НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

«На правах рукопису»

УДК 004.852

До захисту допущено:

В.о. завідувача кафедри ШІ

О.І. ЧУМАЧЕНКО

«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 122 «Комп'ютерні науки»
на тему: «Розпізнавання фейкових новин, використовуючи методи штучного інтелекту»

Виконав:
студент 2 курсу, групи КІ -з11мп
Степенко Богдан Валентичнович _____

Керівник: доцент кафедри ММСА,
к.ф-.м.н., доц. ШубенковаІ.А.

_____

Рецензент:

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ-2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Рівень вищої освіти — другий (магістерський)
Спеціальність (ОПП) — 122 «Комп'ютерні науки» («Системи і методи штучного інтелекту»)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри ШІ

_____ О.І. ЧУМАЧЕНКО

«___»_____2022 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Степенку Богдану Валентиновичу
(прізвище, ім’я, по батькові)

1. Тема дисертації:«», науковий керівник дисертації Шубенкова Ірина Анатоліївна, к.ф.-м.н., доцент, затверджені наказом по університету від «02» листопада 2022 р. № 4040-с.

2. Термін подання студентом дисертації: 14.12.2022 р.

3. Об’єкт дослідження: Методи та моделі ШІ.

4. Предмет дослідження: Системи розпізнавання фейкових новин на основі методів штучного інтелекту.

5. Перелік завдань, які потрібно зробити:

1) здійснити огляд літератури за темою роботи;

2) дослідити актуальність обраної теми;

3) ознайомитись із існуючими методами та моделями;

4) здійснити порівняльний аналіз існуючих методів;

5) розробити та реалізувати систему;

6) провести експеримент, що засвідчує працеспроможність запропонованої моделі, виконати аналіз результатів;

7) провести аналіз ринкових можливостей запуску стартап проєкту;

8) зробити висновки;

9) підготувати ілюстративний матеріал;

10) оформити пояснювальну записку.

6. Перелік ілюстративного матеріалу.

7. Дата видачі завдання:1 вересня 2022 року.

Календарний план:

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання магістерської дисертації	Примітка
1.	Вивчення літератури за темою роботи.	02.09.2022 –14.09.20212	Виконано
2.	Підготовка першого розділу.	15.09.2022 – 29.09.2022	Виконано
3.	Підготовка другого розділу.	30.09.2022 – 18.10.2022	Виконано
4.	Розробка програмного продукту.	19.10.2022 – 05.11.2022	Виконано
5.	Підготовка третього розділу.	06.11.2022 – 18.11.2022	Виконано
6.	Підготовка частини стартап-проєкту.	19.11.2022 – 22.11.2022	Виконано
7.	Концептуальні висновки. Перспективи розвитку отриманих рішень.	23.11.2022 – 25.11.2022	Виконано
8.	Оформлення пояснювальної записки.	26.11.2022 – 03.12.2022	Виконано

Студент Богдан Степенко

Науковий керівник дисертації Ірина ШУБЕНКОВ

РЕФЕРАТ

Магістерська дисертація: 78 с., 19 рис., 32 табл., 37 джерел.

FAKE NEWS DETECTION, BiLSTM, Long Short-Term Memory, Natural Language Processing.

Об’єкт дослідження – Методи та моделі ШІ.

Предмет дослідження – Системи розпізнавання фейкових новин на основі методів штучного інтелекту.

Мета роботи – дослідити та зробити власну модель розпізнавання фейкової інформації, зробити аналіз з найсучаснішими суміжними роботами, що показали найкращий результат.

У минулому для виявлення фейкових новин здебільшого використовували класифікаційні та регресійні моделі, в яких вхідними даними були або зміст статті, або шляхи поширення чуток. Однак у цій роботі представлено модель глибокої нейронної мережі на основі двонаправленої довготривалої короткочасної пам'яті (LSTM).

Дана робота має неабияку актуальність в сьогодення, а саме у військовий час, коли кожен може отримувати величезну кількість інформації з соціальних мереж, але не кожен вміє її правильно аналізувати чи сприйняти, що може створювати негативні настрої в суспільстві. Проте використовуючи відповідну розробку, наприклад як телеграм бот чи додаток в браузері користувач може побачити певну інформацію як потенційно недостовірну, хоча вона і матиме гучні заголовки.

ABSTRACT

Master’s thesis explanatory note: 78 p., 19 fig., 32 tables, 37 sources.

FAKE NEWS DETECTION, BiLSTM, Long Short-Term Memory, Natural Language Processing.

Object of research - Methods and models of AI.

Subject of research - The systems of recognition of fake news are on the basis of methods of artificial intelligence.

The purpose of the work is to investigate and make our own model for recognizing fake information, to make an analysis with the most modern related works that have shown the best result.

In the past, fake news detection mostly used classification and regression models, in which the input data was either the content of the article or the path of the rumor. However, this paper presents a deep neural network model based on bidirectional long short-term memory (LSTM).

This work is of great relevance today, namely in wartime, when everyone can receive a huge amount of information from social networks, but not everyone knows how to analyze or perceive it correctly, which can create negative sentiments in society. However, using the appropriate development, such as a Telegram bot or a browser application, the user can see certain information as potentially unreliable, although it will have loud headlines.

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

NN - Neural Networks

DNN - Deep Neural

RNN - Recurrent Neural Networks

GRU - Gated Recurrent Unit

LSTM - Long Short Term Memory

BiLSTM - Bidirectional Long Short Term Memory

CNN - Convolutional Neural Networks

TF - Term Frequency

IDF - Inverse Document Frequency

NLP - Natural Language Processing

DL - Deep Learning

ML - Machine Learning

SMOTE - Synthetic Minority Over-sampling Technique

CDF - Cumulative Distribution Function

SVM - Support Vector Machine

NB - Naive Bayes

NLTK - Natural Language ToolKit

RE - Regular Expressions

TP - True Positives

TN - True Negatives

FP - False Positives

FN - False Negatives

RMSProp - Root Mean Square Propagation

AdaGrad - Adaptive Gradient

AdaM - Adaptive Moment

ВСТУП

Згідно з Кембриджським словником[1], фейкові новини - це "історії, які виглядають як новини, поширюються в Інтернеті або за допомогою інших засобів масової інформації, зазвичай створюються з метою впливу на політичні погляди або як жарт. Це проблема, що викликає все більше занепокоєння в сучасному світі. Зазвичай фейкові новини поширюються ботами в геометричній прогресії в соціальних мережах. Створювати новини, маніпулюючи відео та зображеннями, стає все легше і легше, тому неправдиві новини стали проблемою, що викликає занепокоєння, особливо в соціальних мережах, де фейкові новини поширюються за допомогою мемів.

Багато фейкових новин щодо вірусу коронавірусу з Китаю. Найпоширеніші з них - про те, що китайці заразилися через вживання в їжу кажанів, що не відповідає дійсності, оскільки існує сім видів коронного вірусу, а той, що спричинив спалах, є новим видом коронного вірусу. Коли йдеться про фейкові новини під час виборів, увага здебільшого переключається на Захід, залишаючи без уваги глобальний південь. Це призвело до збільшення на 40 відсотків поширення фейкових новин в Індії під час виборів.

У Бразилії у 2019 році більшість фейкових новин, поширених у WhatsApp під час президентських виборів, сприяли перемозі ультраправого кандидата Жаїра Болсонару [2] З 11957 вірусних повідомлень, поширених у 296 групових чатах на платформі миттєвих повідомлень у період передвиборчої кампанії, приблизно 42% матеріалів правого спрямування містили інформацію, яка була визнана фактчекерами неправдивою. У Південній півкулі найпопулярнішим додатком для обміну повідомленнями є WhatsApp. Він пропонує зашифровані пірінгові повідомлення, і його дуже складно контролювати, на відміну від таких додатків, як Facebook. Моніторинг розмов користувачів порушує конфіденційність користувачів,

але є корисним для виявлення фейкових новин та їхніх джерел. Для боротьби з фейковими новинами було вжито низку ініціатив, таких як французький закон про боротьбу з маніпулюванням інформацією [3].

Закон запобігає впливу на результати виборів, що було зроблено, наприклад, під час голосування за Brexit, коли газети розпалювали ненависть до іммігрантів та ЄС[4].

Закон запобігає політичному впливу, вимагаючи зобов'язання щодо прозорості для цифрових платформ, які повинні повідомляти про будь-який спонсорський контент, публікуючи ім'я автора та сплачену суму. Платформи, що перевищують певну кількість переглядів на день, повинні мати юридичного представника у Франції та публікувати свої алгоритми"[3].

Закон також вимагає від судді кваліфікувати фейкові новини на основі наступних трьох критеріїв:

- Фейкові новини повинні бути явними.
- Фейкові новини мають розповсюджуватися цілеспрямовано і масово.
- Фейкові новини повинні призвести до порушення громадського порядку або поставити під загрозу результати виборів.

Закон також вимагає співпраці між різними цифровими платформами під час виборчого періоду. Французький орган з питань телерадіомовлення відповідає за посилення правозастосування цієї вимоги. Крім того, Еммануелю Хугу, колишньому президенту Французького агентства преси, було доручено створити орган з питань етики преси.

Заходи з протидії фейковим новинам також були вжиті у Фінляндії, Малайзії та Сінгапурі. Фінляндія запустила ініціативу по боротьбі з фейковими новинами, яка має на меті навчити розпізнавати фейкові новини, ще у 2014 році[5]. Уряд Малайзії ухвалив Закон про боротьбу з фейковими новинами у 2018 році[6]. Однак Закон був звинувачений у прагненні придушити критику адміністрації. У 2019 році в Сінгапурі був прийнятий ще один закон проти фейкових новин під назвою "Закон про захист від

неправдивої інформації та маніпуляцій в Інтернеті". Цей закон отримав такі ж звинувачення, як і закон проти фейкових новин з Малайзії[7].

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Вплив фейкових новин

Фейкові новини мали величезний вплив на індустрію новин з моменту появи Інтернету. Журналісти перебувають під дедалі більшим тиском, щоб виробляти більше матеріалів швидше, і сьогодні вони, як правило, використовують соціальні медіа як джерело інформації [8]. При використанні соціальних мереж як джерела інформації перевірка достовірності новин є великою проблемою з таких причин, як тиск, пов'язаний з необхідністю виробляти більше. Коли журналісти неправильно перевіряють джерела новин, репутація їхнього бренду підривається, а люди втрачають довіру до ЗМІ. На створення репутації бренду можуть піти роки, а на її руйнування - лічені секунди. Тиск на швидкий випуск матеріалу також призводить до менш серйозної журналістики.

Фейкові новини є багатогранним і складним питанням. Вони можуть поширюватися для розпалювання конфліктів, для отримання економічної вигоди, дифамації або в політичних цілях. Крім того, фейкові новини призводять до того, що люди в цілому стають менш поінформованими.

У 2019 році було поширено фейкову новину про смерть 22-річного китайського студента Алекса Чоу, в якій стверджувалося, що він не покінчив життя самогубством, а був або переслідуваний, або виштовхнутий з парковки поліцією Гонконгу [9]. У новинах також стверджувалося, що офіцери заблокували швидку допомогу, яка не могла дістатися до Чоу. Метою цих фейкових новин було підбурювання до антиурядових протестів.

У 2013 році фейкові новини про два вибухи в Білому домі, спрямовані проти попереднього президента США Барака Обами, призвели до падіння промислового індексу Доу-Джонса на 143,5 пункти, що відповідає загальній вартості акцій на суму 130 млрд доларів США [10]. Дональд Трамп оголосив CNN і Washington Post "фейковими новинами" через те, що вони не підтримували його (називали його послідовників сектою і стверджували, що він неодноразово "зачаровував" республіканську партію)[11].

Фейкові новини завжди використовувалися в політичному контексті, наприклад, поширення неправдивих даних опитувань з метою відвернути людей від участі в голосуванні під час політичних виборів, наприклад, у Мексиці в 1988 році PRI (Інституційно-революційна партія), але ніколи в такому масштабі та обсязі, як з використанням сучасних технологій. Під час президентських виборів 2016 року в США, які завершилися обранням Дональда Трампа президентом США, спостерігався сплеск фейкових новин у соціальних мережах. Розслідування після президентських виборів вказують на російський вплив під час кампанії[12].

Інший випадок використання фейкових новин як засобу політичного впливу стався у 2014 році, коли ІДІЛ почав поширювати пропаганду в усіх можливих соціальних мережах. Демократія побудована на довірі до здатності громадськості до аргументованої комунікації. Цифрові технології зробили інформацію більш доступною для громадськості, однак, оскільки глибокі фейкові новини стають все більш просунутими, люди стають менш поінформованими, що ставить під загрозу демократію. Політичний вплив, який здійснювався PRI, Росією та Болсонару, як зазначалося вище, також загрожує демократії, оскільки централізує владу та забирає її у народу.

1.2. Визначення проблеми

У цій роботі досліджується, чи можна використовувати графові нейронні мережі для покращення виявлення фейкових новин. Для цього використовується двонаправлений LSTM з метаданими, вмістом статті та шляхами поширення чуток як вхідними даними для маркування статей як справжніх або фейкових (вміст статті та шляхи поширення чуток здебільшого використовувалися окремо в існуючих роботах). Результати роботи порівнюються з кількома класифікаторами глибокого навчання та машинного навчання.

1.3. Виклики

Виявлення фейкових новин пов'язане з багатьма проблемами. Платформи фейкових новин зазвичай імітують справжні новинні платформи за дизайном і контентом. Фейкові новини не завжди на 100% фейкові, в них можуть бути правдиві твердження, змішані з неправдивими.

За сучасних технологій зображення та відео можуть бути сфабриковані, що унеможливлює миттєву перевірку достовірності новин. Також легко створювати фейкові суб'єкти для поширення фейкових новин. Для цього в Інтернеті є безкоштовні інструменти, а для підміни облич можна використовувати наявні фотографії[13]. Статті фейкових новин, як правило, мають або романтичний, або драматичний зміст, і часто містять більше вищого ступеня порівняння та емоційно-забарвлених слів (для гри на емоціях), ніж справжні новини, отже, обробка природної мови (НЛП) повинна бути життєздатною для виявлення фейкових новин. Однак NLP є складним і трудомістким процесом, оскільки слова можуть мати різні значення в різних контекстах, а кожна мова і діалект вимагають окремої версії NLP. Кількість доступних даних обмежена, а реальних новин більше, ніж фейкових. Це впливає на навчання моделей класифікації.

1.4. Аналіз методів для розв’зку задачі

У минулому для виявлення фейкових новин здебільшого використовували класифікаційні та регресійні моделі, в яких вхідними даними були або зміст статті, або шляхи поширення чуток.

Однак у цій роботі представлено модель глибокої нейронної мережі на основі двонаправленої довготривалої короткочасної пам'яті (LSTM), яка використовує контент статті, шляхи поширення чуток у вигляді часових рядів і метадані як вхідні дані. Вміст статті - це вкладені слова. Метадані складаються з низки описових параметрів як для статей, так і для відповідних твітів і ретвітів, таких як ідентифікатор статті, твіт і ідентифікація ретвітів, контент твітів та ретвітів, кількість підписників.

Описові параметри в метаданих є безперервними числовими атрибутами, вони нормалізовані та дискретизовані. Часові ряди будуються на основі шляхів поширення чуток. Нарешті, вміст вбудованої статті, дискретизовані безперервні атрибути та часові ряди об'єднуються і подаються в двонаправлену мережу LSTM, яка класифікує новинні статті з набору даних Politifact як справжні або фейкові. CNN, C-LSTM (на основі моделі C-LSTM від [14] з використанням метаданих та шляхів поширення чуток на додаток до змісту статті, CNN+LSTM (на основі моделі CNN+GRU з [15] з використанням змісту статті та метаданих на додаток до шляхів поширення чуток), Multinomial NB, SVM та моделі з [15] і [14] з найкращими результатами також реалізовані та протестовані для порівняння.

1.5. Суміжні роботи

За останні кілька років було проведено багато досліджень щодо виявлення фейкових новин у соціальних мережах. Багато з цих робіт використовували методи машинного навчання, такі як дерева рішень та лінійні машини опорних векторів (SVM), як [16] та [14].

У роботах було досягнуто точності прогнозування для виявлення фейкових новин у соціальних мережах 82,7% та 65% відповідно з використанням дерев рішень. При використанні SVM було досягнуто точності 75,5% та 66%. Деякі з цих робіт зосереджені на контенті статей, як, наприклад, [17], які досягли точності 62,4%, використовуючи Fakebox (модель машинного навчання) та fake-real-newsdataset McIntire's fake-real-newsdataset.

Останнім часом також проводяться дослідження з використанням глибинного навчання. Ці роботи зосереджені переважно на шляхах поширення чуток, а не стільки на контенті, наприклад [15]. [15] стверджує, що може виявляти фейкові новини з точністю 85% та 92% у Twitter та Sina Weibo відповідно через п'ять хвилин після початку їх розповсюдження. Робота на основі графових нейронних мереж, зосереджена на даних зображень, отримала точність 94%, використовуючи модель на основі згорткових нейронних мереж (CNN) [17]. [17] також отримали точність 91%, використовуючи LSTM на основі змісту та заголовків статей.

У роботі використовувалися фейкові новини з набору даних "Getting Real About Fake News", а також справжні новини з таких джерел, як "The New York Times" і "The Washington Post". Деякі роботи також застосовували TF-IDF для виявлення фейкових новин, наприклад, [14], де точність 95% була досягнута за допомогою уніграми Naive Bayes на комбінації наборів даних "Liar Liar" і "Fake or Real News". Згадані результати свідчать про те, що

виявлення фейкових новин, як правило, є найбільш успішним з використанням глибокого навчання (особливо LSTM) порівняно з алгоритмами машинного навчання, але також і про те, що алгоритми машинного навчання можуть бути кращими для невеликих наборів даних.

1.5.1. Роботи на основі машинного навчання

- Ефективна система виявлення фейкових новин з використанням машинного навчання [16] виявляє фейкові новини за допомогою наступних класифікаторів машинного навчання: SVM, K-Nearest Neighbors (KNN), Decision tree, Random forest (RF). Найкраща точність була отримана при використанні алгоритмів NLP та класифікатора RF на наборі даних з Kaggle Fake News Challenge. Причина, чому алгоритми машинного навчання виявилися навіть кращими, ніж моделі глибокого навчання в цьому випадку, може полягати в тому, що моделі глибокого навчання вимагають заповнення або стиснення даних, або, можливо, в тому, що набір даних "Брехун-брехун" містить висловлювання, а не повні статті. Набір даних містить кількість статей, а також ідентифікатор статті, назву, автора та мітку.
- Еталонне дослідження методів машинного навчання для виявлення фейкових новин [14] виявляє фейкові новини за допомогою наступних класифікаторів машинного навчання: SVM, логістичної регресії (LR), дерева рішень, Adaboost, наївного Байєса та K-найближчих сусідів. Було використано комбінацію набору даних "Liar liar pants on fire" та набору даних конкурсу фейкових новин Kaggle Fake News. Набір даних 'Liar liar' містить висловлювання з набору даних Politifact. Найкраща точність була досягнута на рівні 95% при використанні мультимножинного наївного байєсівського (NB) класифікатора з

функціями уніграми TF-IDF. Дисертант рекомендує використовувати n-грамові TF-IDF функції з NB для невеликих наборів даних та одноrрамові для великих наборів даних і стверджує, що LSTM класифікатори краще долають перенавчання, ніж NB класифікатори.

1.5.2. Роботи на основі глибокого навчання

- Виявлення фейкових новин за допомогою глибокого навчання [14] виявляє фейкові новини за допомогою LSTM, CNN та Bidirectional Encoder Representations from Transformers (BERT). BERT, використовуючи архітектуру Google "BASE" та токенізацію слів "Wordpiece", показав найкращі результати, а CNN мав дещо кращу точність, ніж LSTM. BERT - це мережева техніка НЛП. Фейкові новини зібрані з набору даних "Getting Real About FakeNews", а справжні статті - з авторитетних джерел, таких як "The New York Times" та "The Washington Post".
- Раннє виявлення фейкових новин у соціальних мережах за допомогою класифікації шляхів поширення за допомогою рекурентних і згорткових мереж [15] виявляє фейкові новини на ранніх стадіях після того, як статті з'являються у Twitter, використовуючи CNN, RNN, SVM, Difficult-to-Treat Resistance (DTR), Gated Recurrent Unit (GRU), RF, Propagation Tree Kernel (PTK) для класифікації шляхів розповсюдження. Було використано три набори даних, що містять дані з соціальних мереж: Sina-Weibo, Twitter 2015 та Twitter 2016. Запропонована в дисертації модель є комбінацією RNN (GRU) та CNN з використанням максимального об'єднання. Використання тільки RNN дало дещо кращі результати, ніж CNN окремо, але різниця є незначною.

- Порівняльне дослідження методів машинного навчання для виявлення фейкових новин [14] впроваджує та оцінює різні методи машинного навчання та глибокого навчання для виявлення фейкових новин. Було використано набір даних Макінтайра "Fake or Real News", набір даних Ванга "Liar liar, pants on fire" та їх комбінацію. У роботі зроблено висновок, що хоча моделі глибокого навчання, а саме C-LSTM (CNN-LSTM) та двонаправлений LSTM, дають найкращі результати, алгоритми машинного навчання (наприклад, лінійний SVM) можуть бути кращими для невеликих наборів даних за умови правильного відбору ознак з урахуванням швидкості навчання.

РОЗДІЛ 2 ОПИС НЕЙРОННИХ МЕРЕЖ ДЛЯ РОЗВ’ЯЗКУ ЗАДАЧІ

2.1. Нейронні мережі

Нейронні мережі засновані на людському мозку в тому сенсі, що вони є алгоритмами, що складаються з декількох вузлів, з'єднаних ребрами, що імітують нейрони [18]. Нейронні мережі з декількома прихованими шарами називаються глибокими нейронними мережами. Приховані шари - це шари між вхідним і вихідним шарами (див. рис. 2.1). Глибока нейронна мережа - це нейронна мережа, яка має декілька прихованих шарів. Глибокі нейронні мережі дозволяють комп'ютерам штучно навчатися і є підкатегорією штучного інтелекту (AI) [19].Глибокі нейронні мережі можуть, наприклад, використовуватися для вивчення особливостей з таких даних, як текст, зображення, звук, або вони можуть використовуватися для класифікації та регресії.

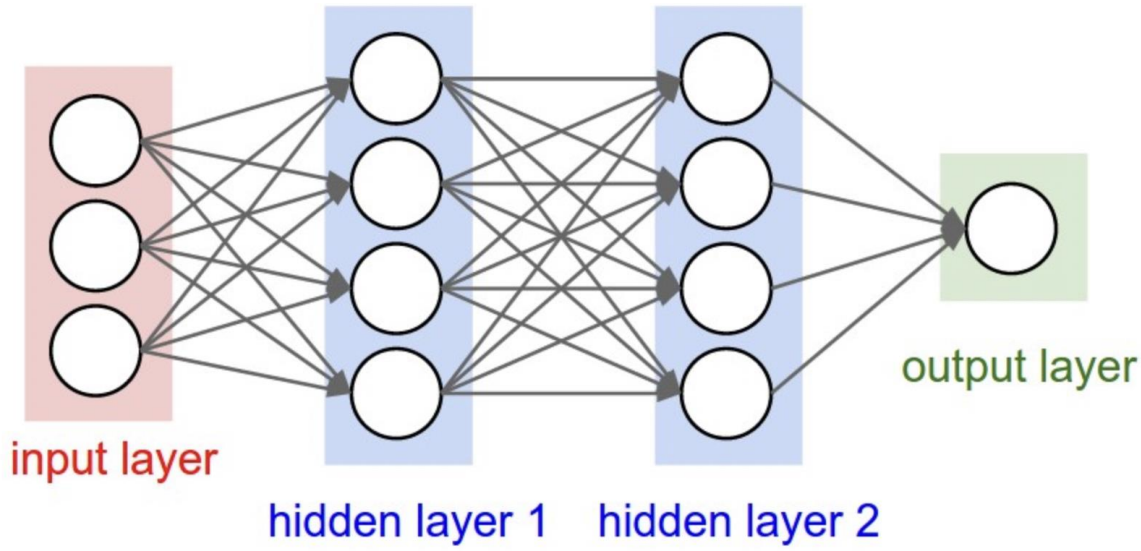

Рис. 2.1. - Шари нейронних мереж

Як зазначалося вище, нейронні мережі мають багато застосувань, але в даному випадку цільові значення є або реальними, або фальшивими, що означає, що це задача бінарної класифікації. Всі вузли в прихованих шарах і вихідному шарі мають свій власний класифікатор. Класифікатори активують вузли на основі входів з попереднього шару та функції активації. Існує багато функцій активації, і вибір функції активації залежить від багатьох факторів. В даній роботі використовується ReLU (Rectified Linear Unit - випрямлена лінійна одиниця) з наступним сигмоїдом. ReLU є найбільш поширеною функцією активації.

2.1.1. Feed-Forward (Forward-Propagation)

Перший прихований шар отримує вхідні дані від вхідного шару, а потім класифікаційні оцінки передаються від шару до шару. Тільки активовані вузли передають свої класифікаційні оцінки наступному шару. У вихідному шарі цільові значення (в даному випадку реальні або фейкові) класифікуються на основі остаточних класифікаційних оцінок, що означає, що на виході отримується прогнозований клас. Цей процес називається прямим розповсюдженням (feed-forward) або прямим поширенням (forward-propagation). Виходи вузлів, H_i, обчислюються шляхом попереднього обчислення мереж вузлів, Z_i, які є сумою вхідних ваг (W_{ij}), помножених на їх відповідні входи (I_{ij}), та зміщень (b_i)[20]:

$$Z_i = \sum(W_{ij} * I_{ij} + b_i) \tag{2.1}$$

Далі, якщо цільові класи не є лінійно сепарабельними, функції активації та виходи вузлів задаються через:

$$H_i = \frac{1}{1+E^{-Yi(Hi)}} \quad (2.2.)$$

де E - сумарна помилка,

Hi - приховані вузли.

2.1.2. Back-Propagation

Зворотне поширення - це процес обчислення градієнтів по відношенню до ваг (див. рівняння 2.4). В той час як пряме поширення використовується для обчислення вихідних даних, метою зворотного поширення є мінімізація функції втрат. Функція втрат являє собою різницю між прогнозованими значеннями та цільовими значеннями. Це робиться шляхом коригування вагових коефіцієнтів та зсувів у зворотному напрямку. Ваги коригуються шляхом попереднього обчислення загальної похибки (E) [21]:

$$E = \Sigma(0.5 * (T_i - H_i)^2) \quad (2.3)$$

де Ti - цільові значення.

Далі від заданих ваг віднімається швидкість навчання (Li), помножена на похідну сумарної похибки w.r.t ваг:

$$\frac{dE}{dW_i} = \frac{dE}{dH_{i+1}} * \frac{dH_{i+1}}{dH_1)} * \frac{dH_i}{dW_i} \quad (2.4)$$

де:

$$W_i = W_i - (L_i * \frac{dE}{dW^i})$$

Слід вибирати швидкість навчання, яка дає найкрутішу функцію втрат, або можна поступово зменшувати, якщо після навчання є епохи.

2.1.3. Градієнтний спуск

Для незваженої нейронної мережі заданий набір вхідних даних завжди призводить до однакового результату. Однак у зваженій нейронній мережі всі вузли мають різну вагу, кожен з яких призводить до унікального результату. Алгоритм градієнтного спуску виконує кілька раундів або епох прямого та зворотного поширення, щоразу змінюючи ваги для підвищення точності класифікації, як пояснювалося вище[22].

Пряме розповсюдження з великою кількістю епох займає багато часу, тому має бути баланс між точністю та кількістю епох, коли точність максимізується, а кількість епох мінімізується. Занадто велика кількість епох також може призвести до надмірної підгонки. Цьому можна запобігти, побудувавши графік втрат при навчанні та тестуванні. Надмірне пристосування - це коли втрати при навчанні менші за втрати при тестуванні, а недостатнє пристосування - коли втрати при навчанні більші за втрати при тестуванні. Втрати під час тестування та тренування повинні бути приблизно однаковими.

2.2. Нейронні мережі для обробки природної мови

2.2.1. Частота терміну - обернена частота документа (TF-IDF)

TF-IDF є продуктом TF та IDF. TF - це кількість термінів у всіх документах для кожного терміну. IDF - це зворотна кількість термінів у кожному документі. TF-IDF є показником того, наскільки рідкісним є даний термін. Чим вищий TF-IDF, що відповідає терміну, тим більш рідкісним є даний термін. Таким чином, найпоширеніші слова матимуть низький TF-IDF.

2.2.2. Вбудовування слів

Вбудовування слів - це процес перетворення термів у вектори чисел з плаваючою комою [23]. Це робиться для підвищення точності штучних нейронних мереж (ШНМ), оскільки ШНМ будуються для векторів неперервних числових вхідних даних і бачать подібність між неперервними векторами з більшою легкістю, ніж подібність між термами.

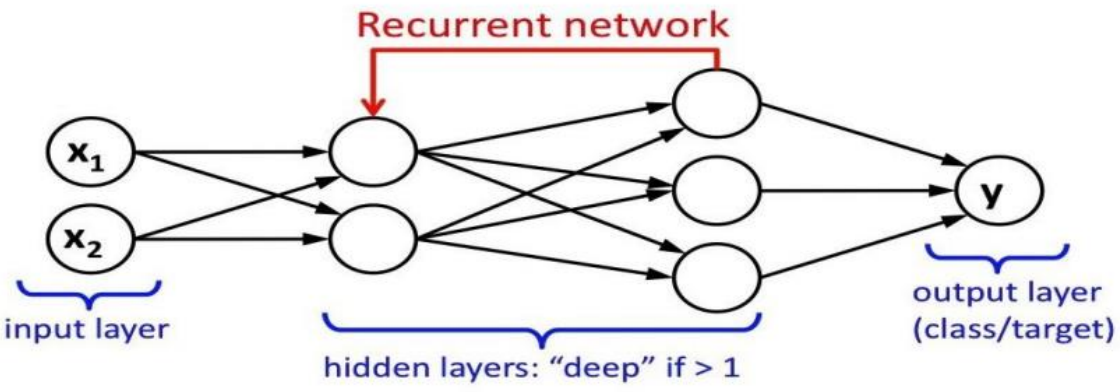

Рис.2.2. - Рекурентна нейронна мережа із циклами

Поширеною технікою вбудовування слів є GloVe (Global Vectors). GloVe - це алгоритм для перетворення немаркованих даних, таких як слова, у неперервні вектори, які зменшують розмірність[24]. Вектори GloVe пройшли попереднє навчання на Вікіпедії та Gigaword 5 і добре передають семантику.

2.2.3. Рекурентні нейронні мережі

Рекурентні нейронні мережі складаються з декількох копій нейронної мережі прямого поширення (FNN), і кожна копія вважається часовим кроком [25]. Рекурентні нейронні мережі використовують алгоритм градієнтного спуску для мінімізації помилки. Однак, оскільки існує багато часових кроків, які мають однакові параметри, замість звичайного розповсюдження використовується розповсюдження через час (BPTT). Між часовими кроками існують втрати, таким чином, виникає додатковий градієнт похибки порівняно зі звичайним зворотним поширенням.

Традиційні нейронні мережі не підходять для класифікації на основі послідовних даних, таких як мова, часові ряди або текст, оскільки вони не мають пам'яті, а послідовні дані є впорядкованими. Рекурентні нейронні мережі вводять пам'ять за допомогою циклів між прихованими шарами. Вихідні дані з циклів зберігаються у внутрішньому стані (див. рис. 2.2).

2.2.4. Gated Recurrent Unit (GRU)

GRU - це архітектура ШНМ, яка контролює потік інформації за допомогою воріт скидання та оновлення, як показано на рисунку 2.33.

Вентиль скидання вирішує, яку інформацію використовувати, а яку відкинути, і, таким чином, його можна вважати сумішшю LSTM вентилів забування та введення [25].

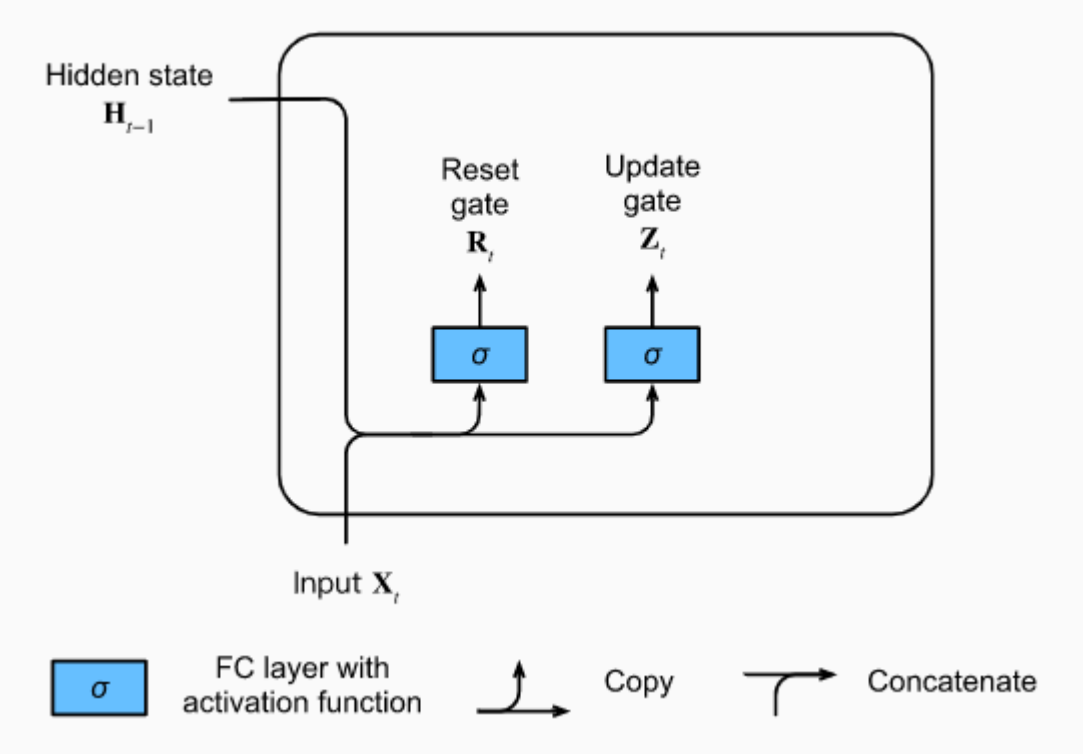

Рис.2.3. - CRU клітина

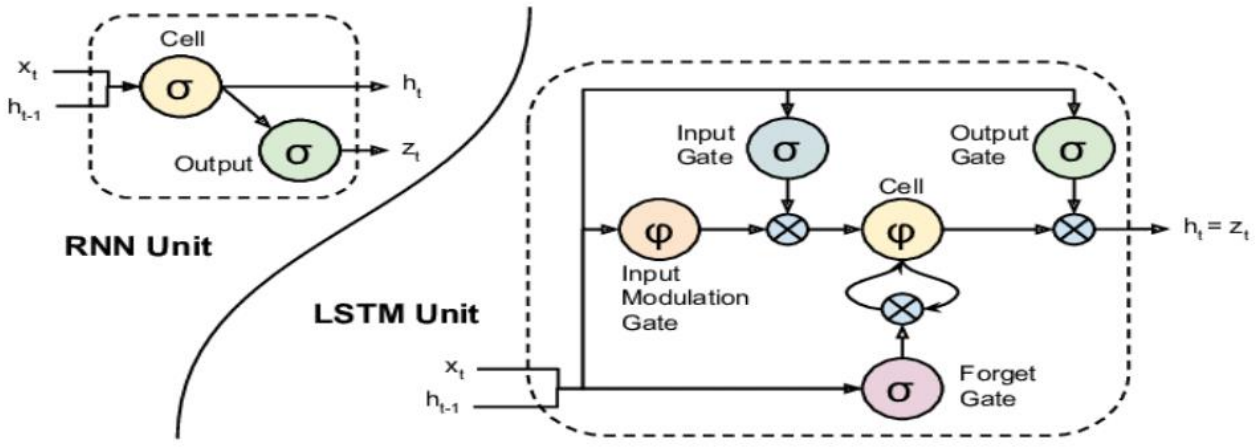

Рис.2.4. - RNN клітина і LSTM гейти

2.2.5. Довга короткочасна пам'ять (LSTM)

Довготривала короткочасна пам'ять (LSTM) - це тип рекурентної нейронної мережі, яка вирішує проблему зникаючого градієнта шляхом введення довготривалої пам'яті. Як обговорювалося в 2.2.3, рекурентні нейронні мережі мають декілька часових кроків і комірку, яка містить один шар мережі (Tan) для кожного часового кроку. LSTM-мережі, з іншого боку, мають комірку пам'яті, яка містить чотири шари мережі для кожного часового кроку (див. рис. 2.4). Три з цих шарів є сигмоїдними, а саме: вхідний, вихідний та затвор забування. Існує також один шар Тана.

Вхідний вентиль вирішує, яку інформацію запам'ятати, в той час як вихідний вентиль вирішує, яку інформацію вивести. Комірки пам'яті LSTM мають два стани, а саме, прихований і стан пам'яті. Прихований стан збігається з вихідним. Стан пам'яті, C, - це місце, де зберігається вхід пам'яті. Рівняння різних вентилів і станів пояснюються на малюнку нижче.

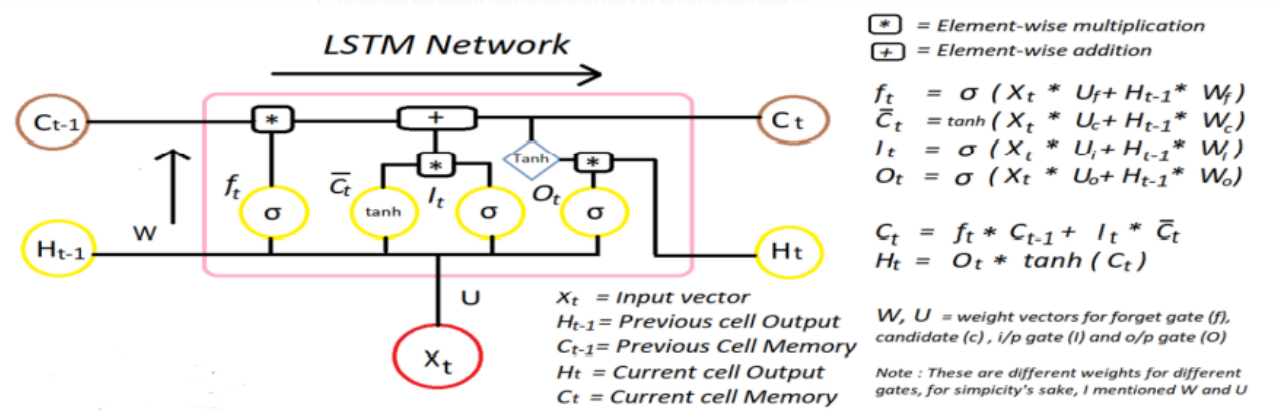

Рис.2.5. - LSTM клітинні рівняння

Ключем до довготривалої пам'яті є затвор забування. У традиційних RNN всі вхідні дані приймаються. У цьому випадку раніше прийнята релевантна інформація може бути замінена новою нерелевантною

інформацією. У LSTM, однак, затвор забування запобігає цій проблемі, фільтруючи нерелевантну інформацію зі стану пам'яті, забезпечуючи таким чином довготривалу пам'ять[26]. Це робиться шляхом множення попереднього стану пам'яті на затвор забуття, f. Якщо f дорівнює нулю, стан пам'яті повністю забувається, і тільки поточний вхід (C't*It):

$$C_t = C_{t-1} * f_t + C'_t * I_t \tag{2.5}$$

Осередки пам'яті також мають вхідний модуляційний вентиль (цей вентиль часто розглядається як підвентиль вхідного вентиля). Цей вентиль нормалізує вхідну інформацію для збільшення швидкості збіжності. ШНМ (включаючи LSTM) використовують функцію активації для обчислення виходів, Ht, де t - кроки часу. Загальною функцією активації є Tan(Ht) (див. рис. 2.5), і вона гарантує, що всі значення залишаються між від'ємною одиницею та одиницею:

$$H_t = Tan_h(C_t) \tag{2.6}$$

Запропонована модель, однак, використовує функцію активації ReLU в ранніх шарах:

$$H_t = \max(0, x) \tag{2.7}$$

ReLU використовується для уникнення проблеми зникаючого градієнта, як пояснюється в наступному розділі. В останніх двох шарах використовується функція активації Sigmoid. Сигмоїд більше підходить для бінарної класифікації, наприклад, для виявлення фейкових новин, ніж тангенс, тому що сигмоїд перетворює значення в діапазон від нуля до одиниці:

$$H_t = \frac{1}{1+e^{-x}} \quad (2.8)$$

Сигмоїдними виходами є ймовірності цільового класу. Недоліком LSTM є те, що він може спричинити надлишкові накладні витрати, оскільки деякі прогнози потребують менше контексту, ніж інші. Прикладом є "сонце - це зірка", при прогнозуванні останнього слова, "зірка", більше інформації не потрібно. Прикладом, коли потрібно більше контексту, а LSTM не є надлишковим, є "Я люблю дивитися мильні опери…Мій улюблений серіал - "Друзі". У цьому випадку перше висловлювання є корисним при прогнозуванні останнього слова другого висловлювання.

2.2.6. Зникаючий градієнт втрат (Vanishing Loss Gradient)

При навчанні ШНМ з декількома шарами може виникнути проблема, яка називається "зникаючий градієнт втрат". Як випливає з назви, проблема полягає в тому, що градієнт "зникає", тобто наближається до нуля.

Проблема виникає, якщо багато градієнтів близькі до нуля, оскільки градієнти є добутками попередніх градієнтів. Проблема зазвичай виникає в більш ранніх шарах (градієнти обчислюються за допомогою зворотного розповсюдження). Це проблема, тому що оновлення ваг є відніманням градієнтів, помножених на швидкість навчання, від самих ваг. Це означає, що якщо градієнти близькі до нуля, то оновлення ваг не матиме майже ніякого ефекту або взагалі не матиме ніякого ефекту. Зникаючий градієнт втрат не є проблемою для функції активації ReLU, тому запропонована модель застосовує ReLU в ранніх шарах і Sigmoid в пізніх шарах (див. розділ 2.1). ReLU перетворює значення вище нуля до початкового значення, а від'ємні числа до нуля, тоді як Sigmoid перетворює значення вище нуля до одиниці, а

від'ємні числа до нуля, і, таким чином, є більш обтяжливою з точки зору обчислень, ніж ReLU. Це також полегшує проблему вибухоподібного градієнта втрат, яка є протилежною проблемі зникаючого градієнта втрат, коли градієнт досягає астрономічно високих значень, оскільки градієнт ніколи не може бути вищим за одиницю.

2.2.7. Двонаправлена довга короткочасна пам'ять (BiLSTM)

Прямі LSTM мають тільки зворотну пам'ять і пам'ятають минуле, в той час як зворотні LSTM навчаються на зворотних входах, і, таким чином, мають пряму пам'ять і пам'ятають майбутнє. Двонаправлені LSTM поєднують в собі прямі і зворотні LSTM, що означає, що вони мають як пряму, так і зворотну пам'ять. Це може покращити продуктивність, надаючи більше контексту і скорочуючи час навчання. Розглянемо наступний сценарій: пам'ять прямого LSTM містить "Хлопчики пішли до...", тоді як пам'ять зворотного LSTM містить "...і потім вони зловили дві рибки". Передбачити наступне слово, "риба", легше при об'єднанні спогадів, ніж при використанні тільки прямої пам'яті LSTM.

2.3. Існуючі підходи

Запропонована в цій роботі модель використовує метадані, шляхи поширення чуток у вигляді часових рядів та вміст статей як вхідні дані, використовуючи двонаправлений LSTM, таким чином поєднуючи підходи до вирішення з [15] та [14].

У [14] спочатку будуються шляхи поширення для кожної статті новин шляхом визначення користувачів, які поширювали кожну зі статей, а потім будується часовий ряд на основі характеристик користувачів. Далі часовий ряд перетворюється на багатовимірні послідовності довжиною n. Якщо розмір часового ряду більший за n, він об'єднується, інакше він випадково передискретизується. Багатовимірні послідовності подаються в мережу Gated Recurrent Unit (GRU) і перетворюються у вектори за допомогою функцій сигмоїда та тангенса.

Нарешті, до виходів з GRU застосовується середнє об'єднання для зменшення послідовності вихідних векторів в один вектор.

Метою роботи [15] було виявлення фейкових новин якомога раніше, а ГРУ є обчислювально ефективними і простими, оскільки їм не потрібен блок пам'яті, як LSTM, саме тому в цій моделі було використано ГРУ, а не LSTM.

Характеристики користувача також перетворюються в багатовимірні вектори ознак за допомогою одновимірної ШНМ з функцією активації ReLU і зводяться до одного вектора за допомогою середньоквадратичного об'єднання.

Виходи з двох мереж об'єднуються і подаються в багатошарову нейронну мережу прямого поширення, яка прогнозує мітки класів для відповідних шляхів поширення з використанням ReLU і Softmax (див. рисунок, взятий з [15]). Softmax - це простий, але ефективний класифікатор для лінійно відокремлюваних проблем.

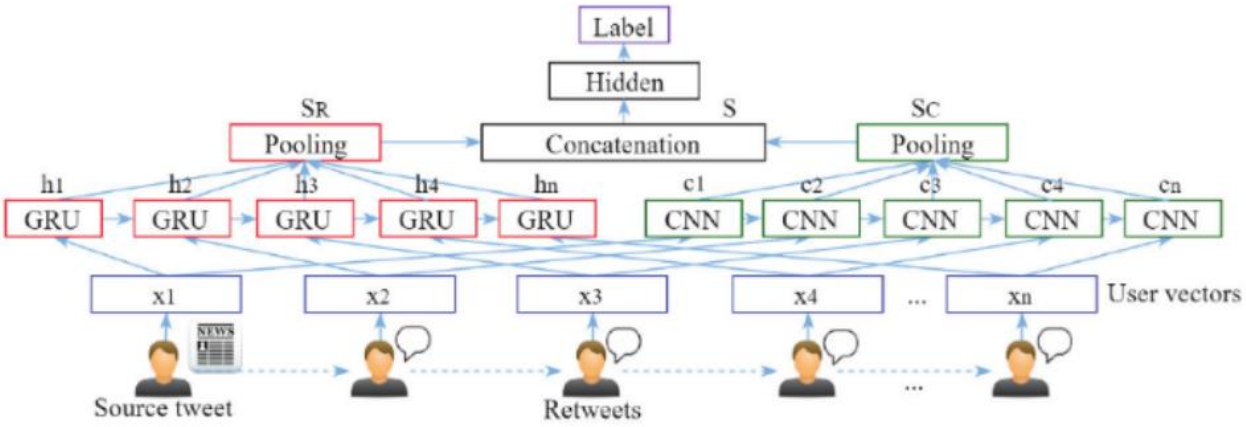

Рис.2.6. - Поєднання архітектури GRU і CNN

[14] спочатку попередньо обробляє текстовий ввід, видаляючи з нього стоп-слова та URL-адреси, видаляючи суфікси, виправляючи написання слів та токенізуючи ввід. Наступні ознаки витягуються за допомогою попередньо навченого вбудовування слів. Вбудовування слів ініціалізуються 100-мірними попередньо навченими вбудовуваннями від GloVe.

Після вилучення ознак двонаправлена модель LSTM навчається протягом десяти епох з розміром партії 128, розмірністю виходу сто, часовими кроками, встановленими на триста, оптимізатором ADAM зі швидкістю навчання 0,0001 і функцією активації Sigmoid в кінцевому щільному вихідному шарі.

2.4. Аналіз

Постановка проблеми в [15] полягала в тому, щоб виявити новини якомога раніше. Модель [15] виявила фейкові новини швидше, ніж найсучасніші моделі, і таким чином вирішила поставлену задачу. Модель виявляє фейкові новини швидко, тому що вона використовує лише часові ряди та характеристики користувачів як вхідні дані, а не складні ознаки, такі

як лінгвістичні особливості. Однак швидке навчання відбувається за рахунок низької точності. Точність можна було б підвищити, використовуючи LSTM замість GRU та додаючи лінгвістичні ознаки як вхідні дані, як це зроблено в запропонованій в цій дисертації моделі. Якщо є доступ до графічного процесора, час не має значення.

[14] дійшли висновку, що LSTM, зокрема C-LSTM (CNN-LSTM), показав багатообіцяючі результати для виявлення фейкових новин на основі NLP з найкращою точністю 95% і що це вимагає подальшої уваги. Як і [15], [14] демонструє хороші результати, але залишає простір для вдосконалення, не використовуючи більше вхідних (наприклад, метадані та шляхи поширення чуток).

РОЗДІЛ 3 РОЗРОБКА МОДЕЛІ

3.1. Запропонована модель

Запропонована модель використовує LSTM (від моделі Keras Sequential), оскільки LSTM ефективно запам'ятовує довгі послідовності та моделює далекі зв'язки, що є кращим при використанні послідовних текстових даних, таких як зміст статті, як вхідні дані. Використовуються функції активації ReLU та Sigmoid (див. 2.2.4). Двонаправлене LSTM, яке є розширенням традиційного LSTM, використовується тому, що він покращує точність за рахунок забезпечення як прямої, так і зворотної пам'яті, як пояснюється в 2.2.6.

Зміст статті, безперервні атрибути та поширення чуток були успішно використовувалися в попередніх роботах, і ця дисертація поєднує ці особливості (див. рис. 3.1). Контент та метадані для твітів, ретвітів та статей спочатку зчитуються з набору даних Politifact у фрейм даних. Потім вміст статті вбудовується в слова, часові ряди для кожного шляху поширення чуток обчислюються за допомогою метаданих, а безперервні атрибути з метаданих нормалізуються та дискретизуються, як пояснюється в наступному розділі.

Потім ці ознаки об'єднуються і використовуються як вхідні дані для двонаправленого LSTM, який використовує вхідні дані для класифікації статей як фейкових або реальних.

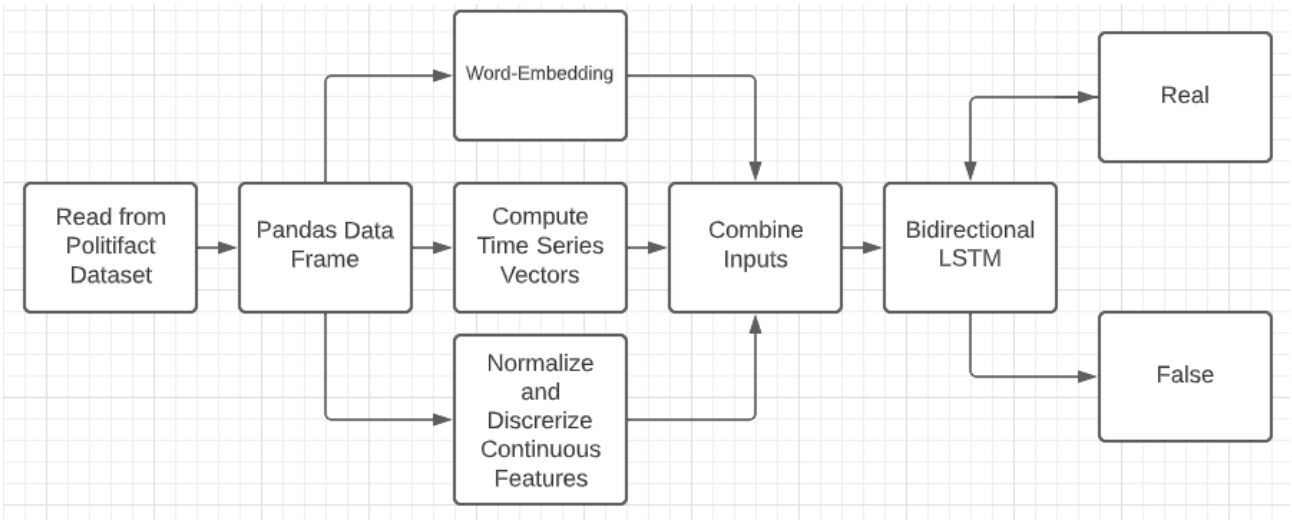

Рис.3.1. - Запропонована модель

3.2. Попередня обробка

У цій роботі дані проходять декілька етапів попередньої обробки. По-перше, вилучаються стовпці та рядки, що містять переважно нечислові значення (NAN). Потім оцінюються значення NAN, що залишилися. Значення NAN для неперервних ознак оцінюються до середнього значення ознаки, тоді як значення NAN для категоріальних ознак оцінюються до найпоширенішого значення для кожної ознаки. Потім видаляються символи, розділові знаки та стоп-слова. Нарешті, вміст статті вкладається в слова, а безперервні ознаки нормалізуються, як описано нижче. Часові ряди також обчислюються на основі метаданих статей, як пояснюється в розділі 3.2.3.

Дані поділяються на навчальну вибірку, що складається з 80% даних, та перевірочну вибірку, що складається з решти 20%. набір, що складається з решти 20%.

3.2.1. Обробка природної мови (NLP)

Вміст статті є вбудованим словом (див. розділ 2.2.2). Зміст кожної статті відповідає вектору довжиною 100000 (включаючи пробіли). Ваги для кожного з векторів генеруються з використанням 6B 100d векторів GloVe. Вектори подаються в шар вбудовування (див. рис. 8) і виводяться у вигляді матриці. Шари вбудовування добре відображають настрої.

3.2.2. Неперервні атрибути

Неперервні атрибути нормалізуються за критерієм min-max та дискретизуються на три біни за допомогою функції cut з бібліотеки Pandas. Міні-максна нормалізація масштабує всі атрибути до одного діапазону (від нуля до одиниці в цій роботі), використовуючи формулу, наведену нижче. Формула є чутливою до викидів.

$$F_i = [X_i - min(X)]/[\max(X) - \min(X)] \qquad (3.1)$$

Нормалізація даних гарантує, що всі атрибути мають однаковий вплив. Для нейронних мереж це зменшує час збіжності градієнтного спуску, а отже, скорочує час навчання. час навчання. Дискретизація зменшує шум даних. Кожен бін згладжує шум на ділянках даних.

Існує 13 безперервних функцій:

- підрахунок улюблених твітів
- кількість вподобаних користувачів
- кількість підписників на користувача
- кількість друзів на користувача

- кількість перерахованих на одного користувача

- кількість ретвітів на твіт

- кількість статусів на користувача

- зміщення початку URL-адрес у текстах твітів

- зміщення, де закінчуються URL-адреси в тексті твіту

- зміщення початку URL-адрес у текстах твітів для кожного користувача

- зсуви, де закінчуються URL-адреси в текстах твітів для кожного користувача

- зсуви, з яких починаються згадки користувача в текстах твітів на користувача

- зміщення, де закінчуються згадки користувача в текстах твітів на користувача

3.2.3. Поширення чуток

Для кожної статті розраховуються вектори часових рядів для всіх відповідних каскадів чуток. Каскад чуток - це ланцюжок ретвітів на даний твіт статті. Елементи часового ряду містять час, що минув з моменту першого твіту для кожного елемента в даному каскаді чуток, і обчислюються шляхом попереднього обчислення різниці в часі між даним ретвітом і попереднім ретвітом або твітом для всіх ретвітів. Ці різниці додаються до вектора D. Елементи часового ряду потім обчислюються шляхом знаходження кумулятивних сум для кожного елемента в D. Експерименти довели, що фейкові статті поширюються швидше, ніж справжні статті, а це означає, що часовий ряд, заснований на поширенні чуток часових рядів є життєздатними як вхідні дані для виявлення фейкових новин.

3.3. Шари

Після об'єднання змісту статті, часових рядів та векторів TF-IDF, ці об'єднані ознаки передаються як вхідні дані для вхідного шару. За вхідним шаром слідує шар вбудовування (див. 3.1.1). Наступним шаром є двонаправлений LSTM-шар з 60 нейронами. Не існує фіксованих правил для встановлення кількості нейронів у прихованих шарах в комірці LSTM, але для того, щоб запобігти перенавчанню, їх кількість повинна бути меншою, ніж:

$$Nh = Ns/(\alpha * (N_i + N_0)) \quad (3.2)$$

де N_i - кількість вхідних нейронів,

N_o - кількість вихідних нейронів,

N_s - кількість вибірок у навчальному наборі даних,

α - довільний коефіцієнт масштабування, зазвичай від двох до десяти.

Після шару LSTM йде шар "GlobalMaxPool1D", який зменшує вибірки, обчислюючи найбільш присутні ознаки (максимальні значення) для кожної карти ознак. Після шару Max-Pooling йде шар відсіву з коефіцієнтом відсіву 0,1, що означає, що десять відсотків нейронів або входів відсіваються. Відсіювання - це метод регуляризації, який запобігає надмірному пристосуванню шляхом відсіювання випадкового набору прихованих одиниць або вузлів при кожному оновленні під час навчання. Вузли відкидаються шляхом встановлення їх в нуль. Нарешті, є два щільні шари з ще одним шаром відсіву між ними для подальшої регуляризації, яка класифікує статті як фейкові або реальні. Перший з цих щільних шарів має 158 нейронів і застосовує функцію активації ReLU. Останній щільний шар має два нейрони і застосовує функцію активації Sigmoid. Функція активації

ReLU застосовується перед Sigmoid, щоб запобігти зникаючому градієнту втрат (див. розділ 2.2.4).

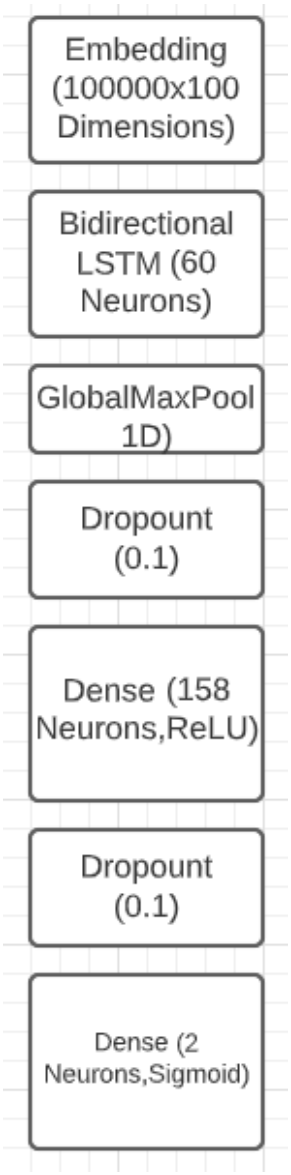

Рис.3.2. LSTM Graph

3.4. Технології

Код, який використовувався для отримання експериментальних результатів у цій дисертації, був написаний на мові Python, оскільки вона має прості синтаксичні правила, що робить код легким для читання, і має багато

бібліотек, корисних для науки про дані в цілому, таких як Pandas, Pickle, Numpy та Pyplot. Pandas добре підходить для представлення даних і може ефективно обробляти великі обсяги даних. Pickle можна використовувати для збереження структур даних в якості резервних копій при виконанні коду, що вимагає багато часу (TQDM може запропонувати індикатори прогресу для таких випадків). Numpy можна використовувати для масштабованих та ефективних обчислень. Pyplot пропонує великий вибір графіків для візуалізації. Python також має хороші бібліотеки спеціально для глибокого навчання, такі як Keras.

3.4.1. Natural Language ToolKit (NLTK)

NLTK - це пакет для обробки природної мови [27]. Пакет можна використовувати, наприклад, для видалення стоп-слів. Стоп-слова – це загальновживані слова, такі як "the", "a" або "and". Вони не надають жодного контексту або значення реченням, і тому повинні бути вилучати.

3.4.2. Wordcloud

Пакет Wordcloud використовується для візуалізації тексту, де розмір тексту відповідає значущості або частоті слів [28]. Для візуалізації хмар слів потрібен Matplotlib.

3.4.3. Keras

Keras має багато переваг. Бібліотека ефективна як на CPU, так і на GPU[Ker(2019)], вона має функції попередньої обробки, такі як токенізація та заповнення, а також має багато моделей, таких як максимальне об'єднання, відсікання, вбудовування та двонаправлений LSTM. Ці моделі можна комбінувати, як це зроблено в даній роботі. Keras навіть має регуляризатори, які можна використовувати, щоб уникнути надмірної підгонки.

- Послідовна модель Keras

Запропонована модель використовує модель Keras Sequential. Послідовна модель Keras модель Keras Sequential проста у використанні і дозволяє будувати моделі шар за шаром, але вона не підтримує множинні входи.

- Функціональний API Keras

Комбінація CNN та LSTM, реалізована для цієї дисертації, базується на моделі RNN+CNN з [15], що використовує статті на додаток до часових рядів та метаданих в якості вхідних даних та LSTM замість GRU. Модель було реалізовано за допомогою Keras Functional API. Функціональний API є гнучким і підтримує декілька вхідних даних. Це означає, що окремі виходи з класифікаторів ДЛ, таких як CNN та LSTM, можуть бути об'єднані.

3.4.4. Регулярні вирази (RE)

RE можна використовувати для перевірки відповідності заданого рядка регулярному виразу, а також для видалення цифр з рядків[29].

3.4.5. Datetime

Пакет Datetime містить класи для маніпулювання датами та часом, наприклад, для віднімання дат з метою обчислення часу, що пройшов[30].

3.4.6. Gridsearch

Gridsearch - це інструмент SKLearn для налаштування гіперпараметрів класифікатора[31].

Гіперпараметри - це аргументи конструктора, які не вивчаються безпосередньо в рамках даного класифікатора. У цій роботі Gridsearch використовувався для оцінки кількості епох та розміру партії для LSTM та CNN. Gridsearch здійснює вичерпний пошук по сітці значень гіперпараметрів для класифікатора і зберігає комбінацію значень з найкращою оцінкою перехресної перевірки. Параметри Gridsearch, протестовані для запропонованої моделі та інших моделей ДН є наступними:

- Епохи 1, 2, 3, 4, 5
- Розмір партії 8, 16, 32, 64, 128

3.5. Набір даних

Всі дані, включаючи статті, твіти та ретвіти, використані для цієї роботи, отримані з набору даних Politifact. Politifact був вперше створений газетою Tampa Bay Times у Санкт-Петербурзі, штат Флорида, у 2007

році[32]. Ним керує Інститут Пойнтера. Politifact - це сайт перевірки фактів, який оцінює достовірність заяв виборних посадових осіб та інших осіб на своєму Truth-O-Meter. Truth-O-Meter - це шкала від "Правда" (100% правди) до "Штани горять" (100% фейк). Мітки в цій роботі, однак, мають бінарний розподіл, або "правда", або "фейк". Набір даних налічує 1056 статей, з яких 432 позначені як підроблені, а 624 - як справжні. Для навчання було використано 19981 запис (11528 фейкових та 8453 реальних) та 15 атрибутів (13 безперервних, часових рядів та змісту статей). 80% набору даних було використано для навчання, а решта - для тестування.

3.5.1. Метод надлишкової вибірки синтетичної меншості (SMOTE)

Дані незбалансовані з більшістю реальних статей. Класифікатори, як правило, мають гірші показники для класу меншин, отже, реальні записи перевибірковуються для вирівнювання розподілу. Це робиться за допомогою SMOTE від Keras. SMOTE генерує нові записи на основі існуючих записів. Запис генерується за допомогою опуклої комбінації даного запису та одного з k найближчих сусідів.

3.5.2. Недостатня вибірка

Недодискретизація запобігає викривленим розподілам, так само, як і методи надмірної вибірки, такі як SMOTE, але робить це за рахунок недодискретизації класу більшості. У цій роботі використовується алгоритм RandomUnderSampler з Python Imblearn API, який видаляє вибірки випадковим чином.

3.6. Експериментальні результати

У цьому підрозділі обговорюються та аналізуються візуалізації даних, згенеровані на основі статей та статистичних даних з набору даних Politifact.

3.6.1. Кумулятивна функція розподілу (CDF)

Кумулятивна функція розподілу (CDF) використовується для знаходження ймовірності того, що випадкова величина буде меншою або дорівнюватиме заданому значенню[33]. Ймовірність, y, обчислюється шляхом ділення кількості подій, які менші або дорівнюють y, на кількість можливих результатів (всі значення менші або дорівнюють y).

3.6.2. Аналіз користувачів

Рисунок 3.3 показує, що користувачі, які твітують фейкові статті, мають менше підписників, ніж користувачі, які твітують справжні статті (користувачі, які твітують справжні статті, мають більшу ймовірність мати багато підписників, ніж користувачі, які твітують фейкові статті).

Рисунок 3.4 показує, що, швидше за все, користувачі, які твітують справжні статті, твітують більше, ніж користувачі які твітують фейкові статті. Поясненням може бути те, що користувачі, які твітують фейкові статті часто створюють нові акаунти, щоб уникнути того, що люди дізнаються, що вони твітують фейкові твіти.

Результати, розглянуті в цьому підрозділі, означають, що метадані, пов'язані з користувачами, є життєздатними як вхідні дані для класифікації.

3.6.3. TFIDF

Рисунок 3.5 показує, що терміни з реальних статей мають вищий TF-IDF, ніж терміни з фейкових статей. Це означає, що фейкові статті мають менший словниковий запас, ніж справжні статті, і що вони мають тенденцію до повторного використання слів. Наприклад, у наборі даних Politifact багато фейкових статей присвячено президентським виборам, що можна побачити на рисунку 3.6 (Трамп і Обама є одними з найпоширеніших слів у фейкових статтях).

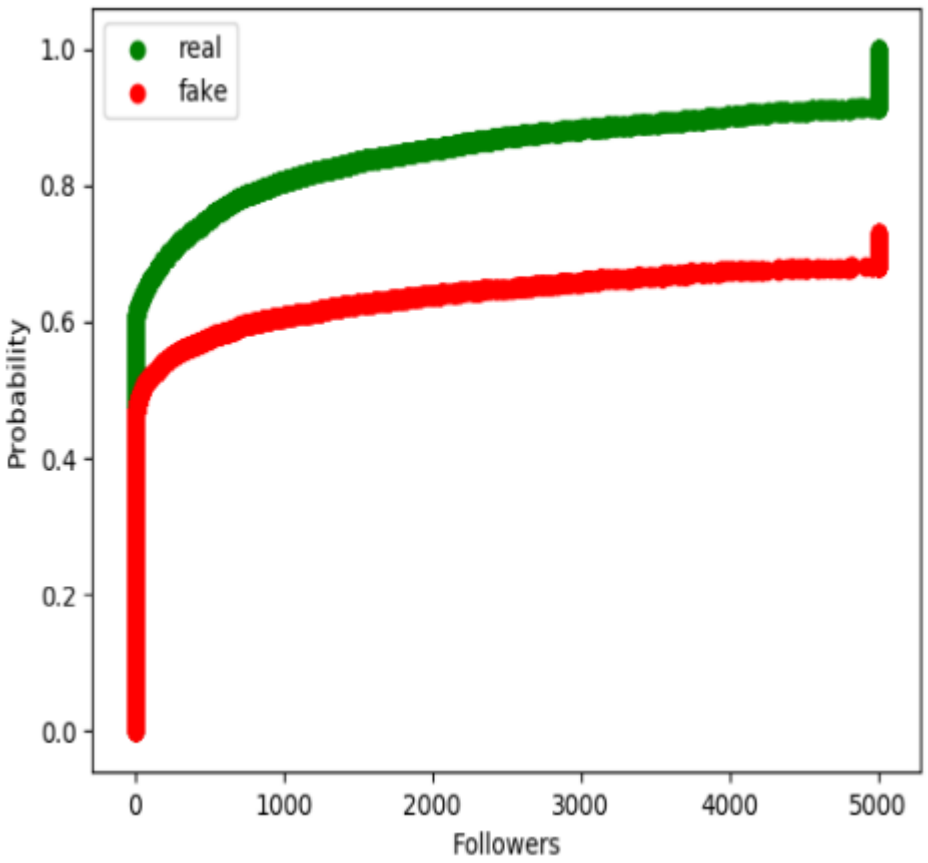

Рис.3.3. Фоловери CDF (Cumulative Distribution Function)

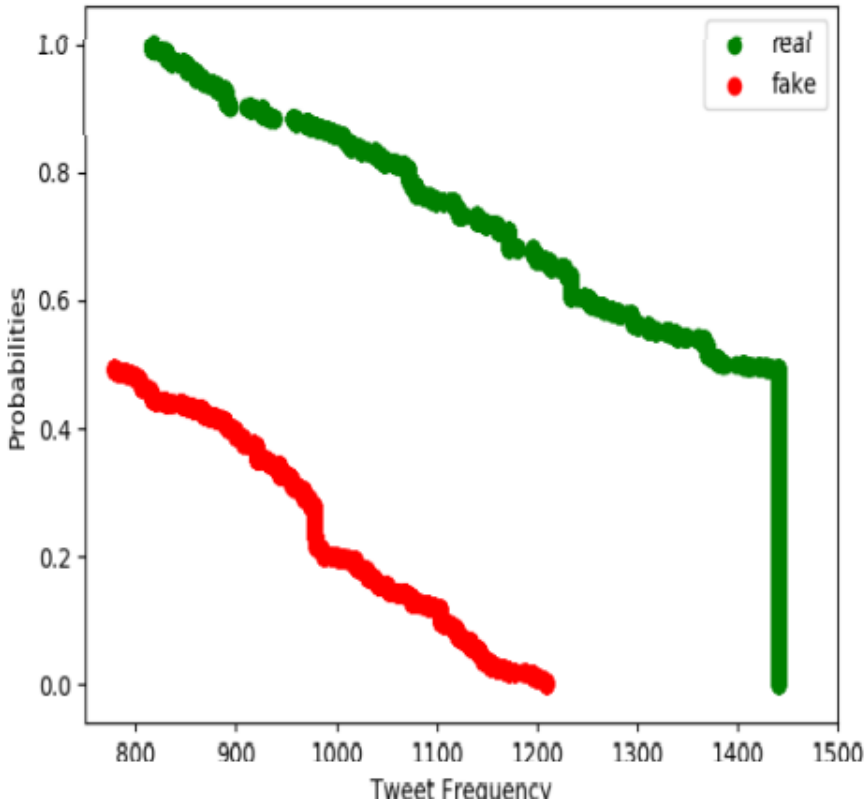

Рис.3.4. Частота твітів CDF

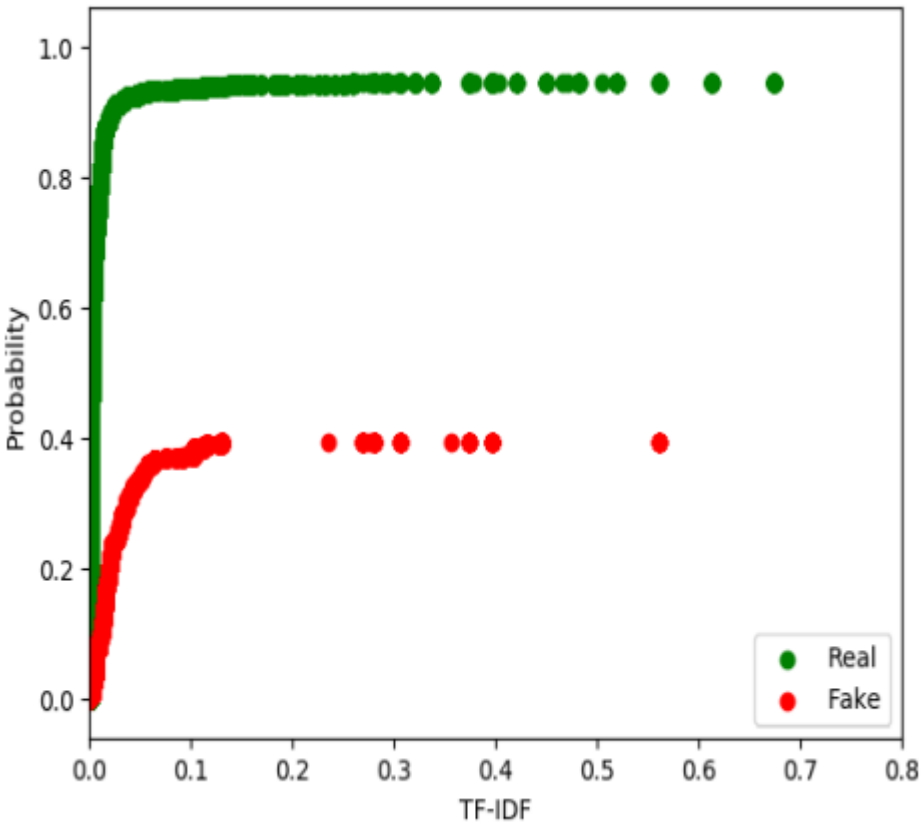

Рис.3.5. TF-IDF CDF

Рисунок 3.6 також демонструє, що фейкові статті, як правило, містять вищі ступені порівняння (наприклад, найвищий) і слова з навантаженням (наприклад, відсторонений і вбитий), а їхній зміст, як правило, є більш новим або драматичним, ніж у справжніх статтях. У деяких фейкових статтях темами є плітки про знаменитостей (наприклад, Dogg, як у Snoop Dogg) або серйозні кримінальні справи (наприклад, killed). На Рисунку 3.7 показано найпоширеніші терміни у справжніх статтях для порівняння, і з нього видно, що мова та зміст, як правило, є буденними та загальними порівняно з фейковими статтями.

3.6.4. Поширення чуток

Як видно з рисунків 3.8. та 3.9, твіти про фейкові новини охоплюють значно більше підписників, а також швидше ретвітуються. Відмінності є суттєвими, особливо для часового каскаду (шлях поширення чуток як функція часу). Фейкові новини також починають поширюватися набагато раніше, ніж справжні.

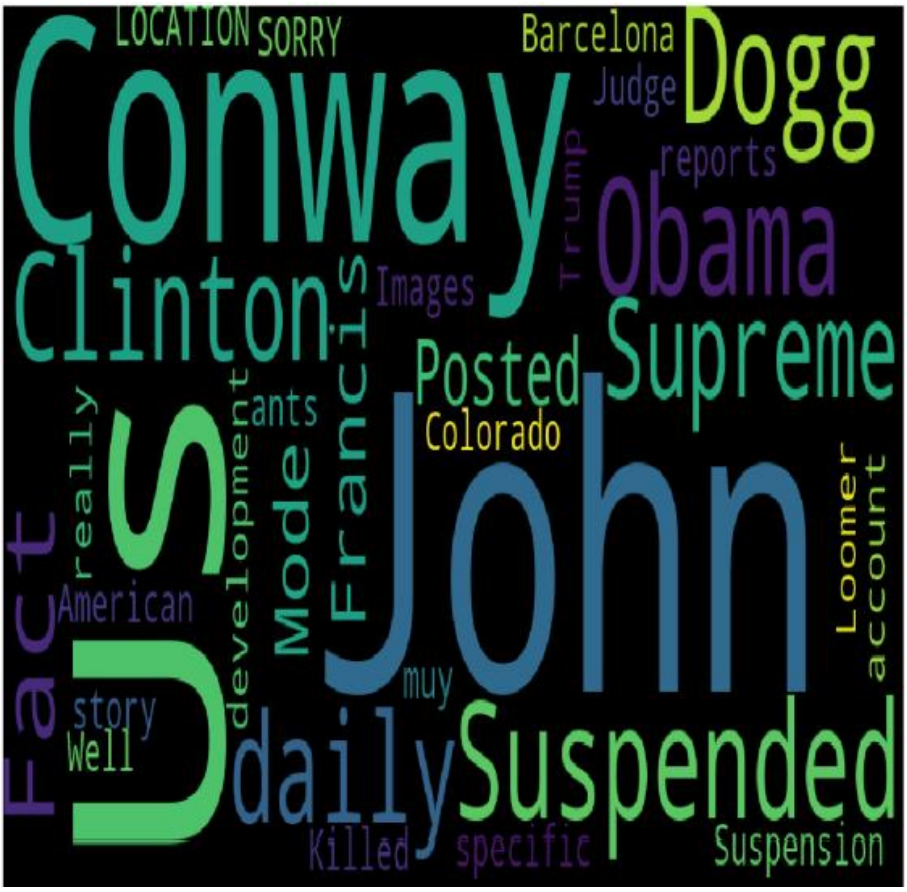

Рис. 3.6. - Фейкові TFIDF Common Wordcloud

Рис.3.7. - Справжні TFIDF Common Wordcloud

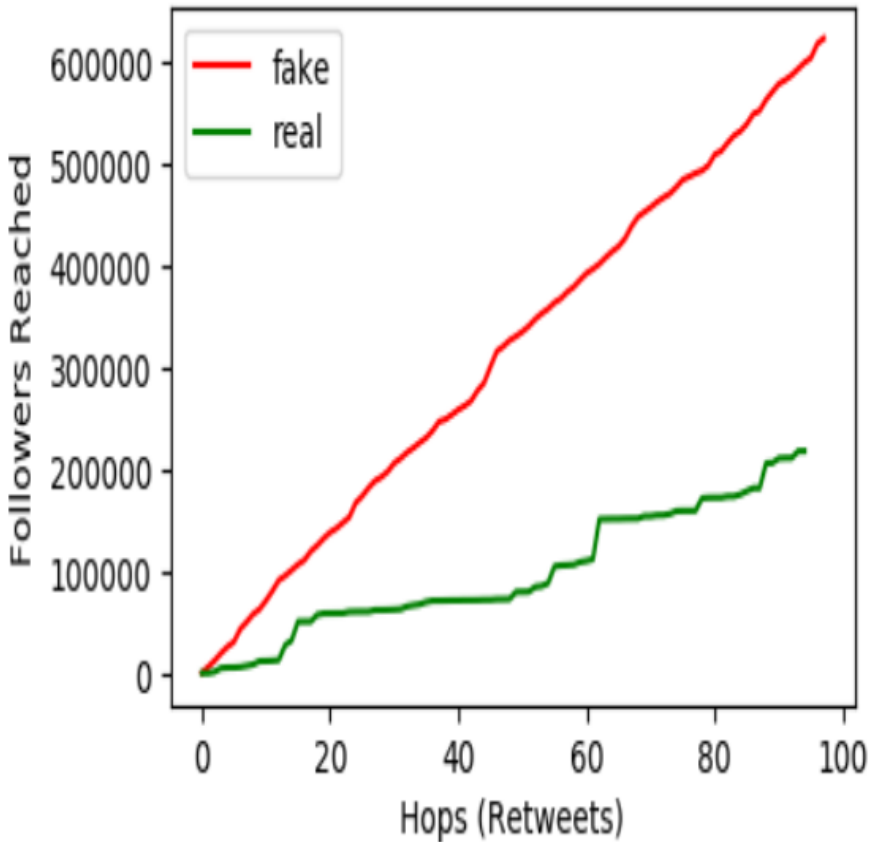

Рис.3.8. - Реакції фоловерів для кожного показника

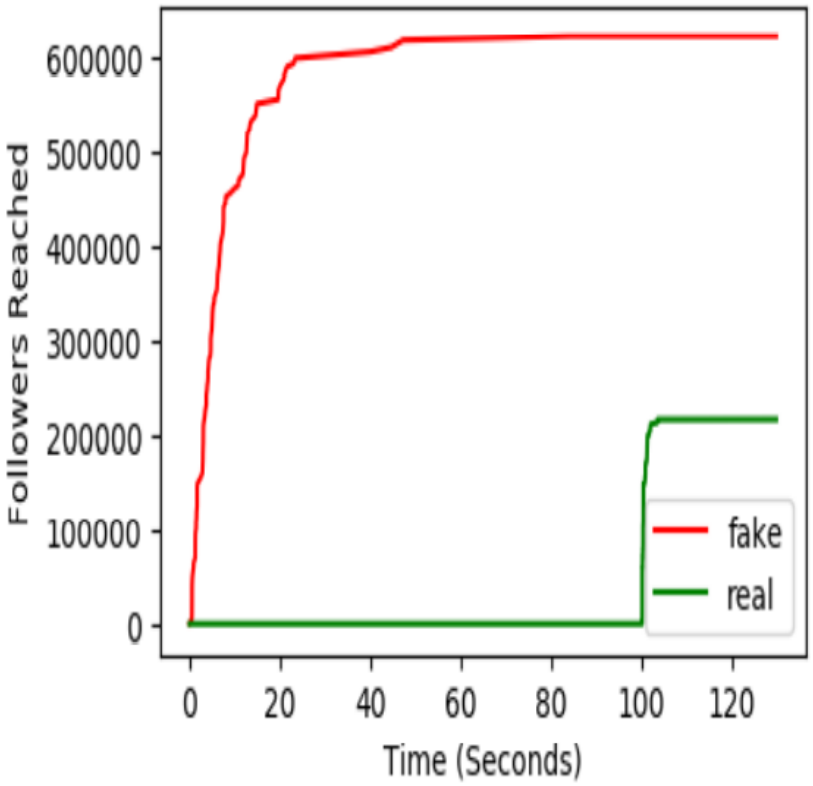

Рис.3.9. - Часовий каскад

Рисунок 3.9 також демонструє, що хоча фейкові новини поширюються швидше, ніж справжні, справжні новини насправді поширюються швидше,

коли вони тільки починають поширюватися. Це може бути пов'язано з тим, що підписники авторів, які часто твітують справжні твіти, як правило, набагато лояльніші, ніж підписники авторів фейкових твітів, і що вони отримують сповіщення, коли з'являються перші твіти в ланцюжку поширення чуток, а отже, всі вони ретвітнуть ці твіти в ланцюжку поширення чуток в один і той же часовий проміжок.

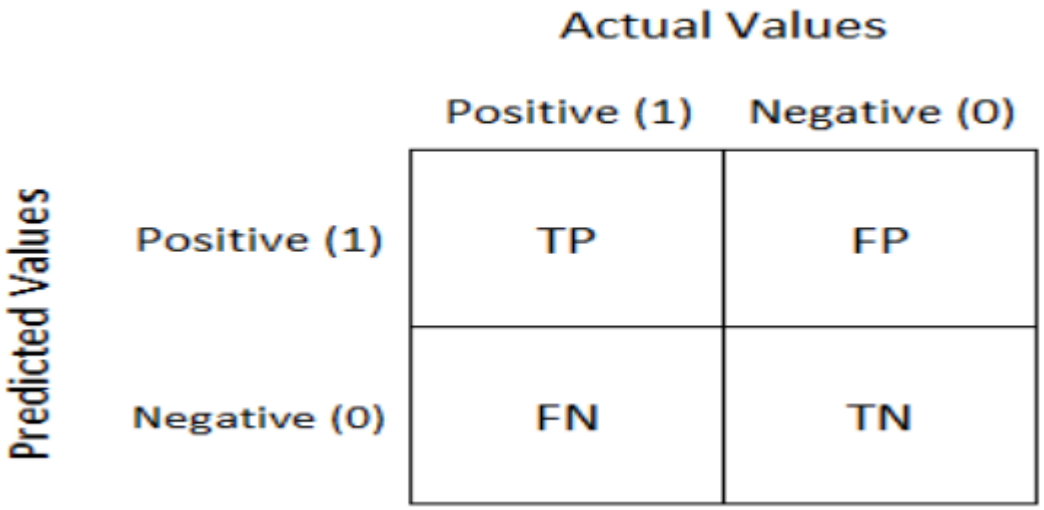

Рис.3.10. - Матриця плутанини

3.7. Статистичні вимірювання

Матриця плутанини є мірою точності для маркованих наборів даних [34]. Матриці плутанини можуть бути обчислені для декількох класів, але для проблеми з двома класами вона матиме чотири комірки, що містять істинно позитивні (ІП), хибно позитивні (ХП), хибно негативні (ХН) та істинно негативні (ІГ) (FN) та хибнонегативні (TN), див. рис. 3.10.

TP - це кількість правильно передбачених позитивних (реальних) записів (твітів). FP – це кількість помилково спрогнозованих позитивних записів. НН - кількість неправильно спрогнозованих негативних (фейкових)

записів. негативних (фейкових) записів. НН - кількість правильно передбачених негативних записів.

Точність - це кількість правильно передбачених позитивних записів, поділена на всі записи, що були передбачені як позитивні:

$$Precision = \frac{TP}{TP+FP} \tag{3.3}$$

Відгук - це кількість правильно спрогнозованих позитивних записів, поділена на всі записи які насправді виявилися позитивними:

$$Recall = \frac{TP}{TP+FN} \tag{3.4}$$

Мікроточність обчислюється так:

$$Micro = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.5}$$

і є відсотком правильно передбачених записів для обох класів.

Макроточність [35]- це середнє значення відкликань:

$$Macro = \frac{\frac{TP}{TP+FN}+\frac{TN}{TN+FN}}{2} \tag{3.6}$$

F1-Score є корисним показником точності, якщо вам потрібен баланс між точністю та пригадуванням, а розподіл класів нерівномірний:

$$F1 = \frac{2*(Precision*Recall)}{Precision+Recall} \tag{3.7}$$

Точність площі під кривою (AUC) вимірює, наскільки добре модель розрізняє два класи. "Крива" в AUC - це ROC-крива, яка є кривою

ймовірності, побудованою з TP на осі y та FP на осі x. Це хороша міра для задач бінарної класифікації з асиметричним розподілом. Формула для AUC така ж, як і для мікроточності, але AUC використовує ймовірності прогнозування як вхідні дані замість прогнозованих міток. Формула для AUC має вигляд

$$AUC = \int_0^1 TPR(FPR^{-1})(x))dx \tag{3.8}$$

Де

$$TPR(T) = \int_T^{\infty} f_1(x)dx$$

$$FPR(T) = \int_T^{\infty} f_0(x)dx$$

f_0 - функція щільності ймовірності для негативного класу,

f1 - функція щільності ймовірності для позитивного класу.

T-критерій Стьюдента визначає, чи відрізняються середні двох сукупностей суттєво, чи ні, за допомогою перевірки гіпотез. Нульова гіпотеза полягає в тому, що середні однакові. Якщо t-значення вище порогового значення, яке залежить від параметра альфа, нульова гіпотеза відхиляється. Існує декілька типів t-тестів, наприклад, одновибіркові, двовибіркові (незалежні) та парні. У даній роботі використовується двовибірковий t-тест, оскільки групи походять з різних популяцій (розподіли ймовірностей прогнозу для різних класифікаторів). Значення t-критерію для парних t-критеріїв визначається за наступною формулою:

$$T = \frac{\mu 1 - \mu 2}{\frac{s(d)}{\sqrt{n}}} \tag{3.9}$$

де μ1 та μ2 - середні значення двох сукупностей,

s(d) - стандартне відхилення різниці між значеннями двох генеральних сукупностей,

n - обсяг вибірки.

3.8. Експериментальна установка

Запропоновані моделі беруть на вхід часові ряди, вміст статей та різні безперервні метадані і класифікує новинні статті як справжні або фейкові за допомогою двонаправленого LSTM. Результати було порівняно з моделями на основі CNN, комбінації CNN та LSTM, SVM, NB, моделлю RNN+CNN з [Liu and Wu(2018)], моделлю Bi-LSTM з [14], та C-LSTM (CNN-LSTM) модель з [14].

3.8.1. Метод опорних векторів (SVM)

SVM класифікує дані, малюючи роздільник у вигляді гіперплощини. Якщо дані дані не є лінійно відокремлюваними, точки даних відображаються у вищий вимір, що робить проблему лінійно відокремлюваною. SVM є ефективним, коли є багато ознак, але його найкраще використовувати на невеликих наборах даних через час навчання. Модель SVM була навчена з використанням 10000 ітерацій та згенерована за допомогою SciKit-Learn (SK-Learn) з параметром 'c' (класифікаційне поле), встановленим на одиницю, лінійним ядром, параметром "degree", встановленим на три, та параметром "gamma", встановленим на у значення auto. Параметр 'degree' є показником

степеня поліноміальної функції ядра і ігнорується, якщо поліноміальне ядро не використовується. Гамма - коефіцієнт ядра для нелінійних ядер. Якщо встановлено значення auto, коефіцієнт ядра буде дорівнювати одиниці, поділеній на n, де n - кількість ознак. У розробленому для даної дисертації SVM класифікаторі було використано лінійне ядро. Найбільш поширені ядра для SVM перераховані та описані нижче.

- Лінійне

Лінійне ядро - це просте ядро, яке використовується, коли дані є лінійно відокремлювані, тобто дані можуть бути відокремлені одним рішенням границею. Ядро здебільшого використовується, коли набір даних містить багато багато ознак.

- Поліном

Поліноміальне ядро використовується для нелінійних даних, оскільки воно враховує схожість записів (а також схожість ознак).

- Радіально-базисна функція (RBF)

Ядро RBF є нелінійним ядром. RBF ядра використовуються для криволінійних границь рішення. Вони, як правило, більш точні і повільніше навчаються, ніж лінійні та поліноміальні ядра.

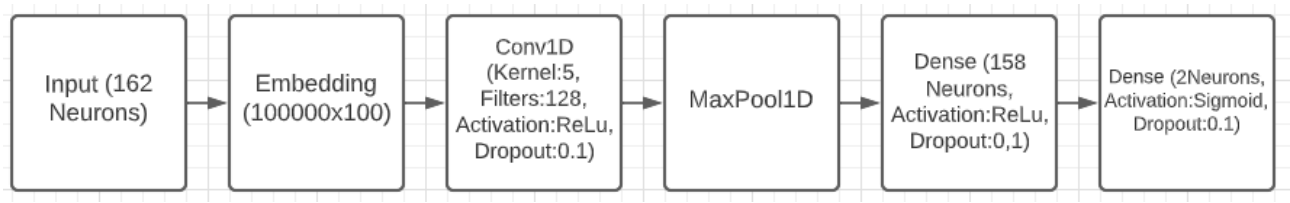

Рис.3.11. - CNN Graph

3.8.2. Наївний Байєс (NB)

Наївні класифікатори Байєса - це прості та швидкі імовірнісні класифікатори, які класифікують мітки на основі ймовірностей з використанням частот [36]. Для NLP, не наївні байєсівські класифікатори обчислюють нульову ймовірність для речень, яких немає в навчальній вибірці. Наївний Байєс класифікатори, однак, розглядають терміни незалежно один від одного і обчислюють ймовірності для кожного терма. NB найкраще застосовувати на векторах TF-IDF, оскільки TF-IDF відображає важливість термінів для кожного документа. Згладжування також важливе при використанні NB, щоб уникнути нульових значень. Мультиноміальний наївний Байєс використовується для порівняння із запропонованою моделлю тому що вставки слів є дискретними величинами, а це означає, що розподіл є мультиноміальний.

3.8.3. Згорткові нейронні мережі (CNN)

CNN - це тип нейронної мережі, яка згортає фільтр ядра для оцінки точок даних. Для кожної згортки фільтр використовується для оцінки значення центральної точки. CNN як правило, найкраще використовується для оцінки пікселів для комп'ютерного зору (зокрема, для класифікації зображень). зокрема, класифікації зображень). Однак він також може бути використаний для NLP. Моделі CNN швидко навчаються і можуть добре працювати, навіть якщо їм не вистачає пам'яті, на відміну від моделей LSTM. CNN можуть бути використовуватися для НЛП шляхом ковзання ковзного вікна або фільтра ядра по ділянках вбудовування матриці вбудовування або матриці однорідних векторів.

У моделі на основі CNN, що використовується для для порівняння в цій роботі, ядерний фільтр ковзає по матриці вбудовування, що складається з

векторів GloVe, яка має розмірність 100000x100. В цілому, LSTM перевершує CNN в задачах NLP, але CNN можуть працювати краще, ніж LSTM на завданнях виявлення ознак у коротких текстах. LSTM найкраще працює на завданнях, які включають довгі послідовності та вимагають певної пам'яті, наприклад, переклади.

3.8.4. Двонаправлена довга короткочасна пам'ять (BiLSTM)

Див. розділ 2 для отримання інформації про BiLSTM та розділ 3 для отримання інформації про пропоновану модель, засновану на BiLSTM.

Модель зображена на рисунку 3.1, а шари що використовуються в запропонованій моделі, перераховані на рисунку 3.2.

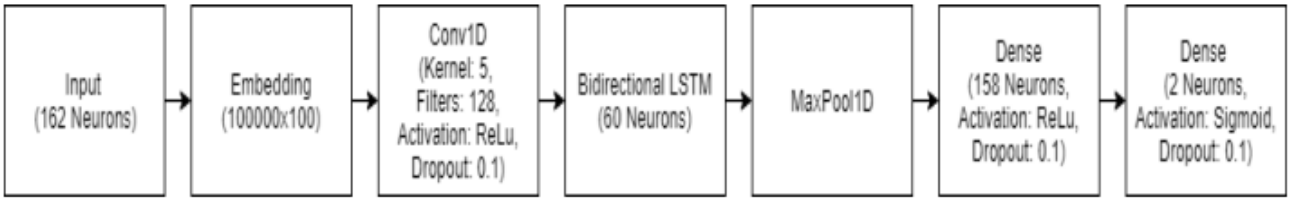

Рис.3.12. C-LSTM Graph

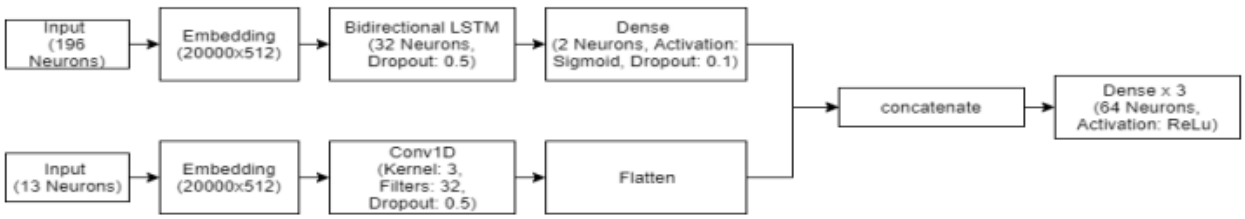

Рис.3.13. CNN_LSTM Graph

3.8.5. C-LSTM

Модель C-LSTM на основі моделі C-LSTM з [14], яка була реалізована для порівняння, поєднує CNN та LSTM шляхом їх стекування (починаючи з шару CNN). Модель використовує ті ж самі вхідні дані, що і запропонована модель. Шари моделі наведені на рисунку 3.12.

3.8.6. CNN-LSTM

Модель CNN-LSTM, заснована на моделі CNN-LSTM з [15], яка була реалізована для порівняння, поєднує CNN і LSTM, так само як і модель C-LSTM, але не шляхом їх складання, як це робить модель C-LSTM. Ця модель вводить вставки слів та часові ряди до шару LSTM, а безперервні ознаки до шару CNN. Шари моделі показані на рисунку 3.13.

3.8.7. Методи оптимізації

Методи оптимізації використовуються для оновлення ваг у нейронній мережі з метою мінімізації функції втрат. Оптимізатори, описані нижче, були протестовані, і той з них, який показав найкращі результати (AdaM), використовується в запропонованій моделі.

- Розповсюдження середньоквадратичного кореня (RMSProp)

Root Mean Square Propagation (RMSProp)[37] - це оптимізаційний алгоритм для нейронних мереж, який використовує різні швидкості навчання для кожного параметра.

Швидкість навчання регулюється за допомогою середніх значень градієнта ваги, отже, метод метод добре працює на зашумлених даних.

- Адаптивний градієнт (AdaGrad)

Адаптивний градієнт (AdaGrad) також використовує швидкість навчання за кожним параметром. Швидкість навчання AdaGrad низька для частих даних і висока для рідкісних даних , і, таким чином, покращує продуктивність на задачах з розрідженими даними (наприклад, задачах НЛП).

- Адаптивний момент (AdaM)

Найкращий результат було отримано при використанні двонаправленого LSTM з оптимізатором Adaptive Moment (AdaM), який показав найкращу точність прогнозування 0,943 без використання вибірки для виявлення фейкових новин з набору даних Politifact. Як і RMSProp, AdaM використовує середні за градієнтом ваги і є ефективним на зашумлених даних. Однак, Adam також використовує дисперсію, таким чином, поєднуючи переваги RMSProp та ADAgrad. Ще однією перевагою методу Adam є те, що він простий у застосуванні та ефективний в обчислювальному плані.

3.9. Висновки до розділу 3

3.9.1. Порівняння з базовими рішеннями

Результати для запропонованої моделі наведені без вибірки, з надлишковою вибіркою SMOTE та з недостатньою вибіркою за допомогою RandomSampler від Imblearn в таблицях нижче. Вибірка даних зроблена тому, що в наборі даних є більше реальних записів, ніж фальшивих у наборі даних.

Таблиця 3.1. – Отримані результати (no Sampling)

	Accuracy		Fake tweets			Real Tweets			
Method	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	AUC
BiLSTM	0.943	0.987	0.989	0.911	0.948	0.894	0.986	0.938	0.989
C-LSTM	0.937	0.985	0.986	0.903	0.943	0.882	0.983	0.930	0.987
CNN	0.938	0.990	0.991	0.903	0.945	0.878	0.989	0.930	0.989
CNN+ LSTM	0.934	1.000	1.000	0.884	0.938	0.869	1.000	0.930	0.990
SVM	0.624	0.619	0.488	0.407	0.444	0.684	0.750	0.716	0.500
NB	0.703	0.677	0.604	0.628	0.616	0.767	0.749	0.758	0.688
GRU+CNN (Liu)	0.585	0.293	0.585	1.000	0.738	0.000	0.000	0.000	0.576
BiLSTM (Khan)	0.939	0.983	0.985	0.906	0.944	0.889	0.982	0.933	0.988
C-LSTM (Khan)	0.938	0.982	0.984	0.907	0.944	0.885	0.980	0.930	0.987

Результати роботи запропонованої моделі порівнюються з моделями на основі CNN, комбінаціями CNN та LSTM, SVM, Multinomial NB, двома моделями з найкращими результатами роботи з [14]. та модель з найкращими результатами з [15]. Міри точності міри точності пояснюються в розділі 3.7.

Таблиця 3.2. – Отримані результати (Oversampling)

	Accuracy		Fake tweets			Real Tweets			
Method	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	AUC
BiLSTM	0.941	1.000	1.000	0.899	0.947	0.877	1.000	0.934	0.988
C-LSTM	0.936	0.987	0.988	0.900	0.942	0.879	0.985	0.929	0.986
CNN	0.937	1.000	1.000	0.892	0.943	0.869	1.000	0.930	0.987
CNN+ LSTM	0.936	1.000	1.000	0.892	0.943	0.866	1.000	0.928	0.991
SVM	0.587	0.803	0.719	0.286	0.409	0.554	0.888	0.683	0.592
NB	0.664	0.725	0.699	0.576	0.631	0.639	0.752	0.691	0.664
GRU+CNN (Liu)	0.580	0.290	0.580	1.000	0.734	0.000	0.000	0.000	0.501
BiLSTM (Khan)	0.932	0.989	0.990	0.893	0.939	0.867	0.987	0.923	0.988
C-LSTM (Khan)	0.934	0.999	1.000	0.887	0.940	0.864	0.999	0.927	0.987

Матриці плутанини, що відповідають моделям з конфігураціями (без вибірки, з надлишковою вибіркою або вибірки, з надмірною або недостатньою вибіркою), які продемонстрували найкращі результати, наведені в таблицях нижче.

Таблиця 3.3. – Отримані результати (Undersampling)

	Accuracy		Fake tweets			Real Tweets			
Method	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	AUC
BiLSTM	0.939	0.981	0.984	0.909	0.945	0.877	0.979	0.931	0.988
C-LSTM	0.920	0.897	0.914	0.950	0.932	0.929	0.879	0.904	0.988
CNN	0.935	0.964	0.969	0.919	0.944	0.894	0.959	0.925	0.987
CNN+ LSTM	0.939	1.000	1.000	0.893	0.943	0.876	1.000	0.934	0.991
SVM	0.926	0.971	0.969	0.880	0.922	0.890	0.972	0.929	0.929
NB	0.700	0.733	0.720	0.653	0.685	0.683	0.746	0.713	0.700
GRU+CNN (Liu)	0.590	0.295	0.590	1.000	0.742	0.000	0.000	0.000	0.499
BiLSTM (Khan)	0.703	0.677	0.9604	0.628	0.616	0.767	0.749	0.758	0.688
C-LSTM (Khan)	0.935	0.984	0.986	0.901	0.941	0.879	0.982	0.928	0.986

Таблиця 3.4. - BiLSTM Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	2073	202
		24	1697

Таблиця 3.5. - C-LSTM Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	2079	223
		29	1665

Таблиця 3.6. - CNN Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	2113	227
		19	1637

Таблиця 3.7. - CNN + LSTM Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	2027	244
		0	1725

Таблиця 3.8. - SVM Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	914	125
		29	1010

Таблиця 3.9. - Multinomial NB Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	648	384
		424	1264

Таблиця 3.10. - GRU + CNN (Liu) Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	2356	0
		1640	0

Таблиця 3.11. - BiLSTM (Khan) Confusion Matrix

		Actual	
Predicted	Positive Negative	Positive 2048 32	Negative 213 1703

Таблиця 3.12. - C-LSTM (Khan) Confusion Matrix

		Actual	
Predicted	Positive Negative	Positive 2088 34	Negative 215 1659

Запропонована модель продемонструвала найкращі результати по продуктивності. Мережі LSTM виявляють шаблони з часом і найкраще підходять для проблем з довгими текстами. CNN також показала багатообіцяючі результати, і, як зазначено в 3.8.3, CNN можуть працювати краще, ніж LSTM, для виявлення особливостей у коротких текстах, що означає, що CNN може працювати краще, ніж LSTM, на іншому наборі даних.

Кількість ретвітів у наборі даних є недостатньою, а отже, і кількість шляхів поширення чуток (більшість твітів не ретвітовані). Низька точність прогнозування GRU+CNN з [15] підтверджує це, оскільки він використовує лише часові ряди та метадані як вхідні дані. Більша кількість ретвітів, швидше за все, призведе до кращої точності прогнозування для моделей, що використовують RNN, оскільки часові ряди є послідовними.

Недостатня вибірка призводить до втрати інформації, оскільки записи з класу більшості видаляються для вирівнювання розподілу. Надмірна вибірка може призвести до надмірного припасування, оскільки нові записи додаються до класу меншості для вирівнювання розподілу, і ці нові записи можуть не бути добрими представниками справжнього класу меншості.

Надмірна вибірка, швидше за все, дає найвищу точність прогнозу (94,1%) порівняно з недостатньою вибіркою через втрату інформації, яка виникає при видаленні записів з класу більшості. Точність прогнозування при використанні неповної вибірки може бути покращена за рахунок використання більшої кількості даних.

Недодискретизація показала кращі результати, ніж передискретизація для інших моделей. Це, ймовірно, пов'язано з тим, що SMOTE синтезує нові записи, використовуючи старі записи, що може призвести до надмірного припасування. Надмірної пристосованості через надмірну вибірку можна уникнути шляхом впровадження k-кратної перехресної перевірки.

Вхідні дані, окрім метаданих, є послідовними, і, як і очікувалося, Bidirectional LSTM продемонстрував найкращі результати з точністю 94,3 %. ШНМ також показав хороші результати, що підтверджує твердження з розділу 3.6.3 про те, що фейкові статті містять більше вищих ступенів порівняння, ніж справжні статті, оскільки ШНМ добре працюють для виявлення ознак (у цьому випадку для виявлення термінів). CNN швидше навчається, ніж LSTM, а оскільки фейкові новини поширюються набагато швидше, ніж справжні новини, CNN може бути кращим вибором для виявлення фейкових новин.

РОЗДІЛ 4 АНАЛІЗ СТАРТАП ПРОЕКТУ

4.1. Опис ідеї стартап-проекту

Спочатку визначимо стартап-проект. В таблиці 4.1 наведено основну ідею проекту.

Таблиця 4.1. - Опис ідеї стартап-проекту

Зміст ідеї	Напрямок застосування	Вигода для користувача
Telegram-бот для перевірки інформації на достовірність	Особисті потреби	Достовірна інформація
	Професійна діяльність	Допомога при аналізі даних, складанні звітів і тд

Далі необхідно визначити конкурентів на ринку і зробити порівняльний аналіз, виявити їх переваги та недоліки. Прямих аналогів не існує, тому вибрали конкурентів, частини функціоналу яких співпадає. В таблиці 4.2 наведені основні характеристики ідеї.

№	Техніко-економічні	(потенційні) товари/концепції конкурентів	W (слабка	N (нейтральна	S (сил

		Мій проект	Credder	The Factual			
1	Лояльність споживачів	Висока в своїй області	Висока в своїй області	Висока в своїй області		+	
2	Ціна	Нижча ринку	Задовільна	Задовільна		+	
3	Функціонал	Широкий	Широкий	Широкий		+	

Таблиця 4.2. - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

4.2.Технологічний аудит ідеї проекту

Наступним необхідним кроком є аудит технологій.

Таблиця 4.3. - Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1	Telegram-бот для перевірки інформації на достовірність	Використання мови програмування C++	Відсутні	Недоступні
2		Використання мови програмування Java	Відсутні	Недоступні
3		Використання мови програмування Python	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: мова програмування Python				

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Стартап не існує в вакуумі, тому необхідно проаналізувати ринок, на який ми виходимо. В таблиці 4.4 наведена характеристика ринку.

Таблиця 4.4. - Попередня характеристика потенційного ринку стартап-проекту:

№ п/п	Показники стану ринку (найменування)	*Характеристика*
1	Кількість головних гравців, од	500 тис.
2	Загальний обсяг продаж	1 млн. $
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	25

Наступним кроком необхідно охарактеризувати основні групи потенційних користувачів продукту і скласти опис вимог кожної такої групи (таблиця 4.5).

Таблиця 4.5. - Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Наявність системи в зручному місці	Всі, хто використовує Телеграм	Читачі новин у Телеграмі	Достовірність
2	Аудиторія формує ринок реклами	Замовники реклами	Початківці	Аудиторія

Далі необхідно проаналізувати загрози, які можуть виникнути та перешкодити вдалому запуску проекту.

Таблиця 4.6. - Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Мала інформованість	Багато людей можуть просто не знати про мережу	Рекламна кампанія в новинних пабліках
2	Залежність від платформи	Орієнтована лише на Телеграм, в разі проблем з даною платформою, даний бот зазнає краху	Розширення

Також необхідно розглянути фактори, які можуть сприяти проекту.

Таблиця 4.7. - Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Побудова аналітики	Отримання ширшої інформації	Самореклама
2	Збільшення аудиторії	Збільшення ціни реклами	Більше інвестицій в розвиток
3	Створення розширення для браузера	Збільшення можливостей для аналітики	Створення позитивного іміджу

Далі необхідно охарактеризувати конкурентне середовище. Результати аналізу наведено у таблиці 4.8. Також виконаємо аналіз конкуренції за моделлю п’яти сил конкуренції Майкла Портера. Результати аналізу зведено в таблицю 4.9.

Таблиця 4.8. - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути*

		конкурентоспроможною)
1. Вказати тип конкуренції: Чиста	Багато систем/ спеціалістів	Розробити відомий продукт
2. За рівнем конкурентної боротьби: міжнаціональна	Є системи з різних країн	Розширити аудиторію
3. За галузевою ознакою: міжгалузева	Застосовується в різних цілях	Розширити можливості
4. Конкуренція за видами товарів: товарно-родова	Конкуренція з іншими подібними інструментами	Розширити функціонал
5. За характером конкурентних переваг: Нецінова	Різні інструменти для моделювання	Збільшення якості та функціоналу
6. За інтенсивністю: Марочна	Бренд надає переваги	Розвивати бренд

Таблиця 4.9. - Аналіз конкуренції в галузі за М. Портером

Складові аналізу	*Прямі конкуренти в галузі*	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
	Spotify	*Спеціалізованість ринку*	*Їх методи та реалізація*	*Їх роботи*	*Більш якісний*
Висновки	Інтенсивна боротьба навряд буде	Є можливості та потенційні конкуренти	Не диктують	Не диктують	Товари замінники відсутні

Результати аналізу підтверджують, що на ринку сприятливі умови для створення і запуску даного стартап-проекту. Були сформульовані та обґрунтовані перелік факторів конкурентоспроможності. Аналіз оформлено в таблицю 4.10.

Таблиця 4.10. - Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння*

		конкурентних проектів значущим)
1	Багатофункціональність	Саме початок не є багатофункціональним, але є шляхи розвитку
2	Доступність програмного продукту	Загальнодоступність
3	Обслуговування	Періодичні оновлення бази та покращення інструменту

Після проведення аналізу можна виділити сильні та слабкі (які потребують вдосконалення) сторони продукту (таблиця 4.11).

Таблиця 4.11. - Порівняльний аналіз сильних та слабких сторін системи

№ п/ п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів*						
			-3	*-2*	*-1*	*0*	*+1*	*+2*	*+3*
1	Багатофункціональність	15			-				
2	Доступність програмного продукту	20					+		
3	Обслуговування	20						+	

SWOT-аналіз дозволяє оцінити можливості та загрози бізнесу, а також сильні і слабкі сторони продукту. Результати наведені у таблиці 4.12.

Таблиця 4.12. - SWOT- аналіз стартап-проекту

Сильні сторони: Доступ до правдивої інформації Зручність і легкість використання Компактність Багатофункціональність	**Слабкі сторони:** Відсутність кросплатформенності Низька обізнаність аудиторії про бот
Можливості: Залучення реклами Вихід на нові ринки Допрацювання функціоналу	**Загрози:** Конкуренція Зміни в функціоналі Телеграму

На основі SWOT-аналізу було спроектовано альтернативну ринкову поведінку для інтеграції стартап-проекту (таблиця 4.13).

Таблиця 4.13. - Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Швидкий вихід з мінімально функціональним продуктом	20%	4 місяці
2	Поступовий вихід з якісним продуктом	80%	8 місяців

4.4.Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії розпочинається з визначення стратегії охоплення ринку. Для цього необхідно охарактеризувати цільові групи потенційних споживачів (таблиця 4.14).

Таблиця 4.14 - Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1	Всі, хто використовує Телеграм	Висока	80%	Низька	Висока
2	Замовники реклами	Висока	20%	Низька	Середня
Які цільові групи обрано:1, 2					

Для роботи в обраних сегментах ринку сформуємо базову стратегію розвитку (таблиці 4.15, 4.16, 4.17).

Таблиця 4.15. - Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку**
1	3	Стратегія спеціалізації	Функціональність, простота входу	Стратегія диференційованого маркетингу

Таблиця 4.16. - Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки**
1	Так	Так	Ні	Стратегія лідеру

Таблиця 4.17. - Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
1	Якість, зрозумілий інтерфейс	Стратегія диференційованого маркетингу	Якість, висока функціональність, велика кількість інформації	Імідж, якість, спеціалізація

4.5. Розроблення маркетингової програми стартап-проекту

Кінцевим етапом є формування маркетингової концепції товару. Спочатку необхідно визначити ключових переваг концепції потенційного товару (таблиця 4.18). Також опишемо три рівні моделі товару (таблиця 4.19).

Таблиця 4.18. - Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)*
1	Достовірність інформації	Правдива інформація	Спеціалізація
2	Реклама	Велика аудиторія	Унікальність

Таблиця 4.19. - Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Бот для перевірки інформації на достовірність		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	Програмний продукт – складна, комплексна система		
	Якість: тестування споживачем		
	Пакування - інсталятор		
	Марка: назва організації-розробника + назва товару		
III. Товар із підкріпленням	До продажу: гнучка оплата, доставка		
	Після продажу: гарантія якості та оновлення		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності			

Далі необхідно визначити цінові межі продукту (таблиця 4.20).

Таблиця 4.20. - Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	100	150	500 млн.	50-200

4.6. Висновки до розділу 4

Був виконаний перший етап розроблення стартап проекту, а саме, маркетинговий аналіз.

Стартап є актуальним, оскільки в мережі присутня велика кількість фейків та чуток, а розроблений телеграм-бот може допомогти із вирішенням цих проблем. Також він має низький поріг входу та велику потенційну аудиторію. Більше того, присутнє поле розвитку та майбутньої оптимізації.

Стартап не потребує великої кількості інвестицій таким чином розвивати проект буде досить легко в разі отримання прибутку.

ВИСНОВКИ

Багато записів містили URL-адреси відео замість статей, і їх довелося відкинути. Ці відео можуть бути використані для класифікації відео за допомогою CNN. Зображення, такі як зображення профілю користувача, також можуть бути додані до набору даних і використані як вхідні дані для класифікації зображень за допомогою CNN.

Може бути реалізована K-кратна перехресна перевірка, щоб уникнути надмірного припасування через надмірну вибірку.

Запропоноване рішення може бути доопрацьоване шляхом створення веб-додатку з графічним інтерфейсом користувача, який використовує вміст статті або декілька ознак як вхідні дані для виявлення фейкових новин.

Більшість твітів взагалі не ретвітиться, а це означає, що для часових рядів поширення чуток доступно мало даних. Більше даних можна зібрати з Твіттера або інших платформ соціальних мереж, або використати інший набір даних з більшою кількістю ретвітів. Набір даних містить багато особливостей, які не були використані в цій роботі, але можуть бути досліджені в подальшому.

Характеристики користувачів можуть бути використані для виявлення наївних користувачів, які легко піддаються на фейкові новини та поширюють їх.

Дані можуть бути додатково деноміновані (наприклад, видалити нерелевантний текст зі змісту статті). З твітів і ретвітів можна видаляти смайлики.

Вміст твітів і ретвітів можна використовувати як вхідні дані разом із вмістом відповідних статей. відповідних статей.

У цій дипломній роботі успішно використано контент статті, метадані та шляхи розповсюдження чуток для покращення виявлення фейкових новин

шляхом моделювання графової нейронної мережі. Запропонована модель, Bidirectional LSTM, показала кращі результати, ніж найсучасніші рішення з суміжних робіт та моделей, реалізованих в даній дисертації, з точністю 94,3%. Результати роботи результатів роботи перевірено за допомогою t-критерію Стьюдента з альфою 0,01 та розподілами ймовірностей прогнозування для кожної з моделей в якості вхідних даних. Результати узгоджуються з висновком про те, що запропонована модель є найбільш точним класифікатором, але також і те, що вона не є суттєво кращою за реалізовану CNN+LSTM модель при використанні заниженої дискретизації.

ПЕРЕЛІК ПОСИЛАНЬ

1.Fake news URL https://dictionary.cambridge.org/dictionary/english/fake-news

2.Daniel Avelar. Whatsapp fake news during brazil election ‘favoured bolsonaro’, 2019. URL https://www.theguardian.com/world/2019/oct/30/whatsapp-fake-newsbrazil-election-favoured-jair-bolsonaro-analysis-suggests

3.Against information manipulation, 2018. URL https://www.gouvernement.fr/en/against-information-manipulation

4.Jon Danzig. How fake news caused brexit, 2017. URL https://europe.ideasoneurope.eu/2017/11/14/fake-news-caused-brexit/

5.Eliza Mackintosh. Finland is winning the war on fake news. What it’s learned may be crucial to western democracy, 2018. URL https://edition.cnn.com/interactive/2019/05/europe/finland-fake-news-intl/

6.Kelly Buchanan. Malaysia: Anti-fake news act comes into force, 2018. URL https://www.loc.gov/law/foreign-news/article/malaysia-anti-fakenews-act-comes-into-force/

7.Casey Newton. Singapore’s fake news law should be a warning to american lawmakers, 2019. URL https://www.theverge.com/interface/2019/12/3/20991422/singapore-fake-news-law-censorship-politics-usa

8.Gerret Von Nordheim. journalists quote social media content ever more frequently, 2018. URL https://en.ejo.ch/research/journalists-quotesocial-media-content-ever-more-frequently

9.How fake news and rumors are stoking division in hong kong, 2019. URL https://www.bloomberg.com/news/articles/2019-11-11/how-fake-news-isstoking-violence-and-anger-in-hong-kong

10. James Cheo. Fake news can make - or break - stock prices, 2018. URL https://www.businesstimes.com.sg/opinion/fake-news-can-make-orbreak-stock-prices

11.Ed Kilgore. Legislators urged to declare cnn and washington post ‘fake news’, 2020. URL https://nymag.com/intelligencer/2020/02/legislatordeclares-cnn-and-washington-post-fake-news.html

12.Abigail Abrams. Here’s what we know so far about russia’s 2016 meddling, 2019. URL https://time.com/5565991/russia-influence-2016-election/

13.Alexander Polyakov. Detecting fake content: One of the biggest challenges for 2020, 2018. URL https://www.forbes.com/sites/forbestechcouncil/2020/01/02/detectingfakecontentone-of-the-biggest-challenges-for-2020/#8c82b4e1219d

14.Junaed Younus Khan, Md Khondaker, Tawkat Islam, Anindya Iqbal, and Sadia Afroz. A benchmark study on machine learning methods for fake news detection. arXiv preprint

arXiv:1905.04749, 2019.

15.Yang Liu and Yi-Fang Brook Wu. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018

16.T. Srinivasa Ravi Kiran A.Lakshmanarao, Y.Swathi. An effecient fake news detection system using machine learning. 2019. ISSN 2278-3075.

17.Álvaro Ibrain and Lara Lloret. Fake news detection using deep learning, 2019

18.Shubham Panchal. Artificial neural networks — mapping the human brain, 2018. URL https://medium.com/predict/artificial-neuralnetworks-mapping-the-human-brain-2e0bd4a93160

19.Bernard Marr. What is deep learning ai? a simple guide with 8 practical examples, 2018. URL https://www.forbes.com/sites/bernardmarr/2018/10/01/what-is-deeplearning-ai-a-simple-guide-with-8-practical-examples/#23fe9e648d4b

20.Casper Hansen. Neural networks: Feedforward and backpropagation explained optimization, 2019. URL https://mlfromscratch.com/neural-networks-explained/#/

21.Matt Mazur. A step by step backpropagation example, 2015. URL https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

22.Suryansh S. Gradient descent: All you need to know, 2018. URL https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

23.Samarth Agrawal. What the heck is word embedding, 2019. URL https://towardsdatascience.com/what-the-heck-is-word-embeddingb30f67f01c81

24.Jeffrey Pennington. Glove: Global vectors for word representation, 2014. URL https://nlp.stanford.edu/projects/glove/

25.Jason Brownlee. Understanding gru networks, 2017b. URL https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

26.Michael Phi. Illustrated guide to lstm's and gru's: A step by step explanation, 2018. URL https://towardsdatascience.com/illustratedguide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

27.Nltk (natural language toolkit) tutorial in python, 2020. URL https://www.guru99.com/nltk-tutorial.html

28.Generating wordclouds in python, 2019. URL https://www.datacamp.com/community/tutorials/wordcloud-python

29.Re - regular expression operations, 2020. URL https://docs.python.org/3/library/re.html

30.datetime — basic date and time types, 2020. URL https://docs.python.org/3/library/datetime.html

31.Gridsearchcv, 2019. URL https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

32.Politifact, 2020b. URL https://www.allsides.com/news-source/politifact

33.Cumulative distribution function, probability density function. URL https://confluence.ecmwf.int/display/FUG/Cumulative+Distribution+Function%2C+Probability+Density+Function

34.Aditya Mishra. Metrics to evaluate your machine learning algorithm, 2018. URL https://towardsdatascience.com/metrics-to-evaluate-yourmachine-learning-algorithm-f10ba6e38234

35.Sebastian Raschka. Machine learning faq, 2020. URL https://sebastianraschka.com/faq/docs/multiclass-metric.html

36.Syet Sadat Nazrul. Multinomial naive bayes classifier for text analysis (python), 2018. URL https://towardsdatascience.com/multinomial-naivebayes-classifier-for-text-analysis-python-8dd6825ece67

37.Ayoosh Kathuria. Intro to optimization in deep learning, 2018. URL https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
#coding: utf-8
import json
import matplotlib.pyplot as plt
from langdetect import detect  #detects what language is written
from  tqdm import tqdm
import nltk
nltk.download('stopwords')

import nltk
import re
from tqdm import tqdm_notebook
from nltk.corpus import stopwords

from tensorflow.keras import regularizers, initializers, optimizers, callbacks
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from keras.utils.np_utils import to_categorical
from tensorflow.keras.layers import *
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from keras.layers import LSTM
from keras.layers import Bidirectional
from keras.layers import GlobalMaxPool1D
from keras.layers import Dropout
from keras.layers import InputLayer
```

```
import pandas as pd
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
import numpy as np
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
df = pd.read_pickle("/local/home/bohdan/Fakenews_Classification/Preprocessing/bigdata_preprocessed.pkl")
df['created_at'] = df['created_at'].astype(str)
df['followers'] = df['followers'].astype(str)
df['following'] = df['following'].astype(str)

cat_Cols = ['text', 'name_user','name_zero_user_mentions_entities', 'location_user', 'description_user', 'contributors_enabled_user', 'default_profile_image_user', 'default_profile_user', 'favorited', 'follow_request_sent_user', 'following_user', 'geo_enabled_user', 'has_extended_profile_user', 'id', 'id_str', 'id_str_user', 'id_str_zero_user_mentions_entities', 'id_user', 'id_zero_user_mentions_entities', 'is_quote_status', 'is_translation_enabled_user', 'is_translator_user', 'lang', 'location_user', 'name_user', 'notifications_user', 'possibly_sensitive', 'possibly_sensitive_appealable', 'profile_background_color_user', 'profile_background_tile_user', 'profile_link_color_user', 'profile_sidebar_border_color_user', 'profile_sidebar_fill_color_user', 'profile_text_color_user', 'profile_use_background_image_user', 'protected_user', 'retweeted', 'screen_name_user', 'screen_name_zero_user_mentions_entities', 'translator_type_user', 'truncated', 'verified_user']

cat_Features =df['text']
print(str(df['text']).encode('utf-8').decode('latin-1'))
```

```
    for col in cat_Cols:
        df[col] = df[col].astype(str)

    label = pd.get_dummies(df['label'])
    label = np.array(label)

    num_Cols = ['favorite_count', 'favourites_count_user',
'followers_count_user', 'friends_count_user', 'listed_count_user', 'retweet_count',
'statuses_count_user', 'zero_indices_zero_urls_entities',
'one_indices_zero_urls_entities', 'zero_indices_zero_urls_url_entities_user',
'one_indices_zero_urls_url_entities_user',
'zero_indices_zero_user_mentions_entities',
'one_indices_zero_user_mentions_entities']
    num_Features = []

    for idx in num_Cols:
        print(idx)
        num_Features.append(df[idx])
    print(type(num_Features[0]))

    import pandas as pd
    from sklearn import preprocessing

    for i,col in enumerate(num_Features):
        if i == 0:
            mat = col
        else:
            mat = pd.concat([mat, col], axis=1)

    x = mat.values #returns a numpy array
```

```
        min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-0.5,0.5))
        x_scaled = min_max_scaler.fit_transform(x)
        num_Norm = pd.DataFrame(x_scaled)

        for col in num_Norm.keys():
            num_Norm[col] = pd.cut(num_Norm[col], bins=3, labels=np.arange(3),
right=False)



        strings = []
        stop_words = set(stopwords.words('english'))

        for line in tqdm_notebook(cat_Features, total=df.shape[0]):
            line = line.replace("""","")
            line = line.replace("\[","")
            line = line.replace(":","")
            line = line.replace("-","")
            line = line.replace("_","")
            line = line.replace("+","")
            line = line.replace("?","")
            line = line.replace(",","")
            line = line.replace("\]","")
            line = line.replace(".","")
            line = line.replace("â€˜","")
            line = line.replace("â€™","")
            line = line.replace("@","")
            line = line.replace("!","")
            line = line.replace("$","")
            line = line.replace("=","")
```

```
            line = line.replace("(","")
            line = line.replace(")","")
            line = line.replace("/","")
            line = line.replace("\\","")
            line = line.replace("&","")
            line = line.replace("#","")
            line = re.sub('\d', '', line)
            line = line.split(' ')
            line = [w for w in line if not w in stop_words]
            line = str(line)
            line = str(line.strip())[1:-1].replace(' ', ' ')
            strings.append(line)

        #encode text as numbers
        tok_Len = 100000 # max number of words for tokenizer
        tokenizer = Tokenizer(num_words=tok_Len)
        tokenizer.fit_on_texts(strings)
        sequences = tokenizer.texts_to_sequences(strings)
        term_Index = tokenizer.word_index
        print('Number of Terms:', len(term_Index))



        sen_Len = 162 # max length of each sentences, including padding
        tok_Features = pad_sequences(sequences, padding = 'post', maxlen =
sen_Len-111)
        print('Shape of tokenized features tensor:', tok_Features.shape)

        indices = np.arange(tok_Features.shape[0])
        np.random.shuffle(indices)
        time_series = df['created_at_retweets']
```

```
time_series.reset_index(drop=True, inplace=True)
print(type(time_series))
time_series = time_series[indices]
print(time_series)
tok_Features = tok_Features[indices]
labels = label[indices]
print(num_Norm)
print(num_Norm.shape)
print(type(num_Norm))

test_Perc = 0.2 #20% data used for testing
num_validation_samples = int(test_Perc*tok_Features.shape[0])
features_Train = tok_Features[: -num_validation_samples]
time_series_Train = time_series[: -num_validation_samples]
num_Norm_Train = num_Norm[: -num_validation_samples]
target_Train = labels[: -num_validation_samples]
features_Val = tok_Features[-num_validation_samples: ]
num_Norm_Val = num_Norm[-num_validation_samples: ]
time_series_Val = time_series[-num_validation_samples: ]
target_Val = labels[-num_validation_samples: ]
print('Number of records in each attribute:')
print('training: ', target_Train.sum(axis=0))
print('validation: ', target_Val.sum(axis=0))


emb_Dim = 100 # embedding dimensions for word vectors
glove =
'/local/home/bohdan/Fakenews_Classification/BiLSTM/glove.6B.'+str(emb_Dim)
+'d.txt'
emb_Ind = {}
```

```
f = open(glove, encoding='utf8')
print('Loading Glove \n')
for line in f:
    values = line.split()
    term = values[0]
    emb_Ind[term] = np.asarray(values[1:], dtype='float32')
f.close()
print("Done---\nMap terms to embedding---")

emb_Mat = np.random.random((len(term_Index) + 1, emb_Dim))
for term, i in term_Index.items():
    emb_Vec = emb_Ind.get(term)
    if emb_Vec is not None:
        emb_Mat[i] = emb_Vec
print("Done")
print('SHAPE EMB MAT')
print(emb_Mat.shape)
print(type(emb_Mat))
print('SHAPE TIME SERIES')
print(time_series_Train.shape)
print(type(time_series_Train))
maxLen = 0
maxIndex = 0

time_series_Train = np.asarray(time_series_Train)

for i,times in enumerate(time_series_Train):
    time_series_Train[i] = np.array(time_series_Train[i])
max_len = max([len(x) for x in time_series_Train])
for i,times in enumerate(time_series_Train):
```

```
        time_series_Train[i] = np.pad(times,(0,max_len-len(times)), 'constant')
    time_series_Mat = np.zeros((len(time_series_Train),max_len))
    for i, times in enumerate(time_series_Train):
        for j, time in enumerate(time_series_Train[i]):
            time_series_Mat[i,j] = time
    features_Train = np.concatenate([features_Train,time_series_Mat],axis=1)

    features_Train = np.concatenate([features_Train,num_Norm_Train],axis=1)



    def create_model():
    # create model
        model = Sequential()
        model.add(InputLayer((sen_Len,),dtype='int32'))
        e = Embedding(len(term_Index) + 1, emb_Dim, weights=[emb_Mat],
input_length=sen_Len, trainable=False)
        print('SEN LEN')
        print(sen_Len)
        print('TERM INDEX')
        print(len(term_Index))
        model.add(e)
        model.add(Bidirectional(LSTM(60, return_sequences=True)))
        model.add(GlobalMaxPool1D())
        model.add(Dropout(0.1))
        model.add(Dense(158, activation='relu'))
        model.add(Dropout(0.1))
        model.add(Dense(2, activation='sigmoid'))

        model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics =
['acc'])
```

```
        return model

    gridmodel = KerasClassifier(build_fn=create_model,epochs=10, batch_size=5, verbose=0)

    batch_size = [8,16,32,64,128]
    epochs = [1,2,3,4,5]
    param_grid = dict(batch_size=batch_size, epochs=epochs)
    grid = GridSearchCV(estimator=gridmodel, param_grid=param_grid, cv=3)
    grid_result = grid.fit(features_Train, target_Train)
    print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
    means = grid_result.cv_results_['mean_test_score']
    stds = grid_result.cv_results_['std_test_score']
    params = grid_result.cv_results_['params']
    for mean, stdev, param in zip(means, stds, params):
        print("%f (%f) with: %r" % (mean, stdev, param))
    print(type(grid_result.best_params_))
    print(len(grid_result.best_params_))
    print(grid_result.best_params_.keys())
    epochs_var = grid_result.best_params_['epochs']
    batch_size_var = grid_result.best_params_['batch_size']

    model = Sequential()
    model.add(InputLayer((sen_Len,),dtype='int32'))
    e = Embedding(len(term_Index) + 1, emb_Dim, weights=[emb_Mat], input_length=sen_Len, trainable=False)
    print('SEN LEN')
    print(sen_Len)
    print('TERM INDEX')
```

```
print(len(term_Index))
model.add(e)
model.add(Bidirectional(LSTM(60, return_sequences=True)))
model.add(GlobalMaxPool1D())
model.add(Dropout(0.1))
model.add(Dense(160, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(2, activation='sigmoid'))
```

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics = ['acc'])
```

```
print(max_len)
history = model.fit(features_Train, target_Train, epochs = epochs_var, batch_size=batch_size_var, validation_split=0.20)
time_series_Val = np.asarray(time_series_Val)

for i,times in enumerate(time_series_Val):
    time_series_Val[i] = np.array(time_series_Val[i])
max_len_val = max([len(x) for x in time_series_Val])
for i,times in enumerate(time_series_Val):
    time_series_Val[i] = np.pad(times,(0,max_len_val-len(times)), 'constant')
time_series_val_Mat = np.zeros((len(time_series_Val),max(max_len,max_len_val)))
for i, times in enumerate(time_series_Val):
    for j, time in enumerate(time_series_Val[i]):
```

```
            time_series_val_Mat[i,j] = time
        combined_Val = np.concatenate([features_Val,time_series_val_Mat],axis=1)
        combined_Val = np.concatenate([combined_Val,num_Norm_Val],axis=1)
        predictions = model.predict(combined_Val)


        import sklearn.metrics as metrics
        from sklearn.metrics import roc_auc_score
        from sklearn.metrics import classification_report
        predictions_bool = np.argmax(predictions, axis=1)
        print(target_Val.shape)
        print(target_Val.shape)
        target_Val = np.argmax(target_Val, axis=1)
        print(predictions_bool.shape)
        print(classification_report(target_Val, predictions_bool,digits=3))
        predictions_prob = model.predict_proba(combined_Val)
        import pickle
        with open('/local/home/bohdan/Fakenews_Classification/T_Test/BiLSTM_proba.pkl','wb') as f:
            pickle.dump(predictions_prob, f)
        target_cat = to_categorical(target_Val)
        auc = roc_auc_score(target_cat, predictions_prob)
        y_pred = (predictions > 0.5)
        matrix = metrics.confusion_matrix(target_Val, y_pred.argmax(axis=1))
        micro = (matrix[0,0]+matrix[1,1])/(matrix[0,0]+matrix[1,1]+matrix[0,1]+matrix[1,0])
        print('micro')
        print(micro)
```

```
print('macro')
macro =
(matrix[0,0]/(matrix[0,0]+matrix[1,0])+matrix[1,1]/(matrix[1,1]+matrix[1,0]))/2
print(macro)
print('auc')
print(auc)
print(matrix)
import matplotlib.pyplot as plt
#%matplotlib inline

val_Loss = history.history['val_loss']
loss = history.history['loss']
epochs = range(1, len(loss)+1)

plt.plot(epochs, loss, label='Training Loss')
plt.plot(epochs, val_Loss, label='Testing Loss')
plt.title('Training and Testing Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.savefig("loss_Function"+".png", bbox_inches='tight')


acc = history.history['acc']
val_Acc = history.history['val_acc']
plt.plot(epochs, acc, label='Training accuracy')
plt.plot(epochs, val_Acc, label='Testing Accuracy')
plt.title('Training and Testing Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
```

```
plt.legend()
plt.savefig("accuracy"+".png", bbox_inches='tight')

#coding: utf-8
import json
import matplotlib.pyplot as plt
from keras.layers import Bidirectional
from langdetect import detect
from  tqdm import tqdm
import nltk
nltk.download('stopwords')
from sklearn.metrics import roc_auc_score
import nltk
import re
from tqdm import tqdm_notebook
from nltk.corpus import stopwords

from tensorflow.keras import regularizers, initializers, optimizers, callbacks
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from keras.utils.np_utils import to_categorical
from tensorflow.keras.layers import *
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from keras.layers import LSTM
from keras.layers import Bidirectional
from keras.layers import GlobalMaxPool1D
from keras.layers import Dropout
```

```
from keras.layers import InputLayer
from sklearn.metrics import classification_report
import pandas as pd
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
import numpy as np
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV

df = pd.read_pickle("/local/home/bohdan/Fakenews_Classification/Preprocessing/bigdata_preprocessed.pkl")
df['created_at'] = df['created_at'].astype(str)
df['followers'] = df['followers'].astype(str)
df['following'] = df['following'].astype(str)

cat_Cols = ['text', 'name_user','name_zero_user_mentions_entities', 'location_user', 'description_user', 'contributors_enabled_user', 'default_profile_image_user', 'default_profile_user', 'favorited', 'follow_request_sent_user', 'following_user', 'geo_enabled_user', 'has_extended_profile_user', 'id', 'id_str', 'id_str_user', 'id_str_zero_user_mentions_entities', 'id_user', 'id_zero_user_mentions_entities', 'is_quote_status', 'is_translation_enabled_user', 'is_translator_user', 'lang', 'location_user', 'name_user', 'notifications_user', 'possibly_sensitive', 'possibly_sensitive_appealable', 'profile_background_color_user', 'profile_background_tile_user', 'profile_link_color_user', 'profile_sidebar_border_color_user', 'profile_sidebar_fill_color_user', 'profile_text_color_user', 'profile_use_background_image_user', 'protected_user', 'retweeted', 'screen_name_user', 'screen_name_zero_user_mentions_entities', 'translator_type_user', 'truncated', 'verified_user']
```

```
        cat_Features =df['text']
        print(str(df['text']).encode('utf-8').decode('latin-1'))
        for col in cat_Cols:
            df[col] = df[col].astype(str)

        print(df['label'])
        label = pd.get_dummies(df['label'])
        label = np.array(label)
        print(np.argmax(label, axis=1))

        num_Cols = ['favorite_count', 'favourites_count_user',
'followers_count_user', 'friends_count_user', 'listed_count_user', 'retweet_count',
'statuses_count_user', 'zero_indices_zero_urls_entities',
'one_indices_zero_urls_entities', 'zero_indices_zero_urls_url_entities_user',
'one_indices_zero_urls_url_entities_user',
'zero_indices_zero_user_mentions_entities',
'one_indices_zero_user_mentions_entities']
        num_Features = []
        for idx in num_Cols:
            print(idx)
            num_Features.append(df[idx])
        print(type(num_Features[0]))

        import pandas as pd
        from sklearn import preprocessing

        for i,col in enumerate(num_Features):
            if i == 0:
```

```
            mat = col
        else:
            mat = pd.concat([mat, col], axis=1)

    x = mat.values #returns a numpy array
    min_max_scaler = preprocessing.MinMaxScaler(feature_range=(0,1))
    x_scaled = min_max_scaler.fit_transform(x)
    num_Norm = pd.DataFrame(x_scaled)

    for col in num_Norm.keys():
        num_Norm[col] = pd.cut(num_Norm[col], bins=3, labels=np.arange(3),
right=False)

    strings = []
    stop_words = set(stopwords.words('english'))

    for line in tqdm_notebook(cat_Features, total=df.shape[0]):
        line = line.replace("'","")
        line = line.replace("\[","")
        line = line.replace(":","")
        line = line.replace("-","")
        line = line.replace("_","")
        line = line.replace("+","")
        line = line.replace("?","")
        line = line.replace(",","")
        line = line.replace("\]","")
        line = line.replace(".","")
        line = line.replace("â€˜","")
        line = line.replace("â€™","")
        line = line.replace("@","")
```

```
        line = line.replace("!","")
        line = line.replace("$","")
        line = line.replace("=","")
        line = line.replace("(","")
        line = line.replace(")","")
        line = line.replace("/","")
        line = line.replace("\\","")
        line = line.replace("&","")
        line = line.replace("#","")
        line = re.sub('\d', '', line)
        line = line.split(' ')
        line = [w for w in line if not w in stop_words]
        line = str(line)
        line = str(line.strip())[1:-1].replace(' ', ' ')
        strings.append(line)

    #encode text as numbers
    tok_Len = 100000 # max number of words for tokenizer
    tokenizer = Tokenizer(num_words=tok_Len)
    tokenizer.fit_on_texts(strings)
    sequences = tokenizer.texts_to_sequences(strings)
    term_Index = tokenizer.word_index
    print('Number of Terms:', len(term_Index))


    sen_Len = 162 # max length of each sentences, including padding
    tok_Features = pad_sequences(sequences, padding = 'post', maxlen =
sen_Len-111)
    print('Shape of tokenized features tensor:', tok_Features.shape)
```

```
indices = np.arange(tok_Features.shape[0])
np.random.shuffle(indices)
time_series = df['created_at_retweets']
time_series.reset_index(drop=True, inplace=True)
print(type(time_series))
time_series = time_series[indices]
print(time_series)
tok_Features = tok_Features[indices]
labels = label[indices]
print(num_Norm)
print(num_Norm.shape)
print(type(num_Norm))

test_Perc = 0.2 #20% data used for testing
num_validation_samples = int(test_Perc*tok_Features.shape[0])
features_Train = tok_Features[: -num_validation_samples]
time_series_Train = time_series[: -num_validation_samples]
num_Norm_Train = num_Norm[: -num_validation_samples]
target_Train = labels[: -num_validation_samples]
features_Val = tok_Features[-num_validation_samples: ]
num_Norm_Val = num_Norm[-num_validation_samples: ]
time_series_Val = time_series[-num_validation_samples: ]
target_Val = labels[-num_validation_samples: ]
print('Number of records in each attribute:')
print('training: ', target_Train.sum(axis=0))
print('validation: ', target_Val.sum(axis=0))


emb_Dim = 100 # embedding dimensions for word vectors
```

```
        glove = '/local/home/bohdan/Fakenews_Classification/BiLSTM/glove.6B.'+str(emb_Dim)+'d.txt'
        emb_Ind = {}
        f = open(glove, encoding='utf8')
        print('Loading Glove \n')
        for line in f:
            values = line.split()
            term = values[0]
            emb_Ind[term] = np.asarray(values[1:], dtype='float32')
        f.close()
        print("Done---\nMap terms to embedding---")

        emb_Mat = np.random.random((len(term_Index) + 1, emb_Dim))
        for term, i in term_Index.items():
            emb_Vec = emb_Ind.get(term)
            if emb_Vec is not None:
                emb_Mat[i] = emb_Vec
        print("Done")
        print('SHAPE EMB MAT')
        print(emb_Mat.shape)
        print(type(emb_Mat))
        print('SHAPE TIME SERIES')
        print(time_series_Train.shape)
        print(type(time_series_Train))
        maxLen = 0
        maxIndex = 0

        time_series_Train = np.asarray(time_series_Train)
```

```
    for i,times in enumerate(time_series_Train):
        time_series_Train[i] = np.array(time_series_Train[i])
    max_len = max([len(x) for x in time_series_Train])
    for i,times in enumerate(time_series_Train):
        time_series_Train[i] = np.pad(times,(0,max_len-len(times)), 'constant')
    time_series_Mat = np.zeros((len(time_series_Train),max_len))
    for i, times in enumerate(time_series_Train):
        for j, time in enumerate(time_series_Train[i]):
            time_series_Mat[i,j] = time
    features_Train = np.concatenate([features_Train,time_series_Mat],axis=1)

    features_Train = np.concatenate([features_Train,num_Norm_Train],axis=1)

    from imblearn.over_sampling import SMOTE
    oversample = SMOTE()
    print(target_Train)
    print(target_Train.shape)
    print(type(target_Train))
    features_Train_Resampled, target_Train_Resampled = oversample.fit_resample(features_Train, target_Train)
    from keras.utils import to_categorical
    target_Train_Resampled = to_categorical(target_Train_Resampled)
    print(target_Train.shape)
    print(type(target_Train))
    print(target_Train)
    print('FEATURES TRAIN')
    def create_model():
    # create model
        model = Sequential()
        model.add(InputLayer((sen_Len,),dtype='int32'))
```

```
        e = Embedding(len(term_Index) + 1, emb_Dim, weights=[emb_Mat], input_length=sen_Len, trainable=False)
        print('SEN LEN')
        print(sen_Len)
        print('TERM INDEX')
        print(len(term_Index))
        model.add(e)
        model.add(Bidirectional(LSTM(60, return_sequences=True)))
        model.add(GlobalMaxPool1D())
        model.add(Dropout(0.1))
        model.add(Dense(158, activation='relu'))
        model.add(Dropout(0.1))
        model.add(Dense(2, activation='sigmoid'))

        model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics = ['acc'])
        return model

    gridmodel = KerasClassifier(build_fn=create_model,epochs=10, batch_size=5, verbose=0)

    batch_size = [8,16,32,64,128]
    epochs = [1,2,3,4,5]
    param_grid = dict(batch_size=batch_size, epochs=epochs)
    grid = GridSearchCV(estimator=gridmodel, param_grid=param_grid, cv=3)
    grid_result = grid.fit(features_Train_Resampled, target_Train_Resampled)
    # summarize results
    print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
    means = grid_result.cv_results_['mean_test_score']
```

```
        stds = grid_result.cv_results_['std_test_score']
        params = grid_result.cv_results_['params']
        for mean, stdev, param in zip(means, stds, params):
            print("%f (%f) with: %r" % (mean, stdev, param))
        print(type(grid_result.best_params_))
        print(len(grid_result.best_params_))
        print(grid_result.best_params_.keys())
        epochs_var = grid_result.best_params_['epochs']
        batch_size_var = grid_result.best_params_['batch_size']

        model = Sequential()
        model.add(InputLayer((sen_Len,),dtype='int32'))
        e = Embedding(len(term_Index) + 1, emb_Dim, weights=[emb_Mat], input_length=sen_Len, trainable=False)
        model.add(e)
        model.add(Bidirectional(LSTM(60, return_sequences=True)))
        model.add(GlobalMaxPool1D())
        model.add(Dropout(0.1))
        #Must use relu first to avoid vanishing gradient (https://towardsdatascience.com/is-relu-after-sigmoid-bad-661fda45f7a2)
        #relu prevents vanishing gradient (https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks)
        model.add(Dense(158, activation='relu'))
        model.add(Dropout(0.1))
        #prevents blowing up activation
        model.add(Dense(2, activation='sigmoid'))

        model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics = ['acc'])
```

```
print(max_len)
history = model.fit(features_Train_Resampled, target_Train_Resampled, epochs = epochs_var, batch_size=batch_size_var, validation_split=0.20)
time_series_Val = np.asarray(time_series_Val)

for i,times in enumerate(time_series_Val):
    time_series_Val[i] = np.array(time_series_Val[i])
max_len_val = max([len(x) for x in time_series_Val])
for i,times in enumerate(time_series_Val):
    time_series_Val[i] = np.pad(times,(0,max_len_val-len(times)), 'constant')
time_series_val_Mat = np.zeros((len(time_series_Val),max(max_len,max_len_val)))
for i, times in enumerate(time_series_Val):
    for j, time in enumerate(time_series_Val[i]):
        time_series_val_Mat[i,j] = time
print(time_series_val_Mat)
print(time_series_val_Mat.shape)
combined_Val = np.concatenate([features_Val,time_series_val_Mat],axis=1)
combined_Val = np.concatenate([combined_Val,num_Norm_Val],axis=1)

target_Val = np.argmax(target_Val, axis=1)

predictions = model.predict(combined_Val)
predictions_bool = np.argmax(predictions, axis=1)
print(predictions_bool.shape)
print(classification_report(target_Val, predictions_bool,digits=3))
predictions_prob = model.predict_proba(combined_Val)
import pickle
```

```
        with
open('/local/home/bohdan/Fakenews_Classification/T_Test/BiLSTM_smote_proba
.pkl','wb') as f:
            pickle.dump(predictions_prob, f)
        target_Val = to_categorical(target_Val)
        auc = roc_auc_score(target_Val, predictions_prob)
        import sklearn.metrics as metrics
        y_pred = (predictions > 0.5)
        matrix = metrics.confusion_matrix(target_Val.argmax(axis=1),
y_pred.argmax(axis=1))
        micro =
(matrix[0,0]+matrix[1,1])/(matrix[0,0]+matrix[1,1]+matrix[0,1]+matrix[1,0])
        print('micro')
        print(micro)
        print('macro')
        macro =
(matrix[0,0]/(matrix[0,0]+matrix[1,0])+matrix[1,1]/(matrix[1,1]+matrix[1,0]))/2
        print(macro)
        print('auc')
        print(auc)
        import sklearn.metrics as metrics
        y_pred = (predictions > 0.5)
        matrix = metrics.confusion_matrix(target_Val.argmax(axis=1),
y_pred.argmax(axis=1))
        print(matrix)
        import matplotlib.pyplot as plt
        #%matplotlib inline
        val_Loss = history.history['val_loss']
        loss = history.history['loss']
        epochs = range(1, len(loss)+1)
```

```
plt.plot(epochs, loss, label='Training Loss')
plt.plot(epochs, val_Loss, label='Testing Loss')
plt.title('Training and Testing Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.savefig("loss_Function"+".png", bbox_inches='tight')
acc = history.history['acc']
val_Acc = history.history['val_acc']
plt.plot(epochs, acc, label='Training accuracy')
plt.plot(epochs, val_Acc, label='Testing Accuracy')
plt.title('Training and Testing Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend()
plt.savefig("accuracy"+".png", bbox_inches='tight')
print(matrix)

#coding: utf-8
import json
import matplotlib.pyplot as plt
from keras.layers import Bidirectional

from langdetect import detect
from  tqdm import tqdm
import nltk
nltk.download('stopwords')
from sklearn.metrics import roc_auc_score
import nltk
import re
```

```
from tqdm import tqdm_notebook
from nltk.corpus import stopwords

from tensorflow.keras import regularizers, initializers, optimizers, callbacks
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from keras.utils.np_utils import to_categorical
from tensorflow.keras.layers import *
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from keras.layers import LSTM
from keras.layers import Bidirectional
from keras.layers import GlobalMaxPool1D
from keras.layers import Dropout
from keras.layers import InputLayer
from sklearn.metrics import classification_report
import pandas as pd
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
import numpy as np
from imblearn.under_sampling import RandomUnderSampler
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
df = pd.read_pickle("/local/home/bohdan/Fakenews_Classification/Preprocessing/bigdata_preprocessed.pkl")
df['created_at'] = df['created_at'].astype(str)
df['followers'] = df['followers'].astype(str)
```

```
df['following'] = df['following'].astype(str)

cat_Cols = ['text', 'name_user','name_zero_user_mentions_entities', 'location_user', 'description_user', 'contributors_enabled_user', 'default_profile_image_user', 'default_profile_user', 'favorited', 'follow_request_sent_user', 'following_user', 'geo_enabled_user', 'has_extended_profile_user', 'id', 'id_str', 'id_str_user', 'id_str_zero_user_mentions_entities', 'id_user', 'id_zero_user_mentions_entities', 'is_quote_status', 'is_translation_enabled_user', 'is_translator_user', 'lang', 'location_user', 'name_user', 'notifications_user', 'possibly_sensitive', 'possibly_sensitive_appealable', 'profile_background_color_user', 'profile_background_tile_user', 'profile_link_color_user', 'profile_sidebar_border_color_user', 'profile_sidebar_fill_color_user', 'profile_text_color_user', 'profile_use_background_image_user', 'protected_user', 'retweeted', 'screen_name_user', 'screen_name_zero_user_mentions_entities', 'translator_type_user', 'truncated', 'verified_user']


cat_Features =df['text']
print(str(df['text']).encode('utf-8').decode('latin-1'))
for col in cat_Cols:
    df[col] = df[col].astype(str)

print(df['label'])
label = pd.get_dummies(df['label'])
label = np.array(label)
print(np.argmax(label, axis=1))

num_Cols = ['favorite_count', 'favourites_count_user', 'followers_count_user', 'friends_count_user', 'listed_count_user', 'retweet_count',
```

```
'statuses_count_user', 'zero_indices_zero_urls_entities',
'one_indices_zero_urls_entities', 'zero_indices_zero_urls_url_entities_user',
'one_indices_zero_urls_url_entities_user',
'zero_indices_zero_user_mentions_entities',
'one_indices_zero_user_mentions_entities']
        num_Features = []
        for idx in num_Cols:
            print(idx)
            num_Features.append(df[idx])
        print(type(num_Features[0]))

        import pandas as pd
        from sklearn import preprocessing

        for i,col in enumerate(num_Features):
            if i == 0:
                mat = col
            else:
                mat = pd.concat([mat, col], axis=1)

        x = mat.values #returns a numpy array
        min_max_scaler = preprocessing.MinMaxScaler(feature_range=(0,1))
        x_scaled = min_max_scaler.fit_transform(x)
        num_Norm = pd.DataFrame(x_scaled)

        for col in num_Norm.keys():
            num_Norm[col] = pd.cut(num_Norm[col], bins=3, labels=np.arange(3),
right=False)

        strings = []
```

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

for line in tqdm_notebook(cat_Features, total=df.shape[0]):
    line = line.replace("""","")
    line = line.replace("\[","")
    line = line.replace(":","")
    line = line.replace("-","")
    line = line.replace("_","")
    line = line.replace("+","")
    line = line.replace("?","")
    line = line.replace(",","")
    line = line.replace("\]","")
    line = line.replace(".","")
    line = line.replace("â€˜","")
    line = line.replace("â€™","")
    line = line.replace("@","")
    line = line.replace("!","")
    line = line.replace("$","")
    line = line.replace("=","")
    line = line.replace("(","")
    line = line.replace(")","")
    line = line.replace("/","")
    line = line.replace("\\","")
    line = line.replace("&","")
    line = line.replace("#","")
    line = re.sub('\d', '', line)
    line = line.split(' ')
    line = [w for w in line if not w in stop_words]
    line = str(line)
    line = str(line.strip())[1:-1].replace(' ', ' ')
```

```
    strings.append(line)

#encode text as numbers
tok_Len = 100000 # max number of words for tokenizer
tokenizer = Tokenizer(num_words=tok_Len)
tokenizer.fit_on_texts(strings)
sequences = tokenizer.texts_to_sequences(strings)
term_Index = tokenizer.word_index
print('Number of Terms:', len(term_Index))


sen_Len = 162 # max length of each sentences, including padding
tok_Features = pad_sequences(sequences, padding = 'post', maxlen = sen_Len-111)
print('Shape of tokenized features tensor:', tok_Features.shape)

indices = np.arange(tok_Features.shape[0])
np.random.shuffle(indices)
time_series = df['created_at_retweets']
time_series.reset_index(drop=True, inplace=True)
print(type(time_series))
time_series = time_series[indices]
tok_Features = tok_Features[indices]
labels = label[indices]
print(num_Norm)
print(num_Norm.shape)
print(type(num_Norm))

test_Perc = 0.2 #20% data used for testing
num_validation_samples = int(test_Perc*tok_Features.shape[0])
```

```
features_Train = tok_Features[: -num_validation_samples]
time_series_Train = time_series[: -num_validation_samples]
num_Norm_Train = num_Norm[: -num_validation_samples]
target_Train = labels[: -num_validation_samples]
features_Val = tok_Features[-num_validation_samples: ]
num_Norm_Val = num_Norm[-num_validation_samples: ]
time_series_Val = time_series[-num_validation_samples: ]
target_Val = labels[-num_validation_samples: ]
print('Number of records in each attribute:')
print('training: ', target_Train.sum(axis=0))
print('validation: ', target_Val.sum(axis=0))

emb_Dim = 100 # embedding dimensions for word vectors
glove = '/local/home/bohdan/Fakenews_Classification/BiLSTM/glove.6B.'+str(emb_Dim)+'d.txt'
emb_Ind = {}
f = open(glove, encoding='utf8')
print('Loading Glove \n')
for line in f:
    values = line.split()
    term = values[0]
    emb_Ind[term] = np.asarray(values[1:], dtype='float32')
f.close()
print("Done---\nMap terms to embedding---")

emb_Mat = np.random.random((len(term_Index) + 1, emb_Dim))
for term, i in term_Index.items():
    emb_Vec = emb_Ind.get(term)
    if emb_Vec is not None:
```

```
        emb_Mat[i] = emb_Vec
print("Done")
print('SHAPE EMB MAT')
print(emb_Mat.shape)
print(type(emb_Mat))
print('SHAPE TIME SERIES')
print(time_series_Train.shape)
print(type(time_series_Train))
maxLen = 0
maxIndex = 0

time_series_Train = np.asarray(time_series_Train)

for i,times in enumerate(time_series_Train):
    time_series_Train[i] = np.array(time_series_Train[i])
max_len = max([len(x) for x in time_series_Train])
for i,times in enumerate(time_series_Train):
    time_series_Train[i] = np.pad(times,(0,max_len-len(times)), 'constant')
time_series_Mat = np.zeros((len(time_series_Train),max_len))
for i, times in enumerate(time_series_Train):
    for j, time in enumerate(time_series_Train[i]):
        time_series_Mat[i,j] = time
features_Train = np.concatenate([features_Train,time_series_Mat],axis=1)

features_Train = np.concatenate([features_Train,num_Norm_Train],axis=1)

from imblearn.over_sampling import SMOTE
undersample = RandomUnderSampler()
print(target_Train)
print(target_Train.shape)
```

```
print(type(target_Train))
features_Train_Resampled, target_Train_Resampled = undersample.fit_resample(features_Train, target_Train)
from keras.utils import to_categorical
target_Train_Resampled = to_categorical(target_Train_Resampled)
print(target_Train.shape)
print(type(target_Train))
print(target_Train)
print('FEATURES TRAIN')
def create_model():
# create model
   model = Sequential()
   model.add(InputLayer((sen_Len,),dtype='int32'))
   e = Embedding(len(term_Index) + 1, emb_Dim, weights=[emb_Mat], input_length=sen_Len, trainable=False)
   print('SEN LEN')
   print(sen_Len)
   print('TERM INDEX')
   print(len(term_Index))
   model.add(e)
   model.add(Bidirectional(LSTM(60, return_sequences=True)))
   model.add(GlobalMaxPool1D())
   model.add(Dropout(0.1))
   model.add(Dense(158, activation='relu'))
   model.add(Dropout(0.1))
   model.add(Dense(2, activation='sigmoid'))

   model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics = ['acc'])
   return model
```

```
gridmodel = KerasClassifier(build_fn=create_model,epochs=10, batch_size=5, verbose=0)

batch_size = [8,16,32,64,128]
epochs = [1,2,3,4,5]
param_grid = dict(batch_size=batch_size, epochs=epochs)
grid = GridSearchCV(estimator=gridmodel, param_grid=param_grid, cv=3)
grid_result = grid.fit(features_Train_Resampled, target_Train_Resampled)
#summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
print(type(grid_result.best_params_))
print(len(grid_result.best_params_))
print(grid_result.best_params_.keys())
epochs_var = grid_result.best_params_['epochs']
batch_size_var = grid_result.best_params_['batch_size']

model = Sequential()
model.add(InputLayer((sen_Len,),dtype='int32'))
e = Embedding(len(term_Index) + 1, emb_Dim, weights=[emb_Mat], input_length=sen_Len, trainable=False)
print('SEN LEN')
print(sen_Len)
print('TERM INDEX')
```

```
print(len(term_Index))
model.add(e)
model.add(Bidirectional(LSTM(60, return_sequences=True)))
model.add(GlobalMaxPool1D())
model.add(Dropout(0.1))
model.add(Dense(158, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(2, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics = ['acc'])

print(max_len)
history = model.fit(features_Train_Resampled, target_Train_Resampled, epochs = epochs_var, batch_size=batch_size_var, validation_split=0.20)
time_series_Val = np.asarray(time_series_Val)

for i,times in enumerate(time_series_Val):
    time_series_Val[i] = np.array(time_series_Val[i])
max_len_val = max([len(x) for x in time_series_Val])
for i,times in enumerate(time_series_Val):
    time_series_Val[i] = np.pad(times,(0,max_len_val-len(times)), 'constant')
time_series_val_Mat = np.zeros((len(time_series_Val),max(max_len,max_len_val)))
for i, times in enumerate(time_series_Val):
    for j, time in enumerate(time_series_Val[i]):
        time_series_val_Mat[i,j] = time
print(time_series_val_Mat)
print(time_series_val_Mat.shape)
```

```
        combined_Val =
np.concatenate([features_Val,time_series_val_Mat],axis=1)
        combined_Val = np.concatenate([combined_Val,num_Norm_Val],axis=1)

        target_Val = np.argmax(target_Val, axis=1)

        predictions = model.predict(combined_Val)
        predictions_bool = np.argmax(predictions, axis=1)
        print(predictions_bool.shape)
        print(classification_report(target_Val, predictions_bool,digits=3))
        predictions_prob = model.predict_proba(combined_Val)
        import pickle
        with
open('/local/home/bohdan/Fakenews_Classification/T_Test/BiLSTM_under_proba
.pkl','wb') as f:
            pickle.dump(predictions_prob, f)
        target_Val = to_categorical(target_Val)
        auc = roc_auc_score(target_Val, predictions_prob)
        import sklearn.metrics as metrics
        y_pred = (predictions > 0.5)
        matrix = metrics.confusion_matrix(target_Val.argmax(axis=1),
y_pred.argmax(axis=1))
        micro =
(matrix[0,0]+matrix[1,1])/(matrix[0,0]+matrix[1,1]+matrix[0,1]+matrix[1,0])
        print('micro')
        print(micro)
        print('macro')
        macro =
(matrix[0,0]/(matrix[0,0]+matrix[1,0])+matrix[1,1]/(matrix[1,1]+matrix[1,0]))/2
        print(macro)
```

```
print('auc')
print(auc)
import sklearn.metrics as metrics
y_pred = (predictions > 0.5)
matrix = metrics.confusion_matrix(target_Val.argmax(axis=1), y_pred.argmax(axis=1))
print(matrix)
import matplotlib.pyplot as plt
#%matplotlib inline
val_Loss = history.history['val_loss']
loss = history.history['loss']
epochs = range(1, len(loss)+1)
plt.plot(epochs, loss, label='Training Loss')
plt.plot(epochs, val_Loss, label='Testing Loss')
plt.title('Training and Testing Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.savefig("loss_Function"+".png", bbox_inches='tight')
acc = history.history['acc']
val_Acc = history.history['val_acc']
plt.plot(epochs, acc, label='Training accuracy')
plt.plot(epochs, val_Acc, label='Testing Accuracy')
plt.title('Training and Testing Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend()
plt.savefig("accuracy"+".png", bbox_inches='tight')
print(matrix)
import sys
```

```
import argparse

sys.path.append('/local/home/bohdan/Fakenews_Classification/Preprocessing')
sys.path.append('/local/home/bohdan/Fakenews_Classification/BiLSTM')
sys.path.append('/local/home/bohdan/Fakenews_Classification/CNN')
sys.path.append('/local/home/bohdan/Fakenews_Classification/BiLSTM_Khan')
sys.path.append('/local/home/bohdan/Fakenews_Classification/C-GRU_Liu')
sys.path.append('/local/home/bohdan/Fakenews_Classification/C-LSTM')
sys.path.append('/local/home/bohdan/Fakenews_Classification/C-LSTM_Khan')
sys.path.append('/local/home/bohdan/Fakenews_Classification/CNN_LSTM')
sys.path.append('/local/home/bohdan/Fakenews_Classification/NB')
sys.path.append('/local/home/bohdan/Fakenews_Classification/SVM')
sys.path.append('/local/home/bohdan/Fakenews_Classification/T_Test')
sys.path.append('/local/home/bohdan/Fakenews_Classification/Visualization')

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("-p", type=str, default=1, metavar='PREPROCESSING |true|false|')
    parser.add_argument("-v", type=str, default=2, metavar='VISUALIZATION |plot|tfidf|timecascade|none|')
    parser.add_argument("-c", type=str, default=3, metavar='CLASSIFIER |bilstm|bilstmkhan|cgruliu|clstm|clstmkhan|cnn|cnnlstm|nb|svm|')
```

```
        parser.add_argument("-s", type=str, default=4, metavar='SAMPLING
|none|smote|undersampling|')
        parser.add_argument("-t", type=str, default=5, metavar='T-TEST |t-
test|none|')
        args = parser.parse_args()
        if len(sys.argv)==1:
            print('See help flag, -h, for help on usage.')
            exit()
        print('')
        try:
            if args.p=='true':
                print('Running with preprocessing.')
                import Preprocessing1
                import Preprocessing2
                import Preprocessing3
                import Preprocessing4
                import Preprocessing5
                import Preprocessing6
                import Preprocessing7
                import Preprocessing8
                import Preprocessing9
            elif args.p=='false':
                print('Running without preprocessing.')
            else:
                print('Running without preprocessing.')
        except FileNotFoundError:
            print('Must run with preprocessing the first time to generate dataset.')
            exit()
        try:
            if args.v=='plot':
```

```
        print('Running with preprocessing.')
        import Plot1
        import Plot2
    elif args.v=='tfidf':
        import TFIDF1
        import TFIDF2
    elif args.v=='timecascade':
        import timecascade
    elif args.p=='none':
        print('Running without visualization.')
    else:
        print('Running without visualization.')
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
    if args.c=='bilstm':
        if args.s=='none':
            import BiLSTM
        elif args.s=='smote':
            import BiLSTM_Oversampling
        elif args.s=='undersampling':
            import BiLSTM_Undersampling
        else:
            print('See help flag, -h, for usage.')
            exit()
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
```

```
    if args.c=='bilstmkhan':
        if args.s=='none':
            import LSTM_Khan
        elif args.s=='smote':
            import LSTM_Khan_smote
        elif args.s=='undersampling':
            import LSTM_Khan_under
        else:
            print('See help flag, -h, for usage.')
            exit()
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
    if args.c=='cgruliu':
        if args.s=='none':
            import Liu
        elif args.s=='smote':
            import Liu_smote
        elif args.s=='undersampling':
            import Liu_under
        else:
            print('See help flag, -h, for usage.')
            exit()
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
    if args.c=='clstm':
        if args.s=='none':
```

```
            import C_LSTM_no_sampling
        elif args.s=='smote':
            import C_LSTM_smote_grid
        elif args.s=='undersampling':
            import C_LSTM_under_grid
        else:
            print('See help flag, -h, for usage.')
            exit()
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
    if args.c=='clstmkhan':
        if args.s=='none':
            import C_LSTM_Khan
        elif args.s=='smote':
            import C_LSTM_Khan_smote
        elif args.s=='undersampling':
            import C_LSTM_Khan_under
        else:
            print('See help flag, -h, for usage.')
            exit()
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
    if args.c=='cnn':
        if args.s=='none':
            import CNN_no_sampling
        elif args.s=='smote':
```

```
            import CNN_smote_grid
        elif args.s=='undersampling':
            import CNN_under_grid
        else:
            print('See help flag, -h, for usage.')
            exit()
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
    if args.c=='cnnlstm':
        if args.s=='none':
            import CNN_LSTM
        elif args.s=='smote':
            import CNN_LSTM_smote
        elif args.s=='undersampling':
            import CNN_LSTM_under
        else:
            print('See help flag, -h, for usage.')
            exit()
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
    if args.c=='nb':
        if args.s=='none':
            import NB
        elif args.s=='smote':
            import NB_smote
        elif args.s=='undersampling':
```

```
            import NB_under
        else:
            print('See help flag, -h, for usage.')
            exit()
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
    if args.c=='svm':
        if args.s=='none':
            import SVM
        elif args.s=='smote':
            import SVM_smote
        elif args.s=='undersampling':
            import SVM_under
        else:
            print('See help flag, -h, for usage.')
            exit()
except IndexError:
    print('See help flag, -h, for usage.')
    exit()
try:
    if args.t=='t-test':
        if args.s=='none':
            import T_Test
        elif args.s=='smote':
            import T_Test_smote
        elif args.s=='undersampling':
            import T_Test_under
    elif args.t=='none':
```

```
            print('Running without T-Test')
        except IndexError:
            print('See help flag, -h, for usage.')
            exit()
if __name__ == "__main__":
    main()
```