Trumer: Rumor Detection on Twitter using Machine Learning and NLP techniques

Grusha Dharod[1], Harshal Dedhia[2], Jaineel Shah[3], Smita Sankhe[4] and Sneh Chitalia[5]

[1, 2, 3, 4, 5]Department of Computer Engineering, K. J. Somaiya College of Engineering, Vidyavihar, Mumbai.

[1]grushadharod@gmail.com

[2]harshaldedhia11@gmail.com

[3]jaineelchamp@gmail.com

[4]smitasankhe@somaiya.edu

[5]csneh4@gmail.com

Abstract. A rumor/rumour is basically a form of a statement whose legitimacy is not yet confirmed. Rumor gives birth to the spread of misinformation in public. These days we get most of the information about what's happening around the world from various online social media websites. Twitter is one of the most popular platforms, where people tweet about various events. One of the major problems we all face on these social media platforms is "RUMORS". There is no authenticity regarding these tweets. In this work, we propose an application "Trumer" that allows the user to input a tweet and calculates the probability of a tweet being a rumor in real-time. This application uses various parameters such as a retweet count, user's followers count, following count, verified status, related tweets, tense of the given tweet to calculate the probability of a tweet being a rumor. This application will allow users to determine the authenticity of the tweet and thus restricting the spread of misinformation.

Keywords: Trumer, rumor, rumor detection, twitter, NLP, natural language processing, hashtags.

1 Introduction

A rumor is defined as a statement, report or a story whose truth is uncertain or doubtful. They circulate from person to person rapidly causing unwanted confusion among people. Rumors can be spread deliberately as a part of propaganda or can be spread due to misunderstandings as well.

Social media has gained a lot of popularity in the last few years. Many social media applications are emerging day by day. Every day, people share millions of posts. But, since everyone has the power to post whatever they want to, the Social Media Platforms have thus become originators of rumors and spread of misinformation.

Twitter is one of the most popular social media platforms today. Approximately, 500 million tweets are tweeted everyday [2]. Additionally, over 85% of all trending topics are news [21]. This makes Twitter one of the major sources to spread the news and also for people to voice their opinions. However, this also paves the way for spreading rumors. These rumors could lead to chaotic situations or even lead to undue losses for a person or an organization. In recent times, there have been many instances of rumors appearing on various social media platforms including Twitter.

A famous example is the 2013 rumor that the White House has been bombed and the then US President Barack Obama is injured. This led to fear among people and also badly affected the stock market, leading to unfavorable economic situations. Another instance was a rumor that 9 Indian banks are going to be shut permanently by the RBI. This rumor led to fear and chaos among the customers of these banks, with many rushing to withdraw all their hard-earned money and shifting to some other bank. Apart from the fear and chaos among common people, it also led to undue losses for these banks.

Therefore, the detection of such misinformation has become the need of the hour. This paper proposes a solution to determine the probability of a tweet being a rumor based on various parameters and present the results in a presentable manner.

2 Related work

Recently, there has been a lot of attention towards detection of rumors, and there have been a lot of methods proposed to do the same. Some methods rely on human efforts while others rely on complex computations to get the desired outcome. Examples of using human-effort for detecting rumors are websites like snopes.com and factcheck.org. These sites have teams of people who investigate the veracity of the rumors by looking up facts and evidence from various sources. Though these methods are very accurate, they are unable to investigate queries from every person on earth. This is where computation-intensive methods come in.

Earlier methods have treated rumor detection as a binary classification problem. One of the examples for rumor being a binary problem is shown in [12]. Slowly, there was a shift towards determining the probability instead of classifying into two buckets. For instance, in [9], the authors proposed such a rumor detection approach. They identified tweets that contain enquiry patterns (the signal tweets), made clusters based on content, created a statement that summarizes the cluster, and then used that statement to pull back in the rest of the non-signal tweets that discuss that same statement. Finally, the candidate rumor clusters were ranked based on various statistical features, in decreasing order of the likelihood of a statement being a rumor.

The approach followed in [13] was a parameter-driven approach, i.e., the detection is dependent on some predefined parameters to better detect different kinds of rumor spread. The authors identified characteristics of rumors on twitter by examining linguistic style used to express rumors, characteristics of people involved in propagating information, and the network propagation dynamics. Then, they noted the key differences in each of these characteristics for the spread of rumor and non-rumor. Finally, these differences were used by a Hidden Markov Model to predict the veracity of real-time tweets.

For keyword extraction, hashtag analysis method was used which outperformed most of the traditional methods. TextRank [5] in which the sentence was tokenized and classified into POS tags. The drawback of this method was only some of the mentioned POS tags could be used as keywords. TF-IDF [7] in which generated keywords based on the frequency of their occurrence. Not all significant keywords have high occurrences in a tweet. SVM Rank [6] is the combination of TextRank [5] and TF-IDF [7], however combining them does not necessarily overcome their respective disadvantages. Word Embedding and Clustering [4] consider semantic relationships for keywords, however it requires a significant amount of pre-processing to be done before the keywords can be extracted.

[1] and [10] are both based on stance detection. Both papers have classified stance on replies and not related tweets. Classes used in these papers are Support, Deny, Comment and Query. The major issue was class imbalance which occurred due to mis-classification of most tweets as “Comment”. Furthermore, their models had difficulty in detecting the stance “Deny”. Rumorlens [6] has not considered many of the parameters such as Tense of a tweet, verified users in related tweets which is the deciding factor, for classification of tweet as a rumor.

3 Dataset

3.1 For Stance Detection

In this step, two datasets were used. The first one was the Sentiment140 dataset [19].

As for the second dataset, it was created manually by taking some controversial tweets and finding related tweets using the Twitter API. It also included the majority of tweets from the PHEME dataset [18]. These tweets were divided into 3 categories and labelled as FOR, AGAINST and NONE. This accounted for a total of 1980 tweets for the stance detection model.

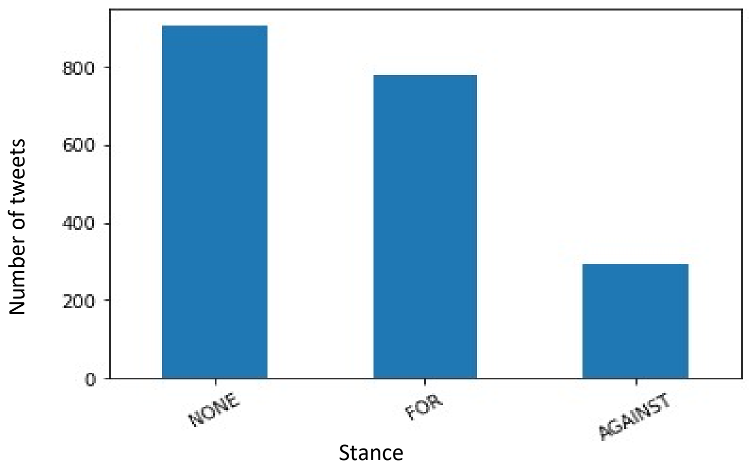


Fig. 1. Class Distribution of dataset

3.2 For Final Model

Data from the PHEME dataset [18] was extracted. Then, for each tweet, the stance was determined using the stance detection model. Dataset consisting of all parameters required for the final model was then formed. This dataset consisted of around 4670 tweets.

4 Architecture

The proposed architecture of our system is shown in figure 2. The user will input a tweet URL in the system. From the input URL, various parameters such as the text of the tweet, hashtags in the tweet, retweet count of the tweet, followers and following count of the user, tense of the tweet and whether the user is verified or not will be extracted.

Using the hashtags extracted, related tweets will be fetched and then the stance of each tweet will be predicted with respect to the source tweet. There will be 3 stances viz. for, against and none.

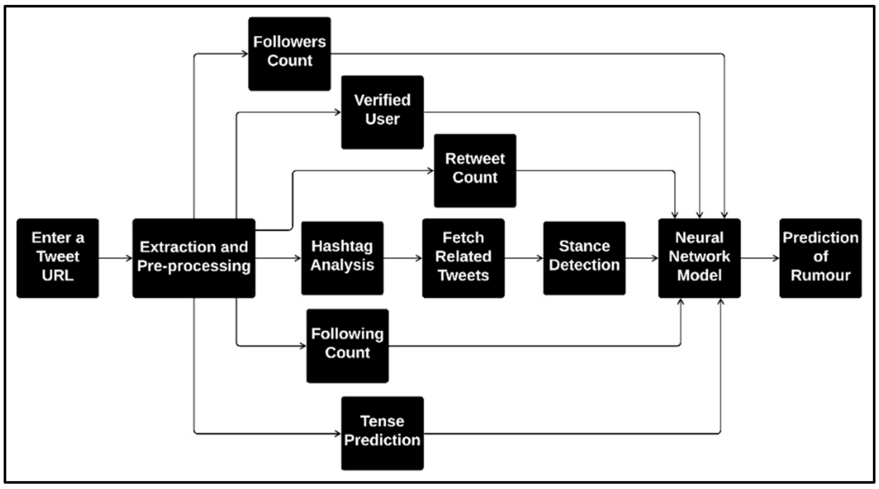


Fig. 2. Trumer Architecture

After this, the parameters extracted earlier along with the stance count is given to a neural network which we have trained to calculate the probability of a tweet being a rumor. Each of these steps are explained in detail in section 5.

5 Implementation

The proposed methodology can be divided into four stages.

5.1 Pre-processing:

I. **Extracting the basic parameters like followers count, verified_status, retweet_count, etc. from the tweet URL.**

The input as the URL of the tweet is taken as an input. First step is to extract the tweet id from the URL. Then using the Twitter API, the following parameters for the tweet are extracted.

- **Tweet text:** This will be useful further for tense detection and stance detection.
- **Verified status:** This will provide a sense of additional validity to the content of the tweet. If the tweeting user is a verified news source or a well-known personality in general, then it is less likely that the tweet is just a rumo r, not backed by any solid evidence.
- **Retweet count:** Usually, retweeting a tweet implies that the user agrees with the post and possibly believes that it is

true. Hence, a higher retweet count might reduce the chances of the tweet being a rumor.

- **Followers count:** The followers count helps to analyze the reach of a user. Generally, rumors are not started by famous users as it might reduce their own credibility if it is found to be a rumor. Hence, a higher followers count would reduce the chances of the tweet being a rumor.

II. Finding tweets related to the source tweet from twitter.

To perform stance detection in Stage 5, the first step is to find a way to fetch related tweets. The most important task here is to extract the keywords, which can be used to search Twitter API to fetch other related tweets.

<u>For this, two methods are used:</u>

1. Hashtag & Proper noun Analysis Method:

A single tweet might not have many hashtags or proper nouns. To counter this, an algorithm is designed which is based on a dual search pattern. Here related tweets are searched for, twice. For the first time, the search is for extracting the major hashtags and proper nouns and the second time to find related tweets using that.

A hashtag is a great tool to easily identify the tweets belonging to a particular topic. As for Proper Nouns, almost all of the tweets would be concerning some person, country, event and so on. Hence, the proper noun will help us to further focus our search with respect to that entity.

A hashtag is an identifier which is written to easily separate content based on a theme/topic. It includes a '#' character followed by one or more words, with no spaces in between. Hashtags are generally used on social media platforms like Twitter, Facebook, Instagram and so on.

- Searching for a particular hashtag helps one discover information on a particular topic, easily.
- Hashtags can also be used to direct information to some target audience.
- They are increasingly used by more and more platforms, thereby indirectly connecting the different sources of information.

This makes Hashtags one of the major parameters to find related tweets. Hashtags and Proper Nouns are extracted from the tweet text. After getting a pool of extracted hashtags and proper nouns, the Twitter API is used to fetch related tweets using this pool. As a result, there would be a bigger collection of tweets to search for the hashtags and proper nouns. Finally, after extracting all the hashtags from this expanded collection of tweets, they are ranked in decreasing order of occurrence.

Lastly, the most frequent hashtags are considered to fetch related tweets using the Twitter API.

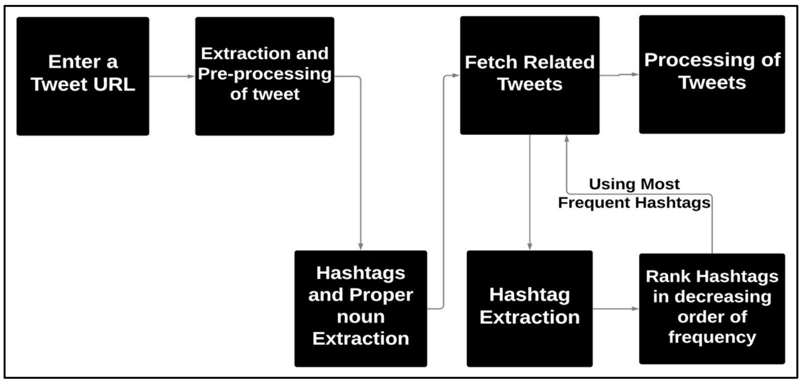

Enter a Tweet URL
Extraction and Pre-processing of tweet
Fetch Related Tweets
Processing of Tweets
Using Most Frequent Hashtags
Hashtags and Proper noun Extraction
Hashtag Extraction
Rank Hashtags in decreasing order of frequency

Fig. 3 Flow diagram of how we obtain the final set of related tweets using hashtags and proper nouns.

2. MonkeyLearn API [14]

In the rare cases where the tweet has neither of Hashtags or Proper Nouns, this method is used to get keywords. On sending the tweet text, the API responds by providing the keywords along with their confidence scores. The keywords with confidence scores greater than 0.5 are considered. Considering these keywords, the Twitter API is used to fetch the related tweets. One of the major advantages of monkeylearn is the identification of key phrases consisting of two or more words. Furthermore, it also provides a confidence level for each keyword. Also, there is no need to pre-process the sentence given for the keyword extraction.

III. Filtering out the tweets based on how much they are related to the source tweet.

Even after fetching tweets using appropriate keywords, there is a possibility that some totally unrelated tweets might get through. To tackle this problem, cosine similarity [20] is used to further determine the level of relatedness of the source tweet with the fetched tweet. So, among the fetched tweets, after removing the stopwords from the tweet the cosine similarity of each tweet is calculated with source tweet. From those, all the tweets are considered which are even very little similar to the source tweet. All those tweets which are completely different are ignored and have cosine similarity of 0. This ensures that completely irrelevant tweets are ignored and tweets having very less similarity are not missed from the consideration for stance detection, thereby yielding better results.

5.2 Identifying the tense of the tweet.

In the second stage, the tense of the tweet is determined. In order to achieve this, first the structure of the sentence is found. For this task, the BLLIP Parser (also known as the Charniak-Johnson parser or Brown Reranking Parser)[9] and the WSJ-PTB3 (WSJ-Penn TreeBank-3)[3] model are used. BLLIP Parser not only has an f-score of 0.91 but also structures sentences as a tree, which allows it to find the tense of multiple sentences with great accuracy[9]. After getting the structure of the sentence, Breadth-First Search is used to find the main verb of the sentence. Finally, the POS-tag of this main verb is used to classify the tweet into Present, Past or Future Tense.

The theory behind this step is that tweets posted in future tense have a higher chance of a tweet being a rumor due to the uncertainty concerning it, and also less availability of any evidence as proof of the event. Now, one might ask that rumors can also be in the past tense, how can one ignore that? Yes, it is possible that the rumor might be in the past tense as well. But, there is a way to counter that. As it is a past event, there will be some evidence supporting or denying it. Some people will know about these facts and can voice their stance by tweeting for/against it. These stances are analyzed in Stage 5, thereby handling the Past Tense Problem.

5.3 Detecting the stance of the related tweets and classifying them into one of FOR, AGAINST, NONE.

Next step is to determine the stance of each related tweet w.r.t to the original tweet. For this task, the ULMFiT[11] algorithm is used for training the model. To train a model for stance detection on tweets, a considerably large dataset would be required. Since the ULMFiT algorithm was used, it was possible for us to achieve considerable results with a relatively smaller dataset. The algorithm uses Transfer learning for text classification. Hence, a large generalized dataset is used to learn and understand the general structure of the language, followed by a smaller dataset focusing on the specific task of stance detection. The datasets used are mentioned in section 3.1.

Table I: Results for stance detection model

FOR			AGAINST			Accuracy
Precision	Recall	F1- Score	Precision	Recall	F1- Score	
0.72	0.63	0.68	0.44	0.32	0.37	64.89%

The results shown in Table I are obtained after applying the ULMFiT algorithm for the Stance detection task. The table contains Precision, Recall, F1-Score values for 'FOR' and 'AGAINST' categories, along with the overall accuracy of the model.

5.4 Final Model:

To determine the probability of the tweet being a rumour, a model was developed which will take followers count, a user is verified or not, retweet count, number of related tweets in support, number of related tweets against the input tweet, number of related tweets which are neutral, number of tweets which are tweeted by verified user and are in support, number of tweets which are tweeted by verified user and are against the input tweet and tense of input tweet as input and give final value for the tweet being a rumour. All the input parameters were normalised. The dataset used to train this model is explained in detail in section 3.2.

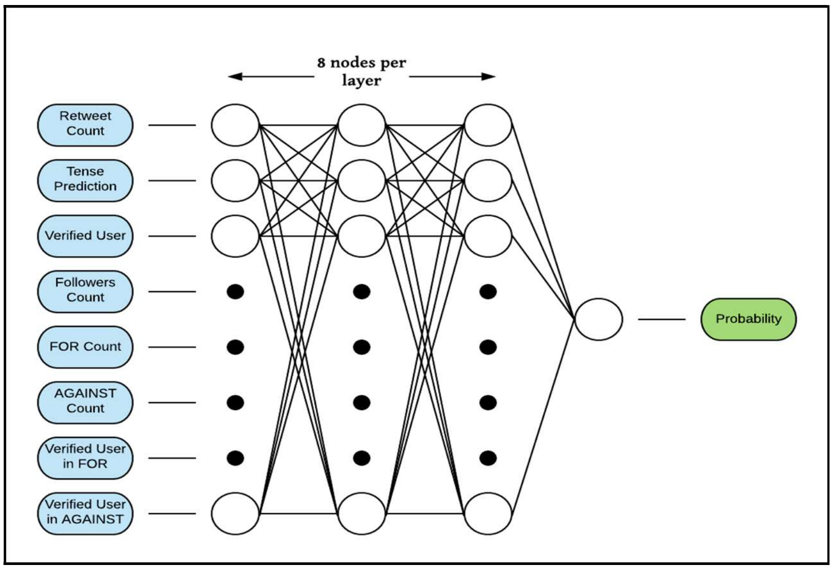


Fig. 4. Neural Network Model Architecture

Initially, a basic Neural Network model was created with 3 hidden layers. Sigmoid was used as the activation function as the output value had to be between 0 and 1 for the probability. The same number of input and output dimensions were kept throughout the network. The loss was set to “mean_squared_error” and optimizer as “adam”. In this scenario, an accuracy of 46% was achieved for 10 epochs which increased to 53.51% for 1000 epochs. To further increase the accuracy, more layers were added, activation function was changed to “ReLU” for all the layers except the final output layer. By doing this an accuracy of 62.48% was achieved. Looking at the data again an observation was made that the number of related tweets which are neutral is making the dataset biased, so that parameter was dropped. By doing this an accuracy of 74.97% was achieved which was the best. However, one of the issues with the Neural Network models was the concentration of probability values close to 0.5. Even for the most obvious instances, the probabilities would rarely cross 0.6. So, instead of neural network, logistic regression was tried, as the probability values were more linearly spread. As input features, normalized values were taken for followers count, retweet count, verified status, supporting tweets, denying tweets, verified supporting tweets, verified denying tweets, neutral tweets and the tense of the tweet. The logistic regression model was trained and an accuracy of 61.24% was achieved. Then, similar to the neural network model, neutral tweets input feature was removed. The accuracy jumped up to 74.84%. Thus, the accuracy was comparable to the neural network model, while having more linear values for probability. To increase the accuracy even further the SMOTE technique was tried to deal with imbalanced classes in the dataset. However, the results actually weren’t as expected and the accuracy dipped to 68.59%. Then the SMOTE technique was used on the neural network. Thus, an accuracy of 77.99% was achieved.

Table II: Results of the final model

Precision	Recall	F1-Score	Accuracy
0.7294	0.8988	0.8038	77.99 %

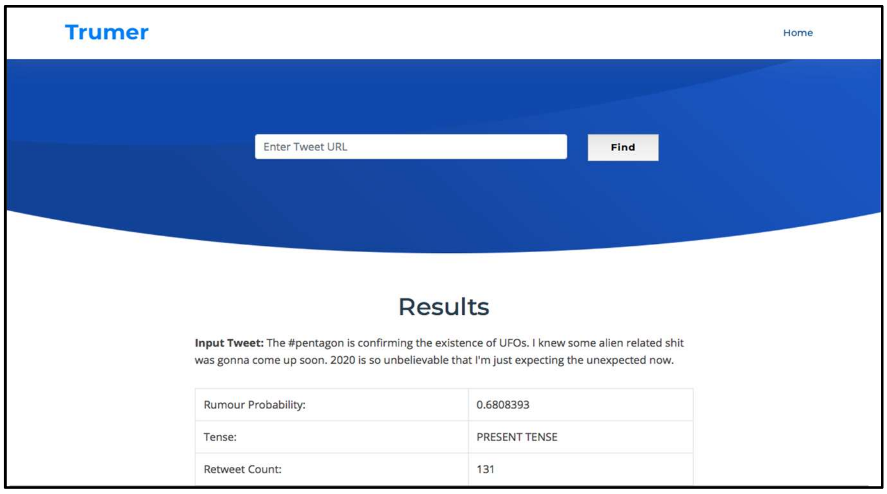
The results of the final model are shown in Table II. The table contains Precision, Recall, F1-Score and the overall accuracy of the model.

6 Results

This method has achieved substantially better results in comparison to SEM-EVAL 2017 results [22] [23]. There were 5 teams which participated in the event and achieved scores 0.393, 0.464, 0.286, 0.536, 0.536 respectively with a baseline score of 0.571.

We tested our application on a few URLs. On tweet "The #pentagon is confirming the existence of UFOs. I knew some alien related shit was gonna come up soon. 2020 is so unbelievable that I'm just expecting the unexpected now" [17]. It shows the rumor probability of 0.6808393 i.e. 68%. It also displays various other parameters such as the followers count, retweet count, if the user is verified or not.

Fig 5(a) shows the output of our system. It displays the tweet text, rumor probability i.e. 0.6808393 along with Tense of the tweet i.e. Present Tense and retweet count i.e. 131.

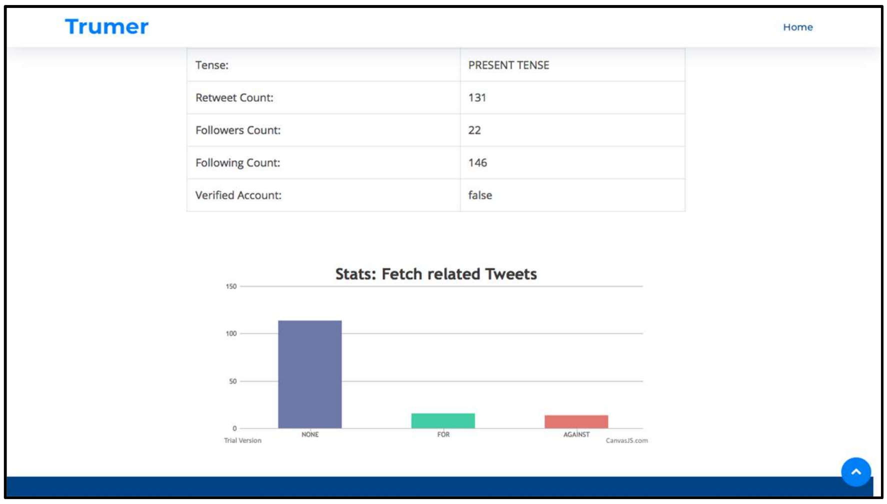


Results

Input Tweet: The #pentagon is confirming the existence of UFOs. I knew some alien related shit was gonna come up soon. 2020 is so unbelievable that I'm just expecting the unexpected now.

Rumour Probability:	0.6808393
Tense:	PRESENT TENSE
Retweet Count:	131

Fig 5(a). Test results - 1



Tense:	PRESENT TENSE
Retweet Count:	131
Followers Count:	22
Following Count:	146
Verified Account:	false

Fig 5(b). Test results - 2

Fig 5(b) shows the other parameters of the result such as Followers count, Following the count, Verified account along with stance detection parameters such as For, against, and neutral.

7 Formulae

The formulae used are as follows:

$$\boldsymbol{P} = \frac{(\boldsymbol{TP_{C1}} + \boldsymbol{TP_{C2}} + .. + \boldsymbol{TP_{Cn}})}{((\boldsymbol{TP_{C1}} + \boldsymbol{TP_{C2}} + .. + \boldsymbol{TP_{Cn}}) + (\boldsymbol{FP_{C1}} + \boldsymbol{FP_{C2}} + \cdots \boldsymbol{FP_{Cn}}))}$$

$$\boldsymbol{R} = \frac{(\boldsymbol{TP_{C1}} + \boldsymbol{TP_{C2}} + .. + \boldsymbol{TP_{Cn}})}{((\boldsymbol{TP_{C1}} + \boldsymbol{TP_{C2}} + .. + \boldsymbol{TP_{Cn}}) + (\boldsymbol{FN_{C1}} + \boldsymbol{FN_{C2}} + \cdots \boldsymbol{FN_{Cn}}))}$$

$$\boldsymbol{A} = \frac{(\boldsymbol{TP_{C1}} + \boldsymbol{TP_{C2}} + .. + \boldsymbol{TP_{Cn}}) + (\boldsymbol{TN_{C1}} + \boldsymbol{TN_{C2}} + .. + \boldsymbol{TN_{Cn}})}{Total}$$

Where,
P is Precision
R is Recall
A is Accuracy
TP_{Cn} is True Positive of nth class
FP_{Cn} is False Positive of nth class
FN_{Cn} is False Negative of nth class
Total is the sum of True Positive, True Negative, False Positive and False Negatives

$$\text{cosine similarity}(\mathrm{A}, \mathrm{B}) = \cos\theta = \frac{A \cdot B}{||\mathrm{A}|| * ||\mathrm{B}||}$$

8 Conclusion

Rumor Detection has become the need of the hour in today's world. In this paper, we have been able to determine the rumor probability of a tweet with considerable accuracy using various parameters such as Stance of Related Tweets, Verified Status of the twitter user, retweet count, and so on. Hence, with this application, we have provided a way for the user to check the veracity of the tweet before believing it and hope to alleviate the issue of spreading misinformation.

9 Future Work

In the future, this method can be further extended to other social media platforms where certain Twitter-specific parameters like Retweet Count, Verification Status, etc. might not be available. Secondly, our system only checks the veracity of the tweets which are input by a user. It can be expanded to look for rumors on the entire platform, thereby also detecting rumorous tweets which have not been checked by the users. Finally, our work is only considering English language tweets. This could be implemented for the major languages of the world.

We are planning to include comments for processing and determining the probability of a tweet being a rumor. Along with comments we also plan to implement image and video processing algorithms. We are also thinking of optimal ways to explore various external links which are generally mentioned in the tweets.

References

1. Endang Wahyu Pamungkas, Valerio Basile, Viviana Patti: *Stance Classification for Rumour Analysis in Twitter: Exploiting Affective Information and Conversation Structure*
2. Twitter Usage statistics https://www.internetlivestats.com/twitter-statistics/
3. Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, Ann Taylor : *Treebank-3*
4. Ping Zeng, Ying Yan & Jianjun Xu: *Automatic Keyword Extraction Using Word Embedding and Clustering* (ICCSEC -2017)
5. Papis Wongchaisuwat: *Automatic Keyword Extraction Using TextRank* (2019 IEEE)
6. Xinggao Cai & Shujin Cao: *A Keyword Extraction Method Based on Learning to Rank*
7. Sungjick Lee, Han-joon Kim: *Automatic Keyword Extraction from News Articles Using TF-IDF Model*
8. Paul Resnick, Samuel Carton, Souneil Park, Yuncheng Shen, Nicole Zeffer University of Michigan School of Information Telefonica Research: *RumorLens: A System for Analyzing the Impact of Rumors and Corrections in Social Media*
9. Eugene Charniak and Mark Johnson: *Coarse-to-fine n-best parsing and MaxEnt discriminative reranking*
10. Elena Kochkina, Maria Liakata, Isabelle Augenstein: *Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM*
11. Jeremy Howard and Sebastian Ruder: *Universal Language Model Fine-tuning for Text Classification*

12. Vahed Qazvinian Emily Rosengren Dragomir R. Radev Qiaozhu Mei: *Rumor has it: Identifying Misinformation in Microblogs*
13. Soroush Vosoughi, Mostafa 'Neo' Mohsenvand, and Deb Roy: *Rumor Gauge: Predicting the Veracity of Rumors on Twitter*
14. MonkeyLearn API: https://monkeylearn.com/
15. Yang Liu, Songhua Xu, and Georgia Tourassi: *Detecting Rumors Through Modeling Information Propagation Networks in a Social Media Environment*
16. Tetsuro Takahashi Nobuyuki Igata: *Rumor detection on twitter*
17. Test tweet URL https://twitter.com/Adderalcoholic/status/1254937695111073793
18. Pheme Dataset link: https://figshare.com/articles/PHEME_dataset_for_Rumour_Detection_and_Veracity_Classification/6392078
19. Sentiment140 Dataset link: https://www.kaggle.com/kazanova/sentiment140
20. Cosine Similarity link https://medium.com/@sumn2u/cosine-similarity-between-two-sentences-8f6630b0ebb7
21. Kwak, H., Lee, C., Park, H., Moon, S.: *What is Twitter, a social network or a news media?* In: International Conference on World Wide Web, Raleigh, USA, pp. 591–600 (2010).
22. Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga: *RumourEval 2019: Determining Rumour Veracity and Support for Rumours*
23. Leon Derczynski and Kalina Bontcheva and Maria Liakata and Rob Procter and Geraldine Wong Sak Hoi and Arkaitz Zubiaga: *SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours*