ABSTRACT

Social networking has become popular in recent years. The vast majority of people are linked to social media and rely on it for news, communication, entertainment, and other purposes. On social media, misinformation, false facts, and conspiracy theories may swiftly spread, encouraging readers to believe what they see. On Twitter, for example, a "trend" is a term, phrase, or issue that attracts a lot of attention. Retweets help to increase the number of people who see a tweet by increasing the number of people who view it. To increase the popularity of a trending subject, new tweets with hashtags relating to it are added to a list of trending tweets. This study provides a mechanism for categorising retweet accounts as fraudulent or authentic based on the value of the original tweet. It would make advantage of the temporal and linguistic characteristics of the retweet. We used NLP and the point process to create a feature vector, which we then fed into a classifier that used Neural networks, Naive Bayes, SVM, and KNN, and analysed the results. In the field of natural language processing, deep learning approaches have lately been discovered to be effective. The findings show that the suggested classification system, which uses LDA and a fully connected neural network model, performs best on the validation dataset, with an accuracy of 94.11 percent.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	**INTRODUCTION**	01
2.	**PROBLEM DEFINITION**	03
3.	**LITERATURE SURVEY**	04
	3.1. Hawkeseye: Detecting fake retweeters using Hawkes process and topic modelling	04
	3.2. Weakly supervised learning for fake news detection on Twitter	04
	3.3. A neural network-based ensemble approach for spam detection in Twitter	05
	3.4. Automatically identifying fake news in popular twitter threads	05
	3.5. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter	06
	3.6. Isolating rumours using sentiment analysis	06
	3.7. Detection of spam-posting accounts on Twitter	07
	3.8. HashJacker-detection and analysis of hashtag hijacking on Twitter	07
	3.9. Exploiting temporal patterns for botnet detection on twitter & quot	07
	3.10. Fake News Analysis Modelling Using Quote Retweet	08
	3.11. Text Mining of Twitter data using a Latent Dirichlet allocation Topic model and Sentiment analysis	09
	3.12. Fake News Identification on Twitter with Hybrid CNN and RNN Models	10
	3.13. Summary	10
4.	**PROJECT REQUIREMENTS SPECIFICATION**	12
	4.1. Product Perspective	12
	4.2. User Classes and Characteristics	12
	4.3. Operating Environment	12
	4.4. General Constraints, Assumptions and Dependencies	13
	4.5. Risks	14

4.6. Functional Requirements **14**

4.7. External Interface Requirements **14**

4.8. Communication Interfaces **15**

4.9. Non-Functional Requirements **16**

5. SYSTEM DESIGN **18**

5.1. Model Architecture **18**

5.2. Activity Diagram **19**

6. PROPOSED METHODOLOGY **21**

6.1. Data Extraction **21**

6.2. Data Cleaning and Pre-processing **21**

6.3. Hawkes Process **21**

6.4. Topic Modelling using LDA **22**

6.5. Classifiers **23**

6.6. Fully Connected Neural Network Model **24**

6.7. Bag of Words Model **25**

6.8. Bi-LSTM Model **26**

6.9. Design Details **27**

7. IMPLEMENTATION AND PSEUDOCODE **29**

7.1. Data Collection and Extraction **29**

7.2. Data Cleaning **31**

7.3. Data Visualization **32**

7.4. Hawkes Process **35**

7.5. LDA **36**

7.6. Neural Network Model **37**

8. RESULTS AND DISCUSSION **39**

9. CONCLUSION AND FUTURE WORK **42**

REFERENCES/BIBLIOGRAPHY **43**

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS **44**

LIST OF FIGURES

Figure no.	Figure name	Page no.
5.1.1	Model Architecture	18
5.2.1	Activity Diagram	19
6.3.1	Hawkes Process	22
6.4.1	Latent Dirichlet Allocation	23
6.6.1	Fully connected neural network	24
6.8.1	Bi-LSTM Model	26
7.3.1.1	Pie chart verified vs non-verified accounts	32
7.3.2.1	Word cloud for real users	33
7.3.2.2	Word cloud for fake users	33
7.3.3.1	No. of Tweets vs Users	34

LIST OF TABLES

Table No.	Title	Page No.
6.6.2	Neural Network Layers	25
8.1	Accuracy of Hawkes Model with Classifiers	39
8.2	Accuracy of LDA Model with Classifiers	40
8.3	Accuracy of Combined Model with Classifiers	40

CHAPTER-1

INTRODUCTION

Through the development of virtual networks and communities, social media is a computer-based technology that facilitates the exchange of information, opinions, emotions and knowledge. By its very nature, social networking is Internet-based, allowing users to instantly share content via electronic methods. The material contains personal information, records, photos, and photographs. Users interact with social media using web-based software or applications on a computer, tablet, or smartphone. The number of active social media users worldwide will reach 3.96 billion in 2020, up 10.9 percent from 3.57 billion in 2018. There were only 2.16 billion users in 2016, an increase of 89.67 percent in just five years. So, this tells us how people are so dependent on the usage of social media these days. Social media particularly twitter is a platform where misinformation, fabricated information and conspiracy theories can spread easily. It facilitates the rapid spread of news whether a story is completely bogus or real. It has been found that a proper system does not exist since it is a research topic. It encourages the rapid dissemination of news, regardless of whether it is absolutely false or true. Retweets are essential for a tweet's popularity to grow.

As a subject becomes prominent, new tweets with hashtags linked to it are added to a list of trending tweets, strengthening the trend's popularity. By sharing the tweet with anyone who looks for the trend, rather than only his or her supporters, Twitter users may reach out to their intended audience. A trending subject on Twitter can be identified by the use of a number of similar hashtags, which can be used together or separately through several tweets. This can be supported by a large number of phoney or fake retweeters. This feature may be strongly influenced by a large number of bogus retweeters. Fake retweeters are described as a Twitter account that retweets spam messages, retweets an unusually large number of tweets in a short span of time, or uses a trending twitter hashtag to encourage activities unrelated to the subject of conversation. We propose to create a new classifier that categorises all retweeters as false or not depending on the original tweet's importance. It makes use of temporal and textual data from retweets. Temporal characteristics (i.e., Time of tweet, Twitter-User-Details, number of tweets per time period etc.) are modelled using Hawkes process and we use Topic modelling techniques for modelling of textual characteristics (i.e. Textual information in tweet or retweet) in the data. Temporal features involve time dependent features like retweet, share, likes etc. Textual content involves actual text content in the tweet or retweet misuse of #Hashtag etc. The relevant temporal and textual features extracted from the model will be used in further classification of retweets as fake or not fake. We could say that the scope of this research is applicable to other social media apps with related reposting features, such as Facebook.

CHAPTER-2

PROBLEM STATEMENT

Social media is a diverse platform where vast amount of information or news could be spread across worldwide in a short interval of time. This information could be real, true to the facts or a chain of false and misleading information. This false news or information often spreads faster than real news. Twitter is one of such platforms where this false news could be spread in order to create chaos, to hurt religious sentiments etc. Twitter has a feature named ‘retweet’ where someone reposts or forwards a post to their own Twitter followers. Retweets are typically credited to their original authors, incentivizing users to make shareable content that expands their Twitter footprint. Retweeting is a mostly used method to boost the reach or popularity of a Tweet.

A considerable number of fraudulent retweeters contribute to this by exploiting a loophole and abusing Twitter's services for their own gain. Spam tweets can be characterized as tweets made by fake Twitter accounts with political motivations, automatically generated material, and meaningless content. Fake retweeters are Twitter users who retweet spam tweets, retweet a very large number of tweets in a short burst of time, or “misuse a trending hashtag to promote events irrelevant to the topic of discussion”. This fake retweeters could be paid users or bots. Bots are mainly used to spread false news and are sophistically designed to dodge any kind of models hence they are difficult to detect so we focus on detecting fake human retweeters.

The main aim of our project would be to classify all the retweet accounts as fake or not fake based on their relevance to the original tweet. We intend to build a classifier model which can classify a retweeter as fake or real based on the textual data as well the temporal data i.e., creation timeline of the user tweet. We mainly focus on account centric information of the users, their previous tweet history etc to classify them rather than their social follower. The model would be trained for limited number of users data but should be able classify any given twitter user as fake or not based on their tweets data.

CHAPTER-3

LITERATURE SURVEY

3.1 Hawkeseye: Detecting fake retweeters using Hawkes process and topic modelling.

Dutta et al. [1], proposed HawkesEye model, which is "a *novel retweeter classification model"* that takes temporal information into account by making use of Hawkes processes, and utilizes Latent Dirichlet Allocation (LDA) based topic modelling to represent the textual content in the tweets. He has used Hawkes estimation, LDA and KNN concepts in the model. Unlike other supervised machine learning models, this model did not require manual feature extraction and also, the model took into consideration the account-centric information of a Twitter user for classification purposes.
There are a few limitations to this paper, including data skewness, which has led to less efficient outcome, and Twitter API constraints while extracting large real time tweets.

3.2 Weakly supervised learning for fake news detection on Twitter.

To classify news as fake or not, Helmstetter et al. [2], proposed a poorly supervised approach. The model extracted features from tweets using LDA and HDA, and then used SentiWordNet to compute the polarity of tweets based on the ratio of positive, negative, and neutral words. The model utilizes two ensemble methods i.e., Random Forest and XGBoost. Since huge training corpora were required to train and evaluate the model for binary classification, manually annotating tweets as fake or not was an expensive and time-consuming task. As a result, they obtained a large dataset in which tweets were labelled as coming from a trustworthy or untrustworthy source, and used this dataset to train the classifier. The main drawback was that, labelling tweets as trustworthy or untrustworthy by their source resulted in a huge, noisy dataset, which might have influenced their F1 score to be very low.

3.3 A neural network-based ensemble approach for spam detection in Twitter.

Madisetty et al. [3], proposed a model that determines whether a tweet is spam or not. The ensemble model employed five CNNs that utilized different word embeddings like Glove and Word2vec of various dimensions, as well as one feature-based model that utilized user-based, content-based, and n-gram features. Spammers have tried to figure out what the detection system looks for in a spam tweet, so that they can adjust the form of spam tweets they send, such that the detected features aren't present. These forms of spam are difficult to detect for programmes that only incorporate feature-based methods. The used algorithm, on the other hand, incorporated both feature-based and deep-learning-based approaches. Even if spammers tried to trick the detection mechanism, the approach was reliable enough to catch some of the spam tweets.

The proposed method outperformed all other approaches, when evaluated on 1KS10KN (imbalanced data) and HSpam (balanced) data sets. When extended to a vast number of unseen tweets, the model trained even though was trained with a limited number of examples, demonstrated good results. But when compared to deep learning methods for only the HSpam14 dataset, which is a balanced dataset, feature-based methods performed poorly.

3.4 Automatically identifying fake news in popular twitter threads.

Buntain et al. [4], proposed a method to identify fake news in popular Tweet threads, the model used Natural Language Toolkit (NLTK) and neural networks. This gave useful and less expensive means to classify true and false stories on Twitter rapidly. Accuracy for this model is quite low (CREDBANK = 65.29%, PHEME = 36.52%). Also, the dataset chosen was not much accurate, because it was done solely by human annotators.

3.5 Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter.

Lukasik et al. [5], developed a Hawkes Process model for time sensitive sequence classification. This method used Hawkes Process, Language Model, KNN, Majority vote, and many other benchmark model. They showed that Hawkes Process is a successful technique that beats a variety of strong benchmark approaches by providing an informative outcome based on temporal dynamics to the multinomial language model that utilised four Twitter datasets focused on rumour stance classification of tweets. Their research emphasised the importance of using the temporal knowledge contained in tweets, which, when combined with the textual material, provided useful information for the model to function well. The key problem was that the approximation used in HP-Approx, violated the Hawkes Process mutual-excitation property.

3.6 Isolating rumours using sentiment analysis.

Sivasangari et al. [6], proposed a mechanism that isolated rumours using sentiment analysis.
The dataset was collected using Twitter API and twitter scraper, a large number of tweets and metadata were scraped using Twitter scraper. Then for the scraped data, a threshold value was determined based on the negative polarity of popular tweets; once the tweets varied from the threshold state, the tweets were automatically classified as a "rumour" or "non-rumour” with the help of a sentiment classifier. This model successfully demonstrated identification of fake retweets using Temporal Analysis. The model also detected false evidence or rumours based on “VADER sentiment analysis”.
The model achieved a high accuracy of 90% on the dataset of fmr. Tamil Nadu Chief Minister Jayalalitha death case. The main drawback of this model was that when something was not published by a verified news agency, there was no option for classifying those rumours manually, only the tweets from authenticated news accounts were considered as genuine information, all other tweets came under the generic tweet category.

3.7 Detection of spam-posting accounts on Twitter.

Inuwa-Dutse et al. [7] in 2018 proposed a mechanism to detect spam accounts on Twitter. The model used Honeypot dataset, which consisted of the automatically annotated and as well as the manually annotated data about the spam users. The model applied Latent Semantic Analysis (LSA) to the Honeypot dataset that yielded results for both querying Twitter and validating spam. Feature selection was done and then used performance metrics for performance comparison across different classification models. This model was not reliant on past tweets, which are often inaccessible on Twitter, made it ideal for real-time spam detection. But spammers were able to adapt quickly in order to evade detection systems; and as a consequence, this model became outdated and unreliable in performing against spammers' new tactics.

3.8 HashJacker-detection and analysis of hashtag hijacking on Twitter.

Jain et al. [8] proposed the Hashjacker Detection model that identified cases of Hashtag hijacking in Twitter. A dataset was selected for a particular #Hashtag, the TF-IDF score was calculated for all the words in the tweets. Then the top scoring words dumped into a dictionary. Each tweet was given a score according to its matched word. The low scoring tweets were the filtered for using wrong hashtags. Though the model performed well for trending #Hashtags that already had a lot of tweets, it failed to achieve good results for new hashtags that was formed recently, as these new hashtags had very less TF-IDF Score and also accounting for the twitter API restrictions on extracting new tweets on a large scale.

3.9 Exploiting temporal patterns for botnet detection on twitter & quot.

Michele et al. [9], proposed Retweet-Buster (RTbust), an automated unsupervised feature extraction-based retweeting bot identification technique. Users were described by latent feature vectors calculated by the LSTM encoder, and they used density-based clustering to look for typical retweeting patterns. Huge groups of users were identified and the model discovered an organised and orchestrated network of accounts, which might have been a retweeting botnet. They labelled bots on some of those accounts that ended up clustered after performing the density-based clustering stage. They marked valid for all those accounts that HDBSCAN treats as noise (i.e., those that aren't clustered).

With F1score = 0.87, RTbust achieved outstanding detection results. When RTbust was implemented to a huge dataset of retweets, two previously undisclosed active botnets containing hundreds of accounts were discovered. RTT plots were also used to clarify the decisions for black-box bot detectors, which made AI more understandable and meaningful. The downside was that, when social bots grow to elude detection techniques, they could eventually elude the proposed model with ease.

3.10 Fake News Analysis Modelling Using Quote Retweet.

Yonghun Jang et al. [10], proposed a method to analyse fake news using Quote Retweet. First the data was collected from Kaggle and twitter API, and then pre-processed. Quote Re-Tweets (QRTs) were extracted, then features were extracted and passed onto a Fake-news classifier that used Neural-Networks. The model was trained and validated on the best features; the neural network-based fake news classifier was used to assess efficiency of the model. This model used a different approach for gathering more data and identifying the best features by extracting Quote-Retweets, to classify false information or news on Twitter with a high accuracy of about 83.05%. When compared to a classification model based solely on traditional Tweet data, the ***'Quote-Retweet'*** based model had greater classification accuracy for fake news, actual news, and overall, respectively, of 4.57 percent, 10.51 percent, and 9.79 percent. For false news, the classification model that extracted and used the best-features had an 8.48 percent better classification accuracy. The accuracy of overall news (real and fake) was only 76 percent, and user response and emotion data were of poor quality.

3.11 Text mining of Twitter data using a latent Dirichlet allocation topic model and sentiment analysis.

Yang et al. [11], in 2018 came up with a mechanism to text-mine Twitter data using LDA topic modelling and sentiment analysis. The model was made to analyse audience evaluations of the film ‘Thor Ragnarok’ on the day it was released, researchers used a latent Dirichlet allocation topic model and sentiment analysis on Twitter data. In the experiment, '***FilterStream***' was utilised to retrieve twitter data. 'streamR' provided this function by connecting to the Twitter Streaming API and opening a stream to record results depending on certain constraints, such as keywords and localities. The data was pre-processed. For sentiment analysis, the syuzhet package was utilised. From the text, this package extracted sentiment and sentiment-derived storey arcs. In this study, sentiment analysis was performed using the "nrc" lexicon. The 'NRC Emotion Lexicon' is a collection of English words that are related with eight different emotions, including anger, fear, sadness, disgust, surprise, anticipation, trust, and joy. There are also two types of feelings: positive and negative. The subjects buried in these tweets were analysed using LDA topic modelling. The "bag-of-words" assumption was used in the topics. This model represented the "presence" of terms in a text file. It was like a "bag" where the syntax was not taken into account. In the bag-of-words paradigm, the only thing that mattered was if a word existed in the text, not where or how it appeared. The number of topics was set at K = 15.

Results showed that the total sentiment of the data could be positive because some high-frequency words in the corpus were positive. The percentage of the total number of the positive terms in the collected data was 61.5%, which referred to an overall positive attitude towards this movie.

This experiment used basic English text. During the data pre-processing step, emoji symbols were deleted. Emoji are frequently used by Twitter users in their posts. Emoji should have been employed because they were a good source for sentiment analysis. They could have explored and compared results using other models such as pLSA, lda2vec. The author was unable to make a precise comparison between the movie review website findings and expected results due to limitations in the experimental data. The experimental data was only collected for 24 hours, and it was only natural language text that was used.

3.12 Fake News Identification on Twitter with Hybrid CNN and RNN Models

Oluwaseun Ajao et al. [12], proposed a model for identifying fake news on Twitter using hybrid CNN and RNN models. This approach suggested employing a hybrid deep learning model that combined LSTM and CNN models to automatically identify features inside Twitter posts without prior knowledge of the subject area or topic of debate. It also used text and picture analysis to determine and categorise fake news posts on Twitter. Using a combination of CNN and LSTM recurrent NN models, this architecture identified and classified false news messages from Twitter Tweets. With 74 percent accuracy and an FMeasure of 39.7%, the Long Short-Term Memory - Convolution Neural Network hybrid model outperformed the dropout regularisation model. The proposed work using a deep learning approach achieved 82% accuracy. These models that used Deep learning allowed for selecting the best features automatically, which allowed for the automatic identification of dependencies between words in fake tweets without manually defining them in the network. The main drawbacks of this model are that it needs more training data, hence faster and accurate classification of fake news is not possible.
The dilemma of weight change over time was a stumbling block for Rnn. The hybrid model has lesser accuracy (74%) than plain LSTM due to insufficient data for training. It requires a lot of computational resources for deep learning.

3.13 Summary

The models described in the survey [1-12] had accuracy levels ranging from 35% to 95%. In most of the papers, the target variable in the dataset has been manually annotated as fake or genuine for training and testing the model. The majority of the datasets were lower in size due to the fact that they were largely annotated by humans. Most papers place a strong emphasis on combining temporal and textual content, which adds up to be extremely valuable. The model's accuracy will be quite low if it is trained on a specific subject dataset. This happened in one of the abovementioned papers on different subject datasets (4th). For training the model, they developed a custom dataset and individually compiled retweeters as false or real. This is a one-of-a-kind dataset that concentrates on a specific yet popular issue and provides a complete chronology of each user during a certain time period. The users were restricted to only test the model by providing their data as input to the model to avoid any corruption or manipulation of the model. To tackle Bot accounts, they were identified based on the timeline of the tweets (bots are programmed to retweet an existing post many times to gain popularity). These insights assisted us in gaining sufficient information to move forward.

CHAPTER-4

PROJECT REQUIREMENTS SPECIFICATION

4.1 Product Perspective:

Social media is a platform where misinformation, fabricated information and conspiracy theories can spread easily. It facilitates the rapid spread of news whether a story is completely bogus or real. It has been found that a proper system does not exist since it is a research topic. The proposed model will be able to classify a group of tweets that are retweeted many times given as input by the user as fake or genuine based on their temporal and textual characteristics.

4.2 User Classes and Characteristics:

The user class and their characteristics are as follows:

- Users (or) Testers: A user is only limited to test the model based on the data provided as input. Based on the provided input appropriate results such as accuracy, F1-score etc will be given as output.
- Model-Trainers: Only the developers of the model will be able to train it to avoid any sort of manipulation, corruption of the model.

4.3 Operating Environment:

Since it is a Machine Learning project it can run on any modern CPUs which has at least 4 cores, 4 threads and 3.5GHz Clock speed. In order to increase the training speed of a model CUDA enabled Nvidia GPU can be used.

4.4 General Constraints, Assumptions and Dependencies:

4.4.1 Regulatory policies:

Since, we are using Twitter API we need to follow their terms and conditions. Due to restrictions in Twitter API, we are not able access complete tweets dataset.

4.4.2 Hardware limitations:

i. Training the model takes time depending upon the hardware we use. Trains faster when we use CUDA enabled Nvidia GPUs.
ii. Minimum storage required to store the dataset.

4.4.3 Limitations:

Bot developers modify the attributes of their accounts as soon as new detection techniques are implemented, allowing them to avoid detection. This "never-ending battle" has resulted in a situation in which social bots have been so complexly constructed that they are nearly indistinguishable from real user accounts. As a result, in the future, bots may be able to dodge the model.

4.4.4 Safety and security consideration**:**

i. Twitter API must not be misused since it is issued for educational and development purposes only.
ii. Not all can train the model but they can use only trained model for getting specified output from given tweet input. Training a model is allowed to administrators only.
iii. As per Twitter API Terms and Conditions following things should not be mis-used
    - Sensitive information (i.e., Political affiliation, Racial, Religious, Trade union membership)
    - Redistribution of Twitter content
    - Multiple applications
    - Automation,
    - spam, and auto-responses
    - Surveillance, privacy, and user protection

4.5 Risks:

i. Classifying the data to be genuine or fake by the annotators may not be accurate which may lead to errors in data.

ii. The proposed model may not be 100% accurate in classifying the tweets to be fake or genuine.

4.6 Functional Requirements:

Data is collected from Twitter API and necessary data Pre-processing steps are done. Then we manually add the target variable column which specifies an account is fake or real. The dataset is split into training and test sets, then data is given to the model for feature extraction. In this phase two features are extracted, i.e., temporal features which are done by Machine Learning techniques and textual features using Topic modelling after training the model test dataset is used to check the accuracy of the model. After that for given input model predicts or classifies account as fake or real or any #Hashtag is misused.

4.7 External Interface Requirements:

4.7.1 User Interfaces:

User interface will be in CLI (Command Line Interface). Sometime GUI required to see some text file. Training the model will take a lot of time after training of model input is converted to output in a very short amount of delay (computational delays). Error may happen due to a model that is not 100% accurate all the time so in very rare cases it will produce error or mis-classified output. Choosing different type of data (different #Hashtags, time period, area) will lead to change in the dataset therefore the model needs to be re-trained for that dataset.

4.7.2 Hardware Requirements:

i. Any Intel (7th gen or higher) or AMD (2nd gen or higher) processors with at least 3.5 GHz base lock

ii. Nvidia CUDA enabled GPU (any GTX and RTX series) will decrease the training time (optional).

4.7.3 Software Requirements:

i. Python

- Version: 3.7 or higher
- Operating Systems: Ubuntu 16.04 or higher, Windows 7 or higher, Mac OS 10 or higher
- Tools and Libraries (Open-source):
    - Tensor flow 2.x
    - Pandas
    - Numpy
    - Sklearn
    - Requests
    - Tweepy
    - Pickle
    - Gensim
    - NLTK

4.8 Communication Interfaces:

HTTP or HTTPS protocol(s) will be used since we are using the Twitter API for fetching tweets data. The most common request limit interval is fifteen minutes. If an endpoint has a rate limit of 900 requests/15-minutes, then up to 900 requests over any 15-minute interval is allowed.

4.9 Non-Functional Requirements:

4.9.1 Performance Requirement:

i. Reliability: Project is dependent on a dataset extracted from Twitter API and performs best on moderately powerful computers.

ii. Robustness: Model we built is capable of classifying the input correctly with acceptable minimal errors.

iii. Availability: Since our model is a freely available open-source model, it can be used by any individual.

iv. Accuracy: We intend to make our model achieve a higher accuracy compared to other previous works by at least 2-4%, thus having a significant improvement over the previous ones.

4.9.2 Safety Requirements:

i. Twitter API must not be misused since it is issued for educational and development purposes only.

ii. Since the proposed system neither stores any user's information nor their testing data hence, they need not worry about any leakage of their information or corruption of the testing data.

4.9.3 Security Requirements:

i. Data poisoning: Since the model depends on the data for learning purposes, hence it should not be corrupted or poisoned, otherwise it could bring the system down to its knees.

ii. Limit probing**:** Users are restricted by testing the system multiple times, since if it were an attacker then it could effectively reduce the rate at which they can devise harmful payloads.

4.9.4 Other Requirements:

i. Maintainability: classification is not 100% accurate some of the accounts may be mis-classified so, those need to be corrected manually.

ii. Portability: Since, we are using pickle a python library model can be portable, therefore re-training the complete dataset is not required.

CHAPTER-5

SYSTEM DESIGN

5.1 Model Architecture:

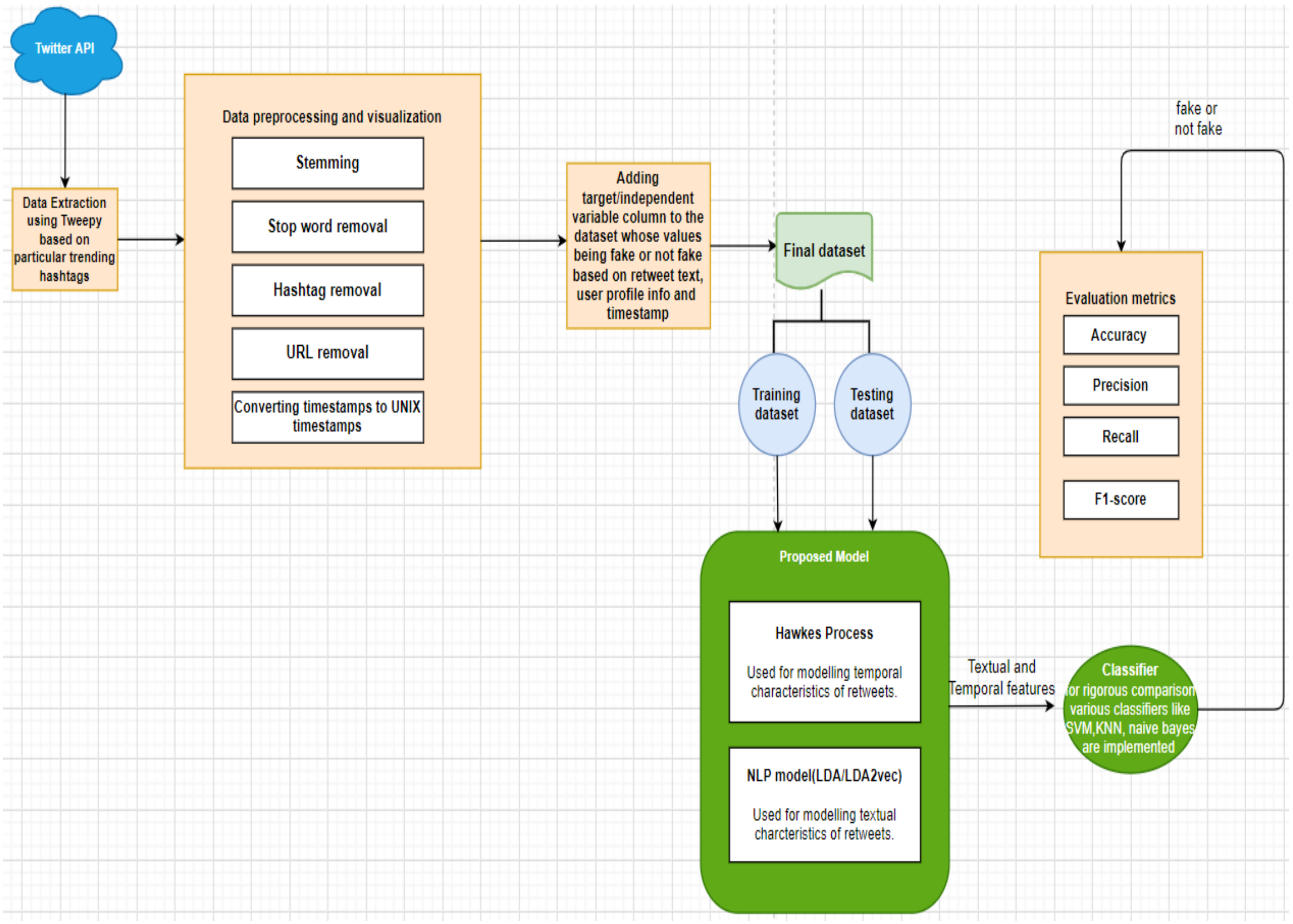
Twitter API
Data Extraction using Tweepy based on particular trending hashtags
Data preprocessing and visualization
Stemming
Stop word removal
Hashtag removal
URL removal
Converting timestamps to UNIX timestamps
Adding target/independent variable column to the dataset whose values being fake or not fake based on retweet text, user profile info and timestamp
Final dataset
Training dataset
Testing dataset
Proposed Model
Hawkes Process
Used for modelling temporal characteristics of retweets.
NLP model(LDA/LDA2vec)
Used for modelling textual charcteristics of retweets.
Textual and Temporal features
Classifier
for rigorous comparison various classifiers like SVM,KNN, naive bayes are implemented
fake or not fake
Evaluation metrics
Accuracy
Precision
Recall
F1-score

Fig 5.1.1 Model Architecture

The above figure 5.1.1 shows the overall system design of the project. Initially, the data is extracted from twitter using twitter API, it is freely available for developers to extract users tweets with certain limitations. Later the tweet's data is further cleaned using various text cleaning process such as stemming, stop word removal, url removal etc. Further, the tweet's data are manually annotated as fake user or not based on their textual information and creation timeline of the tweets. The pre-processed data is given as input to hawkes and LDA model for modelling temporal and textual characteristics of the tweet. The temporal and textual features given as output from the model are fed as input to classifiers which further classifies tweets as fake or not.

5.2 Activity Diagram:

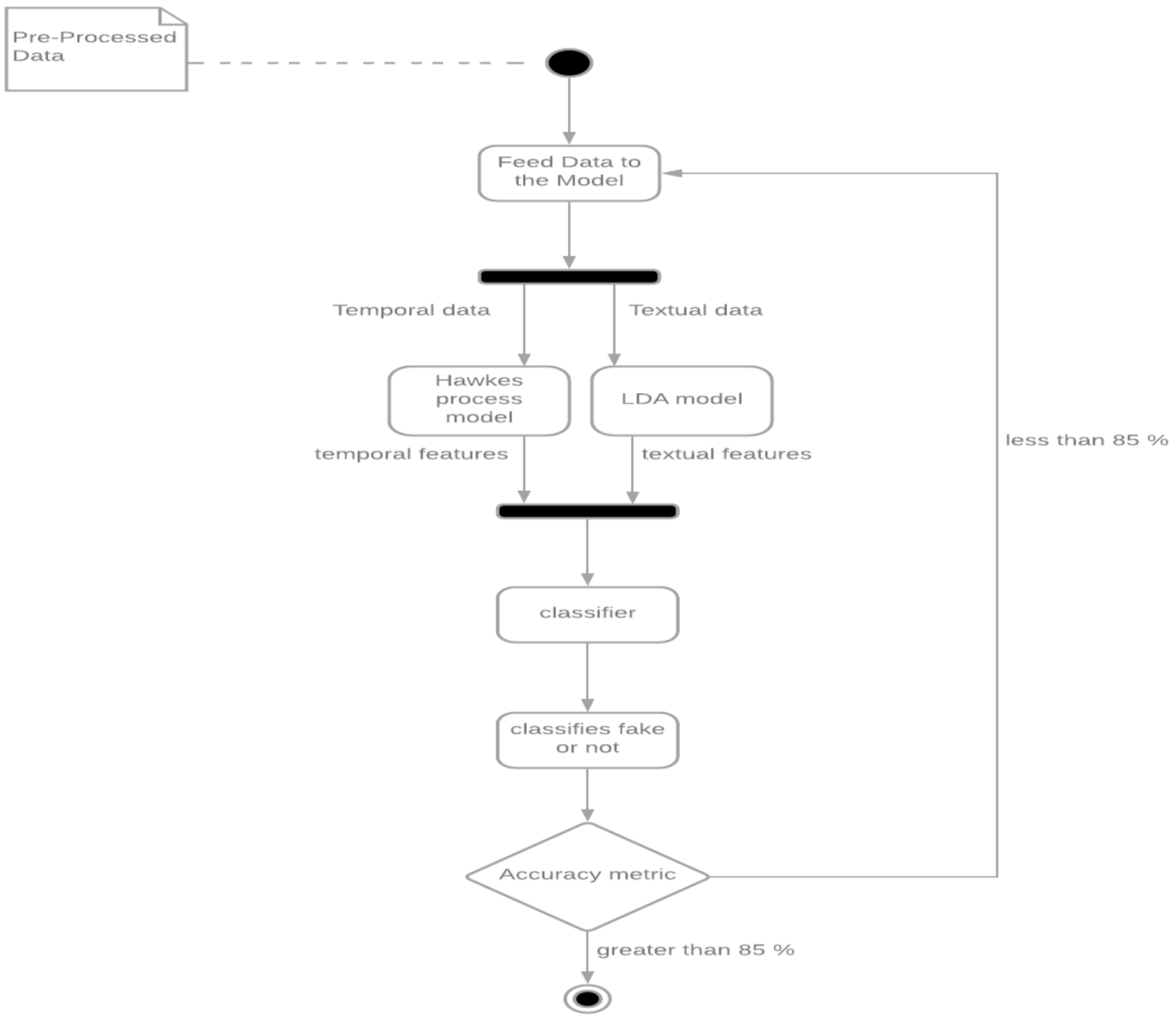

Fig 5.2.1 Activity Diagram

The above figure 5.2.1 depicts the activity diagram, where it shows step by step procedure of how output is generated from the model. The data categorised based on the creation timestamp of the tweet i.e., how many number of times the user has retweeted with certain frame of time is modelled by the Hawkes process and the data categorised based on the user’s tweet text is modelled by LDA model. The topic vectors and the temporal features extracted are further given as input to classifier model such as KNN, neural network etc, to classify it as fake or not fake. Based on the performance metrics the model is made to rerun again and again until the percentage error reduces or accuracy is more than 85 percent.

CHAPTER-6

PROPOSED METHODOLOGY

6.1 Data Extraction:

Developer-access twitter account is used for extracting data using Twitter API. We proceed by creating a filter that will extract tweets based on hashtags/words (ex: COVID, COVID19, CORONA, CORONAVIRUS). The raw data extracted is in JSON format, which is further converted into python Dictionaries. Then it converted from dictionaries to Pandas Data Frames and then finally saved as a CSV file.

6.2 Data Cleaning and Pre-processing:

The CSV file has unrefined information with timestamps and certain ID's. For cleaning the data, we need to remove @mentions, the # symbol, punctuations or any other unwanted symbols, RT (retweet), any hyperlinks if present using certain text cleaning methods such as stemming, stop word removal, url removal, hashtag removal etc. The creation timestamps of the tweets must be normalized between 0 and1 before giving it as input to the model. Also need to remove newlines and multiple blank spaces.

6.3 Hawkes Process:

Main idea to use hawkes process to utilize temporal information is that it supposes the past events can temporarily raise the probability of future events. It complies with the retweeting behaviour of a twitter user where a user who has been frequently tweeting in the recent past is more likely to tweet in the future. In $\Lambda_u(t) = \mu_u + \alpha_u \sum_{j<t} K(t-j)$, which is shown in the figure 6.3.1 below, the term which takes the effect of all the before tweets is the summation term, this allows the effect of past tweets or retweets on the present retweet. '$\mathscr{M}_u$' takes into account the initial retweeting capacity of the user. '$\boldsymbol{\alpha}_u$'

takes into account the effect of past retweets done by the user on the present retweet. As the influence of tweet decays naturally wrt time, the term ‘K(t-j)’ captures the decay in the effect of a tweet at time ‘j’ with respect to current time ‘t’.

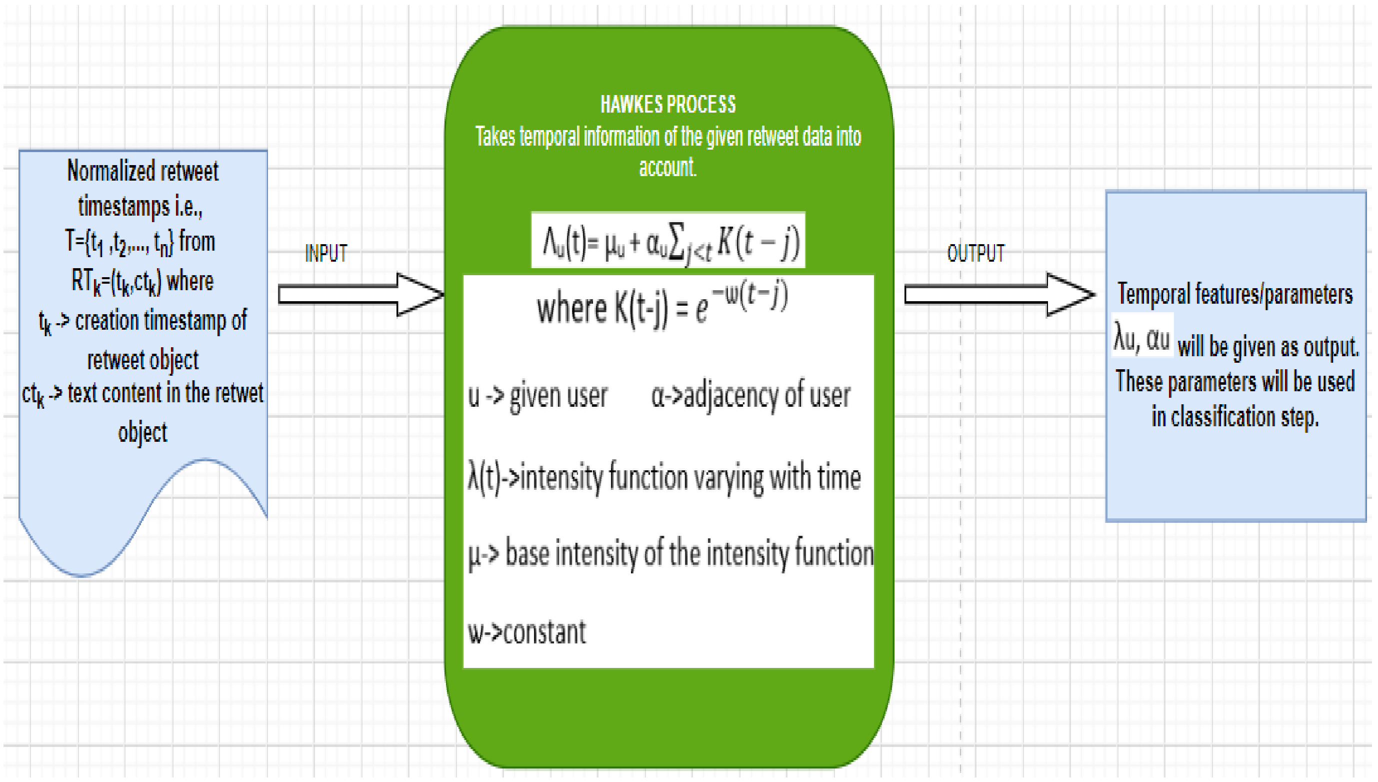
Normalized retweet timestamps i.e., $T=\{t_1, t_2, \dots, t_n\}$ from $RT_k=(t_k, ct_k)$ where
t_k -> creation timestamp of retweet object
ct_k -> text content in the retwet object
INPUT
HAWKES PROCESS
Takes temporal information of the given retweet data into account.
$\Lambda_u(t) = \mu_u + \alpha_u \sum_{j<t} K(t-j)$
where $K(t\text{-}j) = e^{-w(t-j)}$
u -> given user
α->adjacency of user
λ(t)->intensity function varying with time
μ-> base intensity of the intensity function
w->constant
OUTPUT
Temporal features/parameters λu, αu will be given as output. These parameters will be used in classification step.

Fig 6.3.1 Hawkes Process

6.4 Topic modelling using LDA:

Due to the skewness in the data b/w fake and genuine retweeters (a smaller number of fake retweeters will be present compared to genuine), the topic vectors would inevitably be biased towards the genuine users due to unsupervised nature of LDA. Hence 2 LDA models are trained as shown in figure 6.4.1 below, one model is trained on the fake retweeters documents (LDA_f) and other on the genuine retweeters documents (LDA_g).

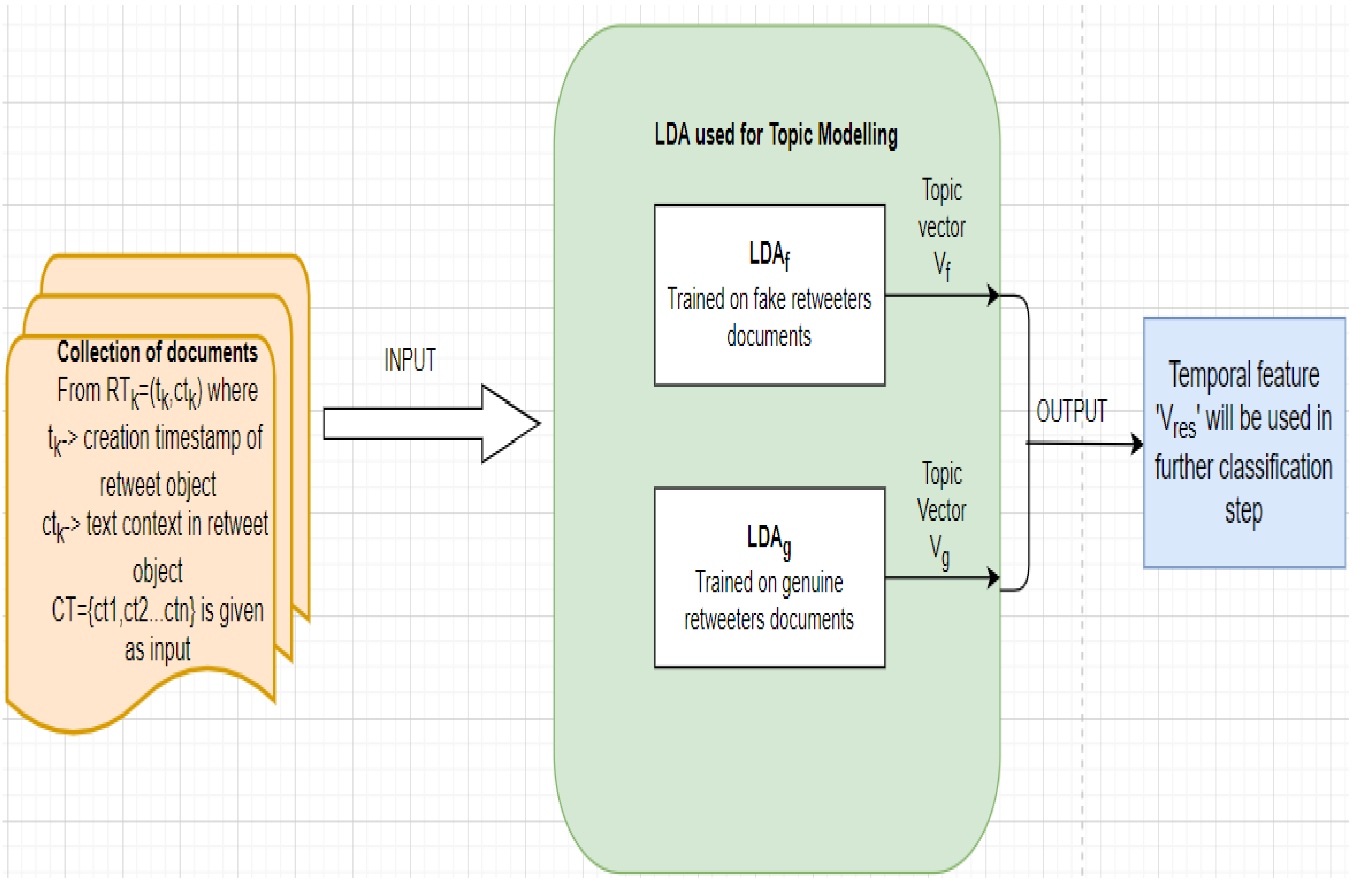
Collection of documents
From $RT_K=(t_K,ct_K)$ where
t_K-> creation timestamp of retweet object
ct_K-> text context in retweet object
CT={ct1,ct2...ctn} is given as input
INPUT
LDA used for Topic Modelling
LDA_f
Trained on fake retweeters documents
Topic vector V_f
LDA_g
Trained on genuine retweeters documents
Topic Vector V_g
OUTPUT
Temporal feature 'V_{res}' will be used in further classification step

Fig 6.4.1 Latent Dirichlet Allocation (LDA)

6.5 Classifier:

We classify retweeters based on the feature vector obtained from the previous stages. We reiterate our intuition that retweeters who are similar in behaviour will tend to have similar feature vectors and hence lesser distance between them due to similarities in retweeting content as well as temporal tendencies. For a more rigorous comparison, we implement various classification models such as Naive Bayes, SVM, neural network, KNN and compare their performances.

6.6 Fully Connected Neural Network Model:

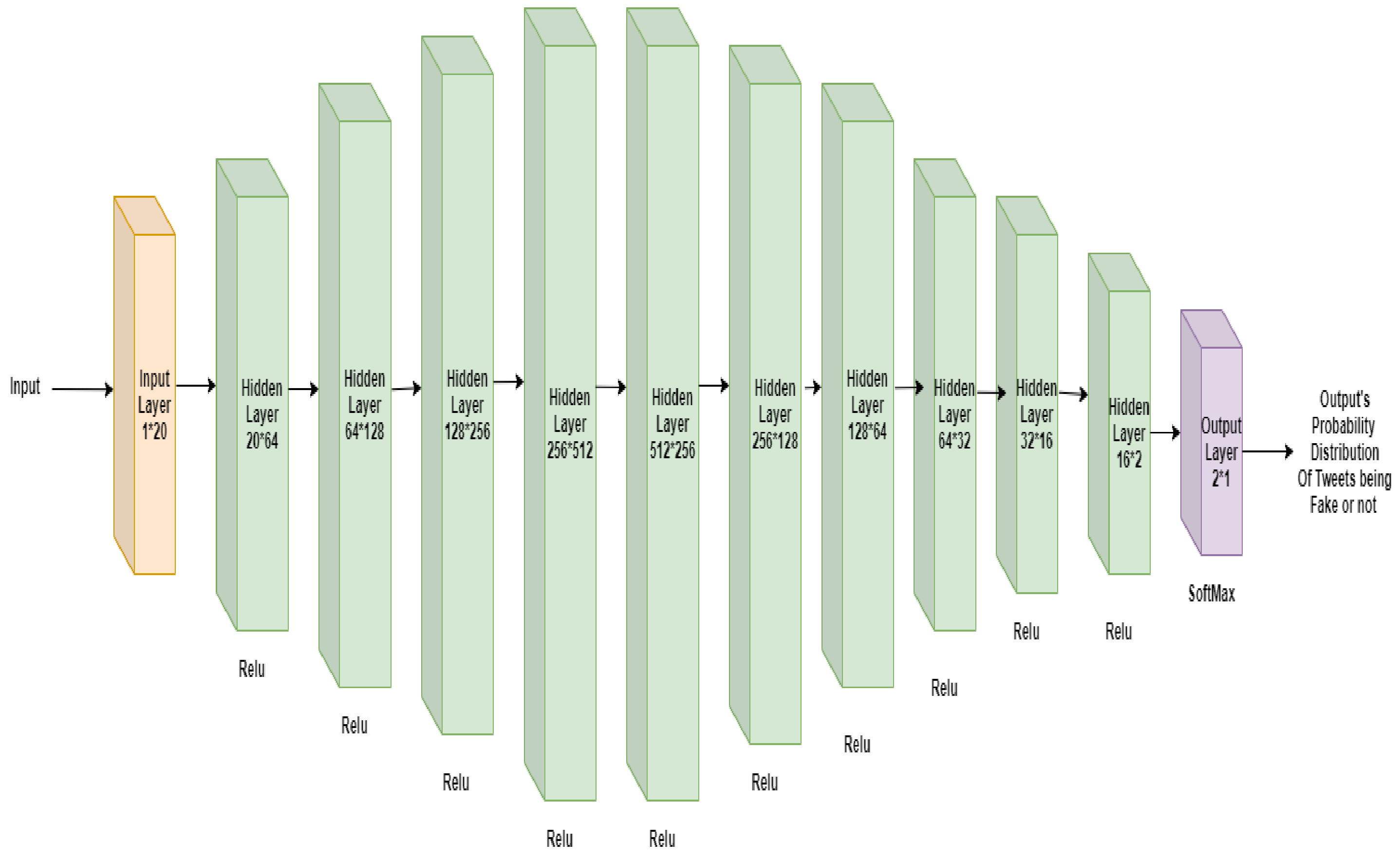

Fig 6.6.1 Fully Connected Neural Network

A fully connected neural network that contains 11 layers is used as one of the classifiers for the combined model. It is shown in figure 6.6.1 above. It uses Relu as its activation function and binary cross entropy as its loss function. It uses softmax function at its output layer which estimates probabilities of whether a user is fake or not. More detailed information is present in table 6.6.2.

TABLE 6.6.2 Neural Network Layers

Layer No	Type of the layer	Input shape	Output shape	No. of Neurons
1	Input	20	64	20
2	Hidden	64	128	64
3	Hidden	128	256	128
4	Hidden	256	512	256
5	Hidden	512	256	512
6	Hidden	256	128	256
7	Hidden	128	64	128
8	Hidden	64	32	64
9	Hidden	32	16	32
10	Hidden	16	2	16
11	Output	2	1	2

- Optimizer: Adam
- Learning Rate: 0.0001
- Number of Epochs: 250
- Libraries Used: Pytorch, and Nvidia CUDA toolkit for Graphics Card support.

6.7 Bag of Words Technique:

This method gets all unique words in the entire dataset to build the corpus or dictionary and replace the words in the text with their index value in the built dictionary. We convert that word index list to a fixed-length list called sequence length so that it can be fed into neural networks because the input layer of the network is fixed; the average number of words in our dataset was 14. We selected the sequence length = 18. If the number of words in the tweet is less than 18 simply append 0s until the length reaches 18. If the number of words in the tweets exceeds 18 simply discard the text that is after 18. Now we have data that converts word sequence (tweet text) to a fixed-length vector of size 18 that consists of index numbers from dictionaries of equivalent words in the tweet.

6.8 Bi-LSTM Model:

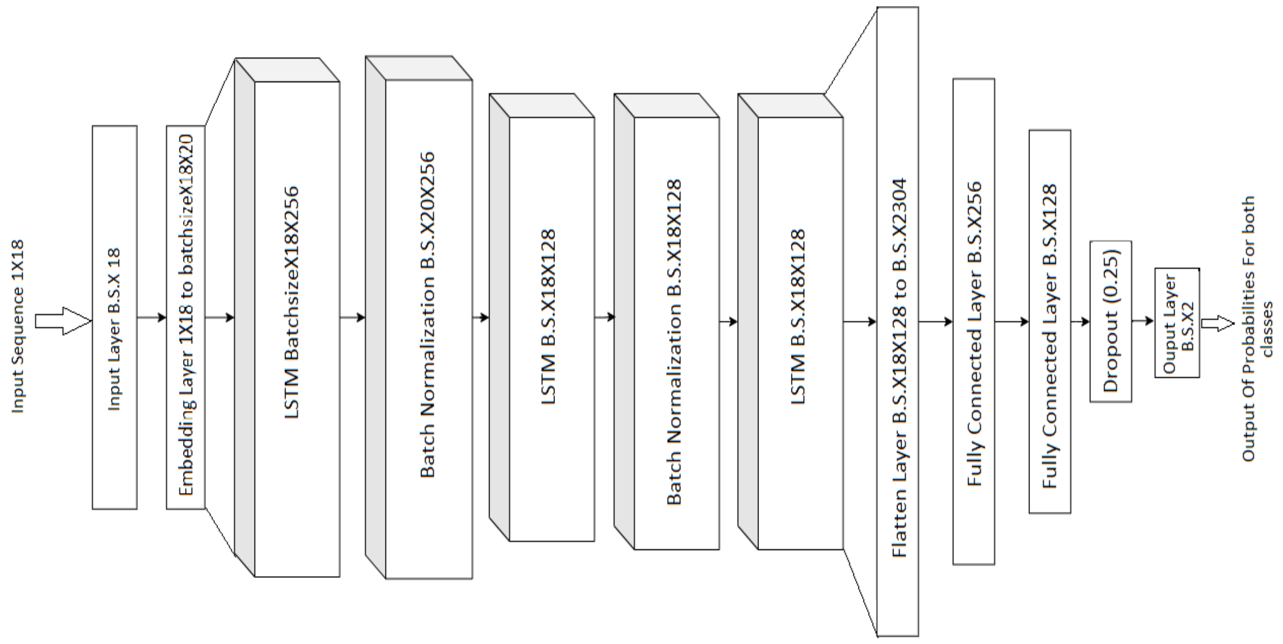
Input Sequence 1X18
Input Layer B.S.X 18
Embedding Layer 1X18 to batchsizeX18X20
LSTM BatchsizeX18X256
Batch Normalization B.S.X20X256
LSTM B.S.X18X128
Batch Normalization B.S.X18X128
LSTM B.S.X18X128
Flatten Layer B.S.X18X128 to B.S.X2304
Fully Connected Layer B.S.X256
Fully Connected Layer B.S.X128
Dropout (0.25)
Ouput Layer B.S.X2
Output Of Probabilities For both classes

Fig 6.8.1 Bi-LSTM Model

Bi-LSTM stands for Bidirectional LSTM. As shown in figure 6.8.1, this Neural network consists of not only Bi-LSTM various layers like Dense (Fully connected), Embedding layer for moving 2D tensor to 3D tensor so that it can be feed into LSTM / Bi-LSTM layers, series Bi-LSTM layers and Batch Normalization layers, a flatten layer for converting 3D tensor to 2D tensor and finally fully connected layers with a dropout of 25% at the hidden layer. Bi-LSTM: Bi-LSTM is more powerful than LSTM or Unidirectional-LSTM. Normal or unidirectional LSTM keeps only past result in it. But Bidirectional LSTM keeps past and future results so that makes it more powerful as well as training becomes more efficient. It is widely used in solving sequence-based problems.

Hyper-parameters, optimizers used:

- Input- Sequence vector of size 18
- Optimizer-Adam
- Learning Rate-0.0001 & no. of epochs-10

- Early stopping with patience -2
- Loss function-Binary cross entropy
- Activation function- Relu in all fully connected layers, sigmoid b/w LSTM layers and sotmax at ouput layer
- Output- probabilities that belongs to classes.

6.9 Design Details:

6.9.1 Novelty:

Detecting the fake retweeters and hashtag misuses using hawkes process and topic modelling**.**

6.9.2 Innovativeness:

For obtaining a better accuracy compared to the previous models we try to use other NLP based model such as LDA2vec etc, for modelling textual information of the tweet. We even compare results based upon different classifiers such as SVM, naive bayes etc.

6.9.3 Interoperability:

The data from twitter is being extracted very easily to the local system using twitter API. The model can be used in any system with minimum requirements.

6.9.4 Performance:

i. Robustness: Model we built is capable of classifying the input correctly with acceptable minimal errors.

ii. Availability: Since our model is a freely available open-source model, it can be used by any individual.

iii. Accuracy: We intend to make our model achieve a higher accuracy compared to other previous works by at least 2-4%, thus having a significant improvement over the previous ones.

iv. Application compatibility: It is compatible on any device which has better or higher processing power. The model could run slower on lower processing power systems.

v. <u>Legacy of modernization</u>**:** The extent of this study is not restricted to Twitter, but applicable to other social media applications such as Facebook and Instagram with similar reposting capabilities

vi. <u>Resource utilization</u>: Training the Deep Neural Network and Natural Language Processing is very much time expensive as well as resource consuming. It requires modern processors and hardware to execute such things.

CHAPTER-7

IMPLEMENTATION AND PSEUDOCODE

7.1 Data Collection and Extraction:

Data collection and Feature extraction is one of the main steps and it takes a lot of time. Selecting the right features will lead the good model accuracy. So, we have collected the data from Twitter. We have created the Twitter developer account for using their API services. Then used ‘Tweepy’ a python library for collecting the data. The collected data was in the form of JSON and JSON is converted into a python dictionary and then python dictionary converted into pandas Data-frame finally pandas data-frame is saved into a CSV file and used for further study and experiments.

7.1.1 Python implementation for Extracting the Tweet data using Tweepy

```
from tweepy import API, OAuthHandler, Cursor
from pandas import DataFrame
from pprint import pprint
from json import loads
class DataCollection:
   API_KEY = “Your API KEY”
   API_SECRETE_KEY = “Your API Secrete Key”
   api = None
   headersWritten = 0
   dataDict = {'tweet-id':[],
               'tweet-created-at':[],
               'text':[],
               'hashtags':[],
               'mentions':[],
               'geo':[],
               'retweet-count':[],
               'user-id':[],
               'user-full-name':[],
               'user-name':[],
               'favourites-count':[],
               'followers-count':[],
               'friends-count':[],
               'is-verified-account':[]
              }
   def __init__(self):
       auth = OAuthHandler(self.API_KEY, self.API_SECRETE_KEY)
       self.api = API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)

   def collectDataOnSearchQuery(self, searchQuery):
       return Cursor(self.api.search, q=searchQuery, per_page = 100, result_type="recent",
lang="en",tweet_mode = "extended").pages(100)

   def appendUsefulDetails(self,searchQuery):
       for page in self.collectDataOnSearchQuery(searchQuery):
           for p in page:
               self.dataDict['tweet-id'].append(p._json['id'])
               self.dataDict['tweet-created-at'].append(p._json['created_at'])
               self.dataDict['text'].append(p._json['full_text'])

               if len(p._json['entities']['hashtags']) == 0:
                   self.dataDict['hashtags'].append('None')
               else:
                   self.dataDict['hashtags'] .append(' '.join([i['text'] for i in
p._json['entities']['hashtags']]))

               if len(p._json['entities']['user_mentions']) == 0:
                   self.dataDict['mentions'].append('None')
               else:
                   self.dataDict['mentions'].append(' '.join((str(i['id']) for i in
p._json['entities']['user_mentions'])))

               if p._json['geo'] == None:
                   self.dataDict['geo'].append('None')
               else:
                   self.dataDict['geo'].append(' '.join(str(i) for i in p._json['geo']['coordinates']))

               self.dataDict['retweet-count'].append(p._json['retweet_count'])
               self.dataDict['user-id'].append(p._json['user']['id'])
               self.dataDict['user-full-name'].append(p._json['user']['name'])
               self.dataDict['user-name'].append(p._json['user']['screen_name'])

               self.dataDict['favourites-count'].append(p._json['user']['favourites_count'])
               self.dataDict['followers-count'].append(p._json['user']['followers_count'])
               self.dataDict['friends-count'].append(p._json['user']['friends_count'])
               self.dataDict['is-verified-account'].append(p._json['user']['verified'])
       return DataFrame.from_dict(self.dataDict)
   def writeToCSV(self,searchQuery,fileName):
       appendUsefulDetails(searchQuery).to_csv(fileName)

searchQuery = 'corona OR coronavirus OR covid OR covid19 OR covid-19'  # Keyword
dc = DataCollection()
df = dc.appendUsefulDetails(searchQuery)
# dc.writeToCSV(searchQuery,'Covid19.csv')
print(df.head())
print('******************************************************************')
print(df.shape)
df.to_csv('New_Covid.csv')
```

7.2 Data Cleaning:

7.2.1 Python implementations for Data cleaning

```
from re import sub, MULTILINE
tweet_text = substitute(r'@[A-Za-z0-9]+', '', tweet_text)
    tweet_text = substitute(r'@[A-Za-zA-Z0-9]+', '', tweet_text)
    tweet_text = substitute(r'@[A-Za-z]+', '', tweet_text)
    tweet_text = substitute(r'@[-)]+', '', tweet_text)
    tweet_text = substitute(r'#', '', tweet_text)
    tweet_text = substitute(r'RT[\s]+', '', tweet_text)
    tweet_text = substitute(r'https?\/\/\S+', '', tweet_text)
    tweet_text = substitute(r'&[a-z;]+', '', tweet_text)
    tweet_text = substitute(r'[^\w\s]', '', tweet_text)
    tweet_text = substitute(r'^https?:\/\/.[\r\n]', '', tweet_text, flags=MULTILINE)
    tweet_text = tweet_text.replace(': ', '')
    tweet_text = tweet_text.replace('_', '')
    tweet_text = tweet_text.replace('\n', '')
    tweet_text = tweet_text.replace('  ', ' ')
    tweet_text = tweet_text.lower()
    return text
def hashtagCounter(h):
    if h == 'None':
        return 0
    else:
        h = h.split(' ')
        return len(h)
```

7.3 Data Visualization:

7.3.1 Pie Chart Showing counts on verified accounts:

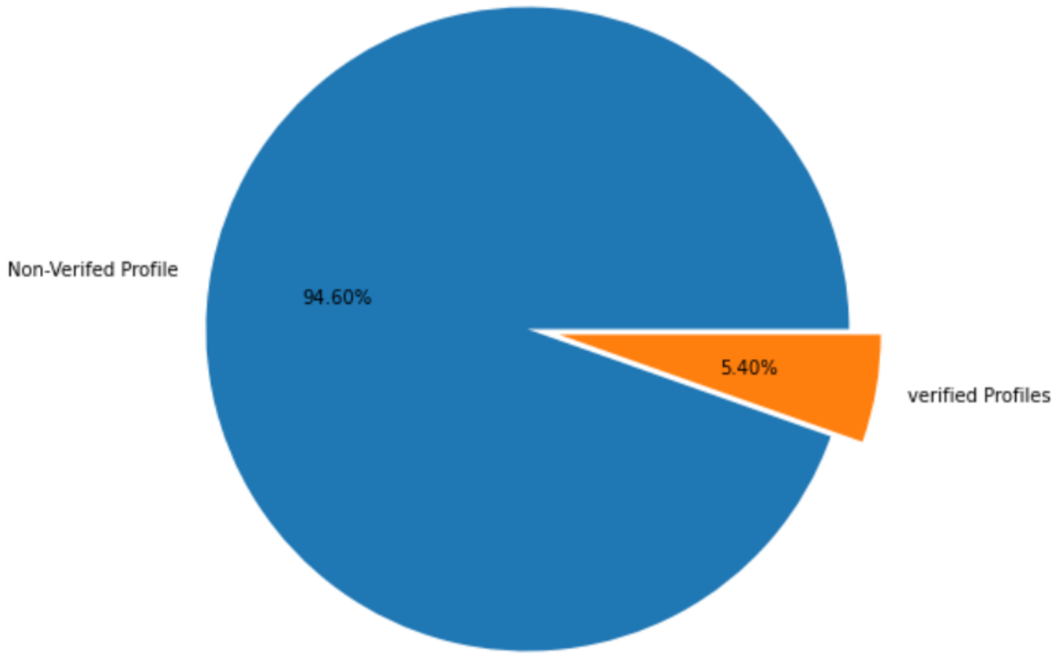

Fig 7.3.1.1 Pie chart verified vs non-verified accounts

The figure 7.3.1.1 shows the percentage of verified accounts vs non-verified accounts of the twitter user's data extracted from twitter. 5.4 percentage of total users data extracted are verified whereas the rest 94.6 percentage are non-verified users. Verified users are the once that are verified by twitter based on the public references or media references etc. This 5.4 percentage of verified users can be manually annotated as real users since there is a very less chance of them being a fake user. But the rest of users have to be checked manually and must be determined whether fake or not.

7.3.2 Word Cloud

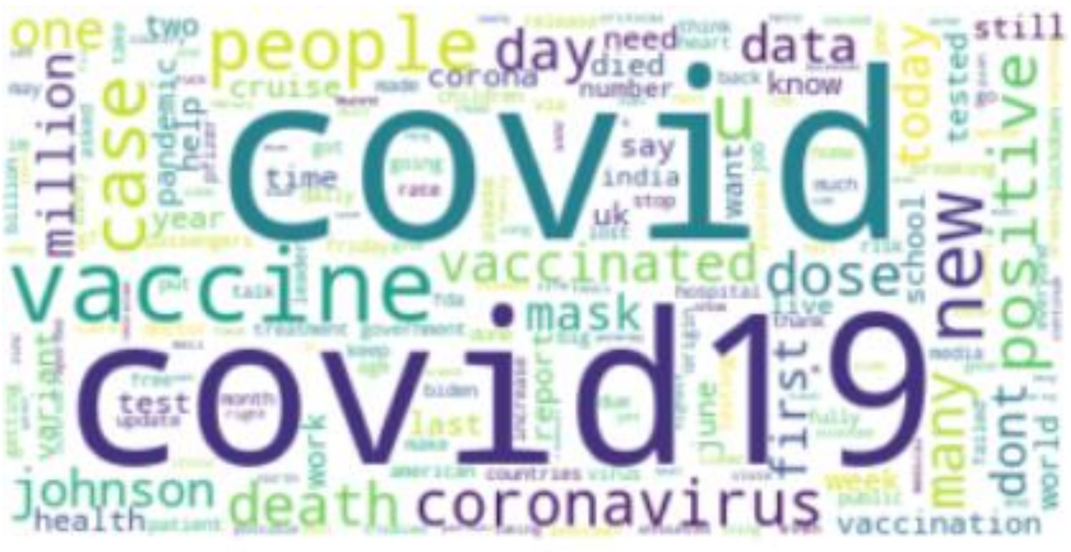

Fig 7.3.2.1 Word Cloud for real users

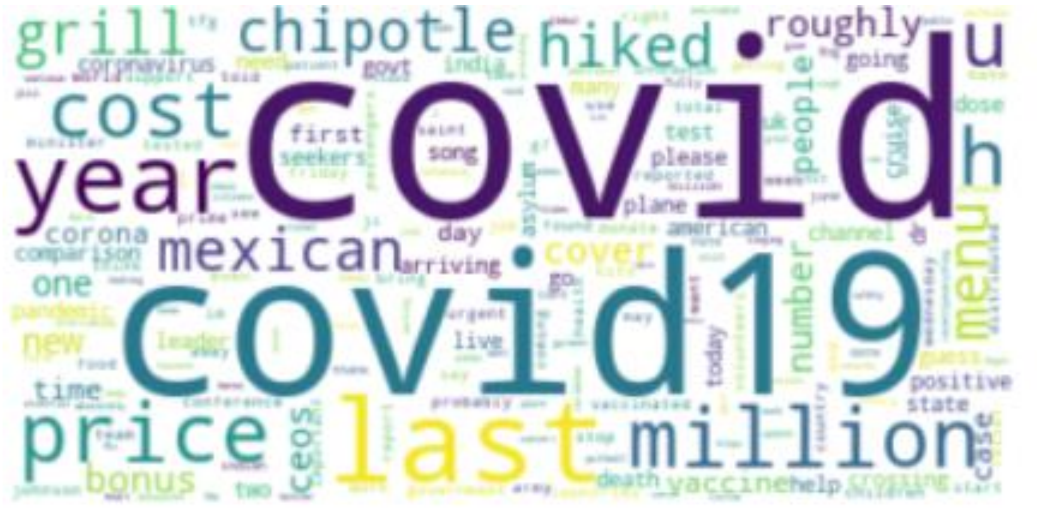

Fig 7.3.2.2 Word Cloud for Fake users

The two figures shown above are word clouds done for real and fake users. Figure 7.3.2.1 shows the word cloud for real users i.e., visual representation of frequency of words that appear in the user tweet text. We can observe that a real user mostly tweets something that is relevant to the topic i.e., #Covid19. Whereas, Figure 7.3.2.2 depicts that a manually annotated fake user tweets something that is irrelevant to the topic #Covid19. A fake user is just misusing trending hashtag to promote or express his/her tweets.

7.3.3 Bar Graph showing Users Having Least No. of Tweets

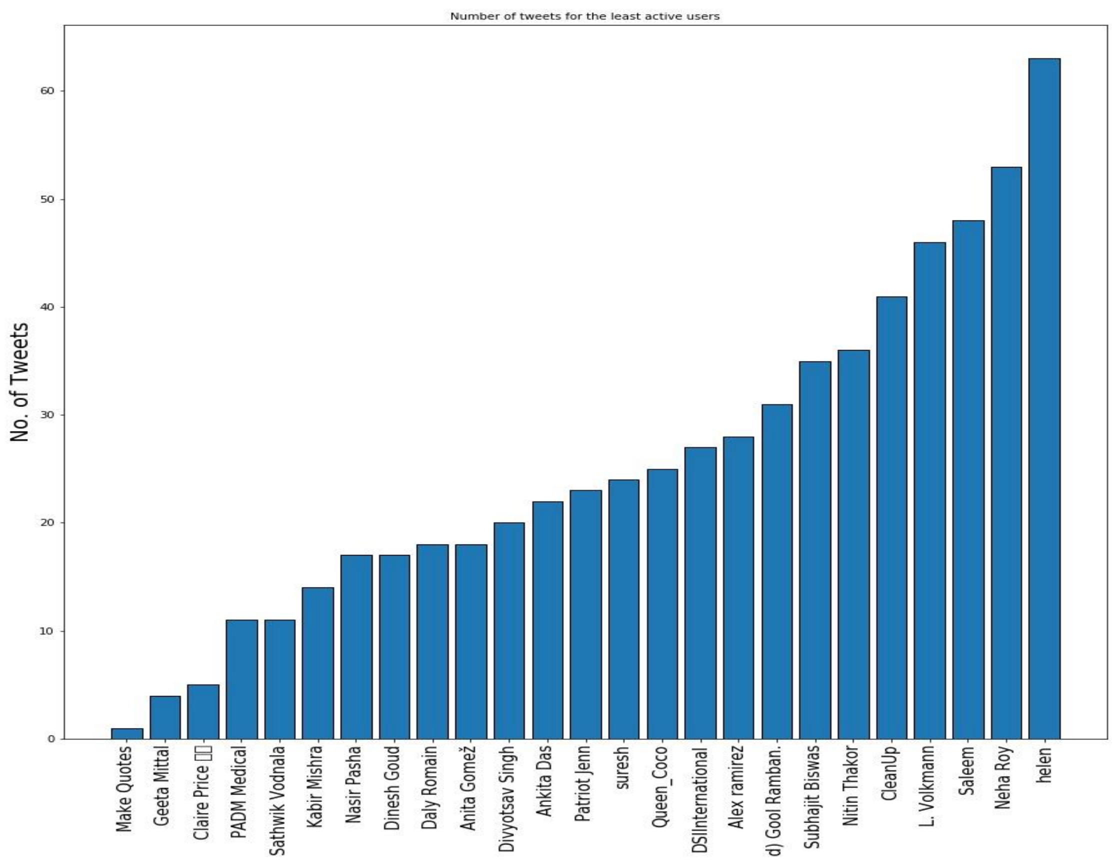

Fig 7.3.3.1 No. of Tweets vs Users

The above Figure 7.3.3.1, shows the users who tweets a smaller number of time or not so active on twitter. These users have small chance of being a fake user since he/she is not abnormally retweeting any tweets. This visualization will be helpful in manual annotation of the users as being fake or not fake.

7.4 Hawkes Process:

```
from tick.hawkes import HawkesExpKern
def get_hawkes_model(timestamps):
    decays = 0.3
    gofit = 'likelihood'
    penalty = 'l2'
    solver = 'agd'
    step = None
    tol = 1e-05
    max_iter = 1000
    verbose = False
    a_kernel = HawkesExpKern(decays, gofit=gofit, penalty=penalty, solver=solver, step=step, tol=tol,
max_iter=max_iter,
     verbose=verbose)
    a_kernel.fit(timestamps)
    baseline = a_kernel.baseline[0]
    adj = a_kernel.adjacency[0][0]
    return baseline, adj
def Hawkes_main_driver():
    baselines = []
    adjs = []
    labels = []
    for i in range(get_number_of_users()):
        i_user_df = get_retweet_df(
            i)  # from 0_Datset folder get each user tweets timeline and filter retweets from them
        i_user_df.created_at = i_user_df.created_at.apply(
            dateTimeCreator)  # String Date Time to python Datetime library
        min_date = min(i_user_df['created_at'])  # time at which first retweet was made
        timestamps = i_user_df.created_at.apply(get_timestamp,
                                                origin_date=min_date).to_numpy()  # time to minnutes passed from the first
retweet
        min_time = np.min(timestamps)
        max_time = np.max(timestamps)
        scaled_time = (np.sort(np.unique((timestamps - min_time) / max_time)))  # Min-Max Scaler (0, 1)
        BaseLine, adj_mat = get_hawkes_model(
            timestamps=[scaled_time])  # fit the model and get the Hawkes Expression kernal model output
        baselines.append(BaseLine)
        adjs.append(adj_mat)
        break
    return baselines, adjs, labels
```

7.5 Latent Dirichlet Allocation (LDA):

```
def get_LDA_trained_Models():
    dataset_path = " "
    df = pd.read_csv(dataset_path + "CompleteAnnotated.csv")
    df.tweet_text = df.tweet_text.apply(text_clean)
    all_docs_genuine = []
    all_docs_fake = []
    labels = []
    complete_docs = []
    for i in range(df.shape[0]):
        labels.append(df.iloc[i]['Annotation'])
        complete_docs.append(df.iloc[i]['tweet_text'].split())
        if df.iloc[i]['Annotation'] == 0:
            all_docs_genuine.append(df.iloc[i]['tweet_text'].split())
        else:
            all_docs_fake.append(df.iloc[i]['tweet_text'].split())
    complete_dict = Dictionary(complete_docs)

    complete_corpus = [complete_dict.doc2bow(text) for text in complete_docs]
    lda = LdaModel(complete_corpus, num_topics=10)
    X = np.array([get_topic_vector(tweet.split(), 10, complete_dict, lda) for tweet in df.tweet_text])
    Y = np.array(labels)
    x_train, x_test, y_train, y_test = train_test_splitter(X, Y)
    return x_train, x_test, y_train, y_test, complete_dict,complete_corpus,lda
```

7.6 Neural Network Model:

```
class NeuralNetworkClassifierModel(nn.Module):
    def __init__(self, input_size, output_size, hidden_size):
        super(NeuralNetworkClassifierModel, self).__init__()
        self.layer1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.layer2 = nn.Linear(hidden_size, hidden_size)
        self.layer3 = nn.Linear(hidden_size, hidden_size)
        self.layer4 = nn.Linear(hidden_size, hidden_size)
        self.layer5 = nn.Linear(hidden_size, output_size)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, inputs):
        out = self.layer1(inputs)
        out = self.relu(out)
        out = self.layer2(out)
        out = self.relu(out)
        out = self.layer3(out)
        out = self.relu(out)
        out = self.layer4(out)
        out = self.relu(out)
        out = self.layer5(out)
        out = self.softmax(out)
        return out

    device = get_Device()

        input_size = 22
        output_size = 40
        hidden_size = 40
        learning_rate = 0.05
        batch_size = 128
        n_epochs = 100

        model = NeuralNetworkClassifierModel(input_size=22,
                                             output_size=2,
                                hidden_size=40)
        model.to(device)
        lossfn = torch.nn.CrossEntropyLoss()
        lossfn.to(device)
        optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

CHAPTER-8

RESULTS AND DISCUSSION

The following results have been obtained by modelling textual and temporal data of the user tweets extracted from twitter. The Performance metrics used to compare the results between different classifiers are accuracy, F1-score and precision. The tables below show the results obtained during the training and testing of different model. Of all the models shown in table 8.1,8.2 & 8.3, the combined-model with neural network models gives better accuracy. LDA with a neural network model gives an accuracy of 94.11% for the validation dataset.

TABLE 8.1: Accuracy of Hawkes Model with Classifiers

Model	**Training Accuracy (in %)**	**Validation Accuracy (in %)**
Hawkes Process + KNN	86.6	88.5
Hawkes Process + SVM	86.81	89.21
Hawkes Process + Naive Bayes	86.81	89.2
Hawkes Process + Neural Network (8 FC Layers + ReLu Activation Softmax at end)	86.81	89.54

Table 8.1 shows different accuracy values obtained by different classifiers with hawkes process. Hawkes process models the temporal data of the user tweets i.e., creation timelines of each tweets which are normalised between 0 and 1. Among KNN(K=5), SVM, Naïve Bayes and neural network classifiers with 8 fully connected layers and softmax function at the output layer, hawkes process with neural network classifier gives higher accuracy value of 89.54 percent and training accuracy of 86.81 percent.

TABLE 8.2: Accuracy of LDA Model with Classifiers

Model	Training Accuracy (in %)	Validation Accuracy (in %)
LDA + KNN	90.82	88.88
LDA + SVM	85.58	88.2
LDA + Naive Bayes	79.52	83.33
LDA + Neural Network (8 FC Layers + ReLu Activation Softmax at end)	91.97	91.83
Bag of Words + Neural Network (LSTM + ReLu Activation Softmax at end)	99.03	88.76
Bag of Words + Neural Network (3 Layers Bi-LSTM, Embedding, Flatten, Dense, Dropout, Batch Normalization + ReLu Activation Softmax at end)	92.391	88.50
Word Embedding (Word2Vec) + Bi-LSTM Neural Network (3 Layers Bi-LSTM, Embedding, Flatten, Dense,Dropout, Batch Normalization + ReLu Activation Softmax at end)	84.62	82.91

Table 8.2 shows various accuracy values of different classifiers with LDA model and different NLP technique. Latent Dirichlet allocation models the textual data of the user's tweet. The tweet text of the user are cleaned using stop word removal, url removal etc and given as input to the model. Among KNN(K=5), SVM, Naïve Bayes, neural network and BI-LSTM model, LDA with Neural network classifier provides higher validation accuracy of 91.83 percent and training accuracy of 91.97 percent.

TABLE 8.3: Accuracy of Combined Models with Classifiers

Model	Training Accuracy (in %)	Validation Accuracy (in %)
Combined Topic Vector + KNN	82.06	76.47
Combined Topic Vector + SVM	75.34	80.71
Combined Topic Vector + Naive Bayes	70.51	69.6

Combined Topic Vector + Neural Network (8 FC Layers + ReLu Activation Softmax at end)	82.55	75.1633

Table 8.3 shows various accuracy values of different classifiers with both LDA and hawkes process model. Among all the classifiers used combined model with SVM classifier provides 80.71 percent for validation data and combined model with neural network classifier provides training accuracy of 82.55 percent.

CHAPTER-9

CONCLUSION AND FUTURE WORK

We have built a classifier that classifies re-tweeters as fake or not based on the context of the tweet as well as the timelines of their tweets with a good accuracy. We trained and tested the model for tweets based on #covid 19 related only, this made it less exhaustive and quite effective. Hawkes process was used for modelling the temporal characteristics of the tweets and LDA model were used for modelling the textual characteristics of the tweets. We trained two LDA models one for genuine tweets and the other for fake tweets each having 10 as the number of topics (K = 10). Now, for a given tweet text we can get 10 probability values that tells us the probability that a given text belongs to that particular topic. This implies for both LDA models. So, for a given tweet text we get 20 probabilities values, 10 from *Genuine* LDA model and the other 10 from *Fake* LDA model. The extracted topic vectors are further given as input to classifier which predicts the user as fake or not based on their tweets. Different classifiers were used and each provided different accuracy values as seen in the above tables. Out of all the classifiers neural network outperforms other classifiers. We would like to extend our project to other social media platforms where hashtags are used such as Facebook, Instagram etc., to classify users as fake or not in the coming future. We would further train the NLP model for other hashtags and tweets for better modelling of textual characteristics of the tweets.

BIBLIOGRAPHY

[1] Dutta, Hridoy Sankar, et al. "Hawkeseye: Detecting fake retweeters using hawkes process and topic modeling." *IEEE Transactions on Information Forensics and Security* 15 (2020): 2667-2678.

[2] Helmstetter, Stefan, and Heiko Paulheim. "Weakly supervised learning for fake news detection on Twitter." *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018.

[3] Madisetty, Sreekanth, and Maunendra Sankar Desarkar. "A neural network-based ensemble approach for spam detection in Twitter." *IEEE Transactions on Computational Social Systems* 5.4 (2018): 973-984.

[4] Buntain, Cody, and Jennifer Golbeck. "Automatically identifying fake news in popular twitter threads." *2017 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2017.

[5] Lukasik, Michal, et al. "Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter." *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2016.

[6] Sivasangari, V., et al. "Isolating rumors using sentiment analysis." *Journal of Cyber Security and Mobility* (2018): 181-200.

[7] Inuwa-Dutse, Isa, Mark Liptrott, and Ioannis Korkontzelos. "Detection of spam-posting accounts on Twitter." *Neurocomputing* 315 (2018): 496-511.

[8] Jain, Nikita, Pooja Agarwal, and Juhi Pruthi. "HashJacker-detection and analysis of hashtag hijacking on Twitter." *International journal of computer applications* 114.19 (2015).

[9] Mazza, Michele, et al. "Rtbust: Exploiting temporal patterns for botnet detection on twitter." *Proceedings of the 10th ACM Conference on Web Science*. 2019.

[10] Jang, Yonghun, Chang-Hyeon Park, and Yeong-Seok Seo. "Fake news analysis modeling using quote retweet." *Electronics* 8.12 (2019): 1377.

[11] Yang, Sidi, and Haiyi Zhang. "Text mining of Twitter data using a latent Dirichlet allocation topic model and sentiment analysis." *Int. J. Comput. Inf. Eng* 12.7 (2018): 525-529.

[12] Ajao, Oluwaseun, Deepayan Bhowmik, and Shahrzad Zargari. "Fake news identification on twitter with hybrid cnn and rnn models." *Proceedings of the 9th international conference on social media and society*. 2018.