змістовну картину того, які саме слова чи фрази мали найбільший вплив на фінальне рішення моделі. Після цього проводиться евристичний аналіз тексту на наявність патернів, характерних для дезінформації. Модуль ImprovedNewsAnalyzer містить словники емоційно забарвленої лексики, слів, що привертають увагу, та конспірологічних термінів. Він також використовує регулярні вирази для пошуку таких ознак, як надмірне використання великих літер або знаків оклику.

Комбінація цих двох підходів дозволяє генерувати деталізований звіт, де підозрілі фрагменти тексту візуально підсвічуються, а для кожного підсвічування надається пояснення. Більше того, система здатна виявляти контрадикції у власній роботі, наприклад, якщо модель з високою впевненістю класифікує текст як фейк, але ані аналіз уваги, ані аналіз патернів не знаходять жодних підозрілих ознак, система генерує попередження про можливу помилку. Це є елементом саморефлексії та значно підвищує прозорість та довіру до результатів роботи системи.

Для розширення функціональних можливостей було реалізовано обробку файлів різних форматів. За цю логіку відповідає окремий файл file_processors.py. Цей модуль здатен вилучати текст із документів PDF, Microsoft Word, Excel, PowerPoint та інших форматів. Система динамічно перевіряє наявність необхідних бібліотек і активує підтримку тих чи інших форматів. Процес обробки файлу, представлений у функції process_file,

включає валідацію розміру, визначення типу файлу, вилучення та фінальне очищення тексту від артефактів. Така реалізація дозволяє системі гнучко адаптуватися до середовища розгортання та коректно обробляти широкий спектр вхідних даних.

3.3 Розробка вебінтерфесу

Клієнтська частина додатку для аналізу фейкових новин реалізована у вигляді одностороннкового вебдодатку, що складається з HTML структури, CSS стилів та JavaScript логіки. Основою вебсторінки є структура, написана мовою HTML5 з використанням сучасних семантичних елементів та адаптивного дизайну.

Основні структурні елементи сторінки розташовані в елементі <body> за принципом логічної ієрархії. У верхній центральній частині сторінки, розміщено заголовковий блок header який містить заголовок «Аналізатор фейкових новин», під яким знаходиться підзаголовок з поясненням функціональності: «Детекція фейкових новин за допомогою BERT, RoBERTa та XLNet» Це зображено у додатку А, рисунок А.1.

Центральну частину інтерфейсу займає контейнер основного контенту .main-content. Цей контейнер містить всі інтерактивні елементи додатку, організовані в систему вкладок для зручної навігації між різними режимами роботи.

У верхній частині інтерфейсу розташована панель статусу API, яка динамічно відображає стан підключення до серверної частини та кількість доступних моделей.

Навігаційна система реалізована у вигляді горизонтального меню вкладок, що дозволяє користувачу перемикатися між трьома основними режимами роботи додатку:

– одиночний аналіз – для аналізу окремих текстів новин;

– пакетний аналіз – для одночасного аналізу декількох текстів;

– аналіз файлів – для аналізу документів різних форматів.

Перша та основна вкладка додатку «Одиночний аналіз» призначена для аналізу окремих текстів новин. Вона містить основні елементи управління для ініціювання процесу аналізу. Користувачу надається можливість за допомогою випадаючого списку обрати одну з трьох доступних моделей – BERT, RoBERTa або XLNet, що зображено у додатку А, рисунок А.2. Цей елемент дозволяє порівнювати результати роботи різних архітектур. Основним елементом взаємодії є багаторядкове текстове поле, куди користувач вводить текст новини для подальшої обробки.

Друга вкладка додатку «Пакетний аналіз» призначена для одночасного аналізу декількох текстів. Вона містить аналогічний селектор моделі та розширене текстове поле з можливістю введення до 8 рядків тексту. Кожен рядок інтерпретується як окремий текст. Результати аналізу відображаються в спеціальному контейнері. Вони включають загальну статистику: кількість оброблених текстів, середню впевненість моделі та загальний час обробки. Ця вкладка зображена у додатку А, рисунок А.5 та рисунок А.6.

Третя вкладка додатку «Аналіз файлів» забезпечує аналіз документів різних форматів. Основним елементом цієї вкладки є поле завантаження файлів, яке підтримує вибір та обробку широкого спектру документів, включаючи PDF, Word, Excel, PowerPoint та інші. Цей елемент зображено у додатку А, рисунок А.7. Для зручності користувача під полем вибору динамічно відображається інформаційна панель, яка завантажується з сервера. Вона інформує про статус доступності бібліотек для обробки кожного конкретного формату, позначаючи їх зеленою галочкоюі або червоним хрестиком. Після аналізу, результати для файлів подаються в розширеному форматі, який, окрім вердикту, включає: назву файлу, його тип, розмір, довжину вилученого тексту та попередній перегляд вмісту.

Результати аналізу відображаються в динамічному блоці, який з'являється під кнопкою аналізу після отримання відповіді від сервера. Цей блок має три можливі стани:

– результат «Фейкова новина»;

– результат «Справжня новина»;

– повідомлення про помилку.

Одним із основних елементів інтерфейсу є система візуального аналізу пояснення рішень моделі, яка реалізована через підсвічування підозрілих частин тексту, що зображено у додатку А.3, А.4 та А.8. Ця функціональність, побудована на взаємодії CSS класів та JavaScript логіки та представлена у вигляді спеціального контейнера, який з'являється після основного результату аналізу. Він містить детальний розбір тексту, де система використовує чотири рівні підозрілості:

– дуже підозріло;

– підозріло;

– помірно підозріло;

– увага моделі.

Для повноцінної інтерпретації результатів візуалізації, під підсвіченим текстом розташовано два допоміжних інформаційних блоки. Перший – це легенда, яка пояснює значення кожного кольору підсвічування. Вона організована у вигляді горизонтального ряду елементів, де кожен містить кольорову плашку та текстовий опис з кількістю знайдених фрагментів відповідного рівня підозрілості. Другим елементом є панель статистики, яка відображає три основні метрики аналізу:

– загальну кількість підкреслених частин;

– загальну кількість слів у тексті;

– відсоток підозрілості.

Для забезпечення прозорості та стабільності взаємодії з користувачем, додаток має розвинену систему візуального зворотного зв'язку. Під час обробки запитів активуються індикатори завантаження, які реалізовані

через CSS keyframes та текстові повідомлення про поточну дію. Одночасно з цим кнопки управління автоматично блокуються, що запобігає повторним натисканням. Крім того, система має надійний механізм обробки помилок, який перехоплює збої мережі, серверні відповіді з помилками та помилки валідації, відображаючи їх у зрозумілому для користувача форматі.

Клієнтська логіка додатку, реалізована на JavaScript, організована навколо асинхронних функцій, які відповідають за взаємодію з серверним API та управління станом інтерфейсу. Ініціалізація додатку відбувається при завантаженні сторінки через обробник події DOMContentLoaded, який викликає функцію checkAPIStatus() для виконання GET-запиту до ендпоінту /health. Це дозволяє в реальному часі перевірити доступність серверної частини, отримати список завантажених моделей та відобразити актуальний статус у відповідній панелі.

Основні функції аналізу, такі як analyzeNews(), analyzeBatch() та analyzeFiles(), інкапсулюють повний цикл взаємодії з користувачем та сервером. Вони відповідають за збір та валідацію вхідних даних з HTML-форм, управління станом інтерфейсу, формування та надсилання POST-запитів до відповідних ендпоінтів API з даними у форматі JSON або FormData для файлів. Найскладнішою є система генерації пояснень, реалізована через функцію generateExplanationHTML(). Вона динамічно перетворює отриманий від сервера масив об'єктів про підсвічені частини тексту в інтерактивну HTML-розмітку, сортуючи фрагменти за позицією та застосовуючи відповідну стилізацію для візуалізації результатів аналізу.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено детальний аналіз предметної галузі, який включає дослідження сучасних проблеми поширення дезінформації, методів обробки природної мови та архітектури глибокого навчання.

Першим етапом стало теоретичне дослідження, в якому було проведено порівняння існуючих підходів до виявлення фейкових новин. Було розглянуто лінгвістичні методи, традиційні алгоритми машинного навчання та сучасні нейромережеві архітектури. Аналіз показав, що трансформерні моделі, такі як BERT, RoBERTa та XLNet, демонструють найвищу точність класифікації.

У практичній частині було реалізовано повноцінний веб-додаток для виявлення фейкових новин з можливістю генерації пояснень аналізу. Клієнтська частина представлена у вигляді зручного та інтуїтивно зрозумілого веб-інтерфейсу, реалізованого засобами HTML, CSS та JavaScript. Серверна частина створена на основі асинхронного фреймворку FastAPI з інтегрованими трансформерними моделями BERT, RoBERTa та XLNet. Архітектура сервера включає валідацію даних за допомогою Pydantic, ефективне управління життєвим циклом моделей та підтримку обробки файлів різних форматів. Також, реалізовано механізм пояснюваності результатів, який комбінує аналіз ваг уваги моделей з евристичним пошуком текстових патернів, що дозволяє візуалізувати та інтерпретувати рішення системи.

Таким чином, створений вебдодаток є завершеним та ефективним інструментом, що поєднує потужність сучасних моделей ШІ з прозорістю та орієнтованістю на користувача. Перспективи подальшого розвитку проєкту включають розширення мовної підтримки, впровадження механізмів безперервного навчання моделей для адаптації до нових методів дезінформації та інтеграцію з зовнішніми системами перевірки фактів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vosoughi S., Roy D., Aral S. The spread of true and false news online. *Science*. 2018. Т. 359, № 6380. С. 1146–1151. URL: https://doi.org/10.1126/science.aap9559 (дата звернення: 15.06.2025).

2. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention Is All You Need – 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (дата звернення: 15.06.2025).

3. XAI–Explainable artificial intelligence / D. Gunning та ін. *Science Robotics*. 2019. Т. 4, № 37. С. eaay7120. URL: https://doi.org/10.1126/scirobotics.aay7120 (дата звернення: 15.06.2025).

4. data.europa.eu. *The official portal for European data | data.europa.eu*. URL: https://data.europa.eu/data/datasets/s2183_464_eng?locale=en (дата звернення: 15.06.2025).

5. data.europa.eu. *The official portal for European data | data.europa.eu*. URL: https://data.europa.eu/data/datasets/s2183_464_eng?locale=en (дата звернення: 15.06.2025).

6. Eisenstein J. Introduction to Natural Language Processing. MIT Press, 2019. 536 с. (дата звернення: 15.06.2025).

7. Goldberg Y. Neural Network Methods for Natural Language Processing. Cham : Springer International Publishing, 2017. URL: https://doi.org/10.1007/978-3-031-02165-7 (дата звернення: 15.06.2025).

8. Martin J. H., Jurafsky D. Speech and Language Processing (3nd Edition). 3-тє вид. Prentice Hall, 2023. 944 с. (дата звернення: 15.06.2025).

9. Hongdan Z., Jiangde Y. Part-of-Speech Tagging Based on Maximum Entropy. *International Journal of Circuits, Systems and Signal Processing*. 2018. № 12. С. 483. (дата звернення: 15.06.2025).

10. Anala S. A Guide to Word Embeddings. *Medium*. URL: https://medium.com/data-science/a-guide-to-word-embeddings-8a23817ab60f (дата звернення: 15.06.2025).

11. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. Т. 521, № 7553. С. 436–444. URL: https://doi.org/10.1038/nature14539 (дата звернення: 15.06.2025).

12. Kim Y. Convolutional Neural Networks for Sentence Classification. 2014. URL: https://doi.org/10.48550/arXiv.1408.5882 (дата звернення: 15.06.2025).

13. Bengio Y., Courville A., Goodfellow I. Deep Learning. MIT Press, 2016. 800 с. (дата звернення: 15.06.2025).

14. Fawcett T. An introduction to ROC analysis. *Pattern Recognition Letters*. 2006. Т. 27, № 8. С. 861–874. URL: https://doi.org/10.1016/j.patrec.2005.10.010 (дата звернення: 16.06.2025).

15. Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473. 2014. URL: https://arxiv.org/pdf/1409.0473 (дата звернення: 16.06.2025).

16. Deep Residual Learning for Image Recognition / K. He та ін. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, м. Las Vegas, NV, USA, 27–30 черв. 2016 р. 2016. URL: https://doi.org/10.1109/cvpr.2016.90 (дата звернення: 16.06.2025).

17. Ba J. L., Kiros J. R., Hinton G. E. Layer Normalization. *arXiv preprint arXiv:1607.06450*. 2016. URL: https://doi.org/10.48550/arXiv.1607.06450 (дата звернення: 16.06.2025).

18. Pan S. J., Yang Q. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*. 2010. Т. 22, № 10. С. 1345–1359. URL: https://doi.org/10.1109/tkde.2009.191 (дата звернення: 16.06.2025).

19. Howard J., Ruder S. Universal Language Model Fine-tuning for Text Classification. *arXiv preprint arXiv:1801.06146*. 2018. URL: https://doi.org/10.48550/arXiv.1801.06146 (дата звернення: 16.06.2025).

20. Why Does Unsupervised Pre-training Help Deep Learning? / D. Erhan та ін. *Journal of Machine Learning Research*. 2010. Т. 11. С. 625–660. (дата звернення: 16.06.2025).

21. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin та ін. *arXiv preprint arXiv:1810.04805*. 2018. URL: https://doi.org/10.48550/arXiv.1810.04805 (дата звернення: 16.06.2025).

22. XLNet: Generalized Autoregressive Pretraining for Language Understanding / Z. Yang та ін. *arXiv preprint arXiv:1906.08237*. 2019. URL: https://doi.org/10.48550/arXiv.1906.08237 (дата звернення: 16.06.2025).

23. RoBERTa: A Robustly Optimized BERT Pretraining Approach / Y. Liu та ін. *arXiv preprint arXiv:1907.11692*. 2019. URL: https://doi.org/10.48550/arXiv.1907.11692 (дата звернення: 16.06.2025).

24. Fake News Detection Datasets. *Kaggle*. URL: https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets (дата звернення: 16.06.2025).