Fake News Detection on Twitter

Karan Mankar
New York University
Email: kmm1110@nyu.edu

***Abstract*— Information quality on social media is of great concern now that more and more of our lives are irrevocable tied to one or the other of those web platforms. This project is an attempt to develop a project that can automate the detection of fake news over twitter and flag it for human moderators. The datasets used are Politifact dataset and the Pheme Rumor dataset, both of which have tweets labelled as real news or fake news by fact checking websites and journalists. The data is cleaned, tokenized, normalized and then trained on classification models to get the performance metrics which can then allow us to test our hypothesis and then deploy the model to the real world if our hypothesis hold true.**

I. INTRODUCTION

Social media is a big part of our daily life today as most of our free time is spent in the digital domain. Social media content influences a lot of our life, right from our fashion sense and choice of restaurants to our discourse on the country's politics and public policies. Our heavy reliance on microblogging sites to present our views or opinions as well as follow influential personalities has off late given rise to the growing problem of misinformation peddling over such websites. Right from the exaggerated claims over the Hillary emails to the NHS bus that was roaming the streets of London and which was heavily publicized by the pro-Brexit camp using Twitter, the problem of fake news to shape the public discourse is of growing concern. Apart from malicious news content, it is also seen that armies of bots controlled by a group of like-minded individuals of organizations are instrumental in peddling such disinformation and giving it the credence necessary to be then passed of the actual news that people tend to believe. India's infamous 'IT cell', backed indirectly by the federal government to disseminate pro government propaganda is a prime example of such coordinated efforts.

This paper aims to use Machine Learning algorithms such as NLP, Anomaly Detection and Text Classification to flag suspicious tweets or news articles. The predictions need to be done with good amount of certainty to then be passed on to dedicated fact-checking teams to confirm the presence of misinformation and immediately act against such articles. The entire project is done based on the CRISM-DM methodology. Computational methods have proven useful in similar contexts where data volumes overwhelm the human analysis capabilities [1]. The system proposed in this paper aims to identify fake news based on the features of a tweet along with the number of times it has been retweeted, the sentiment score of the tweet [2] and the source of the news article. The data used to train and test the model has been collected and labelled by fact checking website Politifact [3] as well as independent journalists and the data is made available through various online sources [4]. The result taken over three different classification algorithm both from the supervised and unsupervised learning domain provide proof regarding our hypothesis proposed by the paper.

II. LITERATURE REVIEW

Research in the field of fake news detection has been intense in the recent years with development of techniques to detect misinformation over social media using Machine Learning and Deep Learning techniques. [5] uses Deep Learning techniques such as LSTM, Convolution Neural Networks, BERT transformer to gain an accuracy of approximately 90% in predicting fake news using data acknowledged as fake news and from reputable tabloids like the New York Times and Washington Post.

[6] suggests using AI-powered analytic tools such as stance-classification to determine whether the headline of the news matches the body, text processing to analyze the author's writing style and image forensics to detect photoshop use. Furthermore, Real time anomaly detection be used to detect anomalies in the author's style of writing indicating a pre-determined text that the owner of the handle just posted after being told to.

While [7] focuses on the same techniques mentioned previously, it highlights the grave issue of lack of enough reliable open-sourced data to make a good misinformation detection system using Machine Learning. It also has created a dataset MisInfoText and asked fellow Data Scientist to contribute to it apart from collecting data from various fact checking websites.

III. BUSINESS UNDERSTANDING

Fake news or Information quality over social media in general is of increasing concern as our reliance on the web platforms grows. Today, social media portals have infiltrated every aspect of our lives as well as control a major portion of it. Apart from entertainment, these portals have recently been shown to be instrumental in coordinating relief effort in times of disaster or natural calamity. During the recent second wave of the coronavirus pandemic, social media influencers in India played a critical role in connecting patients with live saving drugs and oxygen supplies when the country was facing a historic shortage of both these commodities. The presence of fake news during such times could make a already terrible situation worse and so detecting Fake news before it is widely circulated and entrenched into the collective psyche of the masses is of paramount importance.

Through the report a system is proposed that would analyze the tweet content being posted online in real time and look at the retweet patterns along with source information of the news article. This information would them be passed through a machine learning model to classify it as real news or fake news allowing human moderators to take over from there to recheck the piece of information and take a final call whether to classify the news item as fake or not. Unlike other systems which heavily rely on Deep Learning models or user information, the project proposes to build a model only with the help of the source information, tweet text and number of retweets being fed into the system. The hypothesis for such a fake news classifier will be tested through this project. The data will not only contain text information that will require NLP techniques to make them acceptable to any classification models but also numerical data such as sentiment score and length of the tweet, along with the number of times the tweet has already been retweeted.

IV. DATA UNDERSTANDING

The project uses two different datasets collected from two different sources, both containing tweets and their metadata along with a classification done by either a fact-checking website or through trusted fact-checkers. The project makes use of politifact-dataset created while building another such Fake News classifier called FakeNewsNet [8]. The github repository it has code that can be deployed to scrape the latest iteration of the dataset, however since the latest iteration was the same as the already made .csv file present in the repository [3], the politifact-dataset was downloaded for the project. The dataset consists of tweet title, news urls, and tweet ids apart from the row id already present in the data. The tweet ids column was a collection of tweet ids that had retweeted the original tweet with a tab separator. The dataset was fact-checked by fact checking website www.politifact.com and the data was divided into two different folders, each containing either fake or real tweets as labelled by politifact.com [8].

The other dataset used was the Pheme Rumor Dataset [4]. The data was present in the form of a '.tar.gz' file which needed to be downloaded and extracted. The file contains tweets and reactions to those tweets from five major world events. The data is further divided into rumors and non-rumor tweets which can be accessed by clicking on the folder with the source-tweet id. The folder contains reaction folder and source-tweet folder that has the source-tweet information in the form of a JSON file and similarly the reaction folder has a similar structure. The diagram below shows the structure of the Pheme Rumor Dataset:

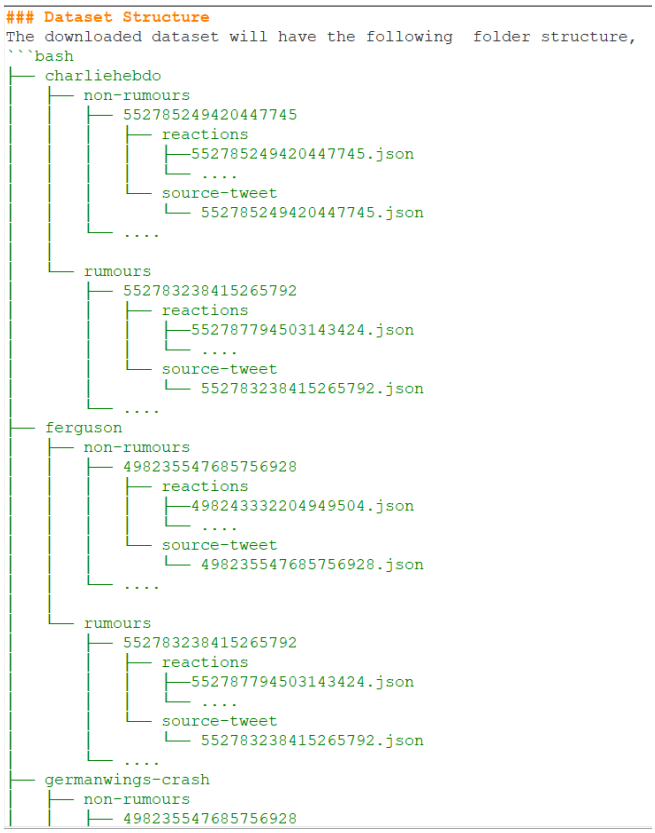

Figure 1

V. DATA PREPARATION

Since the politifact and pheme datasets were not in the same format, the first thing that needed to be done was to get them into the same format. To do this the pheme dataset needed to be explored and relevant fields extracted and made into a new table that would match the structure of the structure of the politifact dataset. To do that a Python script was written, that would explore all the directories, subdirectories, and folder of all the events of the pheme dataset to gather relevant information. The tweet information was stored in the form of a JSON file and so the JSON file had to be opened and fields like 'Text', 'display url' and 'retweets' needed to be extracted to make the database of the same structure. Once the data was extracted from the pheme dataset, work began on the politifact dataset which required conversion of tab separated tweet ids into number of retweets. This was accomplished by splitting the column values into a list and then counting the number of elements in the list and returning only the final number.

The ids from both the datasets were dropped and cleaning began on the text column for undesirable elements like @-mentions, emojis, hashtags etc [9]. Since the politifact dataset was relatively clean that did not require much cleaning, however the pheme dataset was still having raw tweets and so all the text processing and data cleaning. Once both the datasets were cleaned and in similar format the data was joined together to create a new final database that would be used for modeling. Next using NLTK, stop words and punctuations of the text column. Next, Sentiment Analysis is done on the text field using vaderSentiment library [10]. The removal of stop words is important for the next step to create a bag of words for vectorizing the input for the classification algorithms to use them and train. The sentiment analysis score is returned back as a dictionary of four values, positive, negative, neutral and compound. The probability values and present in the dictionary which add to a total of one. The sentiment is identified by setting a threshold of 0.5 taking the key whose value is above the threshold. Finally, a trivial operation of having the length of each tweet and the length is added as a feature along with the sentiment score to the dataset. The news_url column similarly is cleaned for all the strings ahead of the first forward slash to only keep the main page name in the URL field. Words such as http or https or www are removed from the URL too.

VI. EXPLORATORY DATA ANALYSIS (EDA)

Once the data is cleaned and joined together to create the larger final dataset, we can do exploratory data analysis (EDA) to get a basic understanding of the data and see if there are any patterns arising between the different features and the label. We can garner insights from such analysis from the start even before any modeling or prediction is done.

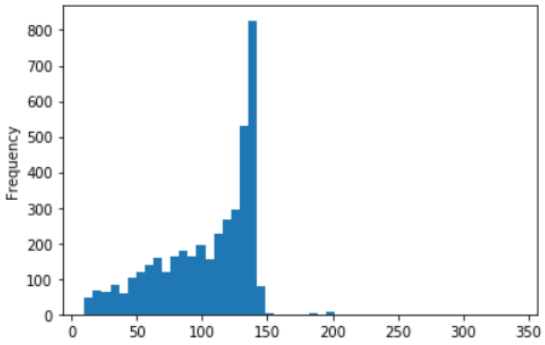

Figure 2 Length of all the tweets

The above figure gives us an idea that majority of the tweets are in the length range of 100 to 150 words no matter their classification.

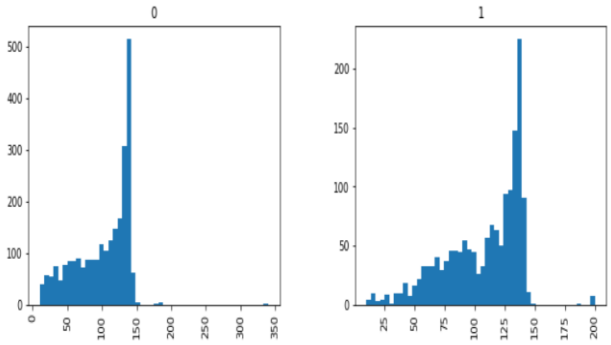

Figure 3 (Length vs Label)

The above histogram reveals to us that there is not direct connection between the length of the tweets and whether they are fake or true. Since the data here show similar distribution for both classes with the highest length as seen above being in the 100-150 range.

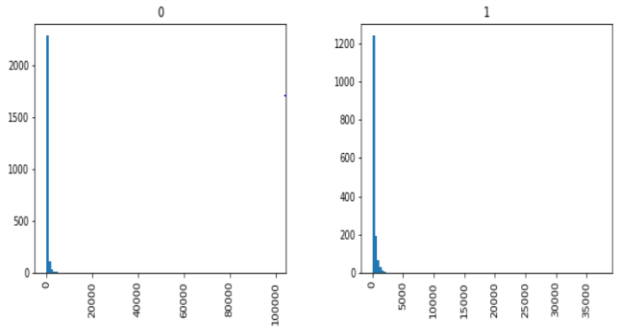

Figure 4 (No_of_retweets vs Label)

Quite similar to Figure 3, even in Figure 4 there is not conspicuous pattern difference between both the classes which leads us to believe that maybe the no_of_retweets have no direct correlation with the tweet being real or fake.

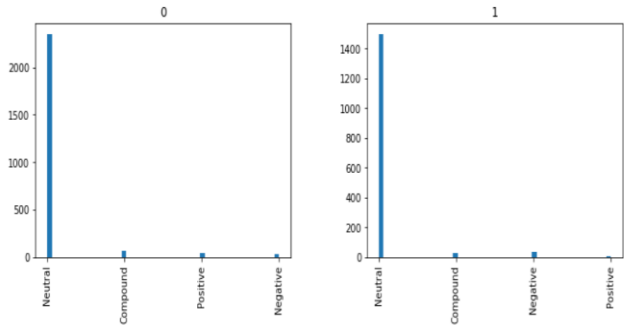

Figure 5 (Sentiment score vs Label)

Figure 5 shows that most of the tweets are generally neutral in nature and the pattern is the same across both the classes, real and fake. The sentiment of the tweets is generally neutral since it is a news article dataset and news articles tend to be neutral in sentiment.

VII. MODELING

After data preparation and cleaning are done, the next step is to vectorize the data. The easiest way to do this is to use the Bag of Words Vectorizer [11]. The text data can be vectorized using Bag of Words or CountVectorizer or tfidf vectorizer. The vectorization algorithms are applied on text data to convert and normalize text data to numerical form which are them accepted by the classification algorithms for modeling. [11]. After converting the text data to bag of words (BOW) vectors, the vectors are further normalized using TF-IDF algorithm. TF-IDF stands for Term

Frequency-Inverse Document Frequency where Term Frequency basically means the frequency of the word appearing in a corpus and the Inverse Document Frequency means logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears [12]. These vectorizers output a Sparse Matrix of the text data.

	Message 1	Message 2	...	Message N
Word 1 Count	0	1	...	0
Word 2 Count	0	0	...	0
...	1	2	...	0
Word N Count	0	1	...	1

Figure 6 (Sparse Matrix) [12]

The Sparse Matrix is then used to train the model to get the predictive output. In case of the data that pertains to this project, there are two text fields that means that the vectorizer needed to be applied twice the Sparse Matrix joined to get the whole normalized corpus.

```
print('Shape of Sparse Matrix: ', tweets_bow.shape)
print('Amount of Non-Zero occurences: ', tweets_bow.nnz)
```

```
Shape of Sparse Matrix:  (4057, 9479)
Amount of Non-Zero occurences:  37214
```

```
print('Shape of Sparse Matrix: ', news_url_bow.shape)
print('Amount of Non-Zero occurences: ', news_url_bow.nnz)
```

```
Shape of Sparse Matrix:  (4057, 1356)
Amount of Non-Zero occurences:  4057
```

Figure 7 (Sparse Matrices)

For the modeling part, the data is trained on three models, two supervised and one unsupervised clustering algorithm. Supervised models always need a label to train while an unsupervised model can work without a variable and needs to cluster datapoints to discern patterns in the data. For the purpose of the project the three models used were

- Naïve Bayes
- Support Vector Machines
- K Means Clustering

After the application of the models the output of different models were as follows:

- Naïve Bayes:

Accuracy	Precision	Recall	F-Score
0.42	0.47	0.48	0.40

```
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
[[148 584]
 [117 369]]
              precision    recall  f1-score   support

           0       0.56      0.20      0.30       732
           1       0.39      0.76      0.51       486

    accuracy                           0.42      1218
   macro avg       0.47      0.48      0.40      1218
weighted avg       0.49      0.42      0.38      1218
```

- Support Vector Machines [13]

Accuracy	Precision	Recall	F-Score
0.60	0.60	0.51	0.39

```
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test, predictions))
```

```
[[726   6]
 [477   9]]
              precision    recall  f1-score   support

           0       0.60      0.99      0.75       732
           1       0.60      0.02      0.04       486

    accuracy                           0.60      1218
   macro avg       0.60      0.51      0.39      1218
weighted avg       0.60      0.60      0.47      1218
```

- K Means Clustering

Accuracy	Precision	Recall	F-Score
0.60	0.30	0.50	0.38

```
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test, predictions))
```

```
[[732   0]
 [486   0]]
              precision    recall  f1-score   support

           0       0.60      1.00      0.75       732
           1       0.00      0.00      0.00       486

    accuracy                           0.60      1218
   macro avg       0.30      0.50      0.38      1218
weighted avg       0.36      0.60      0.45      1218
```

VIII. EVALUATION

After the results of the modeling phase are out in the next phase the models are evaluated and the best model is chosen. In this case Support Vector Machine had the best overall accuracy and other parameters. Support Vector Machine generates an optimal hyperplane in multidimensional space. [14]. Since neither of the model was able to make strong predictions and show good performance the hypothesis that fake news can be predicted using just the text and number of retweets may be incorrect.

IX. PROBLEMS FACED

One of the biggest problems faced during this project was the availability of labeled datasets. Since now even politifact API is not public getting updates on the politifact dataset becomes impossible. A lack of data also means that modeling must be done on a small subset of the actual fake news corpus in the world and that definitely affects the accuracy of the model. Furthermore, applying TF-IDF on two columns instead of the normal one was a problem at the start with respect to the procedure to join the columns back to the original database to use the other features in the computation too.

X. CONCLUSION

This work demonstrates an automated system for detecting fake news on Twitter. The model and data need to be improved upon to get better prediction values and squeeze every bit of performance out of these models. It may be pertinent to experiment more with Support Vector Machine and probably have more metadata related to tweets in the dataset. However, the system can still identify and denote fake news faster than a crowd-sourced worker or a journalist making the process way cheaper to classify fake news on Twitter. These results might also be of value in future studies made on the topic and could be a launching pad for taking the research further and having better performing models.

XI. REFERENCES

[1] J. G. Cody Buntain, "Automatically Identifying Fake News in Popular," 2018. [Online]. Available: https://arxiv.org/pdf/1705.01613.pdf.

[2] M. C. R. K. P. K. M. W. Sebastian Kula, "Sentiment Analysis for Fake News Detection by Means of Neural Networks," 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-50423-6_49.

[3] D. M. Kai Shu, "Github," [Online]. Available: https://github.com/KaiDMML/FakeNewsNet.

[4] G. W. S. H. M. L. R. P. Arkaitz Zubiaga, "Figshare," [Online]. Available: https://figshare.com/articles/dataset/PHEME_dataset_of_rumours_and_non-rumours/4010619.

[5] 1. A. G. a. P. N. Rohit Kumar Kaliyar, "FakeBERT: Fake news detection in social media with a BERT-based deep learning approach," *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7788551/,* 2021.

[6] Z. I. S. U. A. M. G. S. C. a. B. -W. M. Umer, "Fake News Stance Detection Using Deep Learning Architecture (CNN-LSTM)," *IEEE Access,* vol. 8, pp. 156695-156706, 2020.

[7] M. Y. S. Y. ,. a. M. O. A. Iftikhar Ahmad, "Fake News Detection Using Machine Learning Ensemble Methods," *Complexity*, vol. vol. 2020, no. 2020, p. 11 pages, 2020.

[8] D. M. S. W. D. L. H. L. Kai Shu, "FakeNewsNet: A Data Repository with News Content, Social Context and Spatialtemporal Information for Studying Fake News on Social Media," *arXiv:1809.01286,* 2019.

[9] S. Galeshchuk, "Working with Twitter Data in Python," 2019. [Online]. Available: https://medium.com/analytics-vidhya/working-with-twitter-data-b0aa5419532.

[10] "vaderSentimen," [Online]. Available: https://pypi.org/project/vaderSentiment/.

[11] M. Nandu, "Natural Language Processing in Python with Code (Part I)," 2019. [Online]. Available: https://medium.com/@meetnandu996/natural-language-processing-in-python-with-code-part-i-7736e3b112ab.

[12] M. Nandu, "Natural Language Processing in Python with Code (Part II)," 2019. [Online]. Available: https://medium.com/swlh/natural-language-processing-in-python-with-code-part-ii-18c8742762a4.

[13] A. Navlani, "Support Vector Machines with Scikit-learn," 2019. [Online]. Available: https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python.

[14] A. Navlani, "Support Vector Machines with Scikit-learn," [Online]. Available: https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python.