І небо, і землю, і воду вартують,
Ці герої з світанку до ночи працюють.
Завжди вони у серцях, воїни непокірні,
немов чарівники творять неймовірне.

Слава героям, що батьківщину стерегуть,
Що мужньо в бою, свободи клич несуть!
Відвагу, честь, здіймають мов крило,
і не зупинить їх ніяке там …

Дзвони благословляють на світанку,
Молитви летять за тих, хто в битві з ранку.
Хай мужність росте, не знає меж і краю,
На сторожі Вкраїни і всіх передолають!

Дякуєм, браття, за ваші дні і ночі,
За відвагу, за мир, за мову цю співучу.
Україна – мати, в гордості стоїть,
Слава захисникам вічно в серцях звучить!

Пам'ятаймо воїнів, впавших за наш край,
Вічна пам'ять героям, що дали нам свій рай.
Збройні сили міцні, гордо стоїм разом,
За свободу, за мир, за Україну нашу.

Присвячено ЗСУ.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

До захисту допущено:
Завідувач кафедри
_____Вадим МУХІН
«___»_____ 2023 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Інтелектуальні сервіс-орієнтовані розподілені обчислювання”
зі спеціальності 122 “Комп’ютерні науки”
на тему: “Автоматизоване виявлення фейкових новин за допомогою перехресної перевірки джерел з використанням машинного навчання”

Виконав:
студент IV курсу, групи ДА-92
Поплавський Владислав Олегович

Керівник:
кандидат технічних наук
Кислий Роман Володимирович

Консультант з економічного розділу:
доцент, к.е.н.
Рощина Надія Василівна

Рецензент:
д.т.н., професор кафедри ММСА
Кузнєцова Наталія Володимирівна

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2023

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут прикладного системного аналізу

Кафедра системного проектування

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 "Комп'ютерні науки"

Освітньо-професійна програма – "Інтелектуальні сервіс-орієнтовані розподілені обчислювання"

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Вадим МУХІН

«___»_____ 2023 р.

ЗАВДАННЯ

на дипломну роботу студенту

Поплавському Владиславу

1. Тема роботи «Автоматизоване виявлення фейкових новин за допомогою перехресної перевірки джерел з використанням машинного навчання», керівник роботи Кислий Роман Володимирович, кандидат технічних наук, затверджені наказом по університету від

«___»_____20__р. №_____

2. Термін подання студентом роботи – 10 червня 2023 р.

3. Вихідні дані до роботи:

- Теоретичні відомості з літератури
- Середовища розробки Visual Studio та Google Colab, Pyton, CUDA
- Документація на розробку мови програмування Python, бібліотек машинного навчання pytorch, tensorflow

4. Зміст роботи: Робота присвячена дослідженню можливостей сучасних моделей машинного навчання, їх покращення, отримання актуальних наборів даних, пошук інформаційних систем, які можуть допомогти з проблемою автоматизованого виявлення фейкових новин та інформації в інтернеті. Потрібно виконати наступні завдання:

1) Дослідити та описати проблему знаходження фейкових новин, її актуальність, та наявні методи її вирішення.

2) Оглянути існуючі рішення, дослідити можливості вирішення проблеми за допомогою сучасні методик обробки мов.

3) Розглянути наявні доступні набори даних, які можна використати для аналізу фейкових новин. Проаналізувати та підготувати дані для тренування та тестування.

4) Провести експеримент тренуючі різні моделі, проаналізувати результати, зробити оцінку моделей.

5) Описати обрану модель та підхід до збору і обробки даних.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): діаграми алгоритмів опису побудови рішень, скріншоти роботи моделей, їх метріки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В.		

7. Дата видачі завдання ___

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Прибуття студента на практику, оформлення і отримання перепусток	Перший тиждень	
2	Проведення інструктажу з техніки безпеки	Перший тиждень	
3	Проведення екскурсій по підприємству	Перший тиждень	
4	Дослідження та опис проблеми знаходження фейкових новин, її актуальність, та наявні методи її вирішення	Перший тиждень	
5	Огляд існуючих рішень, дослідження можливостей вирішення проблеми за допомогою сучасних методик обробки мов	Другий тиждень	
6	Огляд наявних доступних наборів даних, які можна використати для аналізу фейкових новин.Аналіз та підготовка даних для тренування та тестування	Третій тиждень	

7	Провести експеримент тренуючі різні моделі, проаналізувати результати, зробити оцінку моделей.	Четвертий тиждень	
8	Описати обрану модель та підхід до збору і обробки даних.	П’ятий тиждень	

Студент Поплавський В. О.

Керівник Кислий Р. В.

АНОТАЦІЯ

бакалаврської дипломної роботи Поплавського Владислава Олеговича на тему «Автоматизоване виявлення фейкових новин за допомогою перехресної перевірки джерел з використанням машинного навчання»

Розробка та поширення фейкової інформації та новин в інтернеті стали серйозною проблемою сучасного цифрового віку. У зв'язку з цим, дослідження автоматизованих систем для виявлення та боротьби з цією небезпекою стає все більш актуальним. Окреме занепокоєння викликає значний розвиток генеративних моделей, які за секунди здатні створювати велику кількість текстового та візуального цифрового контенту, що можуть створювати додаткові загрози та проблеми у виявленні такої інформації.

Метою даної роботи було дослідження можливостей сучасних моделей машинного навчання для використання їх у проблемі класифікації фейкової інформації в соціальних мережах, виявлення найкращих підходів, наборів даних, та їх покращення. Оскільки зараз інформація є одним з найпотужніших інструментів, який може використовуватися для низки різноманітних цілей, то ця проблема є і буде залишатися доволі актуальною

Результат роботи: дослідження наборів даних, аналіз, порівняння моделей машинного навчання, створення покращеної моделі для роботи в реальних умовах.

Загальний обсяг роботи 146 с., 45 рис., 9 таблиць, 1 додаток, 56 джерел. Ключові слова: PyTorch, TensorFlow, RNN, Transformer, NLP, датасет, класифікація, аналіз, фейкові новини.

ABSTRACT

bachelor's thesis of Poplavskyi Vladyslav Olegovych on
topic "Automated detection of fake news using cross-checking of sources using machine learning"

The development and distribution of fake information and news on the Internet has become a serious problem of the modern digital age. In this regard, the study of automated systems for detecting and combating this danger is becoming more and more relevant. Of particular concern is the significant development of generative models that can generate large amounts of textual and visual digital content in seconds, which can create additional threats and problems in detecting such information.

The purpose of this work was to investigate the possibilities of modern machine learning models for their use in the problem of classifying fake information in social networks, identifying the best approaches, data sets, and improving them. Since information is currently one of the most powerful tools that can be used for a number of different purposes, this problem is and will remain quite relevant.

The result of the work: research of data sets, analysis, comparison of machine learning models, creation of an improved model for work in real conditions.

The total volume of work is 146 pages, 45 figures, 9 tables, 1 appendix, 56 sources.

Keywords: PyTorch, TensorFlow, RNN, Transformer, NLP, dataset, classification, analysis, fake news.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

NLP (Natural Language Processing) - є галуззю комп'ютерної науки та штучного інтелекту, яка займається розумінням та обробкою людської мови в її природному форматі.

RNN (Recurrent Neural Network) - це клас нейромереж, які мають здатність до обробки та аналізу послідовних даних, зокрема тексту чи мови, шляхом збереження та використання попередньої інформації в контексті поточного вхідного сигналу.

BERT (Bidirectional Encoder Representations from Transformers) - це модель машинного навчання, що використовує трансформери для ефективного представлення тексту, здатна до контекстуального розуміння слів та фраз, заснована на великому масштабі навчання на нерозмічених корпусах текстів.

ML (Machine Learning) - це галузь штучного інтелекту, яка використовує алгоритми та моделі для навчання комп'ютерних систем на основі даних, з метою автоматичного виявлення закономірностей, роботи з прогнозами, прийняття рішень та розробки адаптивних систем, без прямого програмування.

Doc2Vec - це алгоритм у галузі обробки природної мови (NLP), який використовує нейронну мережу для представлення текстових документів у векторному просторі, де кожен документ має унікальний числовий вектор, що зберігає його семантику та контекст.

CUDA (Compute Unified Device Architecture) - це платформа та програмний інтерфейс, розроблені NVIDIA, які дозволяють використовувати можливості графічних процесорів (GPU) для прискорення обчислень.

GPU (Graphics Processing Unit) - це електронний пристрій, спеціально призначений для обробки графічних даних, який володіє великою кількістю паралельних обчислювальних одиниць і використовується для прискорення виконання обчислень у різних областях, включаючи машинне навчання та наукові обчислення.

CPU (Central Processing Unit) - це головний обчислювальний пристрій комп'ютера, відповідальний за виконання інструкцій програм та керування всіма обчислювальними процесами у системі.

ВСТУП

Зі збільшенням кількості інформації в Інтернеті стає дедалі складніше розрізняти правдиві новини від фейкових. Ця проблема не тільки збільшує кількість неправдивої інформації, що поширюється в мережі, але і може призвести до серйозних наслідків, таких як спричинення паніки, руйнування довіри до ЗМІ та інших джерел інформації. Крім того, що фейкові новини можуть призвести до поширення неправдивої інформації, вони можуть також викликати серйозні наслідки у суспільстві. Наприклад, неправдива інформація може призвести до формування негативних стереотипів та передбачень щодо різних груп населення, або створювати напруженість між різними соціальними групами.

Фейкові новини можуть також впливати на виборчі процеси та призводити до змін у громадській думці, що може відображатися на результативності різних політичних партій та кандидатів. Крім того, фейкові новини можуть мати серйозний вплив на економіку, особливо в тому випадку, коли вони стосуються важливих подій та рішень у сфері бізнесу.

Оскільки фейкові новини можуть мати негативний вплив на суспільство, важливо проводити дослідження та розробляти методи для їх виявлення та запобігання. Тільки тоді, коли ми зможемо ефективно боротися з цим явищем, ми зможемо зберегти довіру до надійних джерел інформації та зберегти здоровий інформаційний простір у суспільстві.

Також особливою проблемою, яка додає складності для виявлення класу новини, в останні роки стає генерації фейкових новин стає використання генеративних нейронних мереж. Вони можуть створювати

переконливі фейкові новини, що схожі на правдиві, ускладнюючи їх виявлення, при чому роблячи це зазвичай швидше за людей та з можливістю автоматизації цього процесу для вкидання великих масивів інформації які майже неможливо обробити з використанням лише людських експертів.

Тому, дослідження та розуміння проблеми знаходження фейкових новин є дуже актуальними завданнями для наукового співтовариства. Почнемо вивченню цієї проблеми, а також опису наявних методів її вирішення. Буть досліджені різні підходи та технології, які можуть бути застосовані для виявлення фейкових новин та боротьби з ними, оцінимо їх ефективність та переваги, особливу увагу буде приділено застосуванню методів машинного навчання, дослідженню, здобуттю та обробці даних.

1. ОГЛЯД ПРОБЛЕМИ ФЕЙКОВИХ НОВИН

1.1. Визначення терміну

Фейкові новини - це неправдива інформація, подана як новини, яка може бути оманливою або не містити фактичної основи. Такі новини можуть називатися небажаними, псевдоновинами, альтернативними фактами, фальшивими новинами або нісенітницею. Національний Фонд Демократії визначив фейкові новини як оманливий контент, знайдений в Інтернеті, зокрема в соціальних мережах[1]. Такий контент часто створюють комерційні веб-сайти та сторінки у соціальних мережах, які використовують платформу для прибутку від реклами. Фейкові новини відрізняються від дезінформації, оскільки мотивом створення фейкових новин зазвичай є фінансова вигода чи політичні цілі. Фейкові новини можуть мати певні закономірності в структурі подання інформації, яка грає важливу роль у сприйнятті цих новин[2]. Такі історії зустрічаються в різних сферах, включаючи політику, вакцинацію та вартість акцій.

Також важливою складової маніпулятивної інформації може бути перемішування правдивих фактів та недостовірної інформації у тексті, для отримання певного рівня довіри після чого введення в оману[3]. Ще методами створення маніпулятивної інформації є використання (також наведемо приклади[4]):

1. Нормування мови. Тож "революція" чи "держпереворот"? "Автократія" чи "демократія"?

2. Використання оцінних суджень. Хто тут "ліві" ("праві") "радикали"? То хто тут "популіст"?

3. Метод замовчування.

4. Метод монотонних повторень. "Вода камінь точить". Якщо на гірке весь час говорити "солодке", воно... таки стане солодким.

5. Метод перебільшень. Страху ніколи не буває мало. Якщо ти не приймаєш ідей неоліберaлів, то будуєш концтабори. Лукавство Хайєка та Фрідмана. В іншій версії: "сьогодні ти граєш джаз, а завтра батьківщину продаси".

6. Вісті "з усіх боків" (повідомлення з кожної праски). Блогерів та експертів не буває мало.

7. Якщо всі навколо говорять те саме. то це правда. Або "не довіряй на власні очі". "Не виділяйся"! Ти що, найрозумніший? Тобі це не вигідно.

8. Ефект гойдалок. Про злого Трампа і доброго Обами. Якщо ти борешся зі злом, то від цього ти автоматично не стаєш добрим.

9. Використання соцопитувань на формування громадської думки. Чи ти чув, що люди говорять? Знову не виділяйся!

10. Створення токсичних списків. Трамп, Ердоган, Орбан, Сальвіні, Качинський та, звичайно, Сі. Ти теж хочеш у цю компанію?

11. Вплив на емоції. Як ти можеш так думати (говорити), коли... Поставте після крапки щось страшне і у вас все вийде.

Таким чином певному шматку інформації можна співставити певну оцінку “достовірності” на інтервалі від 0 до 1 (де 0 означає повністю недостовірну та маніпулятивну інформацію, а 1 – надійну інформацію), залежності від того чи використовуються певні мовні конструкції, які використовуються джерела, хто є автором інформації.

1.2. Опис проблеми та її актуальності

Проблема фейкових новин полягає в тому, що вони можуть бути створені з метою поширення неправдивої інформації, що може викликати паніку серед громадськості або впливати на їхні політичні переконання[2][5]. Для виявлення фейкових новин необхідно використовувати різні методи, такі як:

1. Аналіз контексту: Цей метод виявлення фейкових новин передбачає аналіз тексту та з'ясування, чи відповідає він загальному контексту. Наприклад, якщо стаття містить неправдиву інформацію про подію, яка відбулася в певний час та місце, можна перевірити ці дані з офіційних джерел, а також порівняти з іншими повідомленнями про ту саму подію.

2. Перевірка авторства: При виявленні фейкових новин важливо дізнатися, хто їх створив та розповсюдив. Перевірка авторства може включати аналіз даних про автора, які можна знайти в Інтернеті, а також перевірку його інших публікацій та професійної репутації.

3. Перевірка джерела: Процес перевірки джерела інформації, яка була надана в новинах. Якщо джерело ненадійне або неперевірене, то це може бути підозрілим і може бути ознакою того, що новина не є достовірною. Для перевірки джерела варто дізнатися, чи є воно достовірним та чи має воно інші новини, які можна перевірити. Також можна перевірити, чи згадується це джерело в інших надійних джерелах.

4. Перевірка фактів: Процес перевірки тверджень, які містяться в новинах. Якщо твердження є неправдивим або не має достатньої підтримки в доказах, то новина може бути недостовірною. Для

перевірки фактів варто дізнатися, чи існує доказова база для твердження, чи підтримується воно іншими достовірними джерелами. Також можна скористатися факт-чекерами, які перевіряють факти та дають оцінку їх достовірності.

5. Експертна оцінка: процес отримання думки та оцінки від експерта в конкретній галузі на тему новини. Якщо експерт вважає, що новина є недостовірною, то це може бути ознакою того, що новина дійсно не є достовірною. Для експертної оцінки можна звернутися до людей, які мають досвід і знання в певній галузі або відомствах, які мають право на офіційні оцінки новин.

Незважаючи на те, що існують різні методи виявлення фейкових новин, не завжди є можливість однозначно підтвердити або спростувати інформацію. Тому важливо завжди бути критичним до новин, які ми читаємо, та перевіряти інформацію з різних джерел перед тим, як приймати власне рішення.

1.3. Наявні методи вирішення

Отже основні проблеми полягають в тому, що ручні алгоритми перевірки надійності новини чи інформації є доволі трудомістким за часом, ресурсами, кількістю додаткової інформації. Також вимагається певна “інфраструктура” довіри та посилань [6] на надійні джерела, на кшталт схожої інфраструктури публічного ключа, яку вивчає дисципліна безпеки інформаційних систем.

Під час значних подій, які привертають увагу суспільства: вибори, всесвітня пандемія, війна (конвенційна, гібридна), концентрація недостовірної інформації досягає максимальної в інформаційному полі та може призводити до жахливих наслідків. Зараз інформація є найбільш

дієвою зброєю, яка особливо може використовуватися недоброчесно в умовах плюралізму думок, може створювати хибне уявлення про речі у народних мас та впливати на хід історії. При великій концентрації та частоті подачі інформації, ручні методи обробки та перевірки достовірності такої інформації можуть не встигати за швидкістю її розповсюдження та можуть потребувати значних людських ресурсів та кваліфікацій у різних напрямках. Саме тому сучасність потребує швидкої та надійної методики, яка здатна обробляти велику кількість інформації що генерується щосекунди.

Отже, с попереднього пункту приходимо до висновку, що потрібно використовувати сучасні потужні комп'ютери, інформаційні системи, з певною базою правдивої інформації та фактів, які будуть редагуватися та перевірятися незалежними експертами, співставлятися з наявними джерелами інформації, їх достовірністю та надійністю, кожне з яких буде перевірятися за описаним вище алгоритмом. Такі інформаційні системи, вочевидь, вже існують, прикладами можуть бути [7]:

- Fullfact.org – бореться з недостовірною інформацією. Це команда незалежних перевірників фактів і активістів, які виявляють, викривають та протидіють шкоді, яку вона завдає.

- Snopes – це ресурс для перевірки фактів. Коли дезінформація затьмарює правду, а читачі не знають, чому довіряти, перевірка фактів і оригінальні репортажі-розслідування Snopes освітлюють шлях до аналізу, заснованого на доказах і контекстуального аналізу.

- BBC Reality Check – це служба BBC News, присвячена розкриттю фейкових новин і неправдивих історій, щоб знайти правду. Вони досліджують факти та твердження, що стоять за історією, щоб спробувати визначити, правдива вона чи ні.

Також є доволі цікавий підхід розташування ЗМІ у двох координатних осях за їх рейтингом довіри та їх розташуванням на політичному компасі[8]:

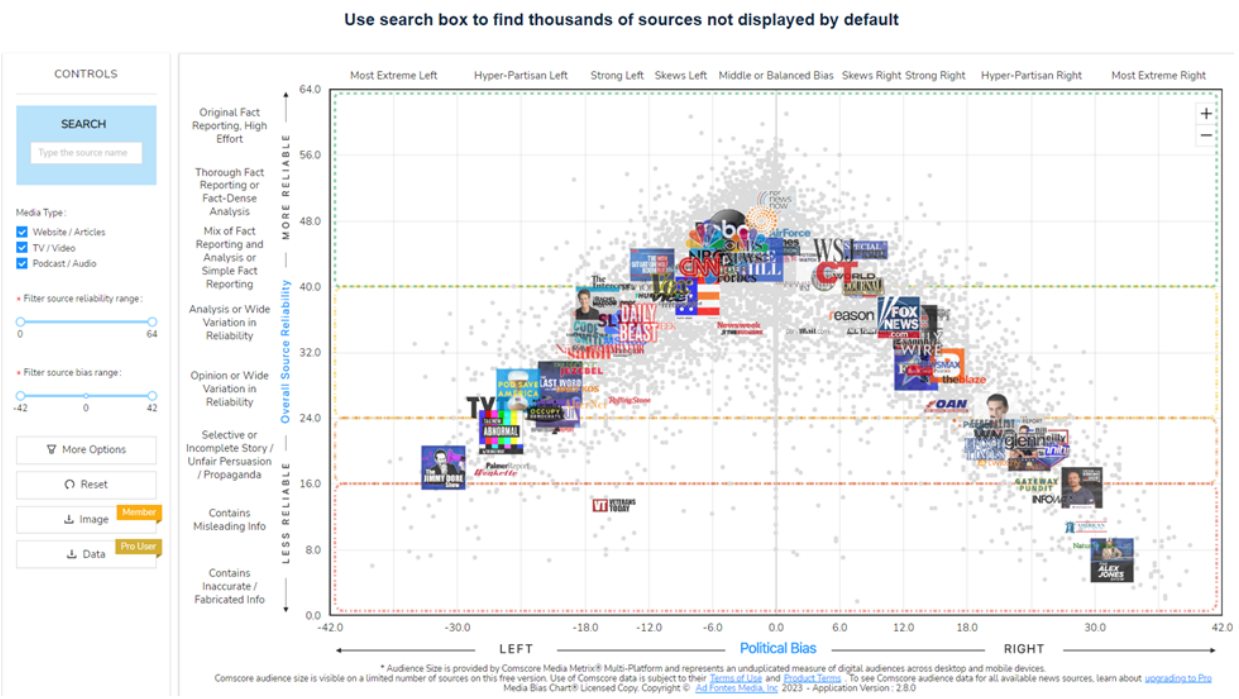

Рисунок 1 — Діаграма упередженості та надійності медіа

На рисунку 1 бачимо на вертикальній осі надписи знизу до гори: “Містить сфабриковану/невірну інформацію”, “Містить інформацію яка вводить в оману”, “Вибіркова неповна історія, нечесне судження, пропаганда”, “Думка чи велика варіація в надійності”, “Змішування донесення фактів та аналітики чи просте висвітлення фактів”, “Чиста аналітика фактів та найдіна перевірка”, “Найвищі зусилля до надання неупередженої оригінальної інформації”. На горізонатльній осі бачимо підписи: “Найбільш екстримально ліві”, “дуже упереджено ліві”, “сильно ліві”, “невеликий зсув до лівих”, “середнє чи збалансовано”, “невеликий зсув до правих”, “сильно праві”, “дуже упереджено праві”, “Найбільш

екстримально праві”. Також можна шукати медіа за назвою та фільтрувати за значеннями найдіності та упередженості.

Ad Fontes Media (Home | Ad Fontes Media) створює діаграму (рис. 1) упередженості медіа, використовуючи двовимірну структуру для оцінювання надійності та упередженості вмісту, шоу та джерел. Горизонтальна вісь (політичне упередження, зліва направо) розділена на сім категорій, три з яких представляють спектр ліворуч, три з яких представляють спектр праворуч і одна посередині. Кожна категорія охоплює 6 одиниць оцінювання, тому загальна числова шкала коливається від -42 зліва до +42 справа. Вертикальна вісь (загальна надійність, зверху вниз) поділена на вісім категорій, кожна з яких охоплює вісім одиниць оцінки, для загальної числової шкали від 0 до 64. Оцінки, зібрані під час аналізу, базуються на припущенні, що ця класифікація є дійсною. Спосіб оцінки джерел новин за параметрами надійності та політичної упередженості. Методологія розвивалася з часом і завдяки внеску багатьох вдумливих коментаторів і експертів.

Процес аналізу включає відбір і аналіз контенту. Ad Fontes Media набирає команди політично різноманітних аналітиків і навчає їх своїй методології. Згодом цей процес перетворився на поточний метод Ad Fontes Media оцінювання вмісту за участю багатьох аналітиків. Усі оцінки, зібрані під час аналізу, базуються на припущенні, що ця таксономія є дійсним способом оцінки джерел новин за параметрами надійності та політичної упередженості.

Такий підхід є доволі зручним, значно спрощує процес перевірки фактів але має декілька суттєвих мінусів:

- Все ще вимагає значної ручної роботи для наповнення та перевірки кожної окремої новини та інформації

- Може займати певний час фактчекінг найбільш свіжих новин
- Не є повноцінно автоматизованим

Для виправлення цих проблем можна використовувати інструменти штучного інтелекту, такі як машинне навчання та аналіз природної мови, щоб автоматизувати процес виявлення фейкових новин та інформаційних спроб маніпулювання. Однак, з огляду на постійне змінення технологій створення та поширення фейкових новин, необхідно постійно вдосконалювати методи їх виявлення.

1.4. Висновки до розділу

Для боротьби з поширенням фейкових новин необхідно використовувати комплексний підхід. Один із методів - це використання програмного забезпечення для автоматизованого виявлення фейкових новин на основі аналізу контексту, перевірки джерел, перевірки авторства, перевірки фактів та експертної оцінки. Важливо зазначити, що жоден з цих методів не є повністю ідеальним, тому їх слід застосовувати разом, щоб максимально зменшити ризики поширення фейкових новин. Важливо створити певну базу та знання світу в системі штучного інтелекту, яка дозволить, використовуючи сучасні архітектури, аналізувати текст на наявність маніпуляцій та недостовірних фактів.

Найбільш ефективним підходом є розробка програмного забезпечення, яке включатиме в себе всі методи виявлення фейкових новин, що описані вище, і забезпечуватиме високу точність виявлення фейків на основі штучного інтелекту та навчання з вчителем та обробки природної мови. Досліджені інформаційні системи можуть допомогти зробити цю модель для передбачення більш неупередженою та

використовуватися як додаткові дані для досягнення найкращих результатів.

Це допоможе не тільки знизити ризики впливу фейкових новин на суспільство, а й зберегти надійність та довіреність засобів масової інформації в очах громадськості. Такий підхід може стати кроком до побудови довірливого та стійкого інформаційного середовища в Інтернеті.

2. ОГЛЯД ІСНУЮЧИХ STATE OF ART РІШЕНЬ

Після дослідження та опису проблеми виявлення фейкових новин, актуальності цієї проблеми, опису основних негативних наслідків, та визначення задачі, ознайомлення з готовими інформаційними системами, які можуть допомогти у вирішенні цієї проблеми, наступним кроком буде пошук існуючих рішень, які будуть базуватися на сучасних методиках обробки мови та машинного навчання.

Надалі ми зосередимося на детальному огляді та аналізі готових рішень для виявлення фейкових новин, розглянемо переваги та недоліки наявних рішень, щоб знайти найкращий спосіб виявлення фейкових новин, а також дослідимо можливості покращення наявних методів та розробки нових підходів, врахуємо недоліки та переваги, щоб розробити власне покращене рішення.

Для цих цілей потрібно буде використовувати метрики, які будуть стосуватися, перш за все, точності віднесення новини до коректного класу, можливості масштабування системи (що стосується можливості використання на різнотипних даних, різних мовах, різних темах та різних джерелах отриманих даних), швидкості роботи нейромережі, кількість її параметрів (що напряму впливатиме на вимоги до потужності CPU/GPU та кількості пам’яті).

Загалом метою буде виявлення найбільш збалансованої системи та можливості її покращення з урахуванням її недоліків.

2.1. Загальний опис алгоритму для впровадження машинного навчання

Для початку ознайомимось з основним процесом, який супроводжує більшість завдань машинного навчання, а саме тих які стосуються Supervised Learning.

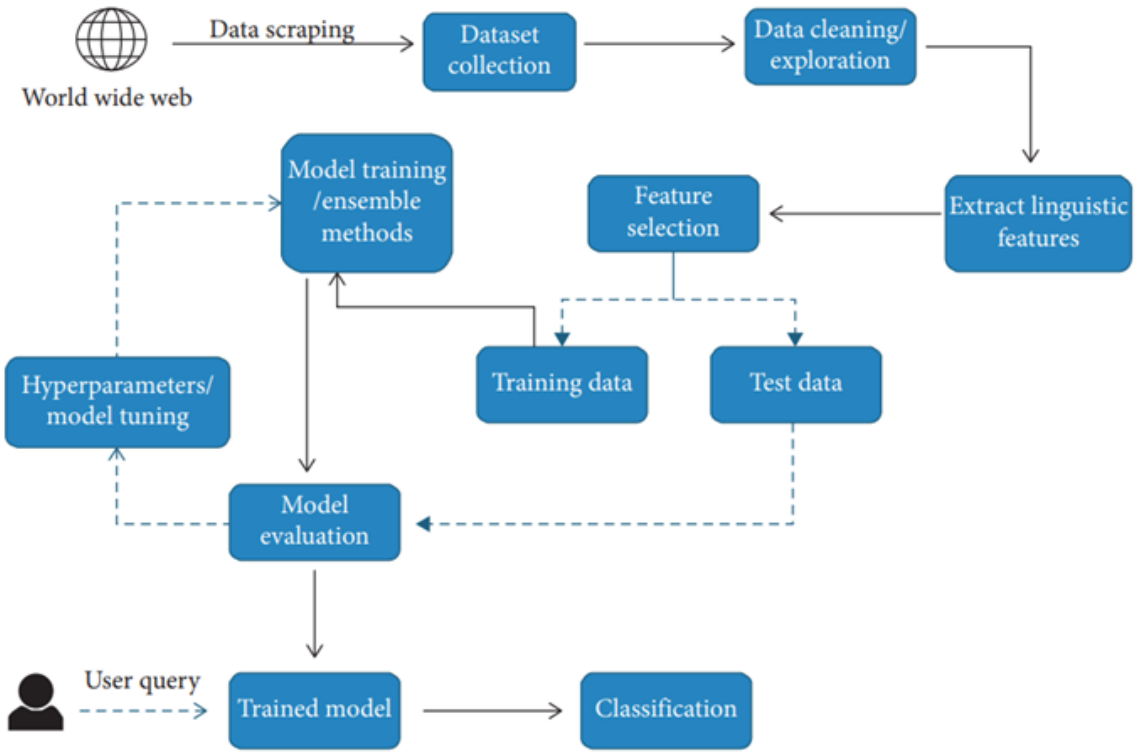

Рис. 2 — Supervised learning підхід до класифікації [9]

На рисунку 2 бачимо діаграму, яка описує основний процес машинного навчання. Зліва зверху бачимо, що процес бере початок з Data scrapping тобто збору даних, далі необхідно очистити дані (Data cleaning). Наступним є процес отримання лінгвістичних змінних/характеристик, які дозволяють мати моделі адекватні уявлення про текст, тобто “Extract linguistic features”, наступним кроком важливо обрати найбільш якісні та пов’язані з класом характеристики тексту тобто Feature Selection. Далі відбувається розбиття даних на train та test data, тобто тестові та

тренувальні дані. Наступним кроком є тренування моделей та можлива агрегація результатів з декількох, тобто “Model training/ensemble methods”. Наступним додатковим кроком є Hyperparameters/model tuning, тобто настройка гіперпараметрів та моделі. Одним з останніх є обчислення результатів моделі (Model evaluation), у випадку успішного результату, який відповідає очікуванням там вимогам, можливо використовувати модель для класифікації даних від користувача (User Query).

З опису (рис. 2) бачимо, перш за все, необхідно збирання якісних даних, які будуть у форматі таблиці, де кожному тексту буде відповідати певне значення (True або False) чи є новина справжньою чи їй ліпше не довіряти. Далі йде обробка та аналіз цих даних (позбавлення зайвих символів, для певних моделей можна прибрати такі частини як прийменник, частку та інші, оскільки зазвичай вони не сильно впливатимуть на загальний сенс речення). Аналіз датасету вбирає в себе розбиття його на теми, перевірку збалансованості класів датасету, перевірку некоректних чи відсутніх даних, перевірка кореляцій появи у окремих класах певних слів, авторів. Тобто вже без прямого застосування машинного навчання можна буде побачити певні залежності, які можуть дати більше уявлення про дані та наштовхнути на нові ідеї. Збирання та підготовка якісних даних – це окремий складний процес, який доволі сильно впливатиме на точність фінального результату, але звісно і без потужної моделі, яка буде здатна якісно вловлювати зв’язки у довгому тексті, краще розуміти сенс речення, вловлювати використання різних мовних конструкцій та лексично-художніх засобів, які були описані раніше, та які можуть найчастіше зустрічатися у маніпулятивних текстах. Тому у наступному розділі приділимо увагу саме аналізу сучасних моделей обробки природної мови.

2.2. Огляд моделей моделей для обробки мови

Спробуємо проаналізувати основні та найпоширеніші моделі, які можуть використовуватися, від найбільш простих до більш складних, які мають давати кращі результати.

2.2.1. Логістична регресія (Послідовні нейронні мережі)

Модель Logistic Regression - це одна з найпоширеніших моделей машинного навчання, яка застосовується в задачах бінарної класифікації.

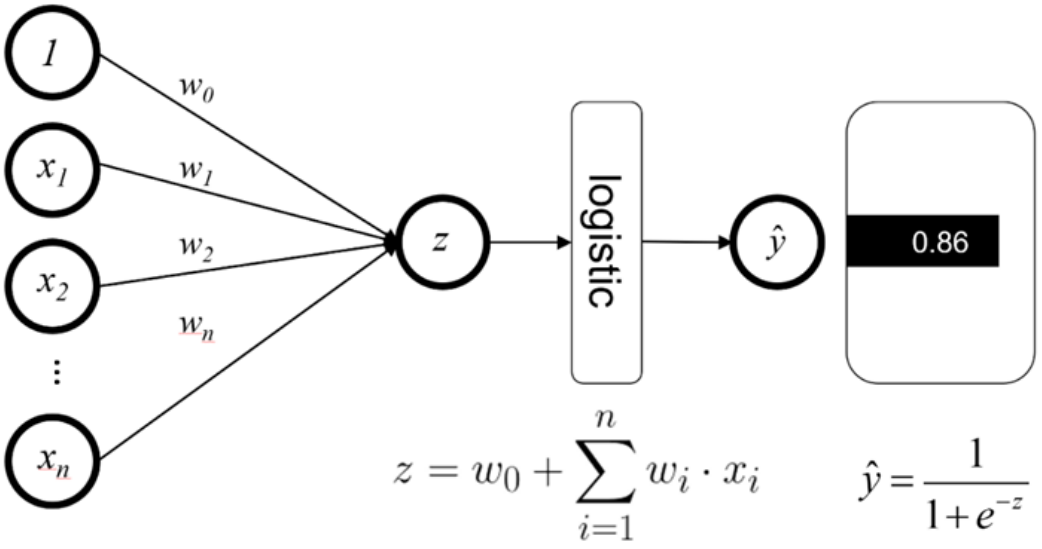

Рисунок 3 — Архітектура логістичної регресії [10]

На рисунку 3 бачимо вхідні оброблені числові дані, які перемножуються на ваги w0, w1, w2, wn., отримується проміжний результат z, яке після чого подається на вхід logistic (логістичного) шару, який переводить цей результат в ймовірність, формули за якими це відбувається наведено нижче.

Формула (1) обчислення вихідних даних з лінійної регресії на основі вхідних даних (x):

$$z^{(i)} = w^T x^{(i)} + b \quad (1)$$

Формула (2) для отримання значення ймовірності на інтервалі [0,1] приналежності до певного класу з використанням сигмоідальної функції:

$$a^{(i)} = \sigma\left(z^{(i)}\right) \qquad (2)$$

де $\sigma(x) = \frac{1}{1+e^{-x}}$

Далі розглянемо функцію втрат(3) , яка використовується в логістичній регресії та її розширеннях, таких як нейронні мережі, визначена як від'ємна логарифмічна правдоподібність логістичної моделі, яка повертає ймовірності y_pred для своїх навчальних даних y_true[11]. Формула Loss функції (бінарна кросентропія) для двох класів:

$$L\left(y^{(i)}, a^{(i)}\right) = -\, y^{(i)} log\left(a^{(i)}\right) - \left(1 - y^{(i)}\right) log\left(1 - a^{(i)}\right) \qquad (3)$$

Також наведемо формули для обчислення градієнтів параметрів по Loss функції, відповідно для z (4) , W (5) , b (6) :

$$dz^{(i)} = a^{(i)} - y^{(i)} \quad (4)$$

$$\frac{\partial L}{\partial w} = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} dz^{(i)} \qquad (5)$$

$$\frac{\partial L}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)} \quad (6)$$

Формули (7, 8) для обчислення нових параметрів, відбувається оновлення з невеликим кроком:

$$w_{new} = w_{old} - \alpha \cdot \frac{\partial L}{\partial w} \quad (7)$$

$$b_{new} = b_{old} - \alpha \cdot \frac{\partial L}{\partial b} \qquad (8)$$

де α - learning rate.

Основна ідея (рис. 3) полягає в тому, що модель бере певні вхідні значення (це можуть бути вектори, матриці, скалярні значення залежно від розмірності та задачі яка стоїть), на виході розділяє дані на дві класи на

основі вхідних факторів. На цьому передбачення завершено. Але якщо моделі в процесі навчання, тобто зменшення loss функції і покращення своїх передбачень, тоді з використанням Loss функції отримає фідбек від справжніх даних та, обчислюючи похідні параметрів моделі відносно loss функції, з невеликим кроком змінює ваги матриць моделі для отримання кращих передбачень, тобто для зменшення Loss функції. Для отримання значення від 0 до 1 використовується найчастіше sigmoid функція. Для отримання класу можна використовувати звичайне округлення, якщо використовується threshold розділення класів у 0.5(більше чи дорівнює 0.5 => класифікуємо як “1”, менше 0.5 => класифікуємо як “0”) , так само його можна встановити іншим, якщо є інформація, що він видає кращі Recall та Precision метрики, це значення обирається відповідно до вимог моделі[12].

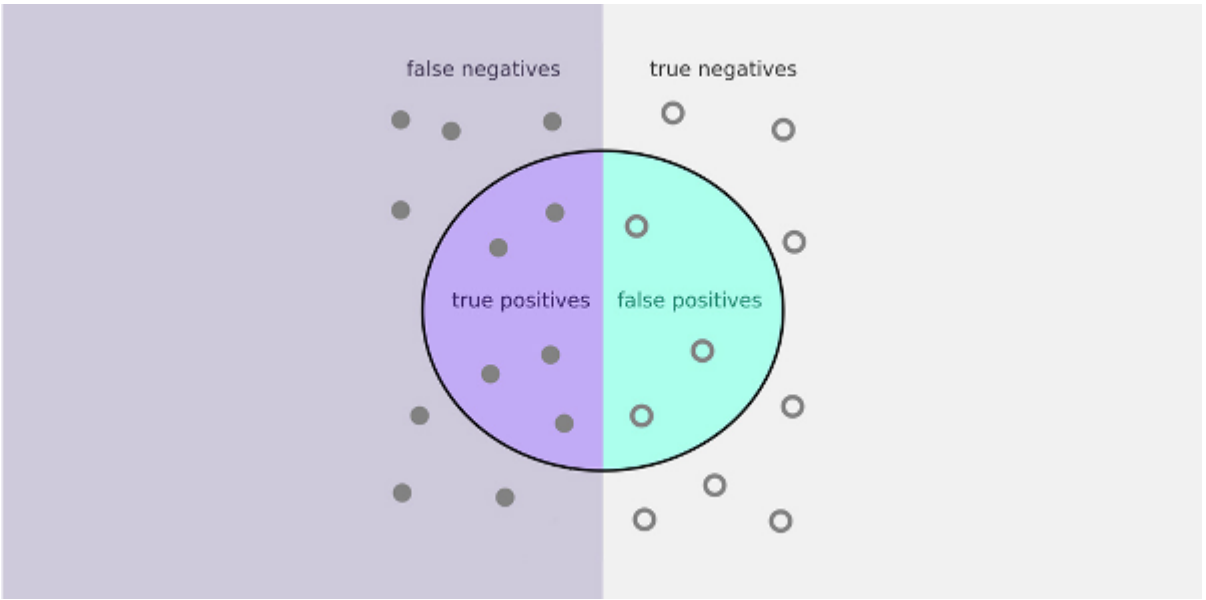

Рисунок 4 — Наочно показано можливі класи в Confusion Matrix[12]

На рисунку 4 бачимо декілька об’єктів, які класифікуються моделлю, в коло входять об’єкти, які мають дійсно “true” класи, але вони розділяються на true positives(на фіолетовому), тобто дійсно отримані

вірні результати, на зеленому бачимо false positives, це ті які є негативними, але модель класифікувала як позитивні. Аналогічно стосується за колом, маємо false negatives, тобто ті, які були класифіковано як негативні, хоча насправді є позитивними, та true negatives, які дійсно є негативними зразками.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (9)$$

Точність (Precision) показує, наскільки точно модель класифікує позитивні приклади. Вона обчислюється як співвідношення правильно класифікованих позитивних прикладів (рис. 4) до загальної кількості прикладів, які модель визначила як позитивні. Висока точність означає, що майже всі приклади, визначені як позитивні, дійсно є позитивними, тобто модель робить мало помилок, класифікуючи негативні приклади як позитивні.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (10)$$

З іншого боку, повнота (Recall) показує, наскільки добре модель виявляє всі позитивні приклади. Вона обчислюється як співвідношення правильно класифікованих позитивних прикладів до загальної кількості дійсно позитивних прикладів. Висока повнота означає, що модель виявляє більшу кількість позитивних прикладів, тобто робить менше помилок, пропускаючи позитивні приклади і класифікуючи їх як негативні.

У практичних застосуваннях, залежно від конкретної задачі, ми можемо визначати баланс між точністю та повнотою. Ми можемо вибрати модель з високою точністю, якщо нам важливо мінімізувати помилки класифікації позитивних прикладів як негативні. Або ми можемо вибрати модель з високою повнотою, якщо ми хочемо максимізувати виявлення всіх позитивних прикладів, навіть за ціною деякої кількості помилок.

Враховуючи спільно точність та повноту, ми можемо зробити більш об'єктивне оцінювання ефективності моделі класифікації в машинному навчанні.

Але тепер виникає питання як використовувати цю модель для аналізу тексту, якщо він може бути довільної довжини, а розмір матриці вагів є фіксованим. Для цього модель отримує вхідний текст та виконує на ньому попередню обробку, наприклад, лематизацію та токенізацію, о яких далі, більше детально піде мова.

Токенізація[13] - це процес розбиття тексту на окремі елементи, які називаються токенами або лексемами. Токенами можуть бути окремі слова, розділові знаки, числа та інші знаки пунктуації.

Лематизація[13] - це процес перетворення слова на його базову форму або лему. Це означає, що всі форми слова (наприклад, дієслова у різних часових формах або іменники у множині) будуть зведені до однієї базової форми. Наприклад, після токенізації речення "Я люблю грати в футбол" буде розбите на токени "Я", "люблю", "грати", "в", "футбол". Після лематизації, слово "люблю" буде перетворено на його базову форму "любити", що допомагає в зведенні різних форм цього слова до одного леми. Після цього векторизує текст, тобто перетворює його у числове представлення, використовуючи методи, такі як Bag-of-words або TF-IDF.

Метод Bag-of-words[14] використовується для перетворення тексту на числовий вектор. Суть методу полягає в тому, щоб зібрати всі унікальні слова з тексту і підрахувати кількість входжень кожного слова в текст. Потім створюється вектор довжиною в кількість унікальних слів у тексті, де кожен елемент вектора відповідає певному слову. Значення кожного елементу вектора відповідає кількості входжень відповідного слова в текст. Цей вектор називається мішком слів або Bag-of-words.

Метод TF-IDF[15] (Term Frequency-Inverse Document Frequency) також використовується для перетворення тексту на числовий вектор, але він підраховує не тільки кількість входжень кожного слова в текст, але і важливість кожного слова в даний текст. Значення TF-IDF враховує кількість входжень слова в даний текст (Term Frequency) і його частоту входження в інші тексті (Inverse Document Frequency). Значення TF-IDF дорівнює добутку значення Term Frequency і значення Inverse Document Frequency. Цей вектор називається TF-IDF вектор.

Відмінність між методами полягає в тому, що TF-IDF використовує ваги, що відображають частоту вживання слова в даних, тоді як Bag-of-words не використовує ваги, тобто розглядає кожне слово як рівнозначне. TF-IDF може бути корисним, коли деякі слова в документі можуть бути більш важливими за інші, такі як ключові слова.

Також можна додати певну максимальну довжину для тексту та представляти кожне окреме слово чи символ як вектор, а кінець тексту та відсутність представляти нульовим вектором. Також непоганим узагальненням цього методу буде використання нейронної мережі[16].

Замість одного шару у логістичній регресії, глибока нейромережа має кілька прихованих шарів. Кожен з цих шарів має свої ваги та функцію активації, що дозволяє моделі отримувати складніші та більш гнучкі взаємозв'язки між вхідними та вихідними даними.

Ключовою особливістю глибокої нейромережі є використання функцій активації в кожному шарі. Функція активації визначає, як будуть змінюватись та передаватись дані між нейронами у мережі. Популярні функції активації включають сигмоїдну функцію, ReLU[17], Tanh[18] та інші. Використання цих функцій дозволяє глибокій нейромережі

моделювати нелінійні залежності у текстових даних та здійснювати більш точну класифікацію.

Архітектура глибокої нейромережі визначає, як шари та нейрони взаємодіють між собою. Зазвичай, глибока нейромережа має один вхідний шар, кілька прихованих шарів та вихідний шар. Кожен прихований шар приймає вхідні дані з попереднього шару, обчислює лінійну комбінацію цих даних з використанням ваг та застосовує функцію активації для отримання вихідних значень. Такий підрядок шарів дозволяє глибокій нейромережі здійснювати більш складну та гнучку обробку текстових даних, що може призвести до покращеної точності класифікації та здатності виявляти складні залежності у тексті.

Усі ці зміни дозволяють глибоким нейромережам бути потужними інструментами для класифікації тексту з використанням TF-IDF. Вони дозволяють моделювати складніші залежності, здійснювати глибоку обробку тексту та покращувати точність класифікації.

Узагальнюючи, модель Logistic Regression чи її покращення у вигляді Feed Forward Neural Network є непоганим інструментом для класифікації текстів в NLP задачах, коли необхідно визначити “baseline performance”, але для досягнення найкращих результатів буде доцільніше використовувати більш досконалі моделі, які більше пристосовані для структури тексту та вловлювання усіх зв’язків між лексемами.

2.2.2. Наївний класифікатор Байєса

Naive Bayes Classifier - це алгоритм машинного навчання, який використовується для вирішення задач класифікації. В основі цього алгоритму лежить теорема Байєса, яка визначає ймовірність того, що подія станеться, на основі ймовірності того, що вона вже сталася[19].

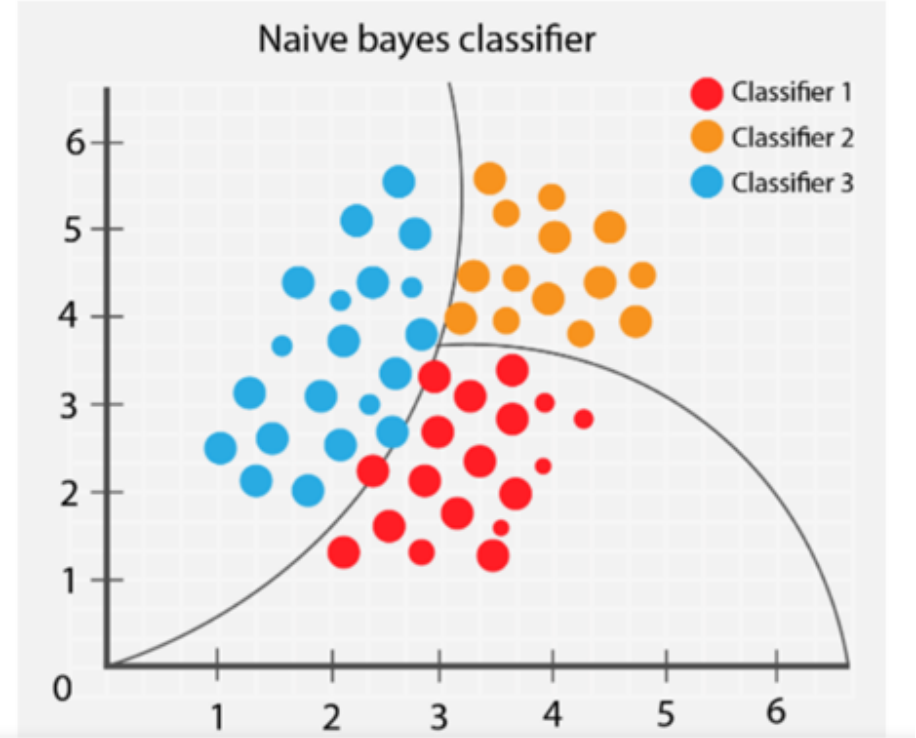

Рисунок 5 — Межи класифікації з використанням NBC [19]

На рисунку 5 бачимо приклад того як можуть розставлятися границі класифікації для декількох класів при класифікації з використанням “Naive Bayes Classifier” (Наівний класифікатор Байєса) двох характеристик: бачимо три класи, які підписані як “Classifier 1”, “Classifier 2”, “Classifier 3”.

Для застосування Naive Bayes Classifier в NLP задачах, спочатку потрібно підготувати корпус текстів, токенізувавши та лематизувавши їх. Потім знаходяться унікальні слова (токени), які використовуються в текстах.

Формули з теорії ймовірностей та статистики, які використовуються для подальших дій:

$$P(C) = \frac{N_C}{N} \quad (11)$$

Попередня ймовірність (11) P(C) представляє ймовірність спостереження класу C у даних навчання, обчислену як відношення кількості екземплярів класу C (N_C) до загальної кількості екземплярів (N).

$$P\left(x_i|C\right) = \frac{count\left(x_i,C\right)+\alpha}{N_C+\alpha\cdot|V|} \quad (12)$$

Умовна ймовірність класу (12) P(x_i | C) представляє ймовірність спостереження ознаки x_i за класу C. Вона оцінюється шляхом підрахунку появи ознаки x_i з класом C, додавання члена згладжування (α) і ділення на суму входжень усіх ознак із класом C і члена згладжування, помноженого на загальну кількість унікальних ознак (|V|).

$$P\left(x_1, x_2, \ldots, x_n|C\right) = \frac{P(C)\cdot P(C)\cdot P(C)\cdot\ldots\cdot P(C)}{P\left(x_1,x_2,\ldots,x_n\right)} (13)$$

Апостеріорна ймовірність (13) P(C | x_1, x_2, ..., x_n) обчислюється шляхом множення попередньої ймовірності та умовних ймовірностей класу кожної ознаки та нормалізації за доказом P(x_1, x_2, ..., x_n)).

У цих формулах наивний класифікатор Байєса обчислює апостеріорну ймовірність P(C | x_1, x_2, ..., x_n) класу C із заданими значеннями ознак x_1, x_2, ..., x_n.

Алгоритм використовує формулу наївного припущення (naive assumption) про незалежність факторів. Отже, вважається, що кожен токен (слово) в тексті внесе свій внесок у його класифікацію незалежно від інших токенів. На практиці, це припущення досить наївне, оскільки токени можуть взаємодіяти між собою в тексті.

Далі для кожного тексту, алгоритм обчислює значення ймовірності належності до кожного з можливих класів (рис. 2.2.2.1.). Ймовірності

обчислюються на основі попередньої інформації про кожний клас та використання формули наївного Байєса. Клас з найвищою ймовірністю вважається "вірним" класом для даного тексту.

Наприклад, у задачі класифікації текстів на "позитивний" та "негативний" класи, алгоритм Naive Bayes знаходить ймовірності належності тексту до кожного з класів на основі використання слова. Якщо у тексті зустрічається слово "хороший", ймовірність того, що текст належить до "позитивного" класу збільшується.

Після підготовки датасету можна перейти до векторизації текстів. Це означає, що кожен текст перетворюється в вектор чисел за допомогою методу Bag-of-words або TF-IDF.

Текст	about	bird	heard	is	the	word	you
	Отримані вектори						
About the bird, the bird, bird.	1	3	0	0	2	0	0
You heard about the bird , you bird?	1	2	1	0	1	0	2
The bird is the word.	0	1	0	1	2	1	0

Таблиця 2.2.2.1. Створення векторів за допомогою BOW

Далі застосовується сам алгоритм Naive Bayes. Він використовується для розрахунку ймовірностей того, що даний текст належить до кожної з категорій, використовуючи векторизований текст. Потім модель вибирає категорію з найбільшою ймовірністю і приписує їй текст.

Одним з головних переваг Naive Bayes Classifier є його швидкодія. Він може обробляти великі обсяги тексту досить швидко, навіть на

стандартному комп'ютері. Крім того, Naive Bayes добре працює з великою кількістю категорій та є доволі простим у реалізації. Однак, зазвичай на складних даних він показує погану точність, якщо у тестовому датасеті тексти містять слова, які не використовуються у навчальному датасеті, а також якщо використовується метод Bag-of-words, який не враховує контекст та порядок слів у тексті. Його можна використовувати як допоміжний алгоритм, але зазвичай його недостатньо для отримання хороших результатів у складних текстах.

2.2.3. Дерево рішень (Випадковий ліс)

Random Forest - це алгоритм машинного навчання для класифікації, регресії та інших завдань, що ґрунтуються на аналізі даних. Алгоритм є поєднанням декількох рішень дерев рішень, які називаються "деревами випадкового лісу".

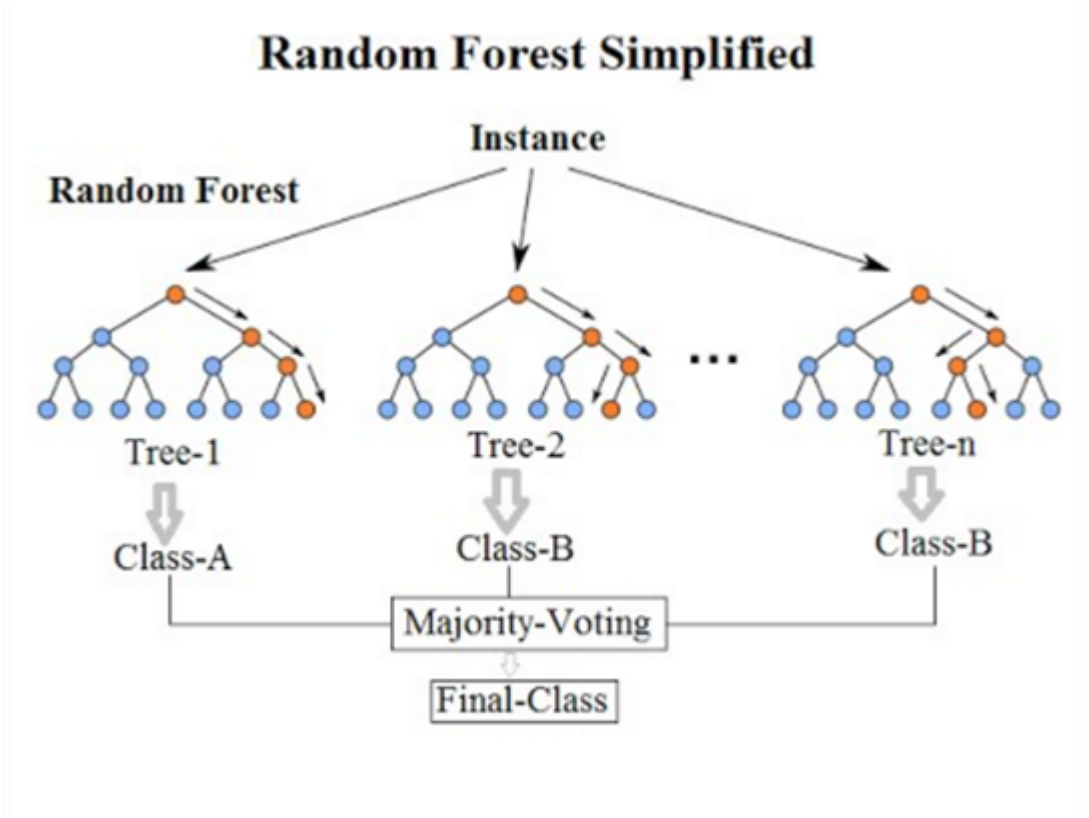

Рисунок 6 — Випадковий ліс (дерев рішень) [20]

На рисунку 6 бачимо підпис “Random Forest Simplified”, тобто спрощена структура випадкового лісу, яка складається з декількох “Instance”, тобто об’єкт класу дерева рішень, після чого створюється декілька таких об’єктів, після навчання, відбувається “Majority-Voting”, тобто голосування більшістю, де в кінці визначається Final Class - остаточний клас, який і є результатом передбачення.

Алгоритм дерева рішень є одним з найпопулярніших методів машинного навчання. В основі його роботи лежить побудова дерева рішень, де кожен вузол представляє один із атрибутів даних, а кожне ребро - можливе значення цього атрибута. На основі цього дерева можна проводити класифікацію або регресійний аналіз даних.

Побудова дерева рішень починається з кореневого вузла, який представляє всю множину даних. Для побудови дерева рішень необхідно визначити, який атрибут має найбільшу інформаційну цінність. Це робиться за допомогою певного критерію інформаційної вигоди, такого як ентропія, джині-індекс або інформаційний приріст. Ці критерії вимірюють, наскільки корисною є кожна змінна у визначенні класу даних.

Після вибору найбільш інформативного атрибута, дерево рішень розбивається на дві частини: одну з підмножин даних, де вибраний атрибут приймає певне значення, і іншу підмножину даних, де цей атрибут не приймає це значення. Цей процес рекурсивно повторюється для кожної підмножини даних, доки не буде досягнуто кінцевої точки, яка є листком дерева. У кожному листі дерева знаходиться прогнозована мітка класу для нових даних, які потраплять у цей лист.

Використання дерева рішень має декілька переваг. Перш за все, це простий і інтуїтивно зрозумілий метод, що дозволяє враховувати взаємозв'язки між змінними.

При створенні Random Forest (рис. 6.) моделі спочатку створюється випадковий набір даних (відбір даних з повтореннями) з оригінального набору даних для тренування. Потім будується дерево рішень на цьому випадковому наборі даних. Цей процес повторюється кілька разів, з кожного разу вибираючи новий випадковий набір даних. Після того, як створено декілька дерев, класифікація нового зразка здійснюється шляхом голосування дерев за більшістю.

Один з головних переваг Random Forest - його здатність працювати з великими наборами даних, враховуючи декілька параметрів. Крім того, він може бути використаний для вирішення проблеми перенавчання, що може виникнути при використанні інших алгоритмів. Наприклад, коли дані містять надмірну кількість ознак, що можуть привести до поганої проекції на майбутні дані. За допомогою Random Forest можна вибрати найбільш значущі ознаки та побудувати більш точну модель класифікації.

2.2.4. Рекурсивні нейронні мережі

RNN (Recurrent Neural Network) - це алгоритм машинного навчання, який використовується для аналізу даних, що мають послідовну структуру. За допомогою RNN можна аналізувати тексти, звукові дані, відео та інші послідовні дані.

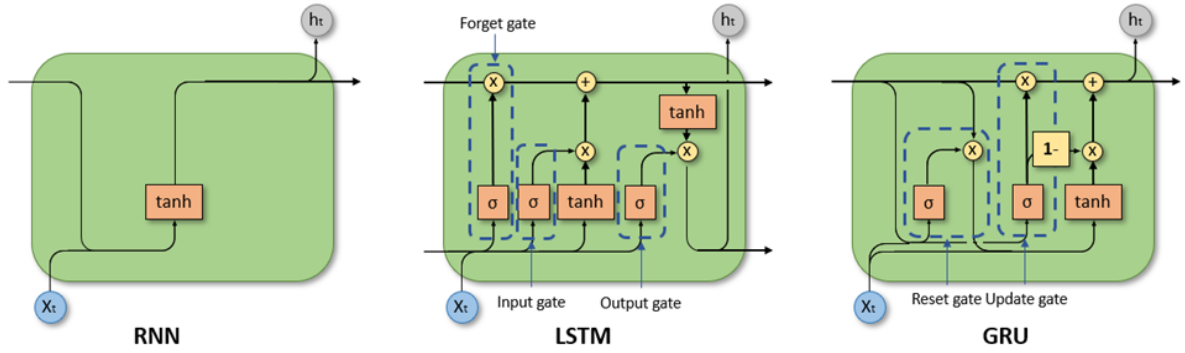

Рисунок 7 — Блок RNN та його архітектурні покращення [21]

На рисунку 7 бачимо приклади архітектур RNN (Reccurent Neural Network), що означає рекурентна нейронна мережа, LSTM (long short-term memory), що перекладається як, Довга короткочасна пам’ять та GRU (Gated Reccurent Units), що перекладається, як вентильні рекурентні вузли.

Основна відмінність RNN від інших алгоритмів машинного навчання полягає в тому, що вона має здатність зберігати стан або інформацію про попередні дані та використовувати її для аналізу наступних даних. Ця здатність дає можливість RNN працювати з послідовними даними будь-якої довжини та зробити прогноз для наступних значень в послідовності.

Архітектура RNN (рис. 7) складається з одного або декількох рекурентних шарів, кожен з яких містить рекурентні нейрони. Рекурентний шар використовує вхідні дані, а також стан з попереднього кроку для генерації вихідних даних та передачі стану на наступний крок.

RNN алгоритм може бути використаний у NLP для обробки послідовностей слів, фраз та речень, таких як текстові документи, розмови, твіти тощо. Він може виконувати завдання, такі як машинний переклад, генерація тексту, класифікація тощо.

Одним з основних застосувань RNN в NLP є моделювання мови. За допомогою RNN можна побудувати модель, яка може передбачати наступний символ або слово в послідовності, враховуючи контекст попередніх символів. Це можна використовувати для генерації тексту, наприклад, для автоматичного писання статей, відгуків або твітів.

Інший застосування RNN в NLP полягає в класифікації текстів. За допомогою RNN можна навчити модель, яка класифікує текст відповідно

до заданої категорії, такої як позитивний або негативний відгук, категорія новин тощо.

Також RNN може використовуватися для машинного перекладу. У цьому випадку RNN будує модель, яка взаємодіє зі вхідним текстом та генерує вихідний текст у цільовій мові.

Існує кілька покращень архітектури RNN для поліпшення її ефективності в задачах обробки природних мов:

- LSTM (Long Short-Term Memory) - це розширення RNN, що дозволяє довші (рис. 7) зв'язки між вхідними та вихідними даними, а також враховує контекст і інформацію з минулого для кращої прогнозування наступних значень.
- GRU (Gated Recurrent Unit) - це більш проста (рис. 7) архітектура, що має менше параметрів, ніж LSTM, але все ще дозволяє моделі зберігати контекст і запам'ятовувати важливу інформацію.
- Bidirectional RNN - це архітектура, що дозволяє моделі аналізувати не тільки попередні значення послідовності, але й наступні значення. Це допомагає моделі зберегти більше контексту та краще розуміти семантику.
- Attention Mechanism - це покращення, що дозволяє моделі зосередитися на певних частинах вхідної послідовності, які є важливими для вирішення конкретної задачі. Це допомагає покращити точність моделі і зменшити кількість параметрів.

Ці покращення можуть бути поєднані разом для створення більш ефективних архітектур, що використовуються в різних задачах NLP, таких як машинний переклад, розпізнавання мови, класифікація текстів і багато інших.

2.2.5. Трансформери

Алгоритм трансформерів (Transformers) є архітектурою глибокого навчання, яка була розроблена для обробки послідовних даних, зокрема тексту, з високою швидкістю та точністю. Він був представлений у 2017 році у статті "Attention Is All You Need" від дослідників компанії Google.

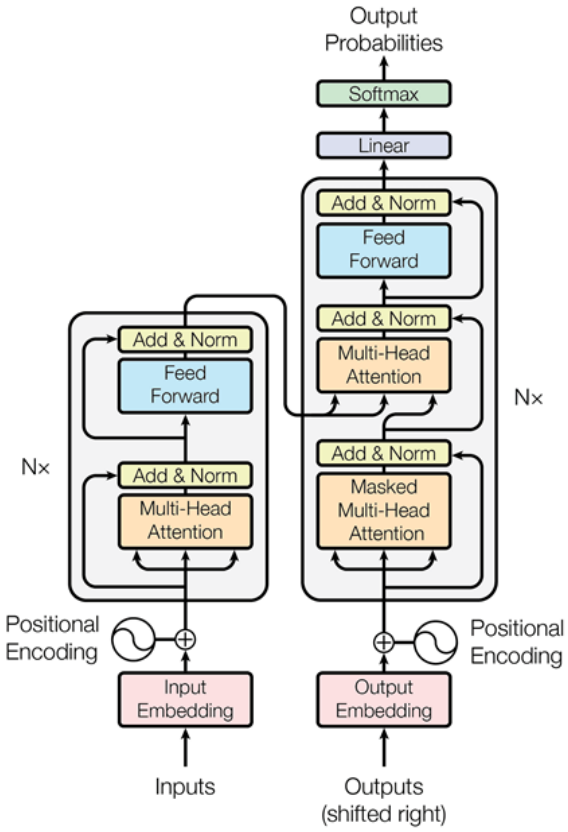

Рисунок 8 — Архітектура трансформер моделі [22]

На рисунку 8 бачимо архітектуру моделі Cell Transformer. Ця модель приймає на вхід послідовність токенів, яку потрібно трансформувати. Ця послідовність токенів спочатку проходить через Input Embedding (вбудовування вхідних даних), що перетворює кожен токен на вектор. Потім використовується Masked Multi-Head Attention (увага з маскою та

декількома головами). Цей шар моделі знаходить взаємозв'язки між токенами в послідовності та приділяє більшу увагу важливим залежностям. Після цього застосовується Add & Norm (додавання та нормалізація), що дозволяє додавати та нормалізувати ваги та зберігати контекстуальну інформацію. Далі використовується Positional Encoding (позиційне кодування), що надає моделі інформацію про позицію кожного токену в послідовності. Це дозволяє моделі розрізняти токени на різних позиціях. Після цього застосовується Feed Forward (прямопропускний шар), який складається з двох лінійних перетворень та нелінійності. Він дозволяє моделі виявляти складні залежності та робити передбачення. Нарешті, використовується Softmax (функція м'якого максимуму), яка перетворює вихід моделі на імовірності для кожного токену. Ці імовірності можуть використовуватися для класифікації або генерації вихідної послідовності. Output Embedding (вбудовування вихідних даних) використовується для перетворення остаточного векторного представлення токенів у вихідний формат або для подальшої обробки.

Основна ідея трансформерів (рис. 8) полягає в тому, що вони використовують механізм уваги (attention mechanism), який дозволяє моделі фокусуватися на важливих частинах послідовності та здійснювати операції з ними паралельно. Це забезпечує високу швидкість обробки та знижує кількість параметрів моделі.

Трансформери складаються з набору блоків, кожен з яких містить кілька шарів, що дозволяє моделі розуміти послідовності на різних рівнях. Кожен блок містить два підблоки - механізм уваги та повністю зв'язний шар (fully connected layer), який здійснює перетворення.

Трансформери можуть бути використані для різних задач обробки мови, таких як машинний переклад, розпізнавання мови та класифікація

тексту. Вони можуть допомогти у роботі з послідовними даними, що мають різну довжину, та покращити якість результатів завдяки використанню механізму уваги.

Декілька найбільш популярних моделей, які використовують архітектуру трансформерів, включають:

- BERT (Bidirectional Encoder Representations from Transformers) - ця модель використовує багатошаровий бідирекційний трансформер для виконання різноманітних завдань NLP, таких як класифікація тексту, розпізнавання іменованих сутностей та машинний переклад. BERT отримав дуже високі результати в багатьох завданнях NLP, завдяки своїй здатності ефективно використовувати контекстуальну інформацію у тексті.
- GPT (Generative Pre-trained Transformer) - ця модель використовує енкодер-декодер архітектуру трансформеру для генерації тексту. Вона була натренована на великій кількості текстів з Інтернету, і може генерувати зрозумілі інтерпретації тексту.
- XLNet - це модель, яка покращує BERT, дозволяючи моделі використовувати контекстуальну інформацію в будь-якому порядку, а не лише зліва направо. Це дозволяє моделі використовувати контекст з більшої кількості речень, що дозволяє досягти кращої точності.
- RoBERTa - це модель, яка є покращенням BERT, збільшуючи розмір моделі та тренуючи її на більшій кількості даних. Вона досягає кращої точності, ніж BERT в багатьох завданнях NLP, завдяки збільшенню кількості параметрів.

2.3. Попереднє порівняння моделей різних архітектур

Було знайдено дослідження [23], які порівнюють результати деяких вище зазначених моделей для класифікації новин. Але автори не зовсім детально описують використані датасети та метод тестування. Тому дані наведені для прикладу, остаточній експеримент буде проведено далі. До нього також планується додати найсучасніші моделі RNN та Трансформери. На рисунку 9 бачимо деякі вище згадані архітектури та точність на вертикальній осі.

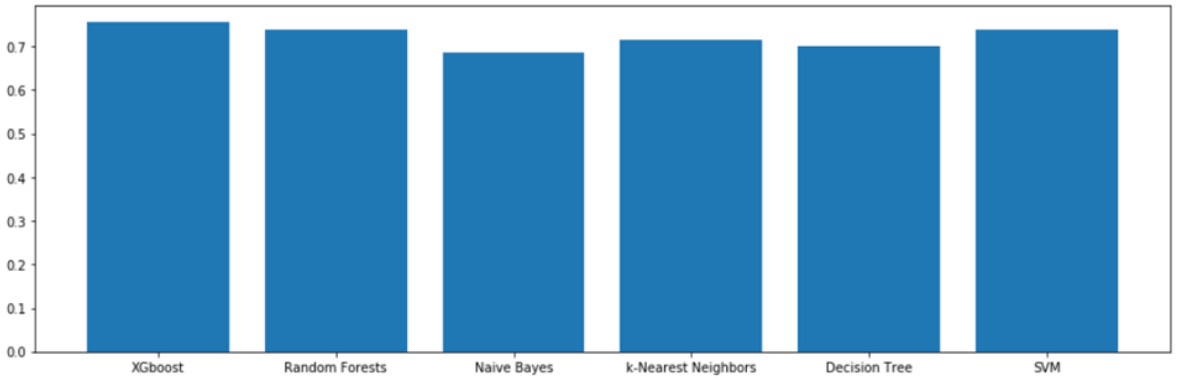

Рисунок 9 — Точність моделей з дослідження [23]

2.4. Огляд готових автоматизованих рішень виявлення фейкових новин

Далі розглянемо декілька сучасних досліджень, які стосуються виявлення фейкових новин, проаналізуємо детально підходи, їх переваги та недоліки, щоб у подальшому запропонувати покращене рішення.

2.4.1. Fakes News Detection Using Machine Learning Approaches

В цій статті [18] описується стандартний підхід до машинного навчання. Автор має певний власноруч створений датасет, тобто новини перевірені та відповідний клас призначений власноруч автором. Після чого

цей датасет розділяється на тренувальний та тестувальний. Тестується декілька моделей вище згаданих сімейств, та декілька їх покращених версій чи не згаданих. Загалом всі вони показують доволі схожі результати.

Цей підхід є найбільш простим та стандартним в машинному навчанні але одразу має декілька недоліків:

- Датасет створений власноруч за незрозумілим алгоритмом, може мати помилки.
- Такий підхід обмежує використання лише схожих за темою новин та за певний проміжок часу, що власне і не буде новинами, тобто це просто класифікація текстових даних
- З попередього випливає, оскільки датасет ніяк не використовує раніше згадані бази даних, знань та перевірених фактів з надійних джерел, він буде вести себе нестабильно та не матиме достатньої експертизи на нових даних, власне навіть на даних, з того самого розподілу, 0.7 точності є далеко не найкращим результатом, тому , наприклад, з використанням найсвіжіших даних з твітеру, його результати навряд чи будуть ліпшими за випадкове передбачення.

Тобто ця стаття та її дані можуть використовуватися як бенчмарк для моделей, але це дослідження не має ніякого сенсу для справжніх даних з соціальних мереж. Розглянемо інші статті, для покращення результатів у реальних умовах.

2.4.2. Predicting Factuality of Reporting and Bias of News Media Sources

Стаття [24] описує дослідження щодо прогнозування правдивості повідомлень та упередженості джерел новинних ЗМІ. Автори експериментують з великим списком новинних веб-сайтів і багатим

набором функцій, отриманих із багатьох джерел, таких як вибірка статей із цільового медіа новин, його сторінка у Вікіпедії, його обліковий запис у Twitter, структура його URL-адреси та інформація про веб-трафік, який він залучає. Експериментальні результати демонструють значний вплив деяких окремих атрибутів на передбачення та їх зв’язок з правдивістю джерела та його розташування на політичних координатах.

Це дослідження важливе, оскільки воно зосереджено на характеристиці цілих джерел новинних ЗМІ, що недостатньо вивчена, але, є важливою дослідницькою проблемою та може відігравати велику роль у створенні таких систем перевірки фактів та автоматизації цього процесу. Він не зовсім підходить для онлайн передбачення правдивості повідомлення, оскільки потребує доволі багато даних, але може використовуватися для автоматичної перевірки джерела та додавання його у певну базу знань, що може стати важливим кроком для класифікації інформації з соціальних мереж. Прогнозування правдивості повідомлень і упередженості для певного засобу масової інформації може бути важливим внеском у цю роботу.

2.4.3. Compare to The Knowledge: Graph Neural Fake News Detection with External Knowledge

Стаття [25] «Compare to The Knowledge: Graph Neural Fake News Detection with External Knowledge» представляє графову нейронну модель під назвою CompareNet для виявлення фальшивих новин. Модель порівнює новини з базою знань (KB). Експериментальні результати на двох контрольних наборах даних демонструють, що CompareNet значно перевершує сучасні методи. Це дослідження робить внесок у сферу виявлення фейкових новин, пропонуючи новий метод, який ефективно включає зовнішні знання та теми для підвищення точності виявлення.

2.4.4. Automated Fake News Detection using cross-checking with reliable sources

В цій статті [26] описується процес використання зовнішніх джерел. У конкретному випадку використовуються твіти від надійних ЗМІ, наприклад таких як: Reuters, BBC, Associated Press. Є певний датасет з trustworthy sources (твіти, новини не зовсім важливо, але це повинна бути доволі велика база, що буде містити багато релевантної інформації чи хоча б буде інформаційно вирівняна з можливим датасетом із твітів). За певну дату береться твіт (можливо і без дати але постають питання швидкої роботи алгоритму, як варіант для поліпшення можливо спробувати використання пошукових рушіїв), у вікні тижня від цієї дати (3 дні назад та 3 дні вперед) беруться шматки інформації з trustworthy джерел. Попередньо відбувається кластер усіх надійних джерел, це робиться для того, щоб для майбутніх твітів, знаходити надійні твіти схожі за змістом та темою. Має сенс крім кластерингу використовувати cosine similarity[27]. Після чого призводиться перевірка таргет твіта з іншими з цього кластеру, чи найближчими з них на рівень схожості семантично, сентиментально та емоційно. Ця інформація дає змогу розпізнати, чи відповідає новина змісту від надійних джерел та чи вся інформація та емоційне забарвлення збігається. Для цього можна використовувати будь-які NLP моделі. Після чого відбувається прийняття остаточного рішення методом голосування.

2.5. Висновки до розділу

Звернемося до останньої проаналізованої статті. Вона найбільш схожа до теми дипломної роботи, та саме такий підхід будемо спробувати реалізувати та запропонувати різноманітні покращення до цього підходу. Поліпшення цієї моделі полягає в тому, що використовуючи такий підхід є безліч варіантів для оптимізації швидкості та точності роботи цього

алгоритму з використанням різних моделей, варіантів кластеризації чи пошуку за базою даних. Є також широка можливість вибору даних, оскільки як було зазначено вище, існують багато інформаційних систем для перевірки фактів, проводилися дослідження для знаходження найбільш правдивих, авторитетних та неупереджених ЗМІ. З використанням сучасних обчислювальних паралельних систем парсінгу можливо зробити ці базу справді великою, коригувати її наповнення з різних джерел, та перевіряти ці джерела за вище зазначеними алгоритмами.

Related Work Description			
Article	Ensemble	Features	Effective Zone
Kwon et al.	102 news topics, each containing at least 60 tweets and their diffusion networks	Characteristics of diffusion network	The collection of tweets that the diffusion network is available for them
Helmstetter et al.	Tweets from a trustworthy source labeled as real news, and tweets from an untrustworthy source labeled as fake news	User, text, and tweet features	Fake news tweeted from users who almost always tweet fake
Buntain et al.	CREDBANK and PHEME datasets	Tweet, characteristics of diffusion network, user, text features	when we have the structure of tweets, many tweets with the same topic, and users who spread only fake or real most of the time.
Agarwal et al.,2019	12.8 K manually labeled short statements	Text features	Since most of the tweets are political, works better in political tweets, but otherwise can be used for other ensembles
Our Approach	4k random tweets	Text features	Topics which are mentioned in reliable sources

Рисунок 10 — Порівняння декількох підходів зі статті [26]

На рисунку 10 бачимо порівняння декількох підходів з наборами даних. Розглянемо їх докладніше з перекладом та описом:

Робота Kwon та інших дослідників розглядала 102 теми новин, які включали не менше 60 твітів та їхні дифузійні мережі. Основними характеристиками, використаними в цій роботі, були характеристики дифузійної мережі, що складалася з твітів, доступних для цієї мережі.

Робота Helmstetter та співавторів використовувала твіти від надійного джерела, які були позначені як реальні новини, та твіти від ненадійного джерела, які були позначені як фейкові новини. Основними

характеристиками в цьому підході були характеристики користувача, тексту та твіту.

Робота Buntain та інших використовувала набори даних CREDBANK та PHEME. У цьому підході розглядались характеристики твітів, характеристики дифузійної мережі, характеристики користувача та текстові особливості. Цей підхід працював ефективно, коли були дані були певної структури, багато твітів на одну тему та користувачів, які поширюють переважно фейкові або реальні новини.

Робота Agarwal та інших дослідників використовувала 12,8 тисячі ручно позначених коротких тверджень. Основною характеристикою в цьому підході були текстові особливості. Цей підхід працював краще у політичних твітах, оскільки більшість з них були політичного характеру, але також може застосовуватися і до інших типів твітів.

Опишемо детальніше ці підходи до класифікації новин. Для передбачення використовуються фічі, які позначають характеристику топології зв’язків користувача з іншими у соціальних мережах, інформація про користувача, довжина твіту, особливості твіта. Але опис “ефективної зони” дає змогу зрозуміти, що ці моделі можуть бути проблемними при використанні на нових даних чи на інших темах твітів, що може особливо помітним при використанні на реальних даних, коли твіт пише людина, яка звернулася до ненадійного джерела за помилкою, чи не має розвинені соціальні зв’язки. Саме тому ідея використовувати найбільш природній спосіб перевірки інформації, звертаючись до джерел роблячи це, майже так само, як би це зробила людина чи експерт в цьому полі, відчувається найбільш потужною, та широким полем для експериментів та різноманітних поліпшень, як і частини що стосується ML так і частини що стосується якості даних.

Тому для наступної практичної реалізації будемо використовувати схожий підхід, ставити експерименти з використанням різних моделей та даних, для внесення додаткового вкладу у цю сферу та для покращення запропонованого підходу.

3. ОТРИМАННЯ ТА ОБРОБКА ДАНИХ

Після детального вивчення можливих підходів до створення системи, та створення свого власного та покращеного, треба проаналізувати наявні набори даних, які можуть використовуватися для цього завдання, які будуть достатньо відповідати вимогам обраного підходу до моделі та мати достатньо різних тем повідомлень у собі, для перевірки моделі на різноманітних даних, достатню кількість даних, для достатнього тренування та тестування з цього набору.

Окремою проблемою може стати достовірність даних, їх точність розмітки та власне саме отримання та їх обробка. Розпочнемо аналіз, відшукаємо один чи декілька датасетів, які можуть бути застосовані до поточної проблеми. Після чого завантажимо обрані та зробимо попередню обробку та їх очищення. Проаналізуємо їх та подивимось, що вже, без використання машинного навчання можна зрозуміти про ці дані.

Крім того, ми дослідимо можливості використання додаткових характеристик тексту, таких як емоції, тон та сентимент, для покращення ефективності моделей. Отримані результати дослідження допоможуть зрозуміти можливості та обмеження підходів до виявлення фейкових новин з використанням аналізу тексту, а також знайти шляхи покращення ефективності таких моделей та отримання даних.

3.1. Оглядних наявних наборів даних

Table 1: Comparison with existing fake news detection datasets

Features / Dataset	News Content		Social Context				Spatiot. Information	
	Linguistic	*Visual*	*User*	*Post*	*Response*	*Network*	*Spatial*	*Temporal*
BuzzFeedNews	✓							
LIAR	✓							
BS Detector	✓							
CREDBANK	✓		✓	✓			✓	✓
BuzzFace	✓			✓	✓			✓
FacebookHoax	✓		✓	✓	✓			
FakeNewsNet	✓	✓	✓	✓	✓	✓	✓	✓

Рисунок 11 — Попереднє порівняння готових наборів даних [26]

На рисунку 11 маємо таблицю яка складається з основних наборів даних та їх характеристик, опишемо їх детальніше.

News Context (Контекст новин):

- Visual (Візуальний): Набор даних містить зображення та ілюстрації, що супроводжують новини.
- Linguistic (Лінгвістичний): Набор даних містить текстові дані, такі як заголовки та описи новин.

Social Context (Соціальний контекст):

- User (Користувач): Набор даних містить інформацію про авторів новин та їх атрибути.
- Post (Публікація): Набор даних містить дані про саму публікацію новини.
- Response (Відповідь): Набор даних містить дані про реакції користувачів на новину.
- Network (Мережа): Набор даних містить інформацію про зв'язки між користувачами.

Spatiot, Information (Просторова інформація):

- Spatial (Просторовий): Набор даних містить просторову географічну інформацію.
- Temporal (Часовий): Набор даних містить часову інформацію новини/запису.

Зі статті бачимо (рис. 11) приблизний огляд найбільш поширених наборів даних, оскільки диплом саме стосується обробки природної мови, то нас не зовсім цікавить наявність візуального контенту, але поки що найбільшим фаворитом є останні нижні набори даних, які стосуються отриманої інформації про пости користувачів з соціальних мереж, що може бути найбільш вірогідним сценарієм використання даної системи. Зробимо більш детальний аналіз для пошуку інших, можливо додаткових наборів даних, які можуть використовуватися для факт чекингу так і для навчання.

3.1.1. LIAR

Це набір даних [28] для виявлення фейкових новин, який включає 12,8 тисяч коротких фраз, позначених вручну за чесністю, темою, контекстом/місцем, спікером, статусом, вечіркою та минулою датою. Він збирався з POLITIFACT.COM протягом десяти років і оцінювався редакторoми politifact.com на його правдивість. Набір даних також можна використовувати для перевірки фактів.

Цей датасет хоча і має бути доволі якісним, нажаль, є більш політично спрямованим, що трохи обмежує його використання у соціальних мережах для аналізу інформації наданої користувачами. Частково ці дані можуть бути використані задля отримання бази знань щодо певних політичних подій, джерел, фраз, особистостей. Але не зовсім

підходять для вдалого застосування на різноманітні теми, об’єм теж є проблемною його стороною.

3.1.2. BuzzFace

Набір даних [29] BuzzFace — це колекція коментарів Facebook, що обговорюють вміст новин, опублікованих у вересні 2016 року. Він складається з майже 1,7 мільйона коментарів Facebook, коментарів плагінів Facebook, коментарів плагінів Disqus і пов’язаного вмісту веб-сторінок із новинними статтями. Набір даних зосереджений на новинах, які анотовані щодо правдивості. Набір даних був створений командою BuzzFeed в результаті дослідження, детально описаного в: Hyperpartisan Facebook Pages Are Publicing False And Leading Information At Alarming Rate. Потенційне використання даних включає оцінку достовірності новин за допомогою машинного навчання, виявлення соціальних ботів і дослідження розповсюдження інформації через кілька різних платформ.

Цей набір даних є доволі цікавим, об’ємним та може використовуватися для дипломної роботи. Проаналізуємо подальші, та зробимо остаточний висновок.

3.1.3. Fakeeddit

Fakeddit — це набір даних [29] фальшивих і справжніх публікацій із Reddit. Він містить понад 170 тисяч публікацій із такими метаданими, як автор, subreddit, дата тощо. Набір даних було створено шляхом поєднання двох існуючих наборів даних: Reddit і LIAR. Набір даних було створено, щоб допомогти дослідникам розробити моделі, які можуть виявляти фейкові новини на платформах соціальних мереж, таких як Reddit. Тобто цей набір даних, там само має змогу використовуватися для аналізу

фейкової інформації в соціальних мережах. Але знову ж, поєднання лише з LIAR трохи обмежує його можливості.

3.1.3. NELA-GT-2019

Набір даних [29] NELA-GT-2019 — це великий набір даних новин із кількома мітками для вивчення дезінформації в новинних статтях. Він містить 1,12 мільйона новинних статей із 260 джерел, зібраних у період з 1 січня 2019 року по 31 грудня 2019 року. Джерела походять із широкого кола основних джерел новин та альтернативних джерел новин. Набір даних включає базові позначки правди на рівні джерела з 7 різних сайтів оцінки, що охоплюють політичну упередженість, фактичність та інші аспекти.

Цей набір даних так само можна використовувати саме для отримання додаткової інформації про новини, для факт-чекінгу, але він не містить згенерованої користувачами інформації.

3.1.4. FakeNewsNet

FakeNewsNet — це набір даних [29] для дослідження виявлення фейкових новин, який містить новинні статті та дані соціальних мереж, пов’язані з ними. Набір даних було створено шляхом збору статей новин з PolitiFact і GossipCop, двох відомих веб-сайтів перевірки фактів, і відповідних даних соціальних мереж із Twitter. Набір даних включає як справжні, так і фейкові зразки новин.

Набір даних не можна поширювати повністю через політику конфіденційності Twitter і авторські права видавців новин. Соціальні взаємодії та інформація про користувачів не розголошуються відповідно до політики Twitter. Однак у вільному доступі є готовий скрипт, за умови отримання доступу до API твітеру, можна завантажити всі необхідні дані.

3.1.5. FacebookHoax

Цей набір даних [30] містить інформацію, пов’язану з публікаціями на сторінках Facebook, пов’язаними з науковими новинами , і конспірологічними сторінками, зібрану за допомогою Facebook Graph API. Набір даних містить 15 500 публікацій із 32 сторінок (14 конспірологічних і 18 наукових) із понад 2 300 000 лайків

3.2. Обґрунтування вибору набору даних

З усіх наявних датасетів, найбільш цікавим та змістовним виглядає саме FakeNewsNet, тому що можна зазначити, що саме він є найбільш великим, та включає у себе інші датасети. Крім політичних даних існують дані про зірок, що може саме по собі включати багато різних тем, і в той самий час одна тема може зустрічатися у багатьох містах в базі знань, що ставить додаткові виклики для обрання найкращого алгоритму пошуку схожих даних у базі знань. Він є проанотованим, відповідно двох сайтів, які перевіряють факти, що свідчить про те, що він має бути доволі точним, що дозволить оцінити точність більш детально.

Парсинг даних з твітеру, може надати можливість більш детально проаналізувати соціальні зв’язки між користувачами, шукати більш і менш найдіних користувачів, та в цілому дає великий простір для аналізу даних. Найцікавішою його рисою є те, що його має бути не дуже складно отримати власноруч шляхом використання Twitter API та пошуку, що дає можливість створити декілька власних даних більш свіжих та перевірити на них алгоритм. Саме тому вибір пав саме на нього. Тепер детальніше розберемось із його завантаженням.

3.3. Завантаження та аналіз набору даних

Цей датасет знаходиться у вільному доступі щодо статей новин в мережі, він містить таблиці у форматі:

id	news_url	title	tweet_ids
politifact15014	speedtalk.com/forum/viewtopic.php?t=51650	BREAKING: First NFL Team Declares Bankruptcy Over Kneeling Thugs	9373494346684989449373793780062822409373800685900554259373844065110056969373
politifact15156	politics2020.info/index.php/2018/03/13/court-orders-obama-to-	Court Orders Obama To Pay $400 Million In Restitution	9726662814418780169726783965755596809728278199935959049728280142694481929728
politifact14745	www.nscdscamps.org/blog/category/parenting/467344/update-	UPDATE: Second Roy Moore Accuser Works For Michelle Obama Right NOW -	9294057407328706569294394504002641929294394848060375069294846621452083209294
politifact14355	https://howafrica.com/oscar-pistorius-attempts-commit-suicide	Oscar Pistorius Attempts To Commit Suicide	8869415264583475218870113002781941768870230431214346258870716463764480008870
politifact15371	http://washingtonsources.org/trump-votes-for-death-penalty-f	Trump Votes For Death Penalty For Being Gay	9152056982120407049152420766815068169152498793410600969152825846487818249152
politifact14404	gloria.tv/video/yRrtUtTCfPga6cq2VDJPcgQe4	Putin says: â€˜Pope Francis Is Not A Man Of Godâ€™ | Must-SeeÂ !!	8932909006374830098932909507008020488932909877574696978940719229375160328940

Рисунок 12 — Приклад даних FakeNewsNet у “сирому” вигляді

Тобто кожен рядок (рис. 12) містить унікальний ідентифікатор рядку, посилання на новину, назву її статі, айді твітів, які пов’язані з нею.

Сторінка в GitHub[31] містить власне скрипт, який дозволяє з використанням twitter API завантажити усі твіти. Але оскільки датасет містить більш ніж 2 мільйони твітів, а безкоштовний twitter API дає можливість завантажувати не більше ніж 10 000 твітів на місяць (платний на 100$ на місяць трохи більше, але цього все одно замало), що є доволі малим числом задля аналізу даних та машинного навчання. Академічна підписка наразі скасована твітером та розробляється інша система, тому цей варіант не підходить. Тому вирішено було написати власний скріпт парсингу даних[32].

Оскільки всі айді твітів відомі, основна ідея полягала в тому, щоб оптимізувати паралельне завантаження даних з мережі. Це може бути зроблено за допомогою Python MultiThreading[33] та MultiProcessing[34]. MultiThreading дозволяє виконувати декілька потоків в межах одного процесу, ділячись спільними ресурсами. Потоки можуть бути взаємодіючими і спільно використовувати ресурси програми, що зменшує час виконання та ресурсоємність програми. Однак, через спільне використання ресурсів, можуть виникати проблеми з блокуванням, коли один потік блокує доступ до спільного ресурсу іншому потоку.

MultiProcessing, з іншого боку, дозволяє виконувати декілька процесів одночасно, які можуть бути фізично розташовані на різних процесорах чи серверах. Кожен процес має власний адресний простір та незалежні ресурси, що дозволяє уникнути проблем з блокуванням. Проте, процеси не можуть безпосередньо взаємодіяти між собою, що може ускладнювати деякі задачі в порівнянні з потоками.

Наступною проблемою був процес саме отримання даних з твітера за айді. Оскільки твітер сторінка при встановленні з’єднання віддає майже порожній HTML документ з JavaScript, виникали проблеми автоматизувати це лише за допомогою надсилання запитів, оскільки додатково потрібно було мати інтерпретатор JS та запускати через нього код, це б займало надто багато часу тому це варіант було виключено. Через ті самі причини було виключено варіант з використанням Selenium[35], що є бібліотекою для автоматизації веб-браузера, яка може бути використана для тестування, моніторингу веб-сайтів та інших задач, пов'язаних з браузером. У деяких випадках, коли взаємодія з веб-сайтом вимагає використання динамічних елементів, таких як кнопки, поля введення та інші, Python Requests не може виконати запит до веб-сайту, тому що ці елементи відсутні в HTML-коді сторінки.

У таких випадках можна використовувати Selenium, який емулює поведінку користувача в браузері, щоб взаємодіяти з динамічними елементами. Крім того, Selenium може забезпечити взаємодію з елементами, які залежать від JavaScript, такі як AJAX-запити або взаємодія з DOM-деревом сторінки. Оскільки Selenium взаємодіє з веб-сайтом через браузер, він вимагає більшої обчислювальної потужності та ресурсів, ніж Python Requests, який працює безпосередньо з HTML-кодом сторінки. Крім того, використання Selenium може займати більше часу на виконання завдання порівняно з Python Requests.

Вирішенням цих проблем стало використання Embedding API твітеру[36], яке, не має обмежень за кількістю запитів та не потребує реєстрації з підпискою.

python code

```
requests.get(f'https://publish.twitter.com/oembed?dnt=true&omit_script
=true&url=https://mobile.twitter.com/i/status/{tweet_id}')
```

Publish.twitter.com[37] — це веб-сайт, який дозволяє вставляти вміст Twitter на ваш веб-сайт або в програму. Можна вставляти твіти, часові шкали та колекції на свій веб-сайт або в програму за допомогою коду JavaScript Twitter для веб-сайтів. Також можна використовувати картки Twitter для відображення мультимедіа в твітах, які посилаються на ваш вміст. Було знайдено посилання, яке за twit_id дозволяє отримати JSON документ за мережевим запитом, який містить усі основні дані щодо твіту, а саме текст, автора, тощо.

На моєму github [38] розміщено код, який дозволяє завантажувати ці дані, веде лог файл щодо кількості пройдених новин. Усі дані записуються в CSV (Comma Separated Values) формат, що є доволі зручним при використанні з основними, найбільш відомими бібліотеками для завантаження даних для машинного навчання та їх аналізу. Оскільки є готові функції які дозволяють одним рядком завантижити всі ці дані у пам’ять та використовувати різноманітні функції для їх обробки. Саме мова йде про Pandas DataFrame [39], структурований формат даних, який є одним з ключових елементів бібліотеки Pandas. DataFrame представляє табличні дані, які можуть бути зручно оброблені та аналізовані.

DataFrame складається з двох основних складових: індексу та стовпців. Індекс дозволяє доступатися до даних у DataFrame за допомогою міток, тоді як стовпці представляють окремі змінні або атрибути, які мають різний тип даних.

Його можна створити з різних джерел даних, таких як CSV-файли, бази даних, Excel-файли тощо. При створенні DataFrame, кожен рядок таблиці представлений окремим об'єктом, що містить дані з одного рядка вихідного джерела даних. Основна перевага використання Pandas DataFrame полягає в тому, що цей формат дозволяє легко індексувати та фільтрувати дані, а також здійснювати обробку даних у великому масштабі. DataFrame також забезпечує зручний інтерфейс для збереження та експорту даних, що значно спрощує їх подальший аналіз та обробку.

Парсинг датасету відбувався близько п’яти діб. Було отримано достатню кількість твітів, та новин, з якими вони зв’язані, для попередньої обробки та аналізу. А саме кількість твітів складає близько 2 мільйонів для 23 тисяч новин з різних джерел.

Для початку почнемо з коду для з’єднання цих файлів у один для подальшого аналізу. Для цього використовуємо стандартні python бібліотеки для обробки даних та швидких математичних обчислень, які були згадані раніше.

Після цього можна побудувати графік загального розподілу класів новин (фейкові та правдиві) та їх кількості.

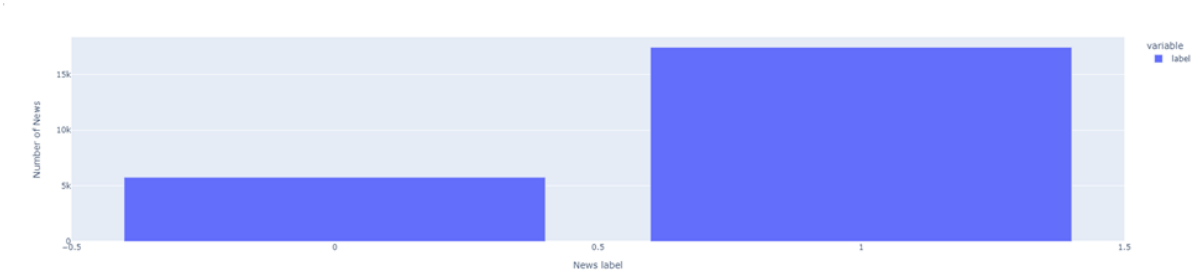

Рисунок 13 — Розподіл класів у новинах

Як бачимо (рис. 13) на осі X зображено клас новини (news label), на осі Y(Number of News) кількість новин. Кількість класів (17 тисяч правдивих новин і 6 фейкових) є доволі незбалансованою, але всього маємо близько 23 тисяч новин, що, враховуючи до декількох сотень твітів на кожну з новин може дати доволі велику кількість різноманітних значень.

Порахуємо максимальну кількість твітів, при вдалому парсингу, маємо значення в 2 526 905, що є доволі великим значенням, що дозволить перевірити велику кількість різноманітних випадків.

Також, враховуючи тему роботи та запропонований алгоритм, порахуємо кількість твітів, які стосуються однієї новини використовуючи агрегаційні функції.

Середня кількість твітів на новину	Перший квартиль (25%)	Медіана	Третій квартиль (75%)
88.95	11	37	65

Таблиця 3.3.1. Статистичні дані за твітами на новину

25-й відсотковий ранг (Q1) та 75-й відсотковий ранг (Q3) визначають діапазон, в межах якого знаходиться середня половина даних, тоді як 50-й відсотковий ранг (Q2) представляє медіану, розділяючи дані на дві рівні частини. Тобто можемо чітко бачити (таблиця 3.3.1.), що кожна новина містить хоча б декілька твітів, що дозволить навчати модель використовуючи підхід перехресної перевірки джерел.

Оскільки в наборі даних міститься url адреса новини, можемо розпарсити url новини та подивитися, які основні джерела новин використовуються, та оцінити їхню достовірність.

Спробуємо розрахувати відносну кількість класів новин та абсолютну кількість новин для кожного джерела та виведемо найбільш часто вживані, та розподіл в них по класам.

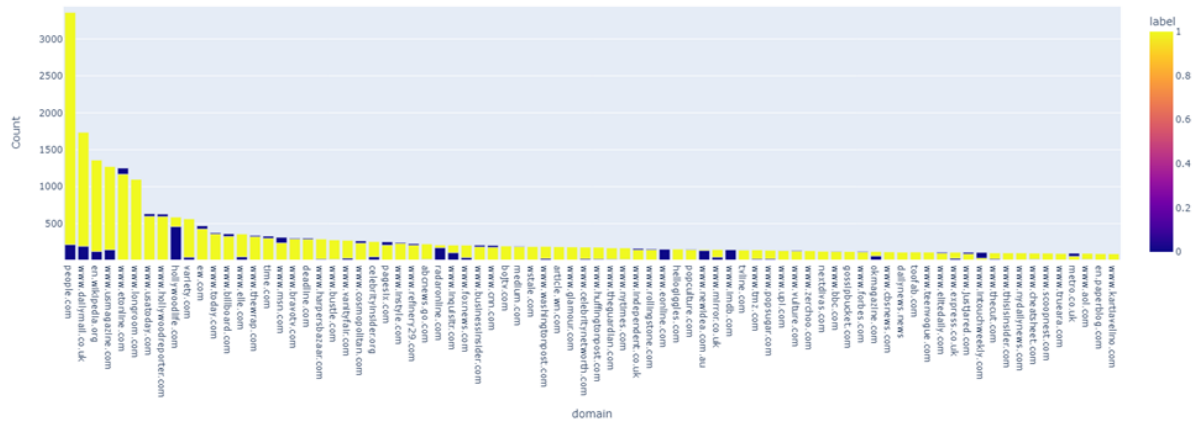

Рисунок 14 — Розподіл фейкових/справжніх новин в найчастіше цитованих доменах

Бачимо (рис. 14), що зазвичай фейкові новини складають орієнтовно до 5-10% від усіх новин з сайту, але є виключення. Але чи зберігається ця тенденція щодо сайтів, які цитуються рідше?

Прораховуємо частоти появи обох класів для кожного сайту.

Отримаємо відсортований (рис. 15), за відносною кількістю фейкових новин список сайтів. Спочатку перевіримо сайти, які містять більше 15 згадок.

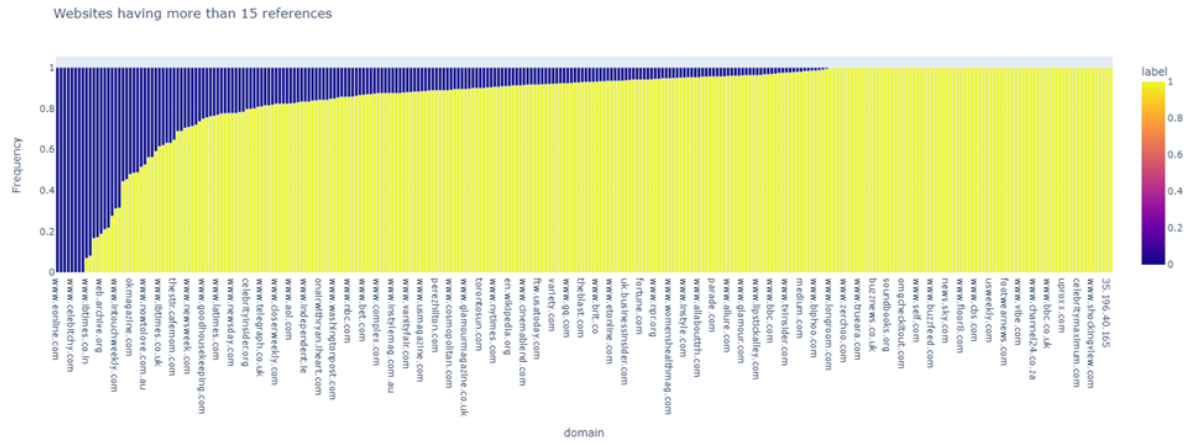

Рисунок 15 — Відсортовані за відносною кількістю новин домени, які зустрічаються більше 15 разів

Тепер перевіримо розподіл для менше ніж 15 згадок про сайт в датасеті.

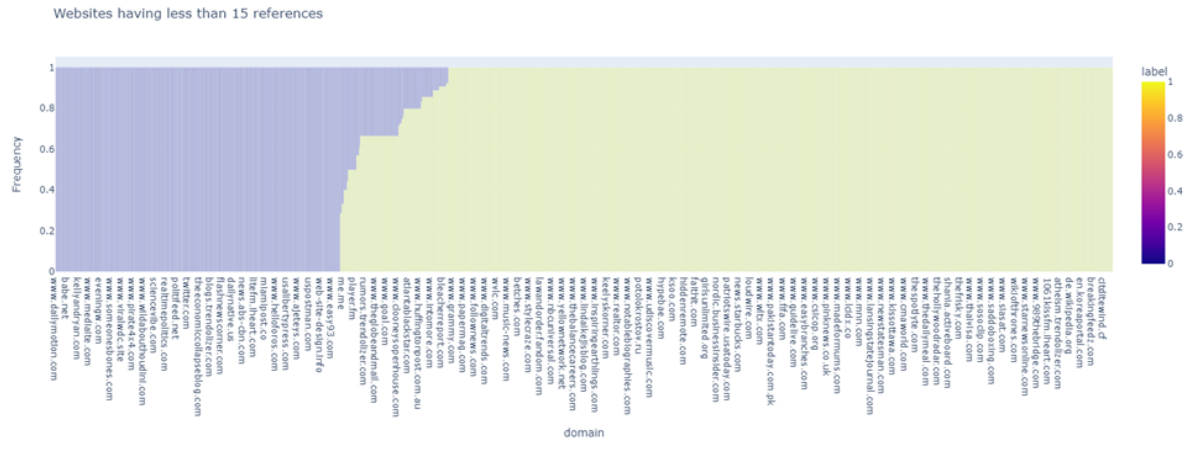

Рисунок 16 — Відсортовані за відносною кількістю новин домени, які зустрічаються менше 15 разів

Як бачимо (рис. 16), сайтів, які містять лише фейкові новини вже значно більше, можливо це статистична похибка, але можемо бачити певну закономірність, що серед сайтів які частіше цитуються, тобто скоріш за все є більш популярними, значно менша кількість має стовідсоткову ймовірність запостити фейкову новину.

Цей висновок звісно потрібно підтверджувати вже напряму аналізуючи дані сайти, та отримувати більш великий датасет, але це може наштовхнути на певні висновки та рішення. Тобто скоріш за все, чим менш відомий сайт, тим більша ймовірність, що він може намагатися набрати аудиторію шляхом випуску фейкових новин, але враховуючи загальну кількість посилань на них, схоже за все, що ця стратегія не дуже працює. Таким чином висуваємо гіпотезу, що частіш цитовані сайти мають меншу схильність до випуску фейкових новин.

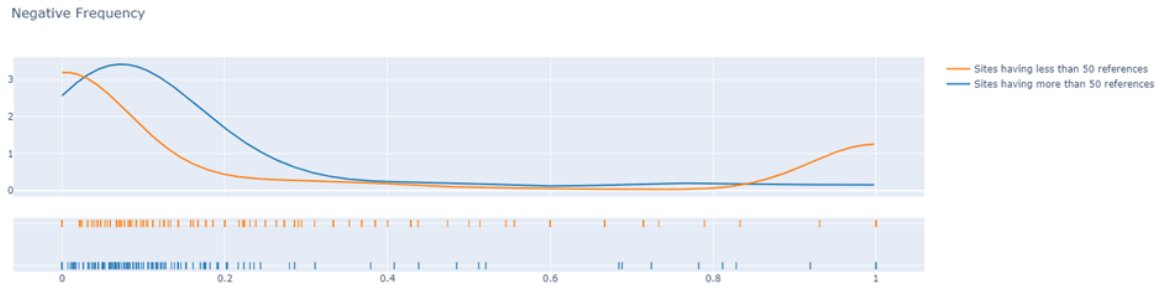

Рисунок 17 — Графік KDE для різних категорій сайтів

Як бачимо (рис. 17), гіпотеза лише підтверджується. Більш популярні сайти мають тенденцію постити більше правди, основні значення фейкових новин на таких сайтах знаходяться в межах до 30% фейкових новин. Що стосується менш популярних сайтів, то таких, що постять лише фейкові новини значно більше.

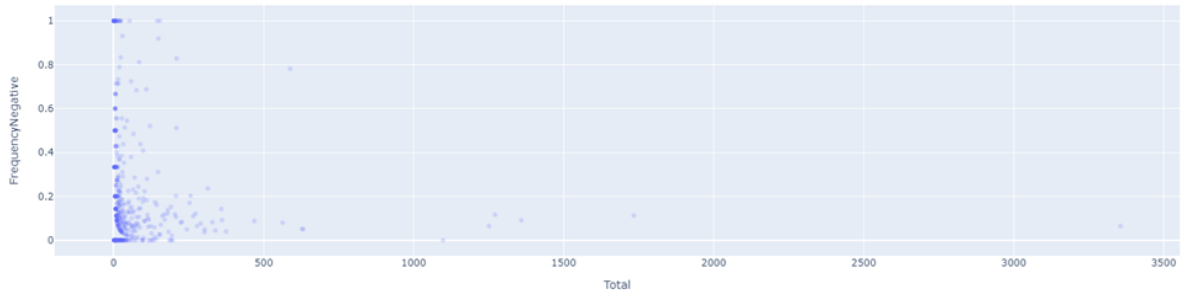

Рисунок 18 — Scatter plot

Ще раз (рис. 18) підтверджуємо отримані висновки з використанням scatter plot.

Перейдемо до обробки текстових даних та отримання з них додаткової інформації на етапі аналізу.

Для цього виведемо частоту використання слів для різних класів, слова які найчастіше зустрічаються для окремих класів. Зробимо такий аналіз саме для новин, потім, аналогічним чином проведемо аналіз саме для твітів.

Для обробки даних твітів використовуємо такий самий код, що і для новин[40], додаємо лише видалення посилань, оскільки зазвичай твіти отримані таким чином містять посилання на джерела чи на медіа, яке йде у твіті.

Для кращого розуміння даних, оскільки вони виходять з різних джерел, проаналізуємо окремо політичні дані та дані з сайту про знаменитостей.

Бачимо найбільш вживані слова:

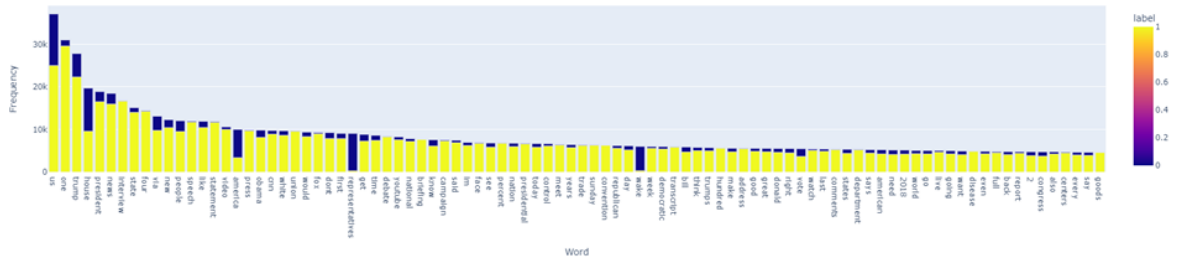

Рисунок 19 — Найбільш вживані слова

Також для слів які зустрічаються більше 4500 разів виведемо їх зустріч у різних твітах за класом, можна бачити (рис. 19) за якими темами найбільше всього фейків.

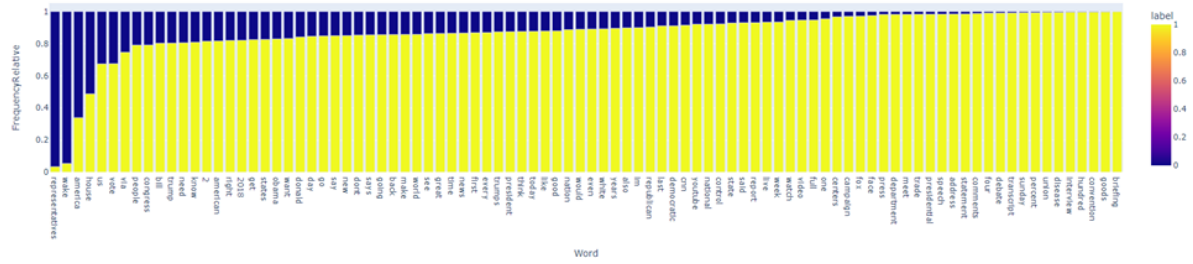

Рисунок 20 — Найбільш “фейкові” з популярних слів

Аналогічно (рис. 20) для наступного набору даних.

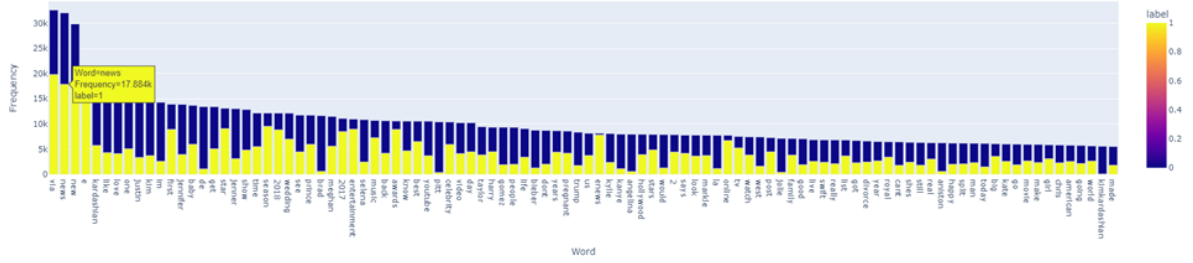

Рисунок 21 — Найбільш вживані слова

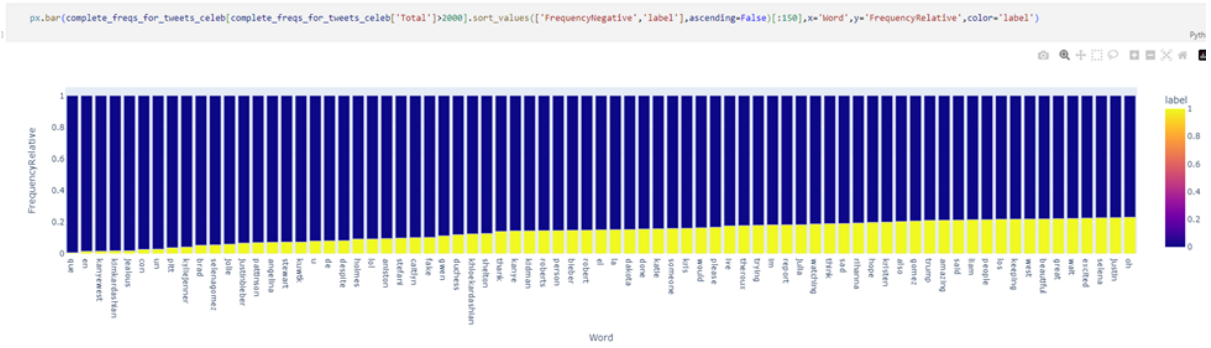

Рисунок 22 — Найбільш “фейкові” з популярних слів

Можемо бачити слова найбільш вживані, та їх загальний зв’язок з твітами та класами новин.

Також виведемо частоти в вигляді word cloud в наборі даних з politifact [41], де розмір слова залежить від того, наскільки часто воно зустрічається.

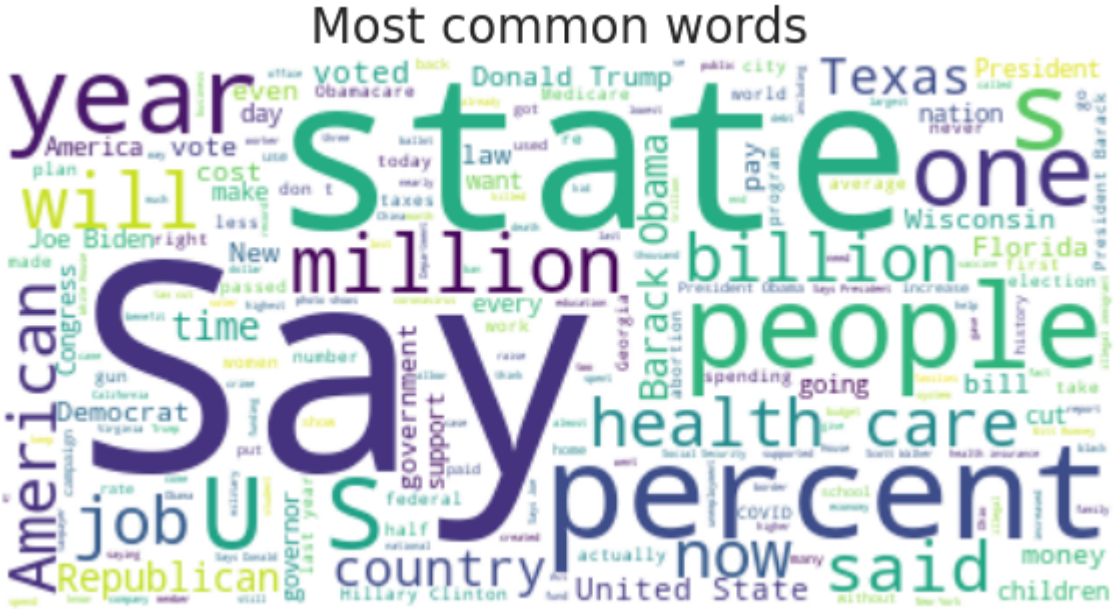

Рисунок 23 — Найбільш вживані слова[41]

Рисунок 24 — Найбільш вживані слова в реальних новинах[41]

Рисунок 25 — Найбільш вживані слова в фейкових новинах[41]

3.4. Висновки до розділу

Було проведено аналіз декількох наборів даних, які використовуються для виявлення фейкових новин. Одним з основних

завдань було визначення частоти вживання певних слів, які можуть вказувати на те, що новина є фейковою. Наприклад, такі слова, як "people", "trump" можуть вказувати на те, що новина є фейковою.

Окрім того, ми проаналізували складність парсингу даних, зокрема, труднощі, які можуть виникнути при зборі даних з різних джерел. Наприклад, зіставлення даних з різних джерел може бути складним, оскільки вони можуть містити суперечливу інформацію або не однакову структуру, окремою проблемою було завантаження даних з твітера.

Було обрано найкращий з можливих наборів даних, який містять достатньо інформації для виявлення фейкових новин та мають досить високу якість даних. Також було використано декілька методів та бібліотек Python для аналізу та обробки цих даних.

Загалом, наші дослідження показали, що вибір наборів даних та аналіз даних - це важлива складова виявлення фейкових новин. Наші результати можуть бути корисними для досліджувачів, які працюють у цій області та можуть допомогти виявляти фейкові новини та зменшувати їх вплив на громадськість.

4. ВИКОРИСТАННЯ РІЗНИХ МОДЕЛЕЙ ДЛЯ ВИЗНАЧЕННЯ ФЕЙКОВИХ НОВИН

Зважаючи на швидкий розвиток сучасних технологій, обробка природної мови є одним з найбільш актуальних напрямів досліджень у сфері комп'ютерних наук. Одним із головних завдань цього напрямку є класифікація текстів. У зв'язку з цим, багато дослідників зосереджуються на тренуванні різних моделей для обробки природної мови, щоб отримати кращі результати в класифікації.

Тепер після детального аналізу предметної області, підготовки та аналізу даних, потрібно провести експеримент з тренування різних моделей для обробки природної мови з метою класифікації текстів. Ми дослідимо результати, щоб зрозуміти, яка модель є найкращою для цієї задачі. Підхід полягатиме в аналізі різних моделей та їх оцінці на основі вимірів точності та швидкості, розмірі моделі, різноманітних метрик, та готовності моделі до класифікації на нових даних. Для цього ми використовуємо зразкові дані та забезпечуємо розбиття на навчальний та тестовий набори даних.

Після проведення експерименту ми отримаємо результати, які мають, після аналізу результатів, підтвердити та зрозуміти, які моделі працюють краще за інших в залежності від задачі класифікації та об'єму даних. Результати дадуть можливість оцінити кращі моделі для класифікації текстів та допоможуть в подальших дослідженнях у цій області.

4.1. Підготовка інструментів машинного навчання

Для досягнення кращої продуктивності та прискорення процесу навчання, використання CUDA [42] (Compute Unified Device Architecture)

може бути важливим кроком. CUDA - це платформа обчислювальних можливостей NVIDIA, яка дозволяє використовувати графічні процесори (GPU) для виконання обчислювальних завдань. У цьому розділі ми розглянемо процес встановлення та налаштування CUDA для прискорення локального навчання моделей.

Першим етапом є перевірка вимог системи. Локально використовується RTX 3050, що має одну з останніх та сучасних архітектур графічних процесорів. Вона підтримує CUDA обчислення, хоча і має не дуже багато пам’яті по сучасних мірках, але може бути використана для пришвидшення навчання моделей з кількістю параметрів до декількох мільярдів, локально з невеликим batch size.

Далі встановлюємо CUDA Toolkit. Завантажено останню версію CUDA Toolkit з офіційного веб-сайту NVIDIA з підтримкою PyTorch. З використанням Windows встановлення трохи ускладнюється тим, що для виконаного файлу драйверів, додатково треба додати його у змінну оточення PATH власноруч, для забезпечення доступу до утиліт та бібліотек.

Для перевірки встановлення використовуємо CLI. Перевіряємо версію та сумісність з відеокартою.

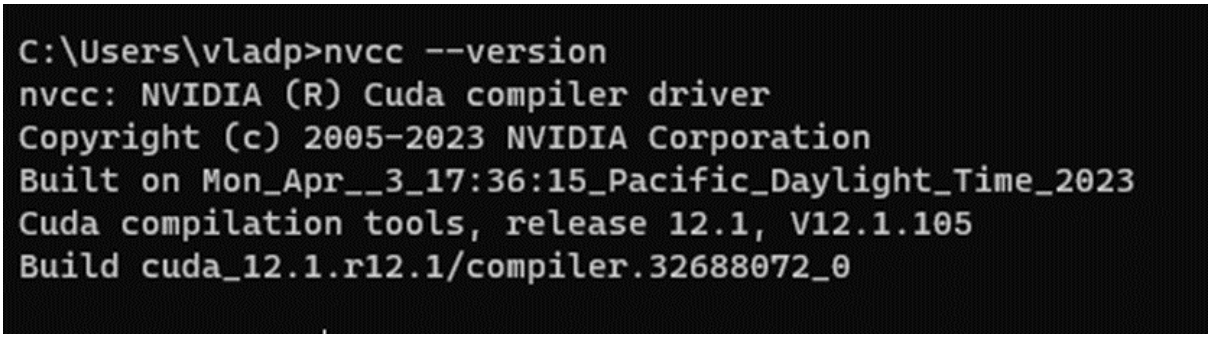

Рисунок 26 — Перевірка правильного встановлення CUDA локально

```
C:\Users\vladp>nvidia-smi
Thu May 18 20:21:09 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 531.79                 Driver Version: 531.79       CUDA Version: 12.1     |
|-------------------------------------+----------------------+----------------------+
| GPU  Name                      TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf            Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                                     |                      |               MIG M. |
|=====================================+======================+======================|
|   0  NVIDIA GeForce RTX 3050 L...  WDDM | 00000000:01:00.0 Off |                  N/A |
| N/A   47C    P0               13W /  N/A|      0MiB /  4096MiB |      0%      Default |
|                                     |                      |                  N/A |
+-------------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Рисунок 27 — Перевірка правильного виявлення GPU

Було встановлено pytorch-gpu [43]. Автоматичним чином ця бібліотека знаходить інформацію про CUDA та сумісний адаптер в системі.

```
[6]
torch.version.cuda

'11.8'
```

```
[7]
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

```
[8]
if device.type == 'cuda':
    print(torch.cuda.get_device_name(0))
    print('Memory Usage:')
    print('Allocated:', round(torch.cuda.memory_allocated(0)/1024**3,1), 'GB')
    print('Cached:   ', round(torch.cuda.memory_reserved(0)/1024**3,1), 'GB')

NVIDIA GeForce RTX 3050 Laptop GPU
Memory Usage:
Allocated: 0.0 GB
Cached:    0.0 GB
```

Рисунок 28 — Перевірка роботи в Python Pytorch

4.2. Навчання та порівняння моделей

Опишемо кожну модель, код, підхід до навчання. Після чого в агреговану таблицю зведемо усі результати, графіки.

4.2.1. Logistic Regression з TF-IDF

Для оцінки ефективності моделей машинного навчання у задачі виявлення фейкових новин, необхідно виміряти різні метрики, такі як F1-міра, Accuracy, ROC AUC (receiver operating characteristic area under curve).

ROC-крива візуалізує зв'язок між чутливістю (Sensitivity) (також відомою як True Positive Rate) та специфічністю (Specisifity) (1 - False Positive Rate) моделі для різних значень порогу (threshold). Уявимо, що у нас є континуум значень порогу, і для кожного значення порогу ми обчислюємо чутливість та специфічність. Потім ми з'єднуємо ці точки, і отримуємо ROC-криву. Ідеальна модель буде мати ROC-криву, яка піднімається дуже швидко вгору, тоді як випадкова модель буде мати ROC-криву, що просто перетинає діагональ.

AUC - це площа під ROC-кривою. Він вимірює загальну ефективність моделі. Чим ближче AUC до 1, тим краще модель робить передбачення. AUC 0.5 відповідає випадковій моделі, а значення більше 0.5 вказує на те, що модель краща за випадковий вибір.

Отже, ROC-крива та AUC надають нам інформацію про здатність моделі розрізняти класи та можливість оцінити її ефективність в класифікації. Чим вище AUC, тим краще модель виконує свою роботу.

Крім того, перед навчанням моделі потрібно виконати обробку текстових даних, щоб перетворити їх на числові представлення. В цьому

розділі ми розглянемо імплементацію основних функцій для цих метрик машинного навчання та обробку текстових даних з використанням алгоритму TF-IDF з бібліотеки sklearn.

Обробка текстових даних з використанням алгоритму TF-IDF:

Після успішної імплементації функцій метрик машинного навчання та обробки текстових даних з використанням алгоритму TF-IDF з бібліотеки sklearn, проводимо навчання та отримаємо результати.

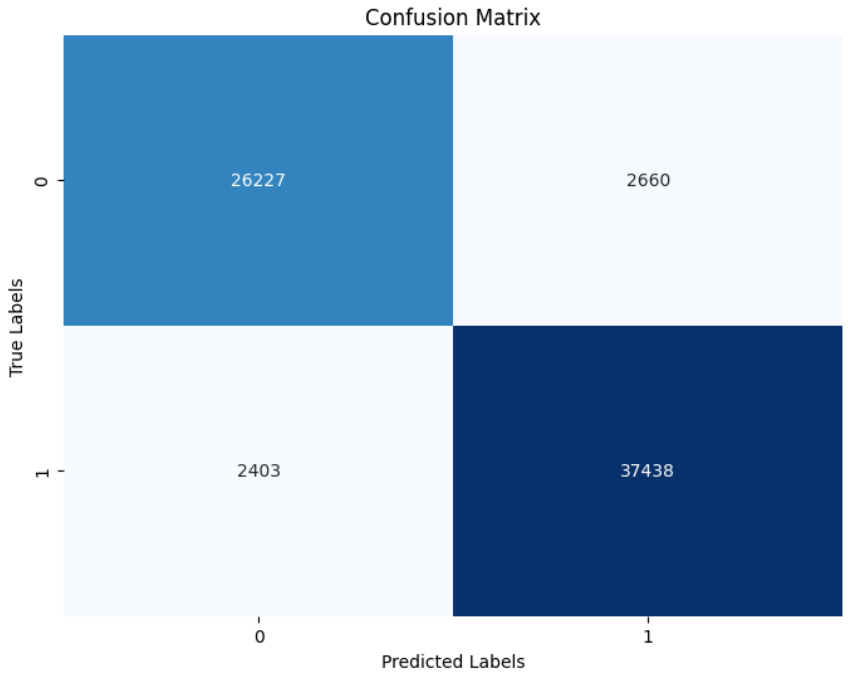

Рисунок 29 — Результати логістичної регресії з TF-IDF

На рисунку 29 і далі будемо бачити раніше описаний підхід з confusion matrix, тобто матриця передбачень моделі, що дозволяє непогано проаналізувати помилки,

4.2.2. Logistic Regression з Doc2Vec

Для ефективної обробки текстових даних і використання їх у моделі машинного навчання, можна використовувати алгоритм Doc2Vec. Він

дозволяє отримати векторне представлення для кожного документа (набору тексту), що зберігає семантичну інформацію, яка відображає його зміст. У цьому розділі ми розглянемо використання алгоритму Doc2Vec для обробки текстових даних і подальше використання отриманих векторних представлень для навчання моделі логістичної регресії.

Проходимо ті самі кроки: завантажуємо набір текстових даних, розділяємо на тестовий і тренувальний. Обробляємо з використанням бібліотеки Doc2Vec (gensim)[44]. Імпортуємо необхідні класи. Перетворюємо кожен текстовий документ у формат TaggedDocument, де кожен документ представляється як список слів з відповідною міткою. Ініціалізуємо об'єкт Doc2Vec з необхідними параметрами, розмірність embedding векторів та кількість епох навчання. Викликаємо метод build_vocab на об'єкті Doc2Vec, щоб побудувати словник слів на основі підготовлених даних.

Цей метод [44] дозволяє перетворити цілий документ (такий як речення, абзац, стаття) у векторну представлення. Спочатку Doc2Vec будує словник, який містить усі слова з нашого набору даних. Воно також присвоює кожному слову унікальний ідентифікатор або індекс. Потім Doc2Vec проходить по кожному документу у наборі даних. Воно розглядає контекст кожного слова в документі, тобто слова, які оточують його. Наприклад, якщо розглядається слово "кіт" у документі, то контекстом можуть бути слова "муркотіти", "пухнастий" або "тварина". Після збирання контексту Doc2Vec намагається побудувати векторне представлення для кожного документа, це може бути елементарно сумою всіх векторів слів, чи можливо більш складна функція. Це вектор - числовий рядок, який кодує семантику документа. Документи, які мають схожі слова та контекст, будуть мати близькі векторні представлення. Для побудови векторів Doc2Vec використовує нейронну мережу. Вона

навчається зв'язувати слова та їх контекст у векторах. Після навчання мережа може генерувати векторні представлення для нових документів, навіть якщо вона не бачила їх раніше. Отже, Doc2Vec дозволяє нам отримувати числові вектори, які представляють значення та семантику документів.

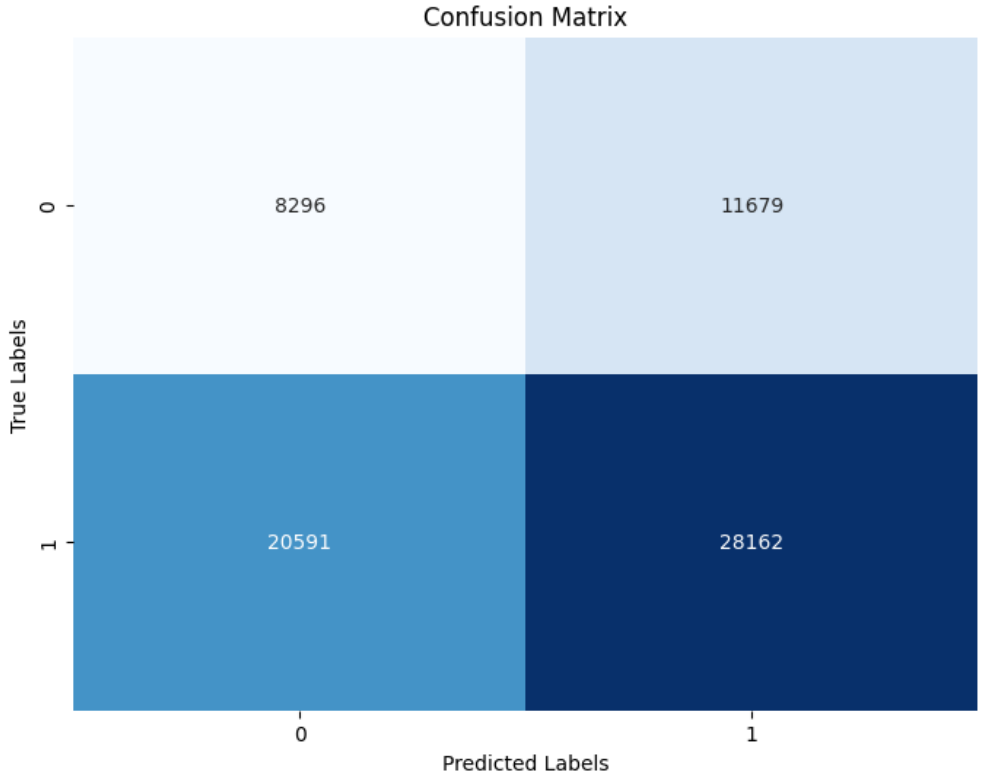

Рисунок 30 — Результати логістичної регресії з Doc2Vec

4.2.3. Bidirectional LSTM

У цьому розділі ми розглянемо використання моделі LSTM (Long Short-Term Memory) з архітектурою Bidirectional для навчання на наборі даних для виявлення фейкових новин. В цей раз сконцентруємося на перевагах використання середовища Google Colab та навчанні на віддаленій GPU [45]. Bidirectional архітектура може допомогти у класифікації, у випадках коли кінець речення може сильно впливати на результат.

Будуємо архітектуру моделі з використанням tensorflow[46], для більш зручного використання API обробляємо дані для розбиття на батчі. Засобами tensorflow зобразимо архітектору графічно.

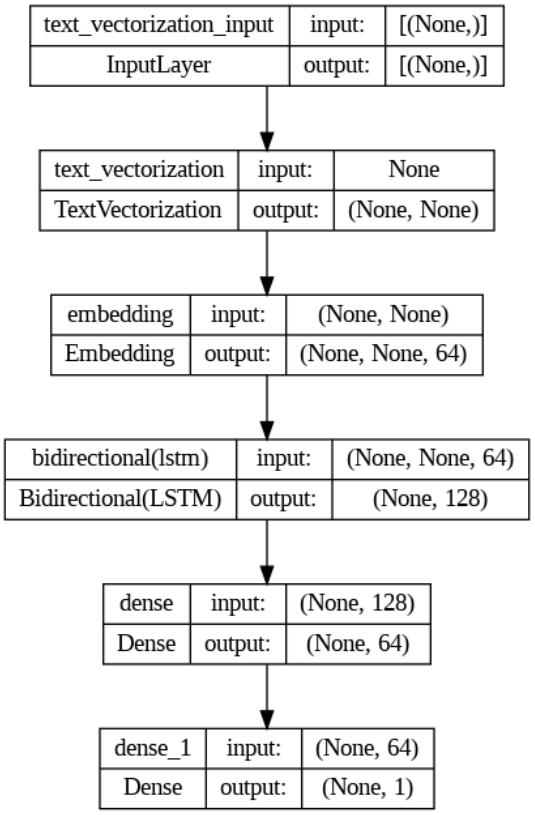

Рисунок 31 — Архітектура моделі

Опишемо детальніше рисунок 31. Спочатку відбувається обробка даних, де найбільш часто вживаним словам призначається певне ціле число, після цього це значення переводиться в one-hot вектор для кожного слова, яке після цього за допомогою embedding шару переводиться в вектор фіксованого значення, де найбільш схожі за значенням слова будуть мати схожі вектори, що може нагадувати Doc2Vec архітектуру. Далі використовуємо LSTM та обретаємо його в Bidirectional шар. Використовуємо розмірність виходного вектору в 64, який потім переводимо в скаляр для класифікації.

Отримано наступні результати

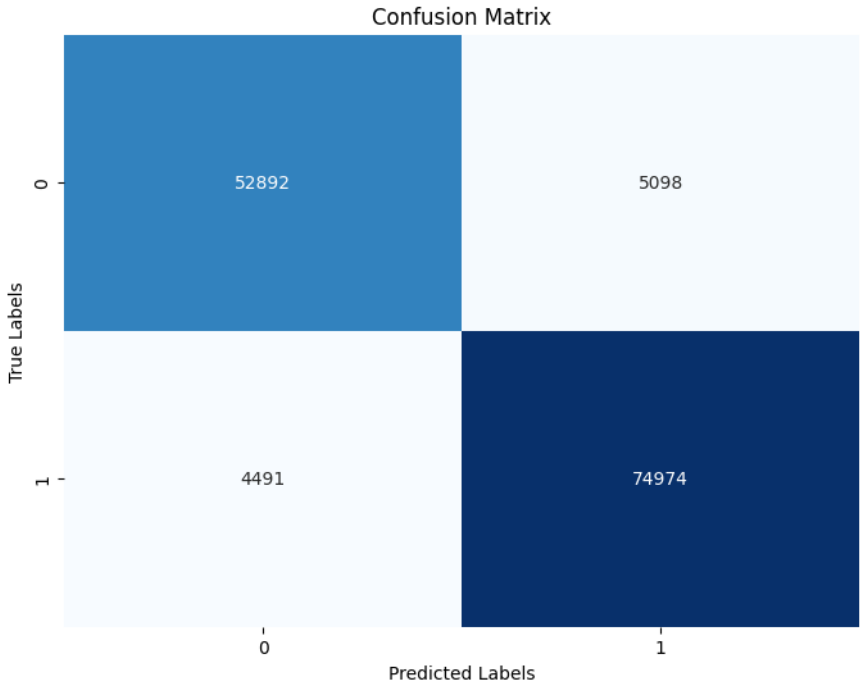

Рисунок 32 — Результати передбачень LSTM

Було використано середовище Google Colab, яке надає можливість безкоштовно використовувати віддалену GPU для прискорення навчання моделі, доступно 16гб відеопам’яті. Єдиним минусом безкоштовної версії є що після довгого бездіяння сервер закриває з’єднання, але для невеликих наборів даних та моделей, такий підхід може бути дуже зручним.

Перевіримо трохи покращену архітектуру з більшою кількістю епох та дропаутом для зменшення ефекту від overfitting.

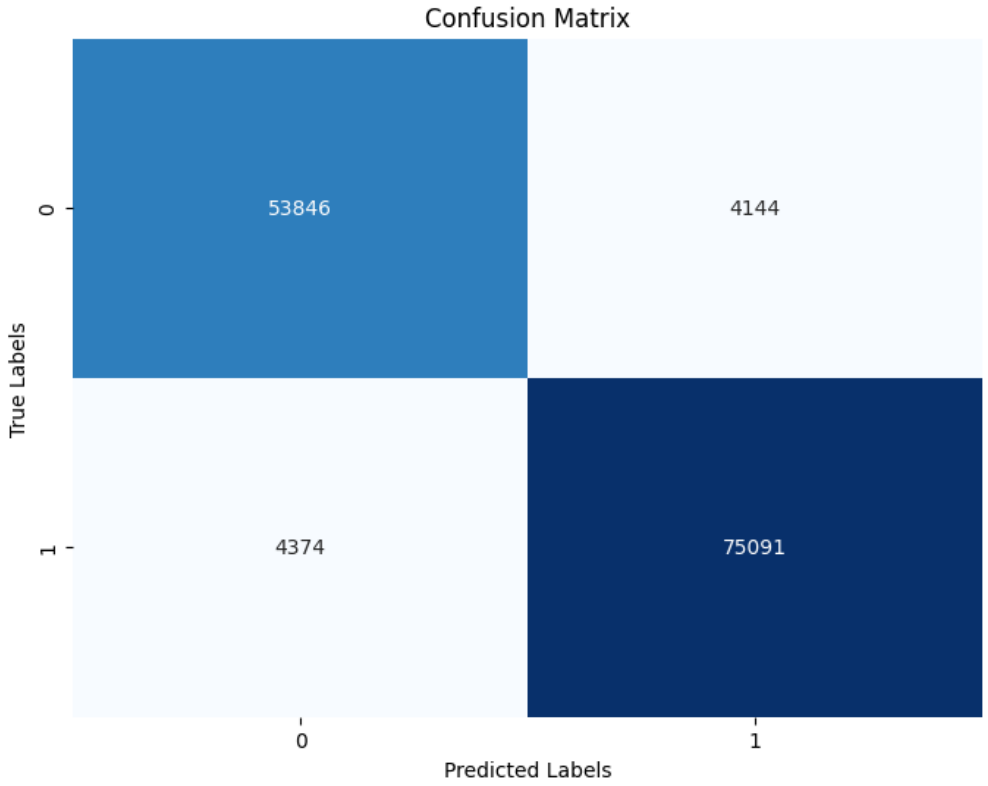

Рисунок 33 — Результати LSTM з покращеною архітектурою

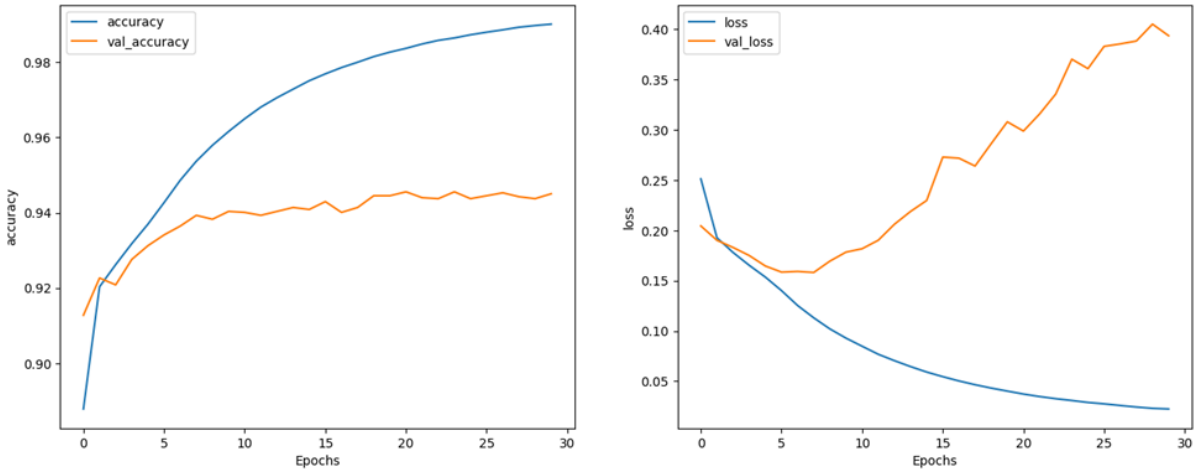

Рисунок 34 — Графіки значень метрик відносно епох навчання LSTM

Використання LSTM Bidirectional для навчання на наборі даних для виявлення фейкових новин на середовищі Google Colab з використанням

віддаленої GPU та Jupyter Notebook з використанням Tensorflow та Keras дозволяє отримати швидке та ефективне навчання моделі. Віддалена GPU дозволяє значно прискорити процес навчання, а середовище Google Colab надає зручність у використанні та розгортанні моделі, не потребуючи додаткових обчислювальних ресурсів на локальному комп'ютері.

4.3. Аналіз результатів

З реалізації різних моделей для виявлення фейкових новин, використовуючи логістичну регресію, TF-IDF, Doc2Vec та LSTM Bidirectional, ми отримали наступні результати та метрики.

Модель	Accuracy Train	Accuracy Test	AUC Train	AUC Test	F1 Train	F1 Test
LR TF-IDF	0.93	0.926	0.939	0.92	0.9473	0.9366
LR Doc2Vec	0.66	0.53	0.66	0.53	0.7405	0.633
LSTM	0.97	0.936	0.97	0.936	0.951	0.9371
LSTM advanced	0.99	0.9456	0.9887	0.9503	0.98750	0.9439

Таблиця 4.3.1. Порівняння метрик моделей

Кожна з цих моделей має свої переваги та недоліки. Давайте розглянемо їх:

Логістична регресія є простою та швидкою у розгортанні моделлю. Вона показує непогані результати та має низькі вимоги до обчислювальних ресурсів. Однак, вона може бути менш потужною у порівнянні з іншими складнішими моделями.

TF-IDF є непоганим алгоритмом для обробки текстових даних. Його використовують для виділення важливих слів у тексті та показує добрі результати. Однак, TF-IDF не враховує семантичну схожість слів та не використовує контекст для розуміння тексту.

Doc2Vec дозволяє отримати векторні представлення для цілого документа. Він здатний враховувати контекст та семантику тексту, що дає кращі результати порівняно з TF-IDF. Однак, Doc2Vec може вимагати більше обчислювальних ресурсів для навчання, особливо на великих наборах даних, саме в нашому випадку був отриманий доволі дивний результат, можливо це пов’язано з кількістю данних, чи невірним підходом до використання.

LSTM Bidirectional використовується для моделювання послідовностей та враховує контекст з обох сторін тексту. Вона показує гарні результати та здатна узагальнювати залежності в тексті, але, в порівнянні з BERT, LSTM Bidirectional може мати меншу точність.

4.4. Висновки до розділу

Загалом, якщо бажана оптимізація розміру моделі та метрик, рекомендовано використовувати TF-IDF або Doc2Vec. Ці моделі показують хороші результати та мають менші вимоги до обчислювальних ресурсів. Однак, якщо є достатньо обчислювальних ресурсів та бажано досягти

максимальної точності, рекомендовано використовувати BERT або LSTM Bidirectional, оскільки вони показують кращі результати, але можуть бути більш вимогливими до ресурсів та складнішими у розгортанні.

Вибір певної моделі залежить від конкретних вимог вашого проекту, доступних обчислювальних ресурсів та балансу між точністю та швидкістю. Рекомендується провести додаткові експерименти та валідацію на вашому наборі даних для визначення найкращої моделі для вашої конкретної задачі виявлення фейкових новин.

5. ФІНАЛЬНА МОДЕЛЬ

Після проведення експерименту було вирішено побудувати модель з використанням BiLSTM та embedding (рис. 35) шару для класифікації тексту. Процес тренування цієї мережі та підбір гіперпараметрів були непростими завданнями, але було застосовано підхід на основі Random та Log Search [47] для досягнення кращої продуктивності моделі.

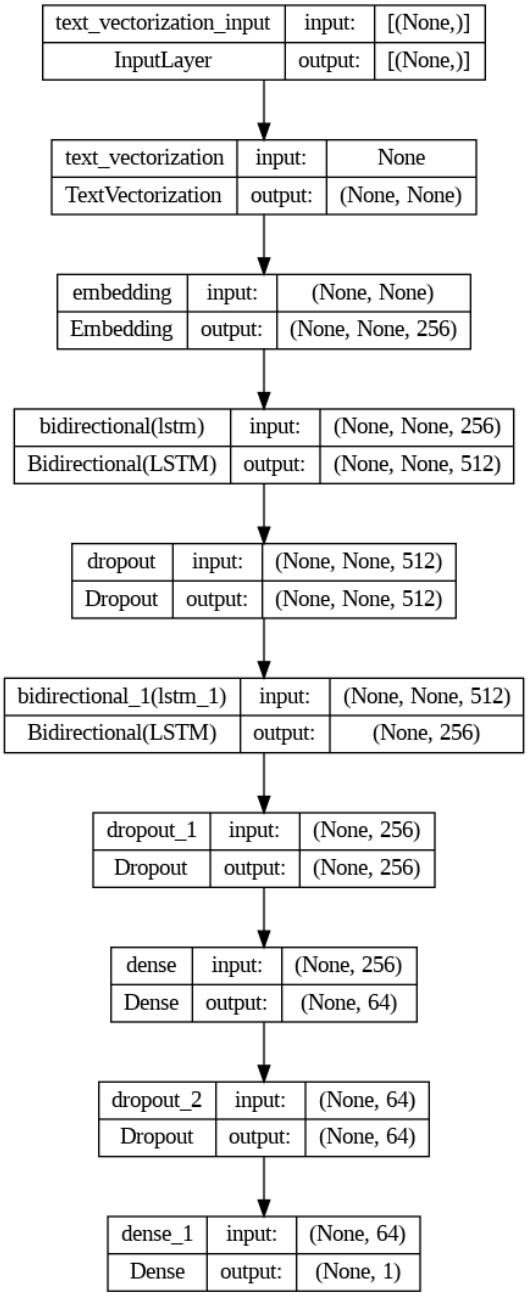

Рисунок 35 — Архітектура отриманої моделі

Спочатку було підготовлено дані для тренування, включаючи текстові дані та відповідні мітки класів, розділено дані на тренувальний, валідаційний та тестовий набори у відношенні 90% для тренувального, 5% на валідаційний, і 5% для тестового. Далі, перетворено слова у векторні представлення за допомогою embedding шару.

Після цього були налаштовані гіперпараметри мережі. Для цього було використано підхід на основі Random Search та Log Search. Замість того, щоб перебирати всі можливі комбінації гіперпараметрів з лінійним шагом, було випадково обрано певну кількість комбінацій на логарифмічній сітці та перевірено їх продуктивність на валідаційному наборі даних.

Під час Random Search вибрано значення для різних гіперпараметрів, таких як кількість LSTM шарів, розмір векторів embedding, кількість нейронів у прихованому шарі, швидкість навчання тощо. Для кожної комбінації гіперпараметрів я тренував модель на тренувальних даних та оцінював її точність на валідаційному наборі.

Після завершення Random Search та Log Search було обрано комбінацію гіперпараметрів, яка мала найкращу продуктивність на валідаційному наборі. Далі модель тренувалася з використанням цих оптимальних гіперпараметрів на тренувальних даних, де за основу для оптимізації було обрано найбільш важливі метріки з тренувального та тестувального набору, а саме F1 Score, Accuracy, ROC-AUC, та безпосередньо їх стабільність між тренувальним та тестовим набором. Для покращення цього параметру було використано Dropout[48] шар, який

дозволяє моделі краще пристосовуватися до невеликих змін в даних та отримувати більш стабільні результати. Він працює таким чином, що може випадковим чином “вимикати” нейрони (прирівнювати до нуля), що дозволяє моделі не пристосовуватися до якихось певних наборів вхідних параметрів а вчитися розуміти більш складну картину відношень в даних в умовах деякої невизначеності чи недоступності даних, це дозволяє зазвичай показувати кращі результати на небачених нових даних чи отримувати меншу різницю в метріках між тренувальним та тестовим набором.

Цей процес тренування був ітеративним: було проведено кілька експериментів з різними комбінаціями гіперпараметрів, щоб знайти найкращу модель для класифікації тексту.

В результаті було отримано навчену модель BiLSTM з embedding шаром, яка має високу точність у класифікації тексту. Цей підхід дозволяє ефективно розпізнавати та класифікувати фейкові новини з використанням даних з trustworthy джерел.

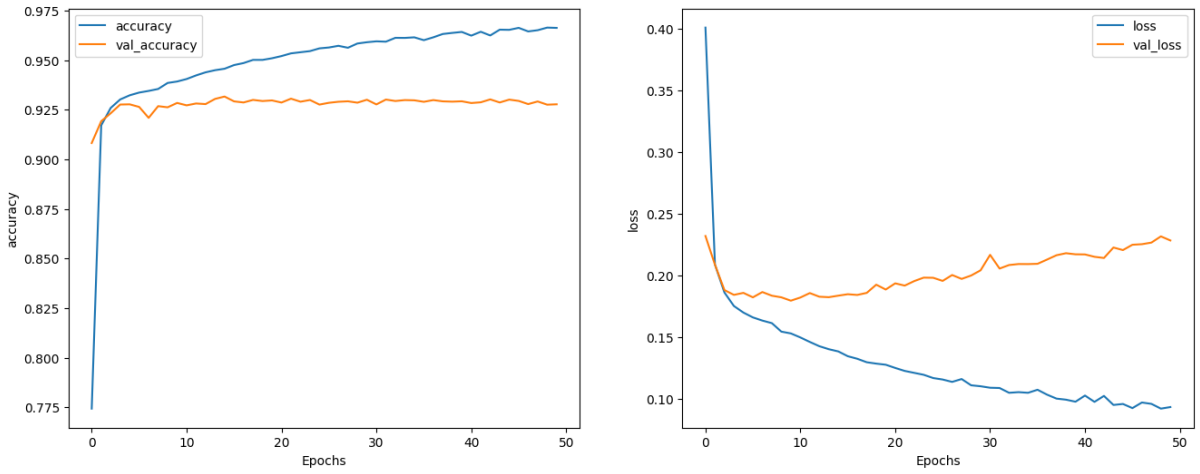

Рисунок 36 — Навчання фінальної моделі

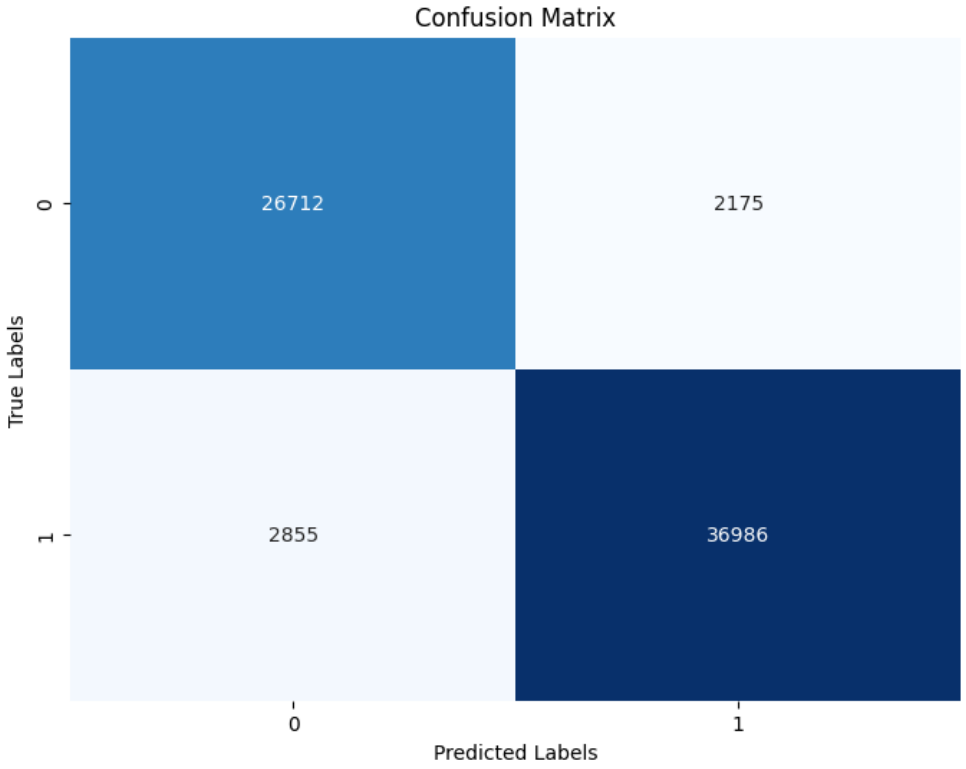

Рисунок 37 — Результати пердбачень фінальної моделі

Тепер спробуємо детальніше розібрати процес класифікації новини, враховуючи джерела (твіти), з якими вона може бути пов’язана. Оскільки вже є натренована модель, надалі переходимо до класифікації усіх твітів для проведення екперименту, для цього, через обмежені обчислювалььні потужності та кількость оперативної пам’яті будемо використовувати раніше підгтовлені ID з набору даних для твітів та для новин. В більш складній системі для пошуку таких відповідних джерел та новин було б доречніше використовувати окремі алгоритми для порівння відповідності вмісту новин та джерел: кластеризацію, алгоритми індексації та пошуку в великих масивах даних (Elastic Search), пошук по TF-IDF та для пришвидшення алгоритми приблизного пошуку найбільш схожих векторів, найближчих сусідів (Locality-Sensitive Hashing), наближеної кластеризації (Approximate Hierarchical Clustering) та інші.

Після класифікації отримано наступний набір даних (рис. 5.3), виведемо приклад з 10 випадкових рядків. Зображено результати до використання sigmoid функції, після її використання: тобто колонка ймовірності належності до класу (true), і остання колонка, яка з використанням 0.5 порогу визначає клас твіту.

	news_id	tweet_id	text	label	TextCleared	predicted_values	pred_sigmoid_applied	predicted_labels
156379	gossipcop-9324688033	523992391682772993	Did @ddlovato Get Engaged? http://t.co/U4bKmcZ...	0	did ddlovato get engaged omygosh its so adorab...	-2.623488	0.067642	0
497325	politifact1731	1064192284353159168	I added a video to a @YouTube playlist https:/...	1	i added a video to a youtube playlist this wee...	8.746694	0.999841	1
48932	gossipcop-606657710	670744566346211330	RT erika_tauber: RT dary87_cardenas: RT BigTit...	0	rt erikatauber rt dary87cardenas rt bigtitbabe...	-0.919345	0.285091	0
138098	gossipcop-1859502594	956874652646699010	Θα είναι η Caitlyn Jenner στο «Dancing with th...	0	caitlyn jenner dancing with the stars	-6.154826	0.002119	0
323479	gossipcop-927607	986074126807859201	Pregnant Cardi B Brings Her Twerking Skills to...	1	pregnant cardi b brings her twerking skills to...	5.862637	0.997164	1
156266	gossipcop-4148349938	1054017170890543104	Remember when Come & Get It went number one an...	0	remember when come get it went number one and ...	-3.471261	0.030141	0
316068	gossipcop-945955	1022982219026116608	Mugshot Madness: Robert Allen (3rd XXXTentacio...	1	mugshot madness robert allen 3rd xxxtentacion ...	5.881655	0.997218	1
362428	gossipcop-923307	978481840007909376	Robin Thicke Shares Sweet Photo of Son Meeting...	1	robin thicke shares sweet photo of son meeting...	6.683986	0.998751	1
351505	gossipcop-843070	852208990440312834	Baseball Cat Is the Miami Marlins' New Unoffic...	1	baseball cat is the miami marlins new unoffici...	1.788658	0.856763	1
11826	gossipcop-1333002237	1033708029349429248	RT thebestofirl "Looks like the reason behind ...	0	rt thebestofirl looks like the reason behind c...	-2.694512	0.063298	0

Рисунок 38 — Результати класифікації

Тепер виконаємо групування по news_id, після чого завдяки визначеним моделлю класам твітів, за допомогою більшості класів (чи середньої суми ймовірностей) визначимо клас новини, в подальшому ці (рис. 38) значення можна запам’ятовувати додатково для подальшої та більш швидкої класифікації новин та джерел (твітітв).

news_id	pred_sigmoid_applied	label
gossipcop-1000240645	0.011591	0.0
gossipcop-1000908841	0.941686	0.0
gossipcop-1009248558	0.002488	0.0
gossipcop-1012123555	0.280369	0.0
gossipcop-1014383679	0.027737	0.0
...	...	...
politifact98	0.660275	1.0
politifact9802	0.946764	1.0
politifact986	0.867008	1.0
politifact99	0.969842	1.0
politifact997	0.980385	1.0

Рисунок 39 — Результати агрегації та справжні класи новин

Виконаємо обчислення точності на цьому наборі, додатково переводимо таким сами чином середню суму ймовірностей на новину в label з використанням threshold. Отримано точність в 0.949 (рис 39), що є покращенням точності на 2-3% (для класифікації було використано не найкращу модель, яка на тренувальному та тестовому наборі набирала приблизно 0.93 точності), таким чином такий підхід з використанням інших джерел, може допомогти додатково покращити точність моделі та отримати більш впевнені передбачення, які базуються на декількох пов’язаних джерелах.

```
[51] news_grouped['label'].value_counts()

     1.0    15955
     0.0     5201
     Name: label, dtype: int64

[52] from sklearn.metrics import accuracy_score

     accuracy_score(news_grouped['label'],news_grouped['according_label'])

     0.9492815276989979
```

Рисунок 40 — Отримані результати точності по новинах

Можливі покращення цього підходу включають використання більш потужних обчислювальних ресурсів, щоб дозволити більший простір пошуку гіперпараметрів, а також використання більш складних архітектур мережі для поліпшення класифікаційних результатів. Оскільки те, що моделі потрібно перенавчатися для постійного отримання добрих результатів при появі нових новин є проблематичним. Цікавим доповненням був би підхід з використанням BERT, кластеризації та пошуку по базі даних. Опишу його детальніше. Використання BERT дозволяє отримати векторне представлення тексту, яке враховує контекст та семантику слів у реченні. Це допомагає покращити якість класифікації тексту і виявлення фейкових новин, але й вимагає чималих обчислювальних ресурсів.

Після отримання векторних представлень тексту за допомогою BERT, можна застосувати алгоритм кластеризації для групування схожих твітів разом. Кластеризація дозволяє виявити подібність між твітами з довіреними джерелами та вхідним твітом. Це може допомогти визначити, наскільки вірогідно, що вхідний твіт є фейковим або має недостовірну інформацію.

Пошук та порівняння з довіреними джерелами: зберігаючи базу даних з довіреними джерелами, можна використовувати пошуковий механізм для швидкого пошуку та порівняння вхідного твіту з цими

джерелами. Це може включати порівняння тексту, перевірку схожості семантики, сентименту або інших характеристик. Якщо вхідний твіт відрізняється від довірених джерел або має негативний сентимент, це може вказувати на його потенційну недостовірність.

Цей підхід забезпечує більш точну і високоякісну класифікацію фейкових новин шляхом використання передових методів обробки тексту, кластеризації та порівняння з довіреними джерелами. Додатковими перевагами є швидкість пошуку та відповіді, що сприяє ефективному виявленню фейкових новин та забезпечує надійну оцінку їх достовірності. Особливо важливою частиною є швидке отримання надійних даних з використанням парсингу перевірених сайтів та стійкість моделі до зміни в даних.

Але звісно така модель вимагає значно більшої кількості обчислювальних ресурсів, значно більшої кількості даних, більшого об’єму сховищ даних, а значить і тривалого часу для збору даних, але така система матиме можливість більш зручного масштабування та єдиною проблемою буде автоматичне отримання свіжих даних та завантаження їх в загальну базу.

6. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі надається оцінка основних характеристик продукту, розробленого для виявлення фейкових новин в соціальних мережах. Даний продукт призначено для використання на окремому сервері до якого через API можна звертатися та класифікувати певний документ (текст, твіт) та виявляти рівень достовірності інформації.

Нижче наводиться аналіз різних варіантів реалізації модулю з метою вибору оптимальної стратегії створення програмного продукту, враховуючи при цьому економічні фактори та основні характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка оцінює реальну вартість продукту або послуги незалежно від організаційної структури компанії. Обидва види витрат – прямі та побічні – розподіляються в залежності від потрібних обсягів ресурсів на кожному етапі.

Метою функціонально-вартісного аналізу є забезпечення найбільш оптимального розподілення ресурсів, які були виділені на виробництво продукції або надання послуг. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Коротко кажучи, даний метод аналізу розробленого продукту починається з визначення послідовності функцій, необхідних для виробництва продукту. Перш за все спочатку йдуть всі можливі функції, які розподіляються по двом групам: ті, що впливають на вартість продукту

і ні. Додатково на цьому етапі оптимізується сама послідовність скороченням кроків, що не впливають на витрати.

Для кожної функції застосовується визначення повного обсягу річних витрат та кількість робочих часів. Базуючись на отриманих оцінках визначається кількісна характеристика джерел витрат, після чого проводиться кінцевий розрахунок витрат на виробництво продукту.

Для кожної функції застосовується визначення повного обсягу річних витрат та кількість робочих часів. Базуючись на отриманих оцінках визначається кількісна характеристика джерел витрат, після чого проводиться кінцевий розрахунок витрат на виробництво продукту.

6.1. Постановка задачі техніко-економічного аналізу

Даний метод ФВА застосовується для проведення техніко-економічного аналізу розробки моделі машинного навчання, та її розгортки в production системі.

Технічні вимоги до фреймворку:

- висока швидкість обробки даних та відповідь користувачеві у реальному часі
- зручність та простота взаємодії
- висока щвидкість розгортання та розробки
- можливість зручного налаштування, масштабування та обслуговування
- досягнення балансу між необхідними обчислювальними ресурсами та точністю системи задля оптимізації витрат на розгортання продукту

6.1.1. Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який вирішує задачу для виявлення фейкової інформації в соціальних мережах. Виходячи з конкретної мети, можна виділити наступні основні функції програмного продукту:

F_1 – вибір мови програмування

F_2 – вибір фреймворку

F_3 – вибір архітектури моделі

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

1) Python
2) C++

Функція F_2:

1) PyTorch
2) TensorFlow

Функція F_3:

1) Logistic Regression
2) RNN
3) Transformer

6.1.2. Варіанти реалізації основних функцій

Варіанти реалізації наявних функцій наведені у морфологічній карті системи на рисунку 41 Морфологічна карта містить всі можливі

комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

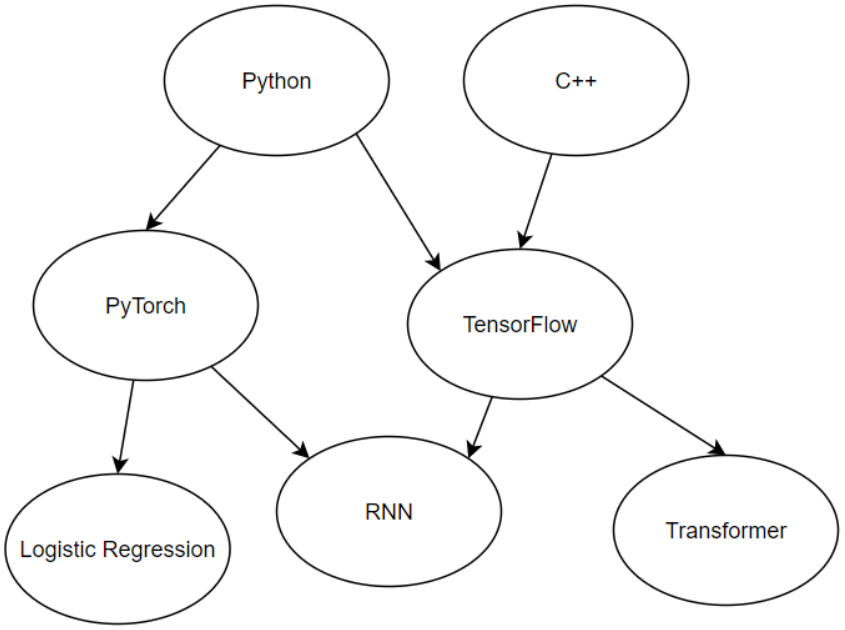

Рисунок 41 — Морфологічна карта варіантів реалізації функцій

На основі цієї карти було побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 6.1.2.1).

Функція	Варіант реалізації	Переваги	Недоліки
F_1	А	Кросплатформена мова програмування, займає менше часу на написання коду	Займає більше часу для виконання коду
	Б	Кросплатформна мова програмування, займає менше часу для виконання коду, є більш оптимізованою.	Займає більше часу на написання коду

F_2	А	Є гнучким інструментом для побудови складних систем, підтримка останніх версій CUDA, open source суспільна підтримка.	Більш висока складність використання, вимагає більшої кількості коду.
	Б	Дозволяє без особливих знань створювати складні моделі з мінімальною кількістю коду.	Менша підтримка та менш кастомізований код.
F_3	А	Є швидкою та малою моделлю машинного навчання.	Погано пристосована до нових даних, середні результати.
	Б	Є доволі швидкою але більш точною моделлю.	Залежно від архітектури може вимагати значної кількості ресурсів для тренування та роботи.
	В	Є найбільш точною моделлю, дозволяє finetuning для швидкого навчання.	Потребує значних ресурсів.

Табл. 6.1 – Позитивно-негативна матриця варіантів основних функцій

Проаналізувавши позитивно-негативну матрицю можна зробити висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти наявні у морфологічній карті.

Функція F_1:

Оскільки реалізація програмного коду для вирішення поставленої задачі потребує підтримку фреймворків, необхідно обрати мову, завдяки якій буде описана реалізація якнайвдобніше, тому варіант Б має бути відкинутий.

Функція F_2:

Описані фреймворки повністю підтримають мову програмування Python, тому підійдуть обидва варіанти, але для меншої кількості рядків та більш швидкої розробки рекомендується використовувати Tensorflow.

Функція F_3:

Кожен з варіантів може бути використаним, залежно від технічних вимог, можливостей та бюджету проекту тому розглядати будемо наступні варіанти.

- F_1(Б) – F_2(Б) – F_3(А)
- F_1(Б) – F_2(Б) – F_3(Б)
- F_1(Б) – F_2(Б) – F_3(В)

6.2. Обґрунтування системи параметрів програмного продукту

6.2.1. Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – об’єм пам’яті для збереження даних
- X2 - об’єм пам’яті для роботи моделі
- X3 – час, який витрачається на виконання прогнозу
- X4 – потенційний об’єм програмного коду, який необхідно створити безпосередньо розробнику

6.2.2. Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту як показано у нище.

Опис параметру	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Об’єм пам’яті для збереження даних	X1	Гб	128	512	1024
Об’єм пам’яті для роботи моделі	X2	Гб	4	16	32
Середній час передбачення	X3	с	15	8	2
Потенційний об’єм програмного коду	X4	строк коду	6000	3000	1500

Табл. 6.2 – Основні параметри програмного продукту

За даними таблиці будуємо графічні характеристики параметрів:

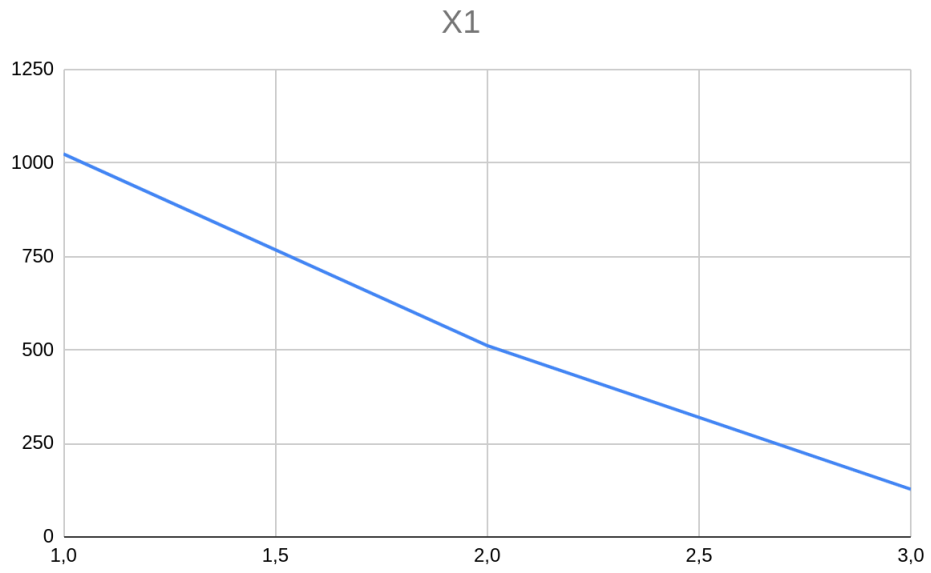

Рисунок 42 – Об’єм пам’яті для збереження даних

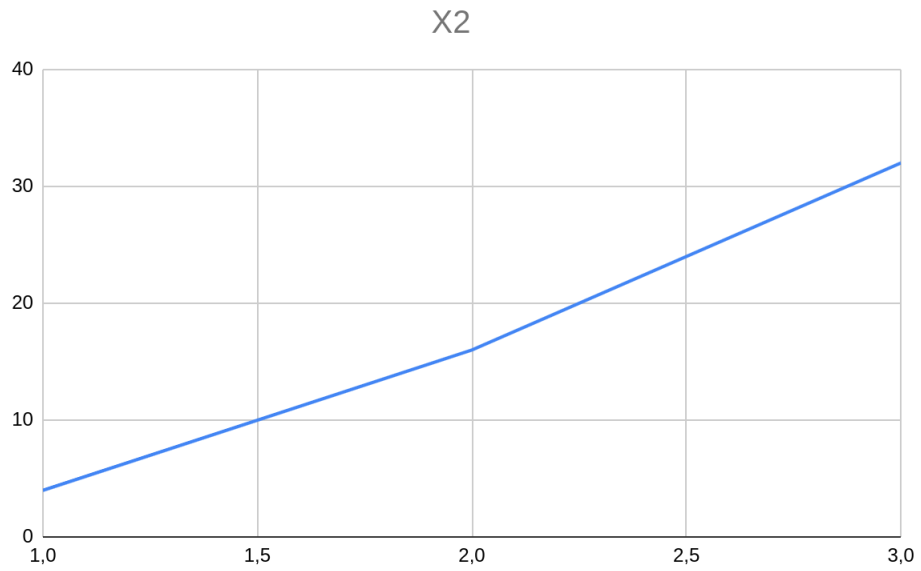

Рисунок 43 – Об’єм пам’яті для роботи моделі

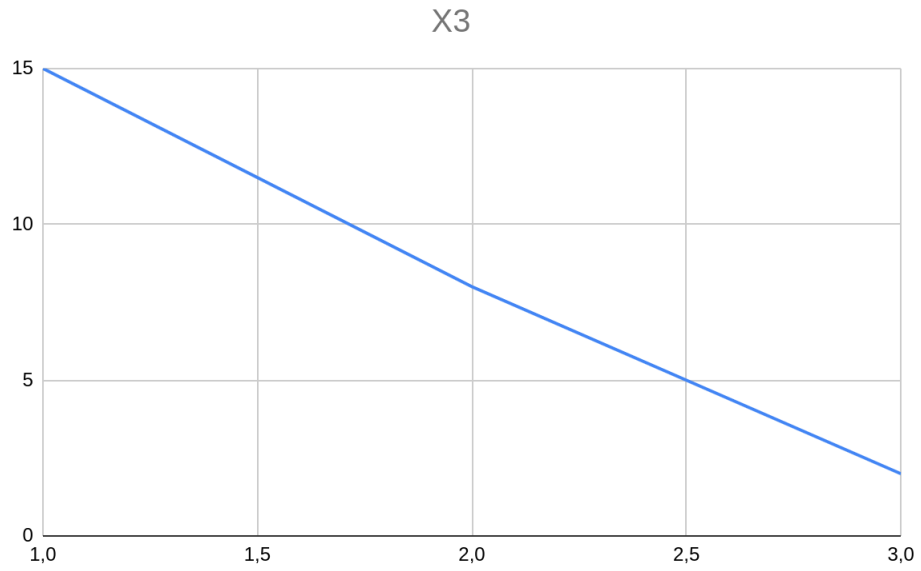

Рисунок 44 – Середній час передбачення

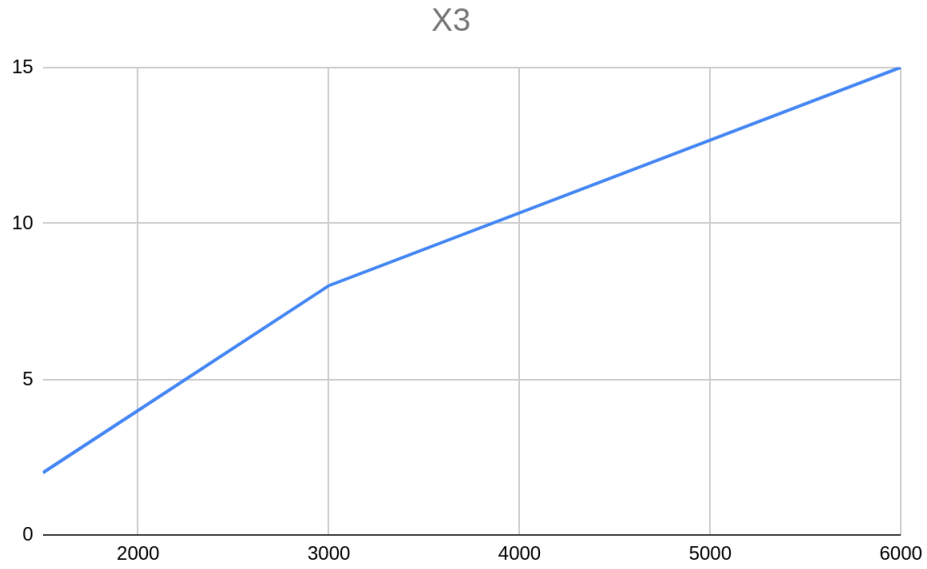

Рисунок 45 – Кількість рядків коду

6.2.3. Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка моделі, який матиме високу точність та покращить досвід від користування продуктом.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значущості передбачає:

1) визначення рівня значимості параметра шляхом присвоєння різних рангів

2) перевірку придатності експертних оцінок для подальшого використання

3) визначення оцінки попарного пріоритету параметрів

4) обробку результатів та визначення коефіцієнту значущості

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів Ri	Відхилення Δi	Δ^2i
			1	2	3	4	5	6	7			

X1	Об’єм пам’яті для збереження даних	Гб	2	2	2	4	2	3	2	17	-0,5	0,25
X2	Об’єм пам’яті для роботи моделі	Гб	3	3	4	2	3	2	4	21	3,5	12,25
X3	Середній час передбачення	с	1	1	1	1	1	1	1	7	-10,5	110,25
X4	Потенційний об’єм програмного коду	кількість рядків коду	4	4	3	3	4	4	3	25	7,5	56,25
	Разом		10	10	10	10	10	10	10	70		179

Таблиця 6.3 - Результати ранжування параметрів

Для перевірки ступеню достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70 \ (6.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів T:

$$T = \frac{1}{n} R_{ij} = 17,5 \ (6.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \ (6.3)$$

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 179. \ (6.4)$$

д) коефіцієнт узгодженості (конкордації):

$$W = \frac{12S}{N^2\left(n^3 - n\right)} = 0,7306 > W_k = 0,67. \ (6.5)$$

Ранжирування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, який рівний 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати заносимо у таблицю 6.4. Числове значення, що визначає ступінь переваги *i*-го параметра над *j*-тим, aa_{iijj} визначається за формулою:

$$aa_{iijj} = 1{,}5 \;\; xx_{ii} > xx_{jj};\; 1{,}0 \;\; xx_{ii} = xx_{jj};\; 0{,}5 \;\; xx_{ii} < xx_{jj}.$$

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	>	<	>	<	<	0,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	<	<	<	>	<	<	<	<	0,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	<	<	>	<	<	<	>	<	0,5
X3 і X4	<	<	<	<	<	<	<	<	0,5

Табл. 6.4 – Результати ранжування параметрів

З отриманих числових оцінок переваги складемо матрицю $A = \|a_{ijj}\|$.

Для кожного параметра розрахунок вагомості $KK_{ВВii}$ проводиться за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (6.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (6.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступною формулою:

$$K_{\text{ві}} = \frac{b_i^{'}}{\sum_{i=1}^{n} b_i^{'}}, \quad (6.9)$$

Як видно з таблиці 6.5, різниця значень коефіцієнтів вагомості після другої ітерації не перевищує 2%, тому додаткові ітерації не потрібні.

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$
X1	1	0,5	1,5	0,5	3,5	0,21875	13,25	0,217
X2	1,5	1	1,5	0,5	4,5	0,28125	17,25	0,282
X3	1,5	0,5	1	0,5	3,5	0,21875	13,25	0,217
X4	1,5	0,5	1,5	1	4,5	0,28125	17,25	0,282
Всього:					16	1	61	1

Табл. 6.5 – Розрахунок вагомості параметрів

6.3. Аналіз рівня якості реалізації функцій

Рівень якості кожного варіанту виконання основних функцій визначається окремо.

Абсолютні значення параметрів *X2* (Об'єм оперативної пам'яті для навчання моделі), *X3* (Середній час передбачення) та *X4* (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X1 – об'єм пам'яті для збереження даних для навчання, обрано не найкращим з можливих і це значення відповідає варіанту 512 гігабайт.

Коефіцієнт технічного рівня якості для кожного варіанта реалізації ПП розраховується за формулою:

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}, \quad (4.11)$$

де n – кількість параметрів;

$K_{\text{в}i}$ – коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Розрахунок показників рівня якості представлено відповідно в таблиці 6.6.

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	A	X1	512	7	0,217	1,519

F3	А	X2	8	9	0,282	2,538
	Б	X3	6	10	0,217	2,17
	В	X4	1000	7	0,282	1,974

Табл. 6.6 – Розрахунок показників якості

За цими даними визначаємо рівень якості кожного з варіантів:

- $F_1(А) – F_2(А) – F_3(А) = 1.519 + 2.538 + 2.17 = 6.23$
- $F_1(А) – F_2(А) – F_3(Б) = 2.538 + 1.974 + 2.17 = 6.682$
- $F_1(А) – F_2(А) – F_3(В) = 1.519 + 2.538 + 1.974 = 6,031$

Отже, найкращим є другий, для якого коефіцієнт технічного рівня має найбільше значення.

6.4. Економічний аналіз варіантів розробки програмного продукту

Для визначення вартості розробки програмного продукту спочатку проведемо розрахунок трудомісткості. [49]

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту
2. Розробка алгоритму збору даних

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується інформація з довідників, а завдання 2 використовує інформацію у вигляді даних. Проведемо

розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється за формулою.

$$T_О = T_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \quad (6.13)$$

де $T_Р$ – трудомісткість розробки програмного продукту;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_Р$ = 60 людино-днів. Поправочний коефіцієнт вхідної інформації для першого завдання: $К_П$ = 1,5. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації рівний $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0,7. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 60 \cdot 1.5 \cdot 0.7 = 63 \text{ людино-дні.}$$

Проведемо аналогічні розрахунки для другого завдання, в якому використовується алгоритм третьої групи складності зі ступенем новизни Б, тобто $T_Р$ = 50 людино-днів, $К_П$ = 0,4, $К_{СК}$ = 1, $К_{СТ}$ = 0,6:

$$T_2 = 50 \cdot 0.4 \cdot 0.6 = 12 \text{ людино-дні.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_o = (63+ 12) \cdot 8 = 600 \quad \text{людино-годин.}$$

В розробці беруть участь програміст з машинного навчання в області природної обробки мови окладом 28000 грн та один інженер даних з окладом 23000 грн. Визначимо середню зарплату за годину за формулою:

$$СЧ = \frac{М}{T_m \cdot t} \text{грн.}, \quad (6.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$СЧ = \frac{28000+23000}{2 \cdot 21 \cdot 8} = 151,78 \text{ грн.} \quad (6.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$СЗП = С_ч \cdot Т_i \cdot КД, \quad (6.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників становить

$$С_{ЗП} = 151,78 \cdot 600 \cdot 1,1 = 100174.8 \text{ грн.}$$

Відрахування на соціальний внесок становить 22%:

$$С_{СВ} = С_{ЗП} \cdot 0,22 = 100174.8 \cdot 0,22 = 22038 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. Оскільки одна ЕОМ обслуговується одним інженером апаратного забезпечення з окладом 18000 грн. та коефіцієнтом зайнятості $К_З$ = 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 18000 \cdot 0,2 = 43200 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 43200 \cdot (1 + 0,1) = 47520 \text{ грн.}$$

Відрахування на соціальний внесок становить 22%:

$$С_{СВ} = С_{ЗП} \cdot 0,22 = 47520 \cdot 0,22 = 10454 \text{ грн.}$$

Амортизаційні відрахування розраховуємо за формулою при амортизації 25% та вартості ЕОМ – 35 000 грн.:

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.4 \cdot 0.25 \cdot 60000 = 21000 \text{ грн.}$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$ – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо за формулою:

$$С_Р = К_{ТМ} \cdot К_Р \cdot Ц_{ПР} = 1,4 \cdot 0,05 \cdot 60000 = 4200 \text{ грн.}$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.6 =$$

$$= 1118,4 \text{ години,}$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_C \cdot К_З \cdot Ц_{ЕН} = 1118,4 \cdot 0,6 \cdot 0,6 \cdot 4,79 = 1928,56 \text{ грн.}$$

де N_C – середньо-споживча потужність приладу;

$К_З$ – коефіцієнтом зайнятості приладу;

$Ц_{ЕЛ}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0,67 = 60\,000 \cdot 0,67 = 40200 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть складати:

$$С_{ЕК} = С_{ЗП} + С_{СВ} + С_А + С_Р + С_{ЕЛ} + С_Н,$$

$С_{ЕК} = 47520 + 10454 + 21000 + 4200 + 1928,56 + 40200 = 125302,56$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 125302.56 / 1118,4 = 112,03 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу складають:

$$С_М = С_{МГ} \cdot Т = 112,03 \cdot 600 = 67222,40 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67 = 47520 \cdot 0,67 = 31838.4 \text{ грн.}$$

Отже, вартість розробки програмного продукту за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{СВ} + С_М + С_Н,$$

$$С_{ПП} = 47520 + 9504 + 10454 + 31838.4 = 99316,4 \text{ грн.}$$

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР} = К_{Кj}/С_{Фj}, \quad (6.21)$$

$$КТЕР = 6.682 /99316,4 = 6.727992557120477e\text{-}05$$

6.5. Висновки до розділу

У даному розділі в результаті виконання економічного розділу були систематизовані і закріплені теоретичні знання в галузі економіки та організації виробництва використанням їх для техніко-економічного обґрунтування розробки методом функціонально-вартісного аналізу. Для цього використовувався методичний матеріал [49]

На основі даних про зміст основних функцій, які повинен реалізувати програмний продукт, були визначені чотири найбільш перспективні варіанти реалізації продукту. Найбільш ефективним виявився третій варіант реалізації функцій ПП, який дає максимальну величину коефіцієнта технікоекономічного рівня, вартість витрат для нього становить $С_{ПП} = 99316,4$ грн.

Цей варіант передбачає:

- застосування мови програмування Python;
- використання фреймворку Tensorflow
- Використання потужної та невеликої моделі LSTM;
- високу точність класифікації;
- високу швидкість класифікації.

ВИСНОВКИ

В даній роботі було проведено дослідження виявлення фейкових новин з використанням перехресної перевірки джерел за допомогою машинного навчання. Проблема фейкових новин є актуальною в сучасному інформаційному середовищі, оскільки маніпулятивна інформація може впливати на суспільство та прийняття важливих рішень.

У роботі були розглянуті основні методи маніпулятивної інформації та проблеми, пов'язані з фейковими новинами. Для вирішення цих проблем була розроблена система, яка використовує перехресну перевірку джерел та машинне навчання для виявлення фейкових новин.

Були використані різні методи та інформаційні системи для збору та аналізу даних, зокрема використовувався набір даних з твітеру. Частота слів, домени новин, їх справжність та фейковість були проаналізовані з використанням різних алгоритмів та моделей машинного навчання, таких як Logistic Regression, TF-IDF, Doc2Vec, LSTM та ROBerta, було отримано багато інформації, щодо частоти джерела, слів, і як вони можуть бути пов’язані з надійністю даних.

Процес експериментів та аналізу даних дозволив вибрати найкращі моделі та гіперпараметри для виявлення фейкових новин. Крім того, було застосовано підхід з кластерізацією та порівнянням сенсу, семантики, сентименту та емоційності з джерелами. Це дозволило покращити точність та надійність системи виявлення фейкових новин.

Результати цієї роботи мають значний внесок у сферу боротьби з фейковими новинами. Використання перехресної перевірки джерел та машинного навчання дозволяє забезпечити більш достовірну інформацію для користувачів та сприяє зміцненню довіри до новинних джерел.

Можливими подальшими діями можуть бути вдосконалення системи шляхом використання більш складних моделей машинного навчання та розширення набору даних для забезпечення більш широкого охоплення новинних джерел. Також, цей підхід може бути використаний в реальних умовах для автоматичного виявлення фейкових новин та надання користувачам достовірної інформації. В цілому, дана робота допомагає зрозуміти проблему фейкових новин, надає реалістичні методи їх виявлення та вносить вагомий внесок у поліпшення якості інформаційного простору.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. National Endowment For Democracy, "Distinguishing Disinformation from Propaganda". [Електронний ресурс] - Режим доступу: https://www.ned.org/wp-content/uploads/2018/06/Distinguishing-Disinformation-from-Propaganda.pdf (дата звернення: 02.03.2023)
2. University of Oxford, "Social Media Manipulation by Political Actors: A Problem on an Industrial Scale. [Електронний ресурс] - Режим доступу: https://www.ox.ac.uk/news/2021-01-13-social-media-manipulation-political-actors-industrial-scale-problem-oxford-report (дата звернення: 08.03.2023)
3. University of Haifa, "Haifa U. Study Finds Winning Recipe for Fake News: 80% Truth, 20% Lies". [Електронний ресурс] - Режим доступу: https://www.asuh.org/news-media/haifa-u-study-finds-winning-recipe-for-fake-news-80-truth-20-lies/ (дата звернення: 15.03.2023)
4. А. Баумейстер. Как обнаруживать и опровергать манипуляции. Практическое руководство. YouTube. [Електронний ресурс] - Режим доступу: https://www.youtube.com/watch?v=I46SJGqotJ8 (дата звернення 12.04.2023).
5. Robert B. Michael, " The relationship between political affiliation and beliefs about sources of “fake news” . [Електронний ресурс] - Режим доступу: https://cognitiveresearchjournal.springeropen.com/articles/10.1186/s41235-021-00278-1 (дата звернення: 09.03.2023)
6. Internet Matters, What is fake news and misinformation? [Електронний ресурс] - Режим доступу:

https://www.internetmatters.org/issues/fake-news-and-misinformation-advice-hub/learn-about-fake-news-to-support-children/ (дата звернення 12.03.2023).

7. University of Exter. Library. Fake News: Fighting fake news. [Електронний ресурс] - Режим доступу: https://libguides.exeter.ac.uk/fakenews/fight_it (дата звернення 12.03.2023).

8. Ad Fontes Media. Interactive Mass Media Bias Chart. [Електронний ресурс] - Режим доступу: https://adfontesmedia.com/interactive-media-bias-chart/ (дата звернення 12.03.2023)

9. ResearchGate. "Fake News Detection Using Machine Learning Ensemble Methods". [Електронний ресурс] - Режим доступу: https://www.researchgate.net/publication/346262009_Fake_News_Detection_Using_Machine_Learning_Ensemble_Methods (дата звернення: 02.04.2023)

10. scikit-learn, "Logistic Regression - scikit-learn". [Електронний ресурс] - Режим доступу: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (дата звернення: 08.03.2023)

11. Machine Learning for Biologists. "Logistic Regression and Artificial Neural Networks". [Електронний ресурс] - Режим доступу: https://carpentries-incubator.github.io/ml4bio-workshop/05-logit-ann/index.html (дата звернення: 25.03.2023)

12. Analytics Vidhya. "Precision and Recall in Machine Learning". [Електронний ресурс] - Режим доступу:

https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/ (дата звернення: 26.03.2023)

13. Mlearning.ai. Medium. NLP: Tokenization, Stemming, Lemmatization and Part of Speech Tagging. [Електронний ресурс] - Режим доступу: https://medium.com/mlearning-ai/nlp-tokenization-stemming-lemmatization-and-part-of-speech-tagging-9088ac068768 (дата звернення: 27.03.2023)

14. Machine Learning Mastery. A Gentle Introduction to the Bag-of-Words Model. [Електронний ресурс] - Режим доступу: https://machinelearningmastery.com/gentle-introduction-bag-words-model/ (дата звернення: 28.03.2023)

15. Wikipedia. TF-IDF. [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/TF-IDF (дата звернення: 28.03.2023)

16. Wikipedia. Neural Network. [Електронний ресурс] - Режим доступу: https://en.wikipedia.org/wiki/Neural_network (дата звернення: 29.03.2023)

17. Wikipedia. ReLU. [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/ReLU (дата звернення: 30.03.2023)

18. Towards Data Science. Activation Functions in Neural Networks. [Електронний ресурс] - Режим доступу: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (дата звернення: 01.04.2023)

19. Analytics Vidhya. "Building Naive Bayes Classifier from Scratch to Perform Sentiment Analysis". [Електронний ресурс] - Режим доступу: https://www.analyticsvidhya.com/blog/2022/03/building-naive-bayes-clas

sifier-from-scratch-to-perform-sentiment-analysis/ (дата звернення: 02.04.2023)

20. Medium. "Random Forest: A Simple Explanation". [Електронний ресурс] - Режим доступу: https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d (дата звернення: 03.04.2023)

21. Towards Data Science. "A Brief Introduction to Recurrent Neural Networks". [Електронний ресурс] - Режим доступу: https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4 (дата звернення: 03.04.2023)

22. Machine Learning Mastery. "The Transformer Model". [Електронний ресурс] - Режим доступу: https://machinelearningmastery.com/the-transformer-model/ (дата звернення: 03.04.2023)

23. Khanam Z.,Alwasel B. N. , Sirafi H., Rashid M. Fake News Detection Using Machine Learning Approaches 2021 DOI:10.1088/1757-899X/1099/1/012040.
URL: https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012040/pdf

24. Ramy Baly , Georgi Karadzhov , Dimitar Alexandrov , James Glass , Preslav Nakov. Predicting Factuality of Reporting and Bias of News Media Sources.
URL: https://arxiv.org/pdf/1810.01765.pdf

25. Linmei Hu, Tianchi Yang, Luhao Zhang, Wanjun Zhong, Duyu Tang, Chuan Shi, Nan Duan, Ming Zhou. Compare to The Knowledge: Graph

Neural Fake News Detection with External Knowledge
URL: https://aclanthology.org/2021.acl-long.62.pdf

26. Zahra Ghadiri, Milad Ranjbar, Fakhteh Ghanbarnejad, Sadegh Raeisi. Automated Fake News Detection using cross-checking with reliable sources.
URL: https://arxiv.org/pdf/2201.00083.pdf

27. Wikipedia. Cosine similarity [Електронний ресурс] - Режим доступу: https://en.wikipedia.org/wiki/Cosine_similarity (дата звернення: 04.04.2023)

28. ActiveLoop AI. Liar Dataset. [Електронний ресурс] - Режим доступу: https://datasets.activeloop.ai/docs/ml/datasets/liar-dataset/ (дата звернення: 05.04.2023)

29. Papers with Code. Fake News Detection Datasets. [Електронний ресурс] - Режим доступу: https://paperswithcode.com/datasets?task=fake-news-detection&page=1 (дата звернення: 08.04.2023)

30. Depak M. Fake News Detection Resources. GitHub. [Електронний ресурс] - Режим доступу: 23 квітня 2023 року. Адреса: https://github.com/mdepak/fake-news-detection-resources (дата звернення: 10.04.2023)

31. GitHub. FakeNewsNet Dataset. [Електронний ресурс] - Режим доступу: https://github.com/KaiDMML/FakeNewsNet (дата звернення: 28.03.2023)

32. Поплавський В. О. Fake News Bachelor Diploma. [Електронний ресурс] - Режим доступу:

https://github.com/hell0ut/fake_news_nlp_diploma_iasa_4th_course (дата звернення: 10.04.2023)

33. Python Docs. Threading. [Електронний ресурс] - Режим доступу: https://docs.python.org/3/library/threading.html (дата звернення: 28.03.2023)

34. Python Docs. MultiProcessing. [Електронний ресурс] - Режим доступу: multiprocessing — Process-based parallelism — Python 3.11.3 documentation (дата звернення: 28.03.2023)

35. Selenium. Documentation. [Електронний ресурс] - Режим доступу: https://selenium-python.readthedocs.io/ (дата звернення: 28.03.2023)

36. Twitter. API. [Електронний ресурс] - Режим доступу: https://developer.twitter.com/en/docs/twitter-for-websites/timelines/guides/oembed-api (дата звернення: 28.03.2023)

37. Twitter.Embedding API. [Електронний ресурс] - Режим доступу: https://publish.twitter.com/# (дата звернення: 28.03.2023)

38. Поплавський В. О. Парсинг даних. [Електронний ресурс] - Режим доступу: https://github.com/hell0ut/fake_news_nlp_diploma_iasa_4th_course/blob/master/twitter_parse.py (дата звернення: 28.03.2023)

39. The Pandas Development Team. Pandas Documentation. [Електронний ресурс] - Режим доступу: https://pandas.pydata.org/docs/ (дата звернення: 28.03.2023)

40. Поплавський В.О. Підготовка даних. [Електронний ресурс] - Режим доступу:

https://github.com/hell0ut/fake_news_nlp_diploma_iasa_4th_course/blob/master/data_preparation.ipynb (дата звернення: 28.03.2023)

41. Kaggle. "Exploring Politifact Dataset", [Електронний ресурс] - Режим доступу: https://www.kaggle.com/code/sophieb/exploring-politifact-dataset (дата звернення: 28.03.2023)

42. NVIDIA. CUDA Documentation. [Електронний ресурс] - Режим доступу: https://docs.nvidia.com/cuda/. (дата звернення: 02.05.2023).

43. PyTorch. Documentation. [Електронний ресурс] - Режим доступу: https://pytorch.org/docs/stable/index.html. (дата звернення: 02.05.2023).

44. Gensim. "Doc2Vec - Gensim". [Електронний ресурс] - Режим доступу: https://radimrehurek.com/gensim/models/doc2vec.html (дата звернення: 04.05.2023)

45. Google. "Colaboratory". [Електронний ресурс] - Режим доступу: https://colab.research.google.com/ (дата звернення: 05.05.2023)

46. Google. TensorFlow. API Docs. [Електронний ресурс] - Режим доступу: https://www.tensorflow.org/api_docs (дата звернення: 06.05.2023)

47. Data. How to evaluate ML parameter tuning. [Електронний ресурс] - Режим доступу: https://web.archive.org/web/20160701182750/http:/blog.dato.com/how-to-evaluate-machine-learning-models-part-4-hyperparameter-tuning (дата звернення: 06.05.2023)

48. Machine Learning Mastery. Dropout for regularizing deep neural networks. [Електронний ресурс] - Режим доступу:

https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/ (дата звернення: 06.05.2023)

49. Пашін В. П. Методичні вказівки до виконання економіко організаційного розділу дипломних проектів (робіт) бакалаврів і спеціалістів для студентів інституту прикладного системного аналізу / В. П. Пашін, В. В. Романов, Н. В. Єгорова. // НТУУ “КПІ”. 2011. С. 118.

50. Coursera. Deep Learning Specialization. [Електронний ресурс] - Режим доступу: https://www.coursera.org/specializations/deep-learning. (Дата звернення: 02.05.2023).

51. Papers with Code. Fake News Detection. [Електронний ресурс] - Режим доступу: https://paperswithcode.com/task/fake-news-detection. (Дата звернення: 02.05.2023).

52. Towards Data Science. 20 Popular Machine Learning Metrics – Part 1: Classification & Regression Evaluation Metrics. [Електронний ресурс] - Режим доступу: https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce. (Дата звернення: 02.05.2023).

53. Hugging Face. Transformers Documentation. [Електронний ресурс] - Режим доступу: https://huggingface.co/docs/transformers/index. (Дата звернення: 02.05.2023).

54. Plotly Technologies Inc. Plotly Python API Reference. [Електронний ресурс] - Режим доступу: 23 квітня 2023 року. Адреса: https://plotly.com/python-api-reference/generated/plotly.data.html (дата звернення: 28.03.2023)

55. Fake news. Wikipedia. [Електронний ресурс] - Режим доступу: https://en.wikipedia.org/wiki/Fake_news (дата звернення 12.04.2023)

56. Towards Data Science. News Category Classification. Fine Tuning RoBERTa on TPUs. [Електронний ресурс] - Режим доступу: https://towardsdatascience.com/news-category-classification-fine-tuning-roberta-on-tpus-with-tensorflow-f057c37b093 (дата звернення: 15.05.2023)

ДОДАТОК

```
import pandas as pd

import numpy as np

import json

from bs4 import BeautifulSoup as bs

import requests as r

from tqdm import tqdm

from csv import writer

from pathlib import Path

from threading import Thread,Lock

import schedule

import os

from time import time


# mutexes for not breaking everything using threads

error_file_mutex = Lock()

news_file_mutex = Lock()

mutex_log = Lock()

mutex_log_max = Lock()

#function to parse one tweet by its id and write to csv file

def get_tweet_text_by_id(filepath,news_id,tweet_id,write_row_func):

    res =
r.get(f'https://publish.twitter.com/oembed?dnt=true&omit_script=true&url=https://mobile.twitter.com/i/stat
us/{tweet_id}')
```

```
    if res.status_code == 200:
        try:
            obj =json.loads(res.text)
            soup = bs(obj['html'],features='lxml')
            return soup.find('p').text
        except Exception as e:
            write_row_func(filepath,news_id,tweet_id,str(e))
            return ''
    else:
        write_row_func(filepath,news_id,tweet_id,res.status_code)
        return ''

df_with_parsed_tweets_columns = ['news_id','news_url','title','tweet_id','text','label']

required_files_with_cols = [('parsed\\news.csv',['id','title','news_url']),
                            ('parsed\\errors.csv',['filepath','news_id','tweet_id','error_text']),
                            ]

FakeNewsDataset_files = ['datasets/FakeNewsNet/gossipcop_fake.csv',
                         'datasets/FakeNewsNet/gossipcop_real.csv',
                         'datasets/FakeNewsNet/politifact_fake.csv',
                         'datasets/FakeNewsNet/politifact_real.csv']
```

```
labels_mapping = {'real':1,'fake':0}


for ds_file_path in FakeNewsDataset_files:
    _,__,dataset = ds_file_path.split('/')
    ds_name = (dataset.split('.'))[0]
    new_file_path = f'parsed\\parsed_tweets_{ds_name}.csv'
    required_files_with_cols.append((new_file_path,df_with_parsed_tweets_columns))
error_file_path = required_files_with_cols[1][0]
ERROR_FILE_DESC = open(error_file_path,'a',encoding='utf-8',newline='')
ERROR_WRITER = writer(ERROR_FILE_DESC)
#writing row to csv error
def write_error_row(filepath,news_id,tweet_id,error_text):
    error_file_mutex.acquire()
    try:
        ERROR_WRITER.writerow([filepath,news_id,tweet_id,error_text])
    finally:
        error_file_mutex.release()
def save_files():
    ERROR_FILE_DESC.close()
    # NEWS_FILE_DESC.close()



logfile = open('parsed\\log.txt','a+',encoding='utf-8')
```

```
logfile.close()


news_parsed = 3842

news_max = 0


START_TIME =time()


starting_points = {'parsed\\parsed_tweets_gossipcop_fake.csv': 'gossipcop-699303448',

                   'parsed\\parsed_tweets_gossipcop_real.csv': 'gossipcop-881670',

                   'parsed\\parsed_tweets_politifact_fake.csv': 'politifact15135',

                   'parsed\\parsed_tweets_politifact_real.csv': 'politifact954'}


def run_parsing(ds_file_path,use_starting_ids=False):

    global news_max

    global news_parsed

    _,__,dataset = ds_file_path.split('/')

    ds_name,ds_label = (dataset.split('.'))[0].split('_')

    new_file_path = new_file_path = f'parsed\\parsed_tweets_{ds_name+"_"+ds_label}.csv'

    label = labels_mapping[ds_label]

    df = pd.read_csv(ds_file_path)

    f = open(new_file_path,'a+',encoding='utf-8',newline='') # w+ to wipe +a to continue filling files

    f_writer = writer(f)
```

```
starting_id_reached = True
if use_starting_ids:
    starting_id = starting_points[new_file_path]
    starting_id_reached = False

try:
    mutex_log_max.acquire()
    news_max += df.shape[0]
    mutex_log_max.release()

    for index,row in df.iterrows():
        # news_url = row['news_url']
        # title = row['title']
        news_id = row['id']
        if news_id == starting_id:
            starting_id_reached = True

        if starting_id_reached:
        #write_news_row(news_id,title,news_url)
            try:
                tweet_ids = row['tweet_ids'].split('\t')
            except:
                pass
```

```
            for tweet_id in tweet_ids:
                tweet_content = get_tweet_text_by_id(ds_name+"
"+ds_label,news_id,tweet_id,write_error_row)
                if len(tweet_content)>0:
                    f_writer.writerow([news_id,tweet_id,tweet_content,label])
            mutex_log.acquire()
            news_parsed+=1
            with open('parsed\\log.txt','a',encoding='utf-8') as flog:
                flog.write(f'Time passed: {time()-START_TIME} seconds. News parsed: {news_parsed} of
{news_max}\n')
            mutex_log.release()

    finally:
        f.close()



threads = list()

for filepath in FakeNewsDataset_files:
    thread = Thread(target=run_parsing,args=(filepath,True))
    threads.append(thread)
    thread.start()

for thread in threads:
```

```
    thread.join()


save_files()


import pandas as pd
```

Також додаю код для завантаження чекпоінтів, тобто номерів новин, на яких було зупинено роботу.

```
FakeNewsDataset_files = ['datasets/FakeNewsNet/gossipcop_fake.csv',
                         'datasets/FakeNewsNet/gossipcop_real.csv',
                          'datasets/FakeNewsNet/politifact_fake.csv',
                         'datasets/FakeNewsNet/politifact_real.csv']



columns = ['news_id','tweet_id','text','label']


starting_points = []



for ds_file_path in FakeNewsDataset_files:
        _,__,dataset = ds_file_path.split('/')
        ds_name,ds_label = (dataset.split('.'))[0].split('_')
        new_file_path = new_file_path = f'parsed\\parsed_tweets_{ds_name+"_"+ds_label}.csv'
        df = pd.read_csv(new_file_path,header=None, names=columns,encoding='utf-8')
```

```
        last_id = df.iloc[-1]['news_id']
        starting_points.append((new_file_path,last_id))

print(starting_points.__str__())
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.figure_factory as ff

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option("display.max_colwidth", None)
starting_points = {'parsed\\parsed_tweets_gossipcop_fake.csv': 'gossipcop-699303448',
                   'parsed\\parsed_tweets_gossipcop_real.csv': 'gossipcop-881670',
                   'parsed\\parsed_tweets_politifact_fake.csv': 'politifact15135',
                   'parsed\\parsed_tweets_politifact_real.csv': 'politifact954'}

parsed_files_paths = list(starting_points.keys())

dfs_dict = {}

columns = ['news_id','tweet_id','text','label']

for parsed_file_path in parsed_files_paths:
```

```
            # for tweets df

            words = parsed_file_path.split('\\')[1].split('.')[0].split('_')
            words.pop(0)
            df_name_tweets = '_'.join(words)

            dfs_dict[df_name_tweets] = pd.read_csv(parsed_file_path,header=None,
names=columns,encoding='utf-8')

            # for news_df

            words.pop(0)
            df_name_news = '_'.join(words)

            file_path_news = f'datasets/FakeNewsNet/{df_name_news}.csv'

            dfs_dict[df_name_news] = pd.read_csv(file_path_news,encoding='utf-8')
dfs_dict['gossipcop_fake']['label']=0
dfs_dict['gossipcop_real']['label']=1
dfs_dict['politifact_fake']['label']=0
dfs_dict['politifact_real']['label']=1

news_dfs =
[dfs_dict['gossipcop_fake'],dfs_dict['gossipcop_real'],dfs_dict['politifact_fake'],dfs_dict['gossipcop_real']]

complete_news_df = pd.concat(news_dfs)
```

```
def count_number_of_tweets(string_object):
        try:
        return string_object.split('\t').__len__()
        except:
        return 0



def count_number_of_words(string_object):
        try:
        return string_object.split().__len__()
        except:
        return 0
complete_news_df['number_of_tweets'] = complete_news_df['tweet_ids'].apply(count_number_of_tweets)
complete_news_df['number_of_tweets'].sum()
from urllib.parse import urlparse


def get_domain(url):
        try:
        if not url.startswith('http'):
        url= 'http://' + url
        return urlparse(url).netloc
        except:
        return np.nan
complete_news_df['domain'] = complete_news_df['news_url'].apply(get_domain)
complete_news_df['Title Lenght'] = complete_news_df['title'].apply(count_number_of_words)
```

```
complete_news_df['Title Lenght'].head()



group_1 = complete_news_df[complete_news_df['label']==0]['Title Lenght'].values

group_2 = complete_news_df[complete_news_df['label']==1]['Title Lenght'].values

X = [group_1,group_2]

group_labels = [f'Fake Article Titles Lenght',f'True Article Titles Lenght']

fig = ff.create_distplot(X,group_labels=group_labels,show_hist=False)

fig.show()

temp_df['Frequency'] = temp_df['Count']/temp_df['Total']

temp_df =
temp_df.merge(temp_df[temp_df['label']==1][['domain','Frequency']].rename(columns={'Frequency':'Freq
uencyPositive'}),left_on='domain',right_on='domain',how='left')

temp_df =
temp_df.merge(temp_df[temp_df['label']==0][['domain','Frequency']].rename(columns={'Frequency':'Freq
uencyNegative'}),left_on='domain',right_on='domain',how='left')

temp_df.head(10)

temp_df.sort_values(['FrequencyNegative','label'],inplace=True,ascending=False)

REFERENCES = 15

px.bar(temp_df[temp_df['Total']>REFERENCES],x='domain',color='label',y='Frequency',height=600,title
=f'Websites having more than {REFERENCES} references')

DIVIDER = 50

group_1 = temp_df[temp_df['Total']>DIVIDER]['FrequencyNegative'].values

group_2 = temp_df[temp_df['Total']<=DIVIDER]['FrequencyNegative'].values

X = [group_1,group_2]

group_labels = [f'Sites having more than {DIVIDER} references',f'Sites having less than {DIVIDER}
references']

fig = ff.create_distplot(X,group_labels=group_labels,show_hist=False)
```

```
fig.update_layout(title='Negative Frequency')

fig.show()

px.scatter(temp_df[temp_df['label']==1],x='Total',y='FrequencyNegative',opacity=0.2)

nltk.download('stopwords')

stemmer = PorterStemmer()

stop_words = set(stopwords.words('english'))


def preprocess_text_for_column(dataframe,column):

    # remove twitter links

    dataframe[column]  = dataframe[column].apply(lambda x: re.sub(r'http\S+', '', x))

    # remove punctuation

    dataframe[column] = dataframe[column].str.replace('[{}]'.format(string.punctuation), '')

    # remove special characters

    dataframe[column] = dataframe[column].str.replace('[^a-zA-Z0-9\s]', '')

    # remove whitespaces

    dataframe[column] = dataframe[column].str.replace(r'\s+', ' ', regex=True)

    # additionaly remove line start and end spaces

    dataframe[column] = dataframe[column].str.strip()

    # remove stopwords

    dataframe[column] = dataframe[column].apply(lambda x: ' '.join([word for word in x.split() if
word.lower() not in stop_words]))

    #removing awkward photo links

    dataframe[column] = dataframe[column].apply(lambda x: ' '.join([word for word in x.split() if not
word.startswith('pictwittercom')]))

    # stem text

    # dataframe[column] = dataframe[column].apply(lambda x: ' '.join([stemmer.stem(word) for word in
x.split()]))
```

```
    # make lowercase
    dataframe[column] = dataframe[column].str.lower()
    return dataframe[column]
from collections import Counter


def count_words_frequency(dataframe,column):
    words = [word for title in dataframe[column] for word in title.split()]
    words_counter = Counter(words)
    words_df = pd.DataFrame(words_counter.items(),columns=['Word','Frequency'])
    return words_df
def create_dataframe_with_statistical_values_for_words(dataframe,columnWithWords):
    real_words_frequency = count_words_frequency(dataframe[dataframe['label']==1],columnWithWords)
    real_words_frequency['label'] = 1


    fake_words_frequency = count_words_frequency(dataframe[dataframe['label']==0],columnWithWords)
    fake_words_frequency['label'] = 0


    complete_freqs_dataframe = pd.concat([real_words_frequency,fake_words_frequency])
    temp_freq_dataframe =
complete_freqs_dataframe[['Word','Frequency']].groupby('Word').sum().reset_index().rename(columns={'
Frequency':'Total'})
    complete_freqs_dataframe =
complete_freqs_dataframe.merge(temp_freq_dataframe,left_on='Word',right_on='Word',how='left')


    complete_freqs_dataframe['FrequencyRelative'] =
complete_freqs_dataframe['Frequency']/complete_freqs_dataframe['Total']
    complete_freqs_dataframe =
complete_freqs_dataframe.merge(complete_freqs_dataframe[complete_freqs_dataframe['label']==1][['Wor
```

```
d','FrequencyRelative']].rename(columns={'FrequencyRelative':'FrequencyPositive'}),left_on='Word',right_on='Word',how='left')

    complete_freqs_dataframe = complete_freqs_dataframe.merge(complete_freqs_dataframe[complete_freqs_dataframe['label']==0][['Word','FrequencyRelative']].rename(columns={'FrequencyRelative':'FrequencyNegative'}),left_on='Word',right_on='Word',how='left')

    complete_freqs_dataframe.fillna(0,inplace=True)


    return complete_freqs_dataframe


import pandas as pd

import numpy as np

import torch

import torch.nn as nn

import torch.functional as F

#import tensorflow as tf

torch.cuda.is_available()


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

torch.version.cuda

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

if device.type == 'cuda':

    print(torch.cuda.get_device_name(0))

    print('Memory Usage:')

    print('Allocated:', round(torch.cuda.memory_allocated(0)/1024**3,1), 'GB')
```

```
    print('Cached:   ', round(torch.cuda.memory_reserved(0)/1024**3,1),
'GB')

# del df['TextCleared']

df = df.dropna()

df.sample(5)

from sklearn.metrics import
accuracy_score,f1_score,precision_score,recall_score,roc_auc_score

def outputmetrics(model,train_X,train_Y,test_X,test_Y):

    y_pred_train = model.predict(train_X)

    y_pred_test = model.predict(test_X)

    f1_score_train = f1_score(train_Y, y_pred_train)

    f1_score_test= f1_score(test_Y, y_pred_test)


    precision_train = precision_score(train_Y, y_pred_train)

    precision_test= precision_score(test_Y, y_pred_test)


    recall_train = recall_score(train_Y, y_pred_train)

    recall_test= recall_score(test_Y, y_pred_test)



    accuracy_score_train = accuracy_score(train_Y,y_pred_train)

    accuracy_score_test = accuracy_score(test_Y,y_pred_test)


    roc_auc_score_train = accuracy_score(train_Y,y_pred_train)

    roc_auc_score_test = accuracy_score(test_Y,y_pred_test)

    print(f'For train set:')
```

```
    print(f'Precision : {precision_train} Recall : {recall_train} f1 :
{f1_score_train} Accuracy : {accuracy_score_train} AUC :
{roc_auc_score_train}')

    print(f'For test set:')

    print(f'Precision : {precision_test} Recall : {recall_test} f1 :
{f1_score_test} Accuracy :  {accuracy_score_test} AUC :
{roc_auc_score_test}')

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split




# Split the dataframe into train and test sets

X_train, X_test, y_train, y_test = train_test_split(df['TextCleared'],
df['label'],train_size=0.9, random_state=42)


# Create a TfidfVectorizer object

tfidf = TfidfVectorizer()


# Fit and transform the training data using the TfidfVectorizer object

X_train_tfidf = tfidf.fit_transform(X_train)


# Create a LogisticRegression object

lr = LogisticRegression(max_iter=1000)


# Fit the model on the training data

lr.fit(X_train_tfidf, y_train)
```

```
# Transform the test data using the TfidfVectorizer object

X_test_tfidf = tfidf.transform(X_test)

from gensim.models.doc2vec import Doc2Vec, TaggedDocument


import nltk

nltk.download('punkt')


tagged_data = [TaggedDocument(words=nltk.word_tokenize(row),
tags=[str(i)]) for i, row in enumerate(X_train)]


model = Doc2Vec(vector_size=64, min_count=2, epochs=40)


model.build_vocab(tagged_data)

model.train(tagged_data, total_examples=model.corpus_count,
epochs=model.epochs)

model.train(tagged_data, total_examples=model.corpus_count,
epochs=model.epochs)

vectors_train = [model.infer_vector(tagged_data[i].words) for i in
range(len(tagged_data))]

vectors_test = [model.infer_vector(tagged_data[i].words) for i in
range(len(tagged_data_test))]

clf = LogisticRegression(max_iter=1000)

clf.fit(vectors_train, y_train)


# evaluate the model on the test set

accuracy = clf.score(vectors_test, y_test)
```

```
print(f'Test accuracy: {accuracy}')

outputmetrics(clf,vectors_train,y_train,vectors_test,y_test)

import multiprocessing


multiprocessing.cpu_count()


from transformers import BertTokenizer



tokenizer = BertTokenizer.from_pretrained('prajjwal1/bert-tiny',
do_lower_case=False)

from transformers import AutoModel


model = AutoModel.from_pretrained("prajjwal1/bert-tiny")

import regex as re

import string


def preprocess_text_for_column(dataframe,column):

    processing_df = dataframe[column].copy()

    # remove twitter links

    processing_df  = processing_df.apply(lambda x: re.sub(r'http\S+',
'', x))

    # remove punctuation

    processing_df =
processing_df.str.replace('[{}]'.format(string.punctuation), '')

    # remove special characters
```

```
    #processing_df = processing_df.str.replace('[^a-zA-Z0-9\s]', '')

    # remove whitespaces

    processing_df = processing_df.str.replace(r'\s+', ' ', regex=True)

    # additionaly remove line start and end spaces

    processing_df = processing_df.str.strip()

    processing_df = processing_df.str.strip('.')

    # remove stopwords

    #processing_df = processing_df.apply(lambda x: ' '.join([word for
word in x.split() if word.lower() not in stop_words]))

    #removing awkward photo links

    processing_df = processing_df.apply(lambda x: ' '.join([word for
word in x.split() if not word.startswith('pictwittercom')]))

    # stem text

    # dataframe[column] = dataframe[column].apply(lambda x: '
'.join([stemmer.stem(word) for word in x.split()]))

    # make lowercase

    processing_df = processing_df.str.lower()

    return processing_df




df['TextCleared'] = preprocess_text_for_column(df,'text')

input_ids_train = []

attention_masks_train = []


# For every sentence...

for sent in df['TextCleared']:
```

```
    # `encode_plus` will:
    #   (1) Tokenize the sentence.
    #   (2) Prepend the `[CLS]` token to the start.
    #   (3) Append the `[SEP]` token to the end.
    #   (4) Map tokens to their IDs.
    #   (5) Pad or truncate the sentence to `max_length`
    #   (6) Create attention masks for [PAD] tokens.
    encoded_dict = tokenizer.encode_plus(
                        sent,                      # Sentence to
encode.
                        add_special_tokens = True, # Add '[CLS]' and
'[SEP]'
                        max_length = 128,           # Pad & truncate
all sentences.
                        pad_to_max_length = True,
                        return_attention_mask = True,   # Construct
attn. masks.
                        return_tensors = 'pt',     # Return pytorch
tensors.
                   )

    # Add the encoded sentence to the list.
    input_ids_train.append(encoded_dict['input_ids'])

    # And its attention mask (simply differentiates padding from
non-padding).
    attention_masks_train.append(encoded_dict['attention_mask'])
```

```
# Convert the lists into tensors.

input_ids = torch.cat(input_ids_train, dim=0)

attention_masks = torch.cat(attention_masks_train, dim=0)

labels = torch.tensor(df['label'])


# Print sentence 0, now as a list of IDs.

print('Original: ', X_train[0])

print('Token IDs:', input_ids[0])

from torch.utils.data import TensorDataset, random_split


# Combine the training inputs into a TensorDataset.

dataset = TensorDataset(input_ids, attention_masks, labels)


# Create a 90-10 train-validation split.


# Calculate the number of samples to include in each set.

train_size = int(0.9 * len(dataset))

val_size = len(dataset) - train_size


# Divide the dataset by randomly selecting samples.

train_dataset, val_dataset = random_split(dataset, [train_size,
val_size])


print('{:>5,} training samples'.format(train_size))
```

```
print('{:>5,} validation samples'.format(val_size))

from torch.utils.data import DataLoader, RandomSampler,
SequentialSampler


# The DataLoader needs to know our batch size for training, so we
specify it

# here. For fine-tuning BERT on a specific task, the authors recommend
a batch

# size of 16 or 32.

batch_size = 16


# Create the DataLoaders for our training and validation sets.

# We'll take training samples in random order.

train_dataloader = DataLoader(

            train_dataset,  # The training samples.

            sampler = RandomSampler(train_dataset), # Select batches
randomly

            batch_size = batch_size # Trains with this batch size.

        )


# For validation the order doesn't matter, so we'll just read them
sequentially.

validation_dataloader = DataLoader(

            val_dataset, # The validation samples.

            sampler = SequentialSampler(val_dataset), # Pull out
batches sequentially.

            batch_size = batch_size # Evaluate with this batch size.
```

```
        )

import gym

import numpy as np

import matplotlib.pyplot as plt




def plot_graphs(history, metric):

  plt.plot(history.history[metric])

  plt.plot(history.history['val_'+metric], '')

  plt.xlabel("Epochs")

  plt.ylabel(metric)

  plt.legend([metric, 'val_'+metric])

df =
pd.read_csv('drive/MyDrive/diploma_data/tweets_no_stemming_no_stopword_
removal_with_original_text.csv')
```

```
df = df.dropna()

import tensorflow as tf

import pandas as pd

from sklearn.model_selection import train_test_split




train_df, test_df = train_test_split(df, test_size=0.2,random_state=42)
```

```
train_dataset =
tf.data.Dataset.from_tensor_slices((train_df['TextCleared'].values,
train_df['label'].values))

test_dataset =
tf.data.Dataset.from_tensor_slices((test_df['TextCleared'].values,
test_df['label'].values))
```

```
BUFFER_SIZE = 10000

BATCH_SIZE = 256

train_dataset =
train_dataset.shuffle(BUFFER_SIZE).batch(BATCH_SIZE).prefetch(tf.data.A
UTOTUNE)

test_dataset =
test_dataset.batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)

VOCAB_SIZE = 10000

encoder = tf.keras.layers.TextVectorization(

    max_tokens=VOCAB_SIZE)

encoder.adapt(train_dataset.map(lambda text, label: text))

model = tf.keras.Sequential([

    encoder,

    tf.keras.layers.Embedding(

        input_dim=len(encoder.get_vocabulary()),

        output_dim=64,

        mask_zero=True),

    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),

    tf.keras.layers.Dense(64, activation='relu'),

    tf.keras.layers.Dense(1)

])
```

```
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True)
,
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])
```

```
history = model.fit(train_dataset, epochs=10,
                    validation_data=test_dataset,
                    validation_steps=30)
model = tf.keras.Sequential([
    encoder,
    tf.keras.layers.Embedding(len(encoder.get_vocabulary()), 64,
mask_zero=True),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128,
return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1)
])
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True)
,
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy',tf.keras.metrics.AUC()])
history = model.fit(train_dataset, epochs=30,
                    validation_data=test_dataset,
                    validation_steps=30)
```

```
      plt.figure(figsize=(16, 6))

      plt.subplot(1, 2, 1)

      plot_graphs(history, 'accuracy')

      plt.subplot(1, 2, 2)

      plot_graphs(history, 'loss')

      !pip install -q -U "tensorflow-text==2.11.*"

      import tensorflow_text as text

      from official.nlp import optimization

      import tensorflow_hub as hub

      import os

      import shutil

      def build_classifier_model():

         text_input = tf.keras.layers.Input(shape=(), dtype=tf.string,
name='text')

         preprocessing_layer = hub.KerasLayer(tfhub_handle_preprocess,
name='preprocessing')

        encoder_inputs = preprocessing_layer(text_input)

         encoder = hub.KerasLayer(tfhub_handle_encoder, trainable=True,
name='BERT_encoder')

        outputs = encoder(encoder_inputs)

        net = outputs['pooled_output']

        net = tf.keras.layers.Dropout(0.1)(net)

               net    =    tf.keras.layers.Dense(1,    activation=None,
name='classifier')(net)

        return tf.keras.Model(text_input, net)
```

```
      epochs = 5

      steps_per_epoch                                                    =
tf.data.experimental.cardinality(train_dataset).numpy()

      num_train_steps = steps_per_epoch * epochs

      num_warmup_steps = int(0.1*num_train_steps)



      init_lr = 3e-5

      optimizer = optimization.create_optimizer(init_lr=init_lr,


num_train_steps=num_train_steps,


num_warmup_steps=num_warmup_steps,

                                                optimizer_type='adamw')
```