POLITICAL ORIENTATION PREDICTION USING SOCIAL MEDIA ACTIVITY

A PROJECT REPORT

Submitted by

ESHWAR PRASAD. S [RA1611003040079]

RAIYAN AHMED [RA1611003040017]

VIVEK CHAURASIA [RA1611003040226]

Under the guidance of

Mrs. S. NIVEDITHA, M.E. (Ph. D)

(Assistant Professor, Department of Computer Science & Engg.,)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING,

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

Vadapalani Campus, Chennai - 26.

MAY 2020

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"POLITICAL ORIENTATION PREDICTION USING SOCIAL MEDIA ACTIVITY"** is the bonafide work of "**ESHWAR PRASAD. S [RA1611003040079], RAIYAN AHMED [RA1611003040017], VIVEK CHAURASIA [RA1611003040226]",** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mrs. S. Niveditha M.E. (Ph. D)	Dr.S.PrasannaDevi, M.E., Ph.D, PGDHRM., PDF(IISc).,
GUIDE	**PROFESSOR &**
Assistant Professor	**HEAD OF THE DEPARTMENT**
Dept. of Computer Science & Engineering	Dept. of Computer Science & Engineering

Signature of the Internal Examiner

Signature of the Internal Examiner

ABSTRACT

Social Networking has risen to a place of prominence as a medium of publishing information. Times are constantly changing, and the power to sway and portray political opinions is shifting from traditional media such as newspapers and television networks to social media platforms like twitter. In this venture we reexamine the problem of measuring and predicting the political orientation of twitter users. We expect to contribute to the study of the political blogosphere by incorporating multiple hypotheses about the behavior of the average twitter user and a registered politician, alike. Incorporating ideas such as tweets, retweets, subtweeting, followers and followees network and degrees of separation helps us understand the twitter political scenario better and helps us better understand how to leverage these sources of information. In recent times, hundreds of researchers take to twitter to analyze the effect of twitter on major political events such as the 2016 and 2020 U.S. elections, and we think that our technical contribution would be the reimagination of the traditional problem of predicting the political leaning of a given user. By studying the political orientation of twitter users, it is possible to target advertisements at individuals, shape digital profiles, and deliver news, articles, views and products that are individualistic and personalized.

ACKNOWLEDGEMENT

First and foremost, we would like to thank God Almighty for giving us the strength, knowledge, ability and opportunity to undertake this project work and to persevere and complete it satisfactorily.

We place our deep sense of gratitude and thank our beloved Chancellor for providing us with a good infrastructure throughout the course.

We take immense pleasure to thank our beloved Pro Chancellors, Sri Ravi Pachamoothoo and Dr. P. Sathyanarayanan for providing a conducive learning environment.

We take the opportunity to extend our heartfelt thanks to our respected Dean (E & T), Dr. K. Duraivelu, for his support and impeccable guidance throughout the four years of our B.Tech. course.

We like to express our special thanks to our Head of the Department, Dr. S. Prasanna Devi for encouraging and helping us throughout the course.

We thank our Project Guide Mrs S. Niveditha for her supervision, feedback, and mentoring, it would have been tough for us to finish this project in a timely manner. Thus, we feel deeply obliged for her support.

We would also like to show our greatest appreciation to our Project Coordinators, Mrs K. Meenakshi & Mr. Manikkannan, We can’t thank you enough for the tremendous support and help offered every week when we had project discussion. Without their encouragement and guidance, this project would not have materialized.

We would like to thank the guidance and support received from all the members including our parents and friends who contributed to this project was vital for the success of the project. We are grateful for their constant support and help.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview of the Project

In the recent years, social media has gained popularity not only as a platform for sharing information but also as a major hotspot for research. Significant research takes place in the political sphere every few years, whenever there is a significant political event is about to unfold on the world stage. Social media has played a central role in political communication, candidates create their narratives, political parties push and pull agendas and the general public's opinion is shaped, all on the platform of online Social Media, such as twitter, reddit, facebook. With the rise of popularity in social media platforms, it is easier than ever for an individual to profile themselves and publish themselves through blogging, microblogging, and posting their opinions online. Especially during major political events such as elections, big amounts of data are published and circulated on the social media sphere, providing researchers with terabytes of data to process and analyze. This type of analysis is usually termed under "Opinion Mining" or "Sentiment Analysis", where in Natural Language Processing is used to examine, identify and pick out opinionated information from various sources of text. Sentiment Analysis is applied to a variety of practical applications such as movie reviews, marketing purposes and customer services. The objective of sentiment analysis is to identify the tone of the writer with respect to the topic of the document or the overall polarity of the said document. The wake of research in sentiment analysis, along with the large amounts of political data published during election times, sentiment analysis finds itself applied in political science. Having stated the above, it is easier to understand why Twitter was chosen as the platform for the analysis of our venture, as Twitter is one of the most socially and politically active platforms, especially during election times. Furthermore, Twitter's enforcement of the content limit of 280 characters facilitates the data to be concise and better for processing purposes. Due to these reasons, we decided to base our analysis on the data that is leverageable from Twitter. The rest of the paper is structured to highlight Related Work, Data Collection, Methodology, Results and Conclusion.

1.2 Aim of the Project

The project's main objective can be concisely captured in the following way: to infer a given users' political orientation incorporating many facets of information available from twitter data, such as Tweets, Retweets, Liked Tweets and the Twitter social graph. Every time there is a politically charged event such as the 2016 or 2020 U.S elections or the Spanish elections, thousands of research ventures take place to use the large amount of data that is produced from these events for political analysis. Computational analysis of political events may be defined as the analysis of political events using computational methods such as Machine Learning and Natural Language Processing. However, there has not been much research in the aspect of individuals. Analysis of individuals' behaviour includes analysing a given individual's post history on the social media space.

In this project, we propose the idea of using the information available for leverage from Twitter, about a single user to determine their position on the political spectrum. Using all the facets of information of a single user gives us a holistic idea about the person. This is different from the previous works of research in the similar vein, which are more focused on collective analysis such as election winner prediction, analysing the percentage of political leaning of the demographic of a particular state or region, etc.

CHAPTER 2

LITERATURE SURVEY

2.1 Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts

The collective survey study done by Grimmer et al., highlights the know-how of automated political analysis. The availability of large amounts of political text, legislative speeches, bill text, parliament proceeding minutes, party statements, manifestos, through electronic medium has enabled the area of automated content analysis. The techniques from these works are not directly applicable to our goal here, as detailed twitter data for users are rarely available from tweets.

2.2 Quantifying Political Leaning from Tweets, Retweets, and Retweeters

Felix et al., propose the idea of quantifying the political leaning of prominent political figures (by nature of their number of retweets), and using them as the measures of 'correctness', since its predetermined knowledge whether any of these figures are democrat or republican. Using these sources, they further find the political leaning of ordinary twitter users (who have retweeted one of the source users at least 10 times).

2.3 What Drives Media Slant? Evidence from U.S. Daily Newspapers

Gentzkow et al. Presented their index [7] to attribute to a media outlet as a measure of its slant, that measures the similarity of an outlet's language to that of a congressional democrat / republican. Their approach to measuring the slant of a newspaper was to compare phrase frequencies in the newspaper with phrase frequencies in the 2005 Congressional Record to identify whether the newspaper’s language is more similar to that of a congressional Republican or a congressional Democrat.

2.3 Visualizing Media Bias through Twitter

Traditional media outlets are known to report political news in a biased way, potentially affecting the political beliefs of the audience and even altering their voting behaviours. Therefore, tracking bias in everyday news and building a platform where people can receive balanced news information is important. An et al. propose a model that maps the news media sources along a dimensional dichotomous political spectrum using the co-subscriptions relationships inferred by Twitter links. By analyzing 7 million follow links, they show that the political dichotomy naturally arises on Twitter when only considering direct media subscription.

2.4 Political Polarization on Twitter

The study by Conover et. al. investigates how social media shape the networked public sphere and facilitate communication between communities with different political orientations. Using a combination of network clustering algorithms and manually annotated data, the authors demonstrate that the network of political retweets exhibits a highly segregated partisan structure, with extremely limited connectivity between left- and right-leaning users. To explain the distinct topologies of the retweet and mention networks they argue that politically motivated individuals provoke interaction by introducing polarizing content into information streams whose primary audience consists of ideologically-opposed users. They conclude with statistical evidence in support of this hypothesis.

2.5 Actionable and Political Text Classification using Word Embeddings and LSTM

In the work by Rao et. al., LSTM architecture (Long Short-Term Memory) was applied to political texts in the form of a text classification problem. They undertake the task of building models for various natural languages to compare the performance of LSTM architectures with word embeddings. They consider two situations, one that of actionability and one of political orientation. In this problem, they have demonstrated the classification of short messages on social media from service industry customers, into actionable or non-actionable. The interesting observation is that the methods that involved word embeddings give better performance than the traditional methods. In terms of the second situation, they undertake the problem of classification of messages into republican or democrat.

2.6 Twitter and Politics: Evidence from the U.S. presidential elections 2016

The 2016 U.S. presidential elections were a riveting political event in terms of social media. The political campaign for the 2016 U.S elections were majorly carried out in the form of social media propaganda, unlike the terms before that where traditional media had a major part to play. This paper is a thorough analysis of the effect of politics on twitter and of twitter on politics. During the 2016 election time, twitter became the hotbed of political discussions, since candidates took to twitter to publicize their views, and establish their policies. The interesting concept that this paper brings across is the feedback loop of journalists and traditional media covering the tweets of candidates, bringing more attention to the tweets and the extra attention leads to more tweets being published.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing Systems

Existing Twitter analysis systems leverage various methods to assess political leanings of a population, a gradual shift in political polarity, predicting potential winner in a given term, etc. Several works of this nature have been documented before:

- Link structure analysis to uncover polarization of political blogosphere
- Sentiment analysis of user comments to infer political orientation of news articles

Works on quantifying political leaning using the twitter follower network:

- Multidimensional scaling on media sources with their pairwise distances from mutual follower sets
- Graph based methods to propagate scores of Congress members on Twitter to media sources through followers

3.1.1 Challenges in Existing Systems

Sentiment analysis on tweets to assess political position has its disadvantages:

- Does not paint a complete and holistic picture of the users' ideological views
- Cannot build a digital profile of a user from a single or even with a temporal series of tweets
- Assessing political leaning of a demography does not serve the purposes and intents of individual orientation

Due to the huge size of most social networks, it is difficult to build a snapshot of a network, Twitter does not allow crawling the network more than few hop neighbourhoods.

3.2 Proposed System

In this proposed framework, we put forth a much more holistic approach than any of the works cited in our research of the related literature. We envision the project as an ensemble classifier problem, consisting of several underlying classifiers, each one handling one aspect of a twitter user. The LSTM (Long Short-Term Memory) architecture of RNN (Recurrent Neural Network) is used, along with several other layers including Word Embeddings and 1-D Convolutional Neural Network layer. We leverage more than just the tweet-retweet ratio maximization or a singular network matrix:

- Activity of a user on the twitter digital space as a whole
- Follower –Followee network
- Tweet Analysis
- Retweet frequency and leaning score
- Erdos Number concept

In regards to the above, our proposed system is curated to provide a more rounded and holistic sense of the individual user, painting an overall picture of her / his digital profile, leading to potential in marketing and business spheres. The fact that the system can integrate any and all kinds of social media information such as friends, followers, public posts, likes, reposts or retweets, makes it adaptable to different platforms and opens up avenues for many applications, such as:

- Election winner prediction of a any given region, given that we can collect social media activity from that specific region over a recent time period.
- Personalized Advertising, based on the prediction of a twitter user, since the products and services availed by people subscribing to different ideologies tend to be different.
- Further political analysis of different demographic based on the aggregate of the political leaning of individual users.

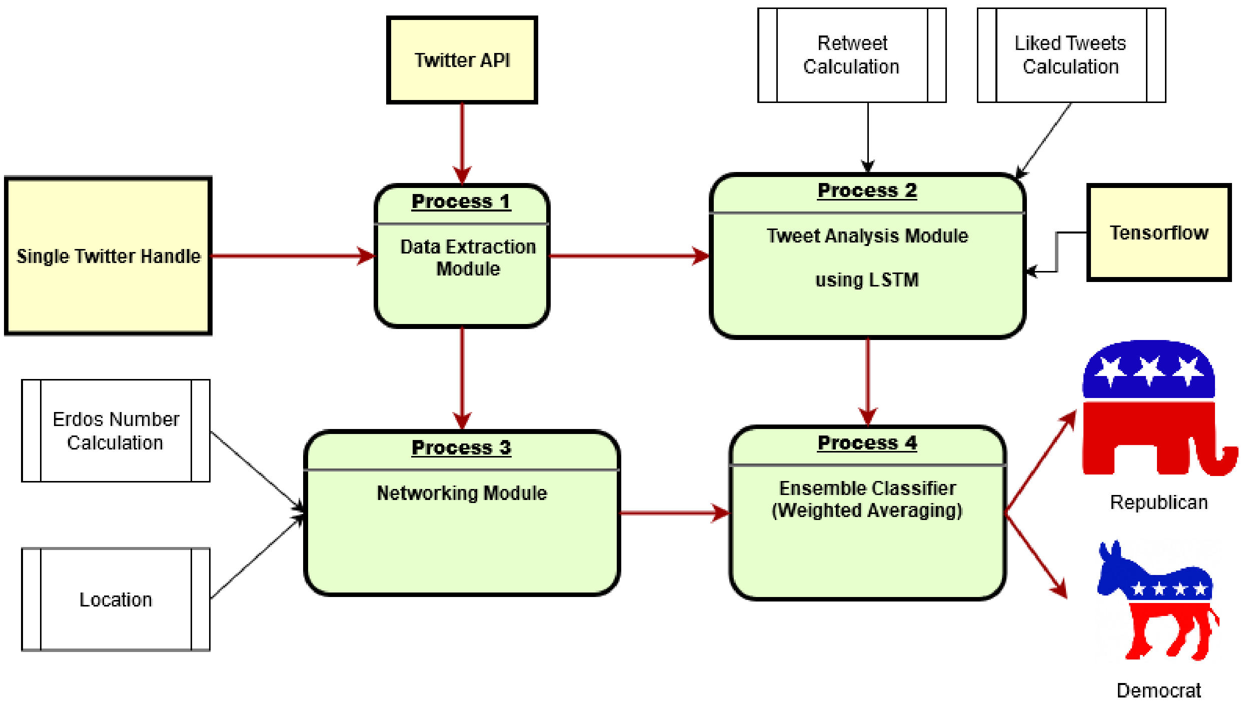


Figure 3.1: System Architecture

3.3 System Architecture

The system architecture is set up in such a way that the 3 initial modules flow seamlessly into the final module titled the "Ensemble Classifier". Initially the data extraction module is tasked with the extraction of large amounts of data from twitter using the Twitter API. Data extraction module is also where we do the exploratory data analysis, determining if any trends exist in the data that was extracted, and if they do, to identify these said trends. The extracted data from this module simultaneously flows into modules 2 and 3, namely Tweet Analysis Module and Networking Module. Each of these modules undertake complicated processes on their own, making use of multiple subprocesses. Tweet Analysis Module is the bread and butter of our project, handling the most intensive task of classifying the user's tweets, retweets and liked tweets as republican or democratic. This module houses the LSTM neural network. Parallel to tweet classification module sits the Networking module, responsible for calculation of the degree of separation of the given user from "ground users" that were decided upon after thorough analysis, deemed to be the most established democratic and republican twitter accounts. The networking module makes use of a probabilistic version of Flajolet-Martin Algorithm. As the figure (figure 1) suggests, the final module is the Ensemble Classifier, which takes as input the output from the previous modules, and computes the final classification based on Bayes' Weighted Average method.

3.4 System Modelling

3.4.1 Use Case Diagram

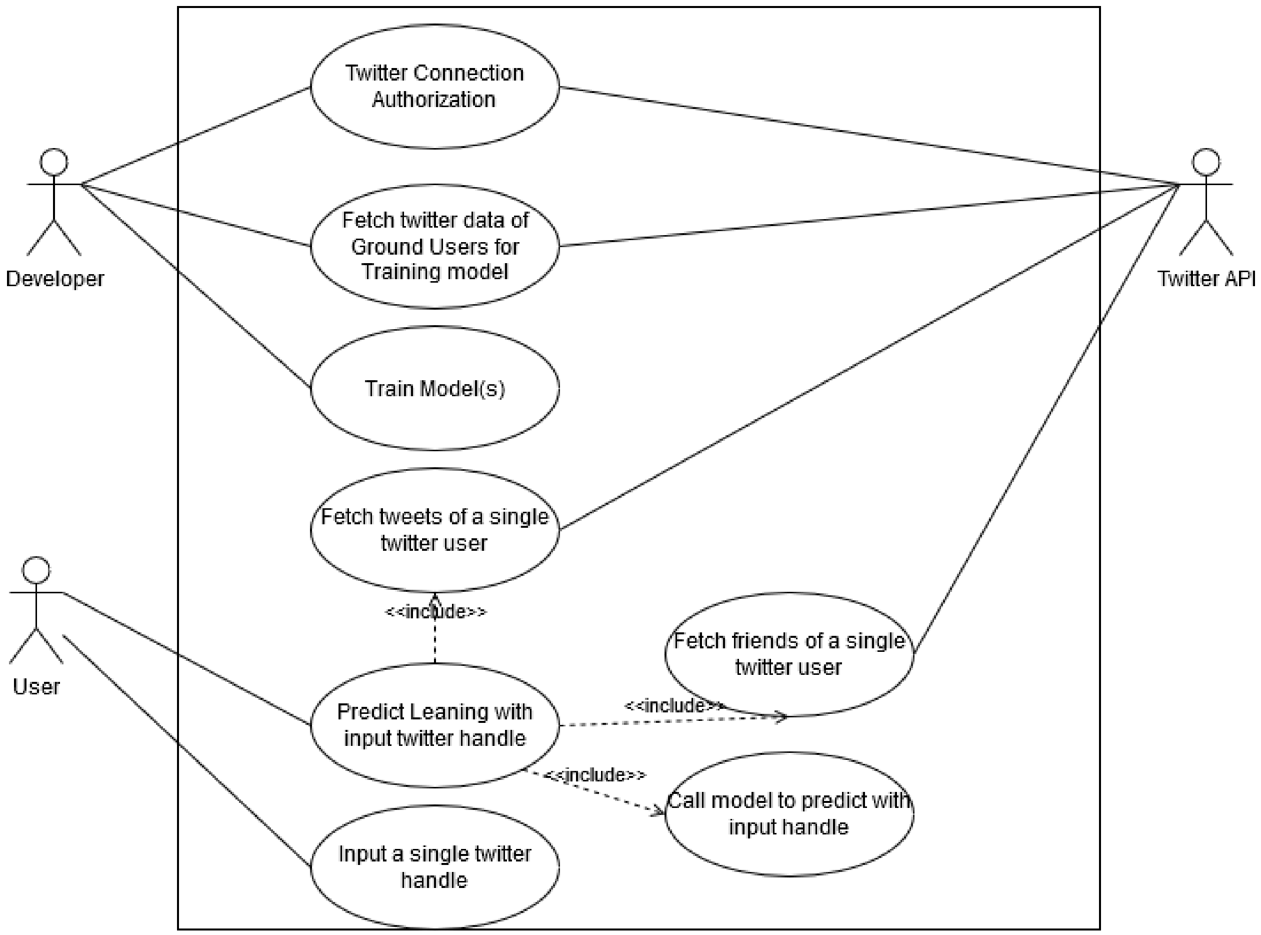

Twitter Connection Authorization
Fetch twitter data of Ground Users for Training model
Train Model(s)
Fetch tweets of a single twitter user
<<include>>
Fetch friends of a single twitter user
Predict Leaning with input twitter handle
<<include>>
<<include>>
Call model to predict with input handle
Input a single twitter handle
Developer
User
Twitter API

Figure 3.2: Use Case Diagram

The above diagram depicts the use cases that the system is expected to handle. The user may input a twitter handle and interact with the system to obtain the prediction of the system.

3.4.2 Activity Diagram

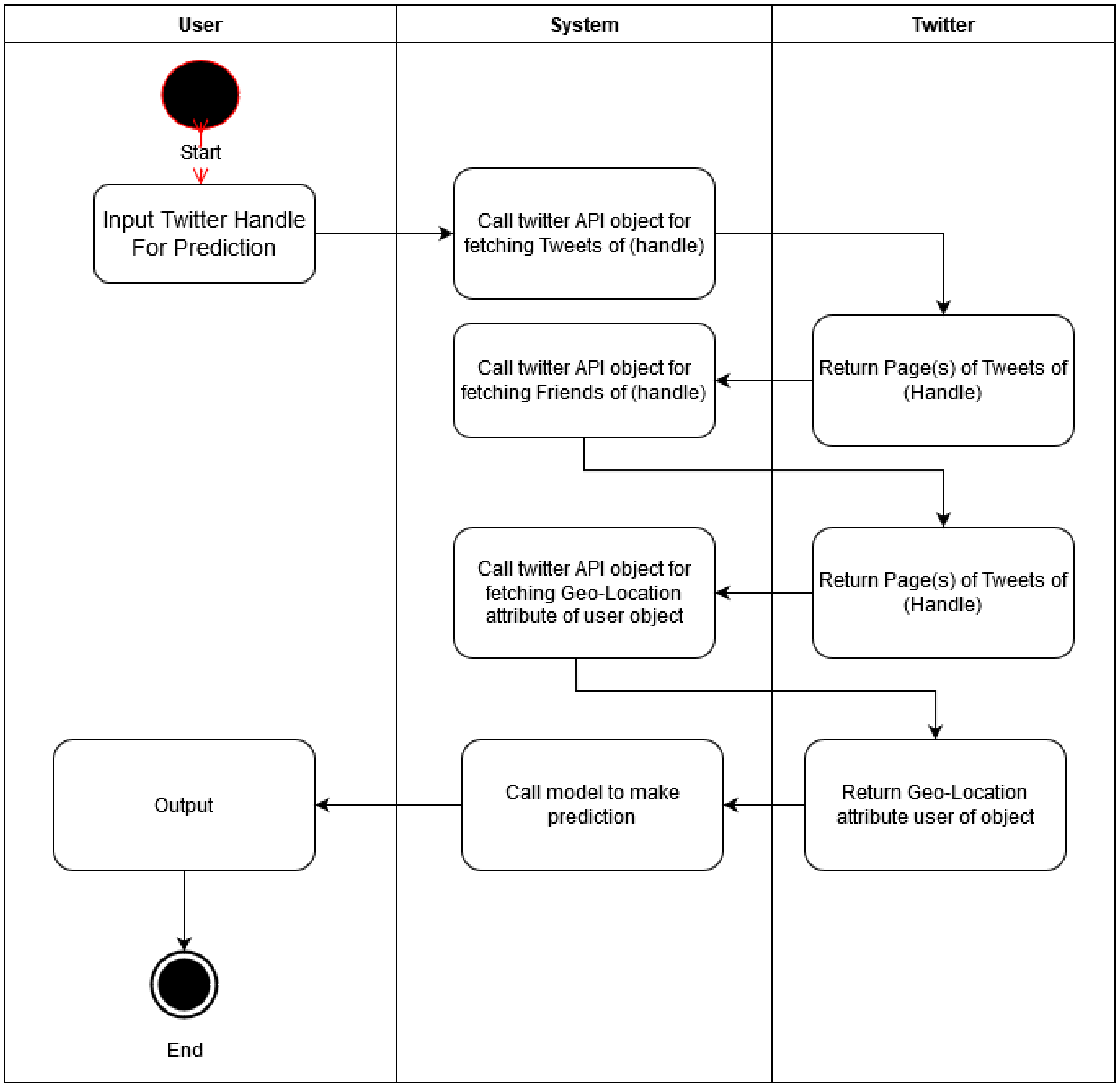

User
System
Twitter
Start
Input Twitter Handle For Prediction
Call twitter API object for fetching Tweets of (handle)
Return Page(s) of Tweets of (Handle)
Call twitter API object for fetching Friends of (handle)
Return Page(s) of Tweets of (Handle)
Call twitter API object for fetching Geo-Location attribute of user object
Return Geo-Location attribute user of object
Call model to make prediction
Output
End

Figure 3.3: Activity Diagram

The above diagram depicts the activity diagram of the political orientation prediction system. The diagram specifically outlines the workflow of prediction on one single twitter handle.

3.4.3 Sequence Diagram

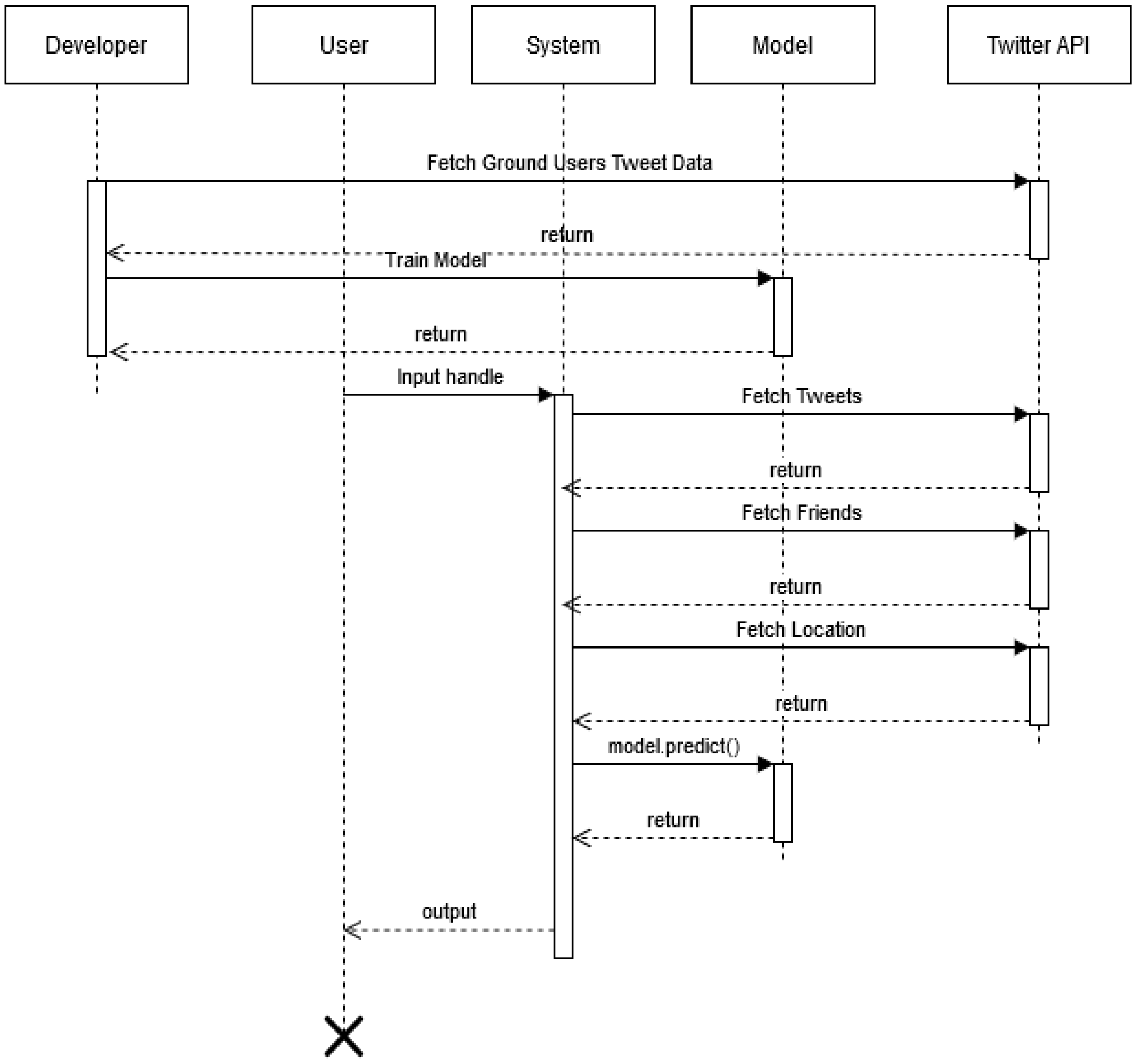


Figure 4: Sequence Diagram

The above diagram depicts the sequence or timeline diagram of the political orientation prediction system. The diagram outlines the working of the system from input to output for making predictions on a single twitter handle.

3.5 Algorithms Used

- LSTM Architecture of RNN, which is mainly used to handle time series data. Although we do not use time series data for our purposes, we have chosen LSTM as it is equipped to handle long distance dependencies in sentences, unlike traditional RNNs.

- Flajolet – Martin algorithm, for finding the degrees of separation between given input twitter user and "ground users".

3.6 Technical Modules

The report henceforth is presented in terms of 3 main modules, namely, Data Extraction Module, Tweet Analysis Module and Networking Module.

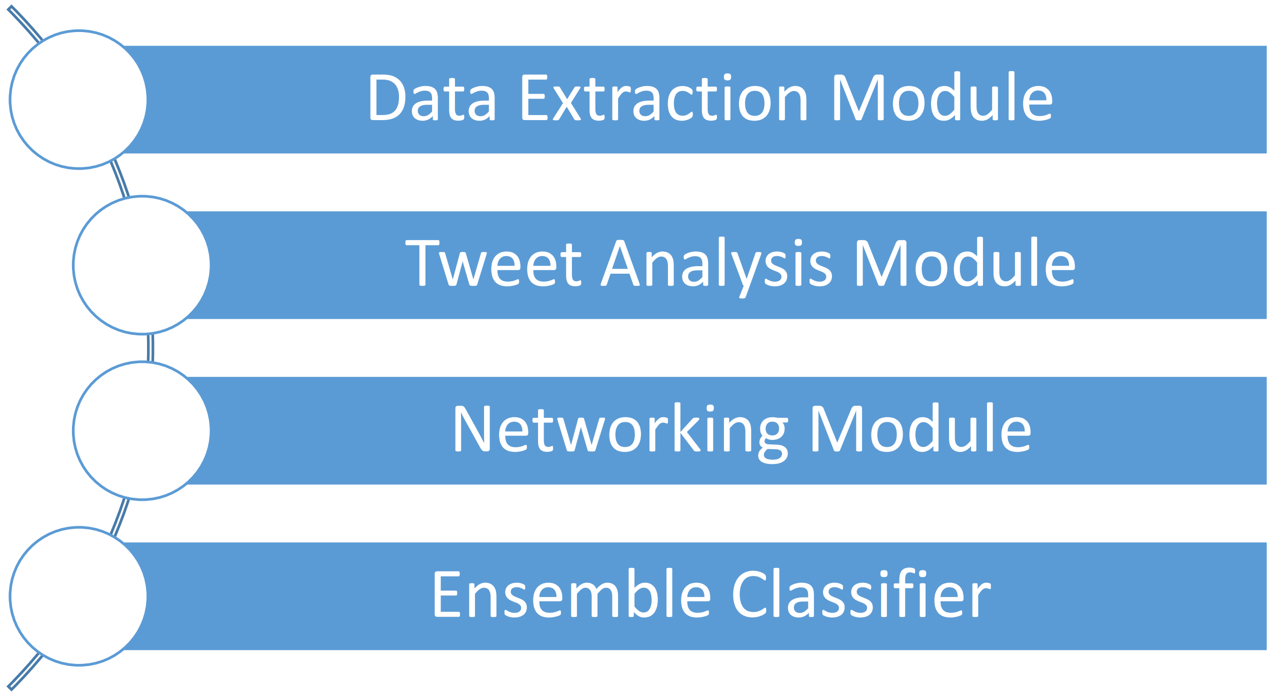


Figure 3,5: Technical Modules

CHAPTER 4

DATA EXTRACTION

In this section, we would like to justify the use of twitter data and describe the Twitter social platform and the characteristics of the twitter data. Twitter is a popular social network platform that allows users to register themselves as users and publish or ‘tweet’ tweets, or short text microblogs, limited to 280 characters as of 2020 (previously 140 characters). Twitter was created in 2006, and consequently gained worldwide popularity, by 2018, averaging 321 million active users every month. Twitter has held a central position in political communication, as was evident from the 2016 US elections.

Twitter may be split into 2 parts, the social network, and the short messages that are published and the social graph consisting of the connections between various twitter users. On twitter, user X can follow user Y (X outgoing link to Y) and we say Y is being followed by X. Unlike facebook, not all links on twitter are reciprocated, i.e. Y does not follow X even though X follows Y.

4.1 Tweets, Retweets and Subtweeting

4.1.1 Twitter Platform

Twitter user handles consist of the following data : screen name, the user profile image, profile description, and a home location. The “User timeline” is said to be the list of tweets tweeted by the user. For example, Vice President Mike Pence (As of 2020) has a twitter account by the handle @VP, boasting 8.6 Million Followers as of March of 2020. Similarly, Prominent democrat Hillary Clinton goes by the handle @HillaryClinton with 24.9 Million Followers as of March of 2020.

4.1.2 Retweets

Retweet is a tweet that is republished by a user which was originally published by a different user, and this is usually done in the context of spreading the word, or agreeing with the original message. Rare as it is, it is also possible that sometimes the original tweet is taken as the subject matter to disagree with. Mentions are another way of interaction between 2 users; User A can reference another user by their twitter handle using the "@" symbol, and when a tweet includes a "mention", the mentioned user receives a notification and the tweet is usually aimed at the said user in a direct manner.

4.1.3 Subtweets

Subtweeting is an informal concept in the twitter sphere, where people tweet out messages that are not directly mentioning someone but with the given context can be identified to be targeted at a particular subject or user.
Twitter API

4.2 Twitter API

The input data, including the tweets by the specified user, their followers, retweets, liked tweets and location data, are all obtained using the Twitter API [12]. The Twitter API has different tiers of rate limiting, and for our purposes, only 15 calls per 15 minute window. Due to the restrictions placed on our usage of the API, further considerations were made to the methodology which will be described later in the paper.

4.3 Exploratory Data Analysis

In order to understand the language used in the tweets published by our target base users (24 of each from Democrats and Republicans), We first analyzed the usage of hashtags and words by frequency, looking for clues in the underlying differences between a republican tweet and a democratic tweet.

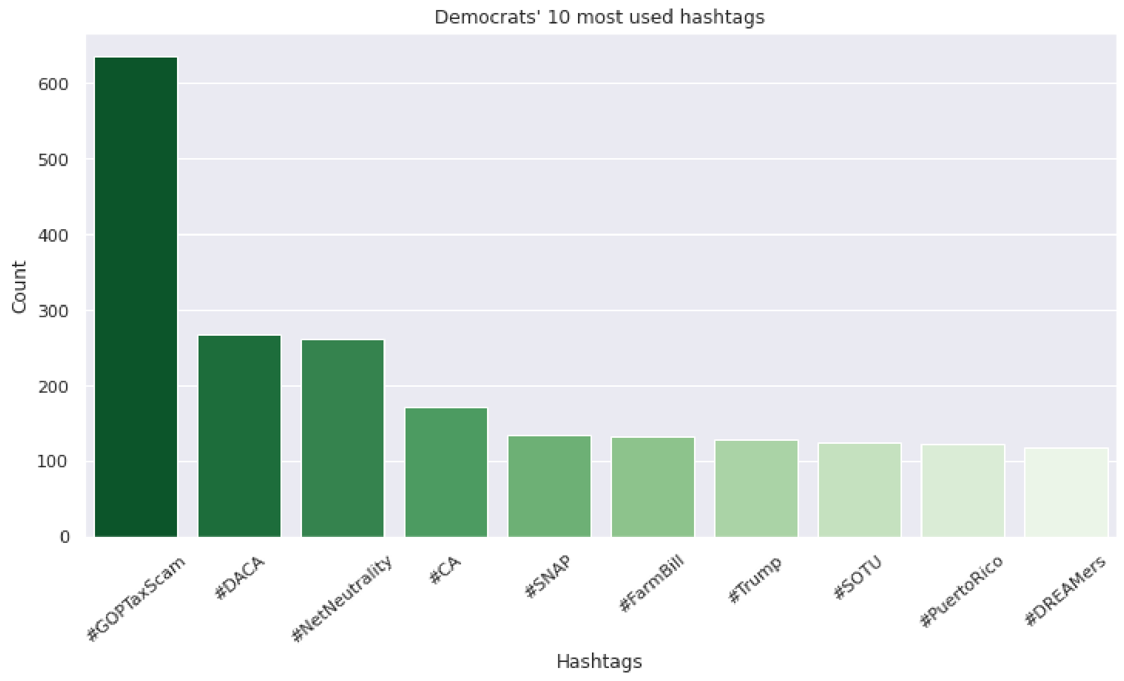


Figure 4.1: Bar plot summarizing top 10 hashtags used in tweets published by democrats.

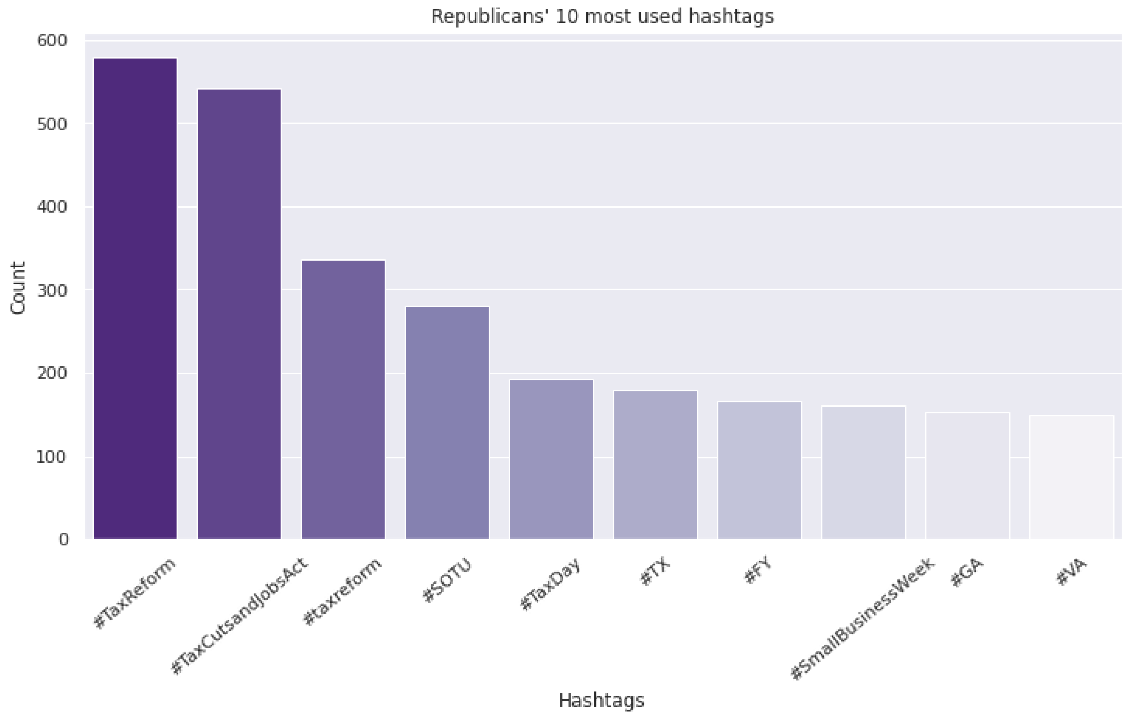


Figure 4.2 : Bar plot summarizing top 10 hashtags used in tweets published by republicans.

Interestingly, the hashtags used by the different parties may be inferred in the following manner. From 2016 and during 2018 congressional elections, the Republican party has been the ruling party, and we can clearly see their hashtags reflecting the policies that were put into action during the 2018 era. And as for the democrats (figure. 4.1), their hashtags seem to express resentment with the policies put into action.

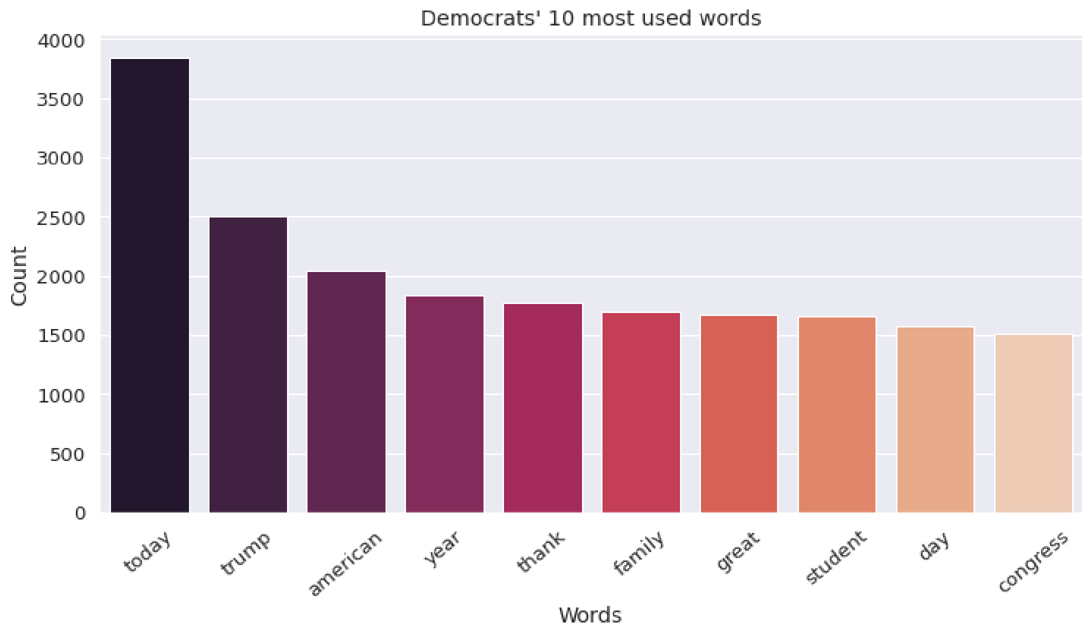


Figure 4.3: Bar plot summarizing top 10 words used in tweets published by democrats.

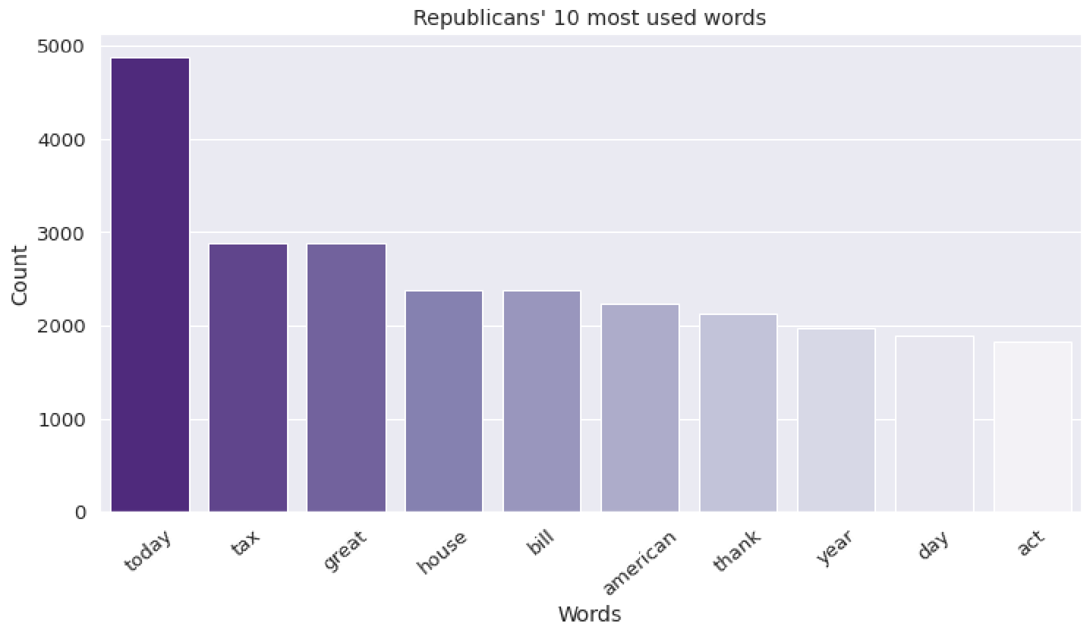


Figure 4.4: Bar plot summarizing top 10 words used in tweets published by republicans.

Consequently we visualized the words that are frequently being used in order to identify the innate differences. We used the seaborn bar plots in python to draw these said graphs. We broke down the words of the tweets into tokens which were then vectorized, and ordered by frequency. Then the top 10 words that were used were picked out for visualization purposes (figures 4.3,4.4).

4.4 Data Extraction Methodology

Our tweet analysis dataset consists of the tweets surrounding the 2018 U.S. Congressional Elections, with the hypothesis following the assumption that politicians are relatively consistent with the language models that they follow while publishing tweets on the twitter platform. More than 86000 tweets were collected from 24 members from each of the parties, house republicans and house democrats.

4.4.1 Data Pre-processing for Analysis

In this section we outline the process of preparing the data for training our core neural network model. Since our methodology involves some NLP processes, we follow the common cleaning up methods involved in any NLP model, including Tokenization, Lemmatization, Removing Stop Words.

4.4.2 Tokenization

Tokenization is the process of splitting a given character sequence into ‘tokens’, while also eliminating any and all special characters such as punctuation. Tokenization is one of the first steps for preprocessing a raw text and we used the TweetTokenizer[13] from the NLTK package which is built for tokenizing tweets.

4.4.3 Lemmatization

Lemmatization is the process of identifying the root form (also known as dictionary form) of a word so they can be analyzed as a single term. Lemmatization is a similar process to stemming but it does not only trim inflections (affixes) but also bring context to the word, hence tracing words with similar meaning but different appearance to one single root word, also known as the Lemma. We use the WordNet Lemmatizer available from the NLTK package.

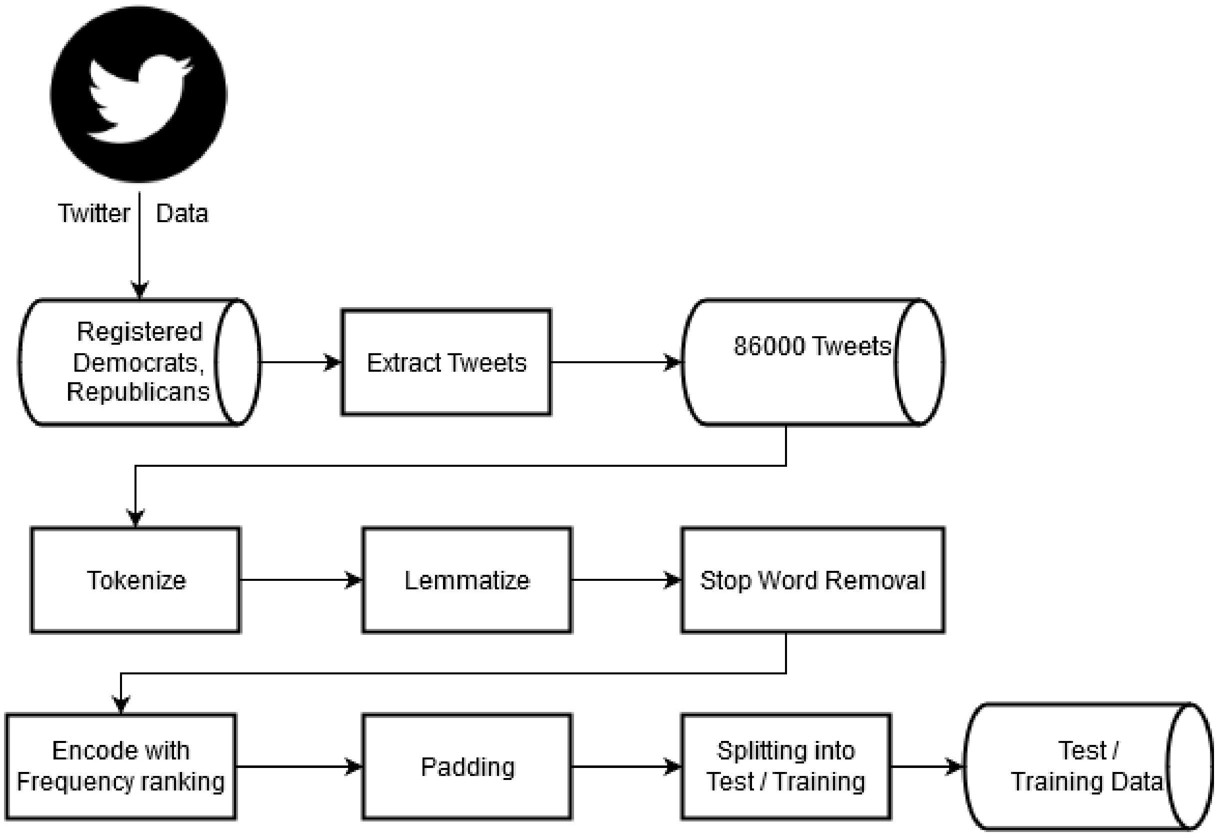


Figure 4.5: Process of data extraction and data pre-processing for tweet analysis.

4.4.4 Preparation of Data for Analysis

The data at this stage was converted to a proper format for preparing the input data to the core LSTM model of the system. 10,000 words were ranked in terms of frequency in the descending order using a frequency distribution function, and each word in each of the input tweets were encoded with their frequency ranks instead. In case a specific word was not in the list of the top 10,000 words, the word was encoded with a 0. Further, the input shape for Keras [14] models are required to be NumPy arrays of the same length, and naturally we modified the input data to be of 50 length. Each of the rows of the dataframe was set to 50, longer rows were truncated and shorter rows were post-padded with zero index values.

CHAPTER 5

TWEET ANALYSIS

5.1 Module Workflow:

Over 86,000 tweets were collected from 24 Democrats and 24 Republicans, and these tweets were:

- Tokenized
- Lemmatized
- Stripped of Stop Words
- Encoded with each token's frequency distribution rank
- Padded for Keras input format (numpy array of 50 length, padded with 0s)
- Split into 80% Training, 20% Testing sets of data.

Multiple algorithms were tested including linear and logistic regression, and the choice finally fell upon Bidirectional RNN LSTM.

We make use of the Tensorflow backend GPU for using the CuDNN LSTM, according to Keras documentation: *"Fast LSTM implementation backed by CuDNN. Can only be run on GPU, with the TensorFlow backend".*

5.2 Neural Network Architecture

Deep Neural Networks have consistently proved to be successful in a variety of applications ranging from text processing to image recognition. With the use of multiple non-linear transformation layers, neural networks are able to capture high levels of abstractions. RNNs (Recurrent Neural Networks) are a class of ANNs (Artificial Neural Network) where the connections between the network nodes can model temporal data. This very nature allows RNNs to handle temporal behaviour of the data; using their memory state, they can process variable length inputs, making RNNs highly applicable in fields such as handwriting recognition and speech recognition. The internal states are also known as ‘memory’ states that lets RNNs retain events past. The problem with traditional recurrent networks is that they fail to handle long-term dependencies between segments of a sentence.

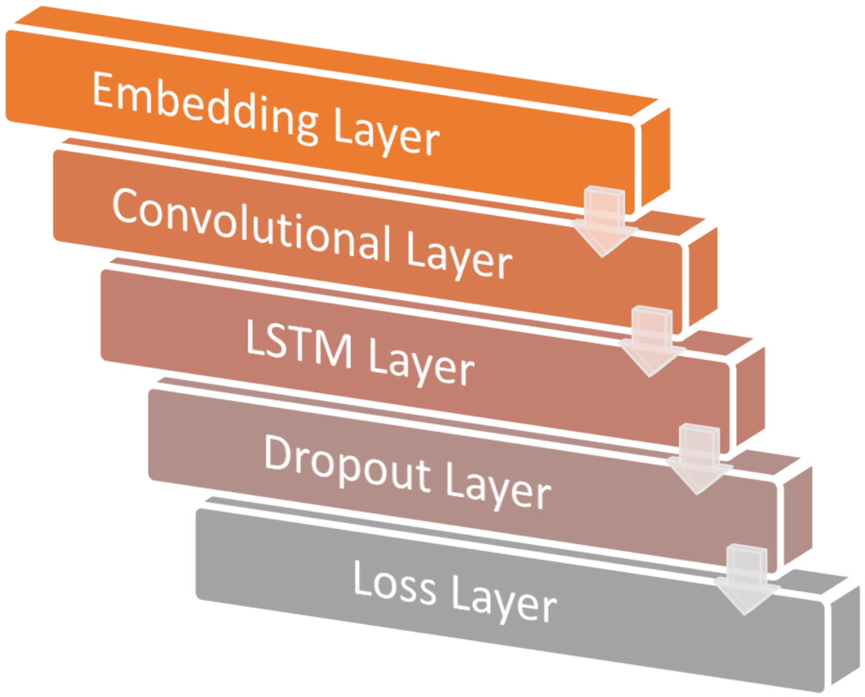


Figure 5.1: Neural Network Architecture

LSTM (Long Short-Term Memory) is an RNN architecture that can circumvent the problem of long term dependencies by the use of 3 regulating gates in each cell. Long term dependencies are an important factor to be considered while learning to classify sentences, as words within sentences display this type of dependencies. Our neural network architecture uses embedding layers to classify the pre-processed input. Our input data has 2 labels, democrat and republic, and we use supervised learning to train the neural network.

5.3 Embedding Layer

The initial layer in our neural network is the Keras embedding layer that encodes the positive integer indices of the input into dense vectors of fixed size. The example provided in the Keras documentation is [[4], [20]] -> [[0.25, 0.1], [0.6, -0.2]]. The embedding layer is only allowed as the first layer of the model, with the purpose of learning the mapping of the words in the entire vocabulary to a one-dimensional space. Basically, the words in the discrete vocabulary are mapped to a one-dimensional vector space and the mapping is learned through this layer. Remembering such mapping has been shown to have significant improvements in performance. Word embeddings capture meaning-based relationships, also known as semantic relationships from the input data, saving us from actually defining and extracting features.

5.4 Convolutional Layer

The embedding layer is immediately followed by a one-dimensional convolutional layer. A CNN (Convolutional Neural Network) typically is used for image classification, in which an input image is accepted as a 2-Dimensional input representing the image's resolution pixels. A similar process is applied in a 1-Dimensional ConvNet layer, where the model is able to extract features from the input data and maps the features of the sequence of the data. 1D Convolutional Neural Networks are shown to be highly effective for feature learning from an input data where the input data is of the same length, and the location of the features is trivial. We use the ReLU (Rectified Linear Unit) activation function for the convolutional layer. Mathematically, it is given as:

$$y = max\ (0, x)$$

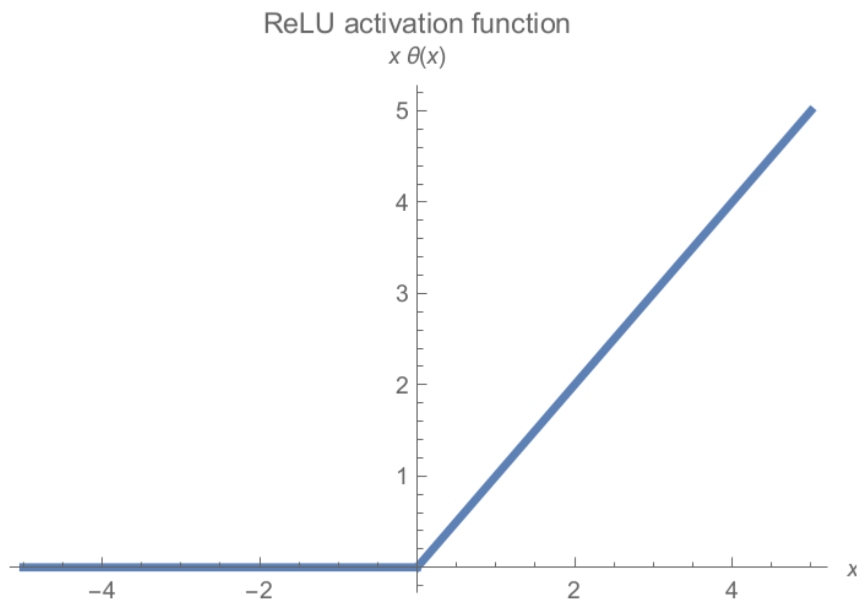


Figure 5.2: ReLU Activation Function.

The ReLU activation is a linear function that outputs the input value if the input is positive, zero, otherwise. It is one of the simpler activation functions, easier to compute and saves training time for the model. ReLU activation function also tends to converge faster as linear functions cannot plateau, unlike the sigmoidal or hyperbolic tangent functions.

5.5 LSTM Layer

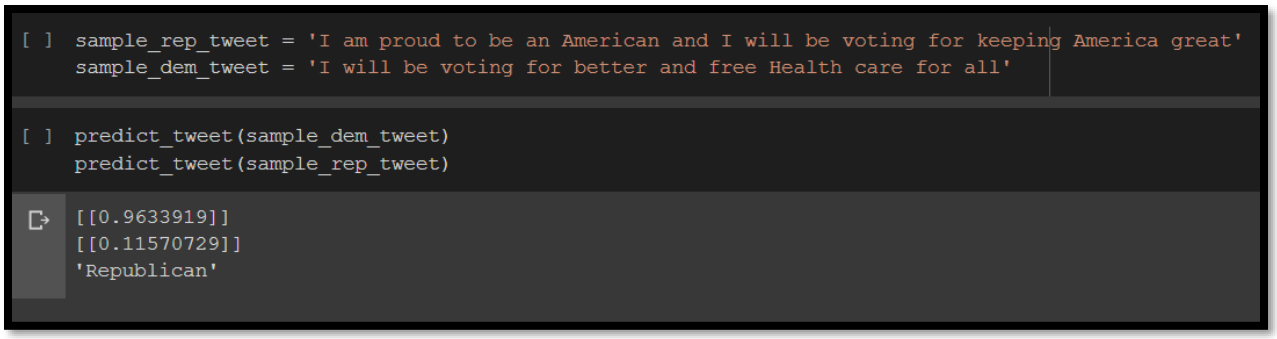
The next layer in our core model is the LSTM Layer with 256 LSTM units. Every LSTM unit is a memory cell, with 3 gates and a recurrent connection. The three gates are input gate, output gate and a forget gate. The input gate can do one of the two things: allow or deny the input from altering the cell state. Consequently, the output gate either allows or blocks the internal state from affecting other units. The forget gate is the interesting one, giving the LSTM architecture units the ability to either remember or forget the internal state, depending on the self-recurrent connection. Increasing the number of LSTM layers tends to increase the accuracy of the model, and the same number of embedding layers tends to be the optimal number of layers.

5.6 Dropout Layer

Dropout Layer is a technique that was introduced by Srivastava et. al. for regularizing neural networks to circumvent the problem of overfitting. Overfitting is usually characterized by the loss of the ability of the network to generalize. The dropout technique is to randomly omit a part of the network nodes while training, and hence preventing the nodes from adapting to each other. The outputs of our LSTM layers are sent to the dropout layer where 50% of the LSTM units are randomly dropped.

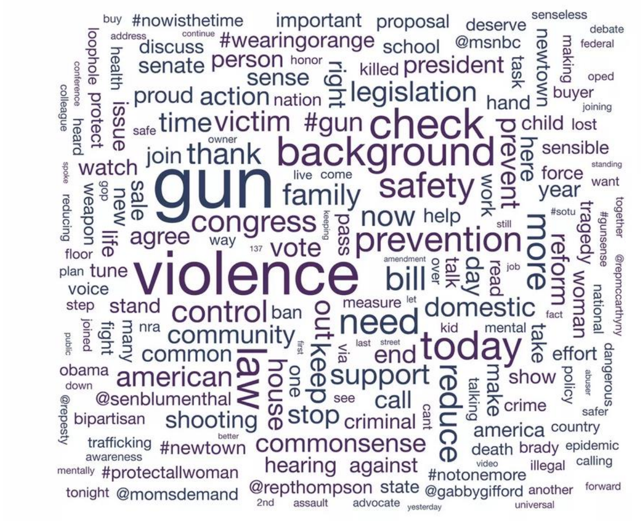
5.7 Loss Layer

The last layer in our neural network architecture is the loss function, which is responsible for penalizing any deviation of the predicted class from the true class. Our problem here is a binary classification problem, and hence a binary cross entropy is an apt loss function for our neural network. Using the thus described characteristics, we trained our neural network for classification of a single tweet from a single twitter user into either republican or democrat. The 100 most recent tweets of the input user are run through the network for prediction and the prediction scores are aggregated for a final classification of the user's tweet content. The same process is repeated for "retweets" which are identified by the prefix "RT" in the extracted dataset, and for favourite tweets which are tweets that were deliberately liked by the input user.



```
[ ]  sample_rep_tweet = 'I am proud to be an American and I will be voting for keeping America great'
     sample_dem_tweet = 'I will be voting for better and free Health care for all'

[ ]  predict_tweet(sample_dem_tweet)
     predict_tweet(sample_rep_tweet)

     [[0.9633919]]
     [[0.11570729]]
     'Republican'
```

Figure 5.3: Prediction of political leaning in action.



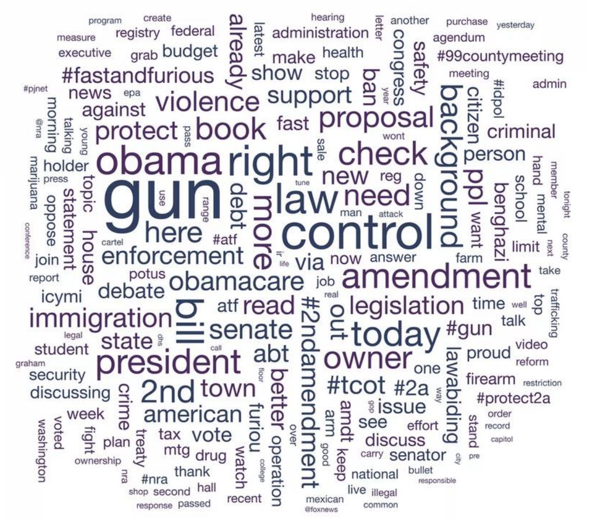


Figure 5.4: Democrats on "Guns"

Figure 5.5: Republics on "Guns"

CHAPTER 6

NETWORKING MODULE

The problem of classifying and assigning relevant and meaningful political classification based on a person’s entire profile activity is a much harder task than what is handled in [6] and [20]. There have been many works on quantifying political opinions and political slant using the twitter social graph, but this area is still relatively under discovery as the task of looking up the temporally dynamic twitter network is a performance cost heavy task. We argue that even if it adds a heavy performance overhead, considering the twitter network data adds a lot of impactful information to the final classification. To understand this simply, consider 2 users X and Y, user X being a relatively active user on twitter with some politically charged tweets, and other tweets that may or may not be relevant to our classification problem. User Y, on the other hand, is what one would consider a bystander user, or a read-only participant. They are also sometimes called ‘lurkers’. In informal Internet, a lurker is defined as a participant of an online community who observes, but does not post or participate. Lurkers are a major part of any community that has online presence. In the traditional twitter classification models, user X would easily be identified, since her user timeline would have the required amount of data to work with. Whereas, user Y’s timeline would have no tweets to work with, and would either be handled in an inappropriate manner, or would be discounted from being made an actual prediction on.

In our system, user X’s prediction would be handled in a similar manner to the other political classification systems, with the exception that their networking information will also be factored in. Additionally, user Y’s account can be classified by our model much better than the traditional models. User Y’s tweet prediction will return null but our networking algorithm can identify where the user lies on the political spectrum, that is, on average what is her degree of separation from the ‘ground users’ (users that we have decided as figureheads of the two ideologies after thorough deliberation).

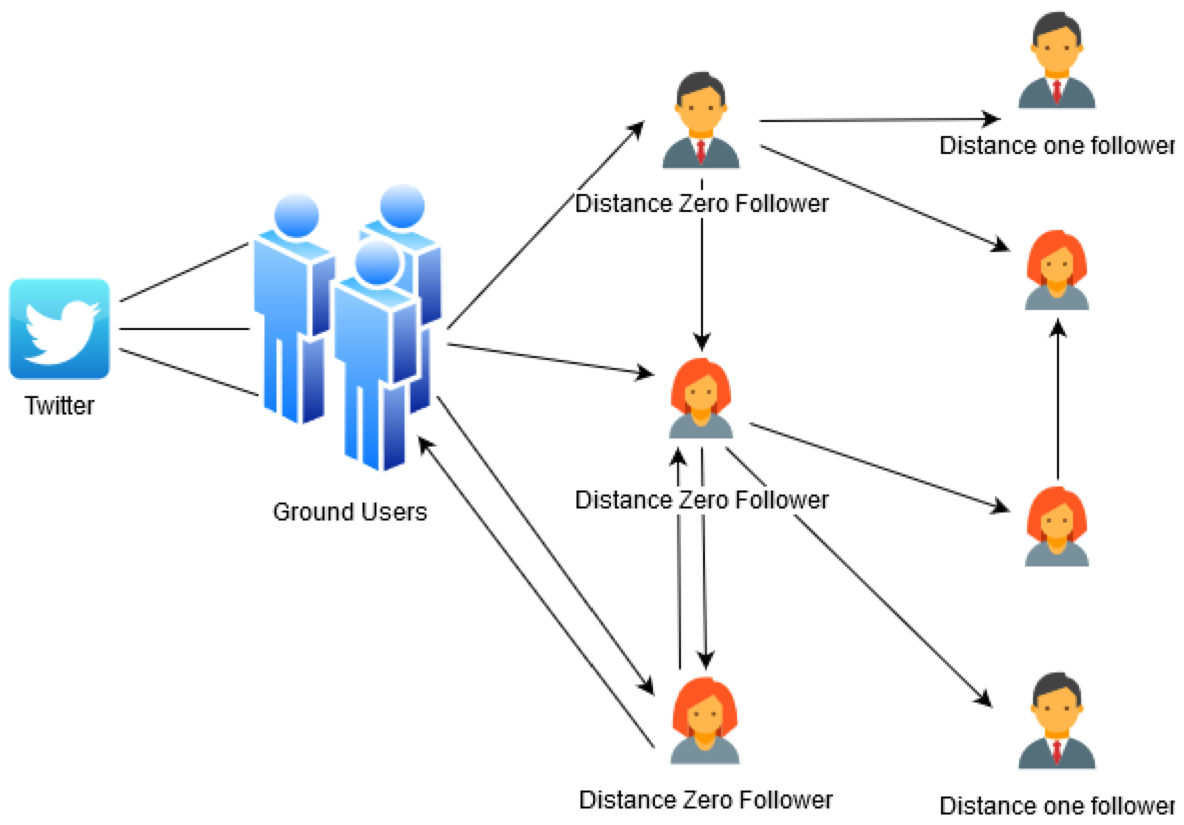


Figure 6.1: Figure depicting Degrees of Separation.

6.1 Module Workflow

6.1.1 Challenges to be addressed:

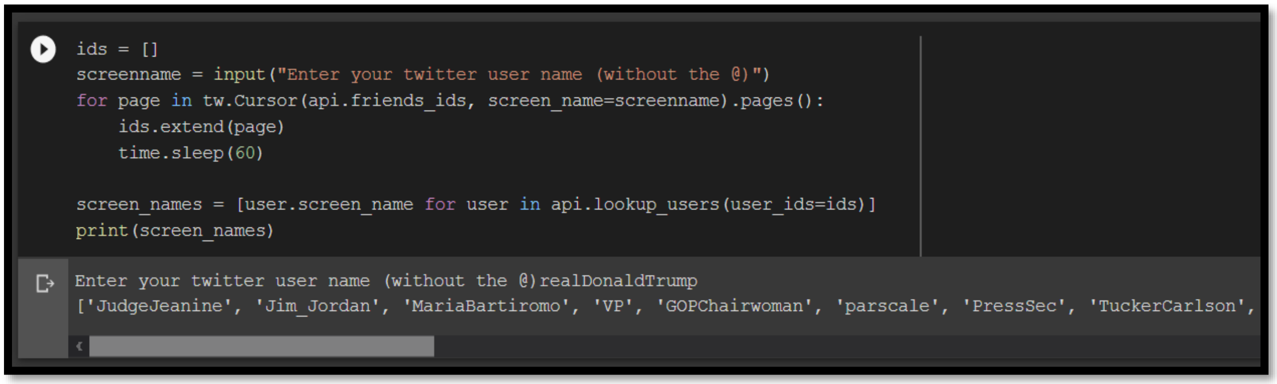
- Building the Twitter Friendship network is a monumental task, considering the number of incoming (followers) and outgoing (Friends) edges.
- Building the Twitter Friendship graph beforehand is also not an option as we want user handles to be input dynamically.
- Twitter API call rate limit of 15 per 15 minutes is a great restriction to route through each connection in a simple BFS fashion.
- Djikstra's shortest path becomes simple BFS in case of equal weights (which is the case for twitter links), and hence becomes extremely inefficient.

6.1.2 Solutions:

- We First attempted the simple BFS, and were challenged by the rate limit. Once the rate limit sleep time was introduced, we were next faced with the connection establishment issues.
- We pivoted to Bidirectional Search, considering that all of our goal nodes are accounted for as ground users.
- Yet still, Bidirectional Search was also proving rather inefficient, considering only 15 API calls are allowed per 15-minute window, and routing through millions of followers from goal end was not a feasible option.
- Finally, we decided to move to a probabilistic search, starting from source end (Input Twitter Handle), and sorting each frontier layer member in the order of maximum outgoing edges.
- In doing so, we greatly increase the likelihood of the path being found, since, theoretically, more outgoing edges lead to a higher likelihood of containing links to the goal nodes.

6.2 Erdos Number

We would like to introduce, in this section, the concept of Erdos Number, also described as the ‘collaborative distance’. The Erdos Number is described as the measure of collaborative distance from any person to the renowned mathematician Paul Erdos, measured by co-authorship of academic mathematical papers. Paul Erdos was a prominent mathematician who authored over a 1500 academic papers. The important aspect of the Erdos Number is that leading mathematicians are shown to have relatively low Erdos numbers. The average Erdos Number is only 3 for the mathematical field medallists. The interesting part about the concept is that it is applicable as a tool to realize how prominent mathematicians collaborate to arrive at solutions. The concept is also central in many studies that use it as a proxy to study the pattern in which new mathematical theories emerge and propagate. We liken our use of the twitter social network to the concept of erdos number; using the following and followee data from Twitter API as ‘collaboration’. Instead of using a single individual as in the case of erdos number, we decided to curate two sets of ground users who will become the root node(s) of the so-called twitter collaboration graph. (Figure 6.1) depicts the same.



```
ids = []
screenname = input("Enter your twitter user name (without the @)")
for page in tw.Cursor(api.friends_ids, screen_name=screenname).pages():
    ids.extend(page)
    time.sleep(60)

screen_names = [user.screen_name for user in api.lookup_users(user_ids=ids)]
print(screen_names)
```

```
Enter your twitter user name (without the @)realDonaldTrump
['JudgeJeanine', 'Jim_Jordan', 'MariaBartiromo', 'VP', 'GOPChairwoman', 'parscale', 'PressSec', 'TuckerCarlson',
```

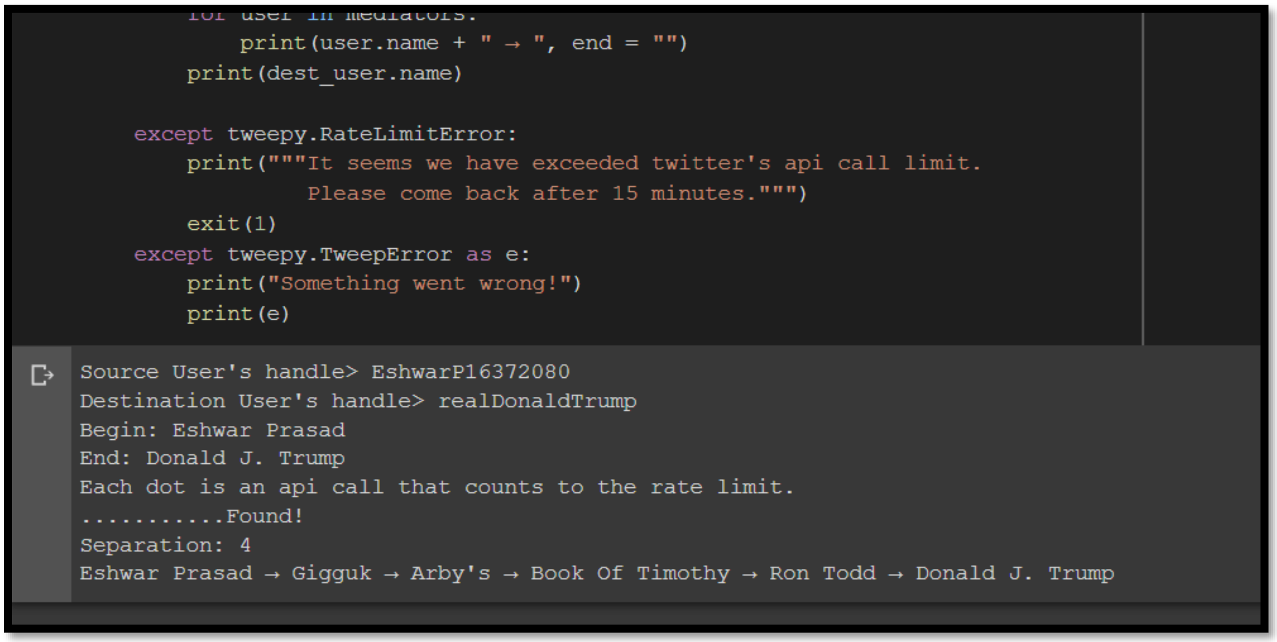
Figure 6.2: Fetching all the accounts followed by a given twitter handle.

6.3 Algorithm Used

The objective of the module is to assign a classification label on the input twitter handle from the two labels, namely, Democrat or Republican. The methodology we have conceived is that the degree of separation will be the lesser of the average degrees of separation from two sets of ground users, specifically, the set of ground democrats and set of ground republicans. Mathematically, it can be represented as:

$$Y = \min\left(\left(\frac{1}{n}\sum_{1=1}^{n}(a_i)\right),\left(\frac{1}{m}\sum_{j=1}^{m}(b_j)\right)\right)$$

where ‘n’ is the number of ground democrats and ‘m’ is the number of ground republicans. To achieve this objective, the core algorithm has to be able handle the simple task of finding the degree of separation between 2 given twitter users. That is, count the number of users that fall intermediate between the 2 users. The two types of edges that are available in the social graph are: Followers, the incoming edges and the people that the user follows known as Friends, the outgoing edges. There have been a few works that have experimented on the degrees of separation in social graphs like Facebook, but the definition of ‘degree of separation’ has been different. Typically, the average number of intermediate links that separate the given node from other nodes in the social graph was the focus of the calculation. Flajolet-Martin algorithm is an effective method to calculate this average. But our task is slightly different, and the closest work to our task has been dealt with in [24]. According to [24], a bidirectional search was the most effective way to handle such a task, but we have some complications in implementing a bidirectional search to our dynamic case, as we cannot build a predetermined social graph. Also, the twitter API does not allow us to make more than 15 API calls per 15 minutes, hence it is of monumental importance to reduce the number of API calls. In order to account for this, we decided to use a slightly different approach. Adding a heuristic to the traditional bidirectional search, we can improve upon the efficiency in our current scenario. Converting the bidirectional breadth-first-search into a probabilistic search further reduces the number of API calls we would have to make. The heuristic we used was to prioritize picking up nodes that had the maximum outgoing edges, but we probabilistically pick nodes that are closer to the starting node in order to emulate the breadth-first-search behaviour.



```
            for user in mediators:
                print(user.name + " → ", end = "")
            print(dest_user.name)

    except tweepy.RateLimitError:
        print("""It seems we have exceeded twitter's api call limit.
                 Please come back after 15 minutes.""")
        exit(1)
    except tweepy.TweepError as e:
        print("Something went wrong!")
        print(e)
```

```
Source User's handle> EshwarP16372080
Destination User's handle> realDonaldTrump
Begin: Eshwar Prasad
End: Donald J. Trump
Each dot is an api call that counts to the rate limit.
...........Found!
Separation: 4
Eshwar Prasad → Gigguk → Arby's → Book Of Timothy → Ron Todd → Donald J. Trump
```

Figure 6.3: Degrees of separation from account "@EshwarP16372080" to "@realDonaldTrump"

CHAPTER 7

TESTS & RESULTS

The predictions for political orientation are accompanied by a confidence score between the range of 0 to 1, which indicates the orientation of each individual message. Meaning, a message with a score 0.8 is considered extremely democratic in nature, and a score of 0.55 is considered slightly democratic, or even ambiguous, depending on the outlook. The model was tested with multiple sets of test cases to eliminate any innate bias. The first set of tests were run on a set of 50 previously untested politicians' twitter handles. The decision to select this set of test cases was made because it is unambiguous as to whether the input users are democratic or republic. It is a sensitive issue to label someone as one or the other unless she has proclaimed herself to be one, or is registered under one of the parties.

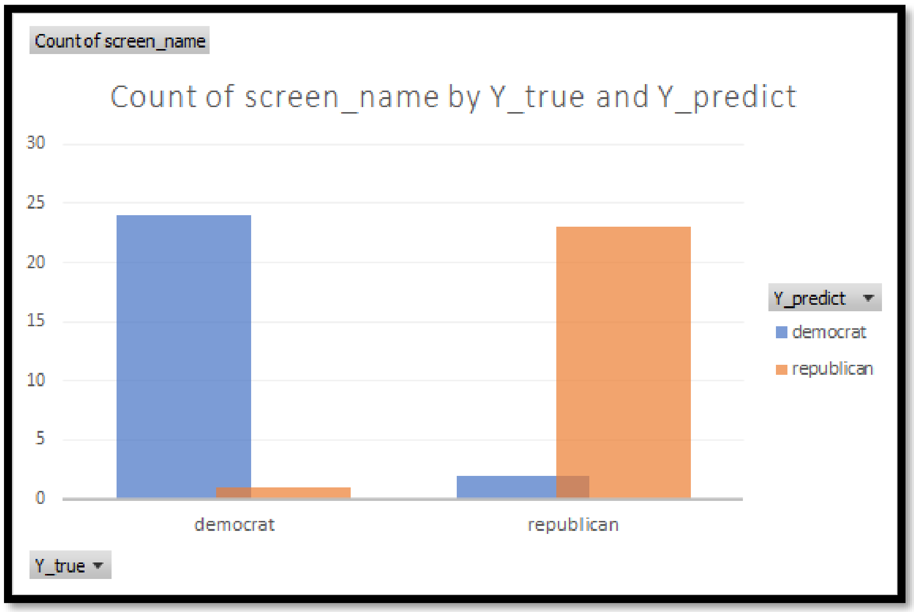


Figure 7.1: Figure depicting the results of classifications made on tweets made by 50 registered politicians.

Each test, i,e, the Tweet classification module, the retweet classification and the liked tweets classification was run on the selected members of the house, and the results were documented. (figure 7.1) depicts the classification results made by our model on 50 different members of the house. Out of the 25 democrats, 24 were properly classified and one misclassified with a score of 49 democratic tweets and 51 republican tweets, which is fairly ambiguous. On inspection into the misclassified handle, it was found to be Thomas Richard Suozzi, an American politician, attorney, and accountant who is the U.S. Representative for New York's 3rd district. His twitter timeline was fairly active, and especially with the recent, politically neutral events such as the Coronavirus outbreak, seems to have tweets that were more related to neutral events than to politically charged events. On the other hand, 23 republicans were properly classified by our model with 2 misclassifications. This is to be noted as it correlates to another hypothesis later in this section below.

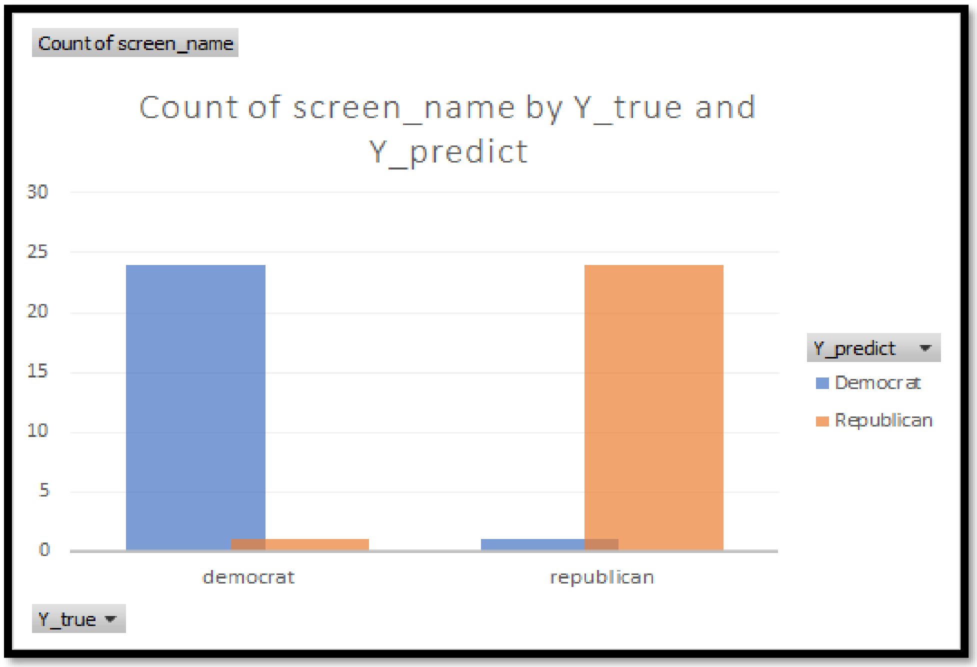


Figure 7.2: Figure depicting the results of classification made on Retweets made by 50 registered politicians.

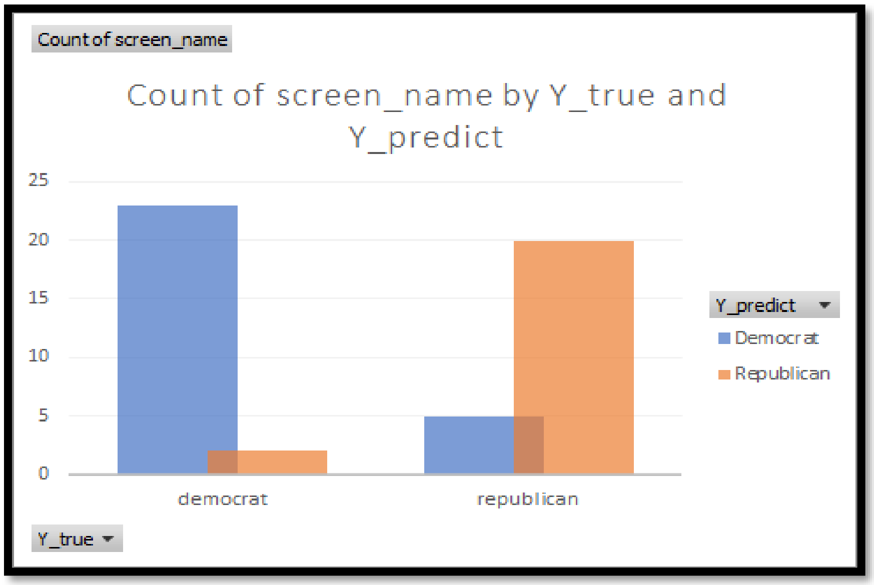


Figure 7.3: Figure depicting the results of classification made on Liked Tweets by 50 registered politicians.

The interesting observation in the tests so far on the registered politicians and on the next set of predictions (figure 7.4) that were run on 'impactful political twitter accounts' [25][26], is that the majority of the misclassifications tend to happen when the Y_true is republican. This brings us to our final hypothesis about the behavior of republicans on twitter, that they tend to be more sarcastic towards the subject matter than democrats on average. Our final set of tests were run on 38 users that are relatively active on twitter but a majority of them do not belong to a registered political party. They were arbitrarily chosen and act as our unbiased test cases.

An interesting observation from the results (figure 7.4) was that 17 out of 38 of them fell under a subtle balance of 40-60% democrat or republican. That is, 44.7% of the test cases were found to be lying in the 40-60% region, bringing us to the inference that the general public's tweets are much more neutrally charged compared to the tweets tweeted out by registered politicians. Our findings from the preliminary tests (fig. 7.1-7.5) support our initial hypothesis that politicians are relatively consistent with the language models that they follow while publishing tweets on the twitter platform. Furthermore, the fact that we used data from the period of 2018 U.S Congressional elections to train our model and tested it with the 100 most recent tweets published by the current house politicians further strengthens our hypothesis. Fig. 7.5 represents the results of our final ensemble classifier with weighted average confidence scores. The very intriguing observation from this graph is that the average confidence score for registered democrats is 0.751, while on the other hand, the average confidence score for registered republicans are just 0.531. This adds to our previous claim that the majority of misclassifications happen in republicans. This can be inferred in a few ways; one is that democrats are more outspoken with their views, and can be more aggressive in relaying their opinions on Twitter, compared to the more amorphous republicans.

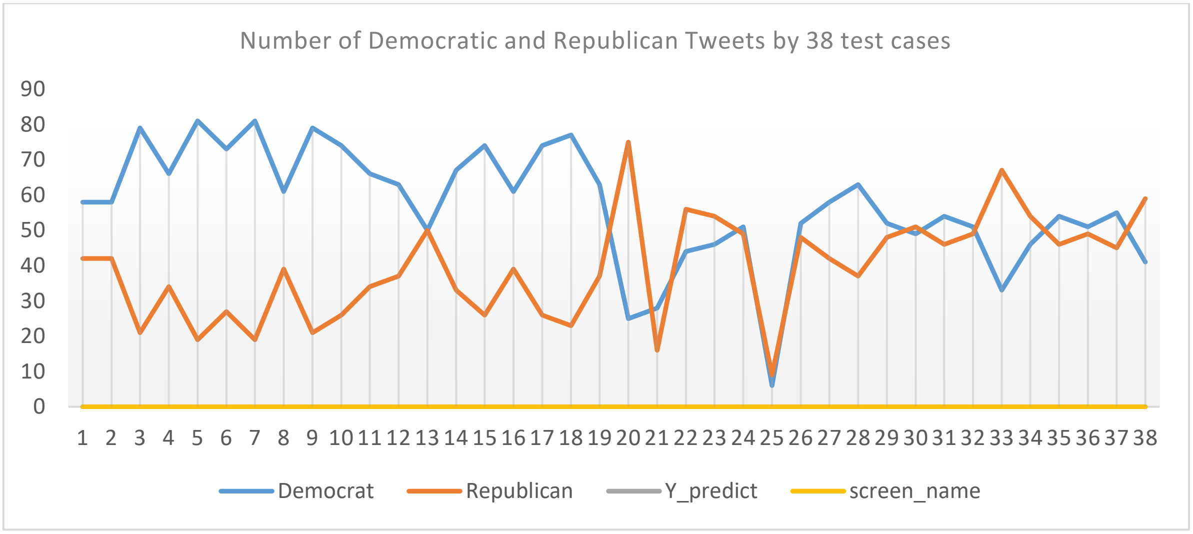


Figure 7.4: Visualization of classification results of *'impactful political twitter accounts'*.

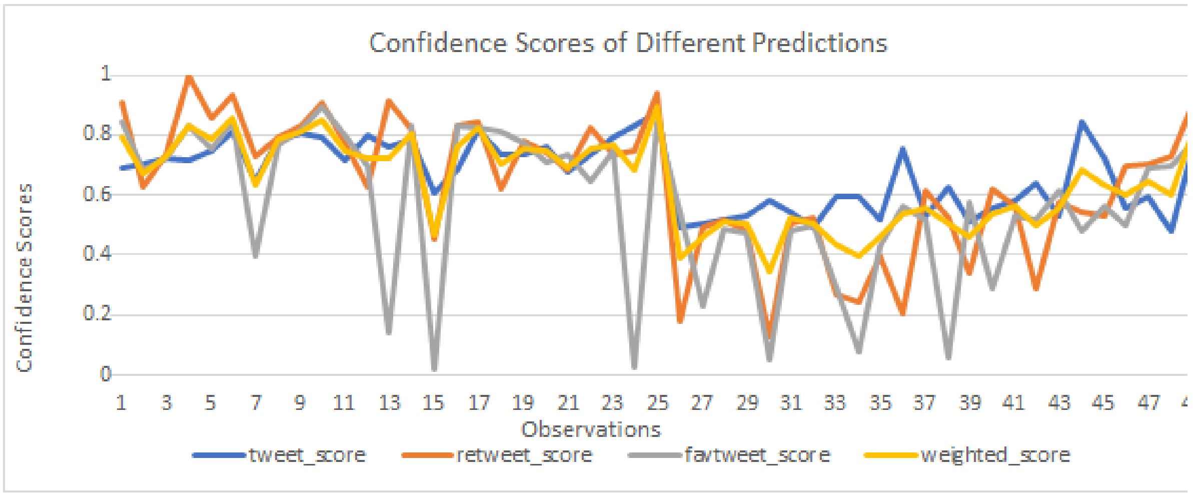


Figure 7.5: Visualization of Confidence Scores of Final Ensemble Classifier.

CHAPTER 8

CONCLUSIONS & FUTURE WORK

8.1 Conclusions & Summary

Determining the position of an individual on the political spectrum is a central research question in the field of political computation. Researchers have been exploring various sources of large amounts of political opinions such as newspapers articles, political manifestos, Reddit, Facebook and Twitter. The bigger the source of data is, the harder it is to parse and process the said data. In this paper, we have studied the case of determining the political orientation of a given twitter user, considering all the information that is available to researchers from twitter database. The type and number of hashtags and words used by politicians during major political events such as the 2018 U.S elections was analysed with over 86,000 tweets taken into consideration. Multiple network architectures were tested and the deep LSTM with Convolutional layers was found to outperform other traditional learning methods. Overall, we see an accuracy of over 81% in the best-case scenarios, and a 76% for worst case scenarios. Various types of testing data were considered, ranging from the most frequent tweets by politicians' twitter handles that the model was originally trained on, tweets by a completely new set of politicians, and tweets by 38 completely randomly picked out, politically active twitter users. Interestingly, the preliminary results correlated with the initial hypothesis that politicians are relatively consistent with the language models that they follow while publishing tweets on the twitter platform. Furthermore, the 38 unseen test cases have added to the second hypothesis that republicans tend to publish more ironic or sarcastic tweets compared to democrats. Consequently, it was also observed that the way in which politicians use twitter differs significantly from the way in which the general public does, since 44% of the arbitrary test cases have fallen in the 40%-60% range of polarity, meaning that arbitrary users tend to be politically amorphous in their tweets.

8.2 Future Work

Having stated these findings, we believe that this is a scientific step towards more research into the leverageable aspects of twitter data in Computational Political Science. The twitter social graph can further be made more effective if a reliable cached version of twitter data could be found that is easy to traverse, considering that the rate limiting on twitter API makes any kind of social networking algorithm ineffective as of now. The handling of tweet content analysis and classification can further be improved to handle spam accounts. It is also to be noted that sarcasm is still to be handled by our model and additions could be made to account for this. Works on detecting sarcasm have emerged [33] and this is a perfect opportunity to employ various NLP tasks, as our model is readily adaptable to any kind of social network that allows messaging or publishing of opinions and links such as friends or followers, such as Facebook, Twitter and Reddit.

APPENDIX A

PROJECT CODE

A.1 Sample Code

TweetClassification.ipynb

```
!pip install chart_studio
import tweepy
import numpy as np
import statistics

import math
import sys
from itertools import zip_longest
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.model_selection import train_test_split
from nltk.tokenize.casual import TweetTokenizer
from nltk.stem.lancaster import LancasterStemmer
from spacy.lang.en.stop_words import STOP_WORDS
from nltk.probability import FreqDist
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from wordcloud import STOPWORDS
from textblob import TextBlob
from nltk.text import Text
from pandas import Series
import seaborn as sns
import nltk, string
import pandas as pd
import numpy as np
import nltk as nlp
import warnings
import sys
import os
import re
```

```
%tensorflow_version 1.x
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.layers import LSTM, Bidirectional, CuDNNLSTM
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
from keras.models import model_from_json

from imblearn.over_sampling import SMOTE
import nltk
nltk.download('punkt')
nltk.download('wordnet')

STOPWORDS.add("rt")
STOPWORDS.add("s")
STOPWORDS.add("u")
STOPWORDS.add("amp")
STOPWORDS.add("th")
STOPWORDS.add("will")
STOPWORDS.add("t")
STOPWORDS.add("m")
from google.colab import drive
drive.mount('/content/drive')
df                    =                    pd.read_csv("/content/drive/My
Drive/tweet/ExtractedTweets.csv")
df.dropna(axis = 0, inplace = True)
df["Party_log"] = [1 if each == "Democrat" else 0 for each in df.Party]
def extract_hash_tags(twt):
       hashtag = re.findall('(\#[A-Za-z_]+)', twt)
       if hashtag:
          return hashtag
       else:
           return ""
df_h = df.copy()
df_h['top_hashtags']        =        df_h['Tweet'].apply(lambda        x:
extract_hash_tags(x))

hashtags_list_rep = []
hashtags_list_dem = []

for n in range(len(df_h['top_hashtags'])):
    if df_h['Party_log'][n] == 0:
        hashtags_list_rep += df_h['top_hashtags'][n]
    elif df_h['Party_log'][n] == 1:
        hashtags_list_dem += df_h['top_hashtags'][n]
Dem_tophashtags                                                                 =
Series(hashtags_list_dem).value_counts().head(n=10)
sns.set_style("white")
sns.set(font_scale=1.0)
plt.figure(figsize=(12,6))
ax1   =   sns.barplot(x=Dem_tophashtags.index,   y=Dem_tophashtags,
orient='v',   palette="Greens_r").set_title("Democrats'   10   most   used
hashtags")
plt.xlabel("Hashtags")
plt.ylabel("Count")
plt.xticks(rotation=40)
plt.show(ax1)
#ax1.set_xticklabels(ax.get_xticklabels(), rotation=40)
Rep_tophashtags                                                                 =
Series(hashtags_list_rep).value_counts().head(n=10)
sns.set_style("white")
sns.set(font_scale=1.0)
plt.figure(figsize=(12, 6))
ax2   =   sns.barplot(x=Rep_tophashtags.index,   y=Rep_tophashtags,
orient='v',   palette="Purples_r").set_title("Republicans'   10   most   used
hashtags")
plt.xlabel("Hashtags")
plt.ylabel("Count")
plt.xticks(rotation=40)
plt.show(ax2)
democrat=df[df.Party=="Democrat"]
republican=df[df.Party=="Republican"]
#add some unnecessary words to STOPWORDS list
STOPWORDS.add("rt")
STOPWORDS.add("s")
STOPWORDS.add("u")
STOPWORDS.add("amp")
STOPWORDS.add("th")
STOPWORDS.add("will")
STOPWORDS.add("t")
STOPWORDS.add("m")
democrat_list=[]
for d in democrat.Tweet:
    d=re.sub(r'http\S+', '', d) #remove links
    d=re.sub("[^a-zA-Z]", " ", d) #remove all characters except letters
    d=d.lower() #convert all words to lowercase
    d=nltk.word_tokenize(d) #split sentences into word
    d=[word for word in d if not word in STOPWORDS] #remove the
stopwords
    lemma=nlp.WordNetLemmatizer()
    d=[lemma.lemmatize(word) for word in d] #identify the correct form
of the word in the dictionary
    d=" ".join(d)
    democrat_list.append(d) #append words to list
#same process as before
republican_list=[]
for r in republican.Tweet:
    r=re.sub(r'http\S+', '', r)
    r=re.sub("[^a-zA-Z]", " ", r)
    r=r.lower()
    r=nltk.word_tokenize(r)
    r=[word for word in r if not word in STOPWORDS]
    lemma=nlp.WordNetLemmatizer()
    r=[lemma.lemmatize(word) for word in r]
    r=" ".join(r)
    republican_list.append(r)
democrat_tweets=str(democrat_list).split()
democrat_tweets=[word.replace("'","") for word in democrat_tweets ]
democrat_tweets=[word.replace("[", "") for word in democrat_tweets ]
democrat_tweets=[word.replace("]","") for word in democrat_tweets ]
democrat_tweets=[word.replace(",", "") for word in democrat_tweets ]

republican_tweets=str(republican_list).split()
republican_tweets=[word.replace("'","") for word in republican_tweets
]
republican_tweets=[word.replace("[", "") for word in republican_tweets
]
republican_tweets=[word.replace("]","") for word in republican_tweets
]
republican_tweets=[word.replace(",", "") for word in republican_tweets
]
#FreqDist records the number of times each words are used.
fdist_democrat = FreqDist(democrat_tweets)
fdist_republican=FreqDist(republican_tweets)
print("Frequency of Words : Democrat")
fdist_democrat
```

```
Dem_freqw=Series(fdist_democrat).sort_values(ascending=False).hea
d(10)
Rep_freqw=Series(fdist_republican).sort_values(ascending=False).hea
d(10)
sns.set_style("ticks")
sns.set(font_scale=1.2)
plt.figure(figsize=(12,6))
ax3  =  sns.barplot(x=Dem_freqw.index,  y=Dem_freqw,  palette  =
'rocket').set_title("Democrats' 10 most used words")
plt.xlabel("Words")
plt.ylabel("Count")
plt.xticks(rotation=40)
plt.show(ax3)
sns.set_style("white")
sns.set(font_scale=1.2)
plt.figure(figsize=(12,6))
ax4  =  sns.barplot(x=Rep_freqw.index,  y=Rep_freqw,  palette  =
'Purples_r').set_title("Republicans' 10 most used words")
plt.xlabel("Words")
plt.ylabel("Count")
plt.xticks(rotation=40)
plt.show(ax4)
top_words = 10000
tokenizer = TweetTokenizer(reduce_len=True)
tweets = df.copy()

# cleaning
tweet_arr = tweets.Tweet.to_numpy()
tweet_list = []
lemma = nlp.WordNetLemmatizer()

for d in tweet_arr:
   d = re.sub(r'http\S+', '', d) #remove links
   d = re.sub("[^a-zA-Z]", " ", d) #remove all characters except letters
   d = d.lower() #convert all words to lowercase
   d = nltk.word_tokenize(d) #split sentences into word
   d = [word for word in d if not word in STOPWORDS] #remove the
stopwords
   d = [lemma.lemmatize(word) for word in d] #identify the correct
form of the word in the dictionary
   d = " ".join(d)
   tweet_list.append(d)
```

```
tweet_arr = np.asarray(tweet_list)

tweets['Tweet'] = tweet_arr

# tokenizing
tweets['Tweet'] = tweets.Tweet.apply(tokenizer.tokenize)

# converting tweet tokens to freq dist ranks
fdist = FreqDist(word for tweet in tweets.Tweet for word in tweet)
terms = [term for term, count in fdist.most_common(top_words)]
tweets.Tweet = tweets.Tweet.apply(lambda tweet:
                              [terms.index(term) if term in terms else 0
                               for term in tweet])
# padding every tweet to max review length

x = tweets.Tweet
y = tweets.Party_log

max_review_length = 50
x = sequence.pad_sequences(x, maxlen=max_review_length)

print(x.shape)
print(y.shape)
# sourced from
import keras.backend as K
def matthews_correlation(y_true, y_pred):
    y_pred_pos = K.round(K.clip(y_pred, 0, 1))
    y_pred_neg = 1 - y_pred_pos

    y_pos = K.round(K.clip(y_true, 0, 1))
    y_neg = 1 - y_pos

    tp = K.sum(y_pos * y_pred_pos)
    tn = K.sum(y_neg * y_pred_neg)

    fp = K.sum(y_neg * y_pred_pos)
    fn = K.sum(y_pos * y_pred_neg)

    numerator = (tp * tn - fp * fn)
    denominator = K.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn))

    return numerator / (denominator + K.epsilon())
def get_stats(y_test, y_pred):
 from sklearn.metrics import matthews_corrcoef, confusion_matrix,
accuracy_score, f1_score

 print('MCC: ', matthews_corrcoef(y_test, np.round(y_pred)))
 print('Accuracy Score: ', accuracy_score(y_test, np.round(y_pred)))
 print('F1 Score: ', f1_score(y_test, np.round(y_pred)))
 print('Confusion Matrix ')
 print(confusion_matrix(y_test, np.round(y_pred)))
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                     test_size=0.2,
                                                     random_state=0)
# serialize model to JSON
model_json = model.to_json()
with open("/content/drive/My Drive/tweet/model.json", "w") as
json_file:
   json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("/content/drive/My Drive/tweet/model.h5")
print("Saved model to disk")
# load json and create model
json_file = open('/content/drive/My Drive/tweet/model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("/content/drive/My
Drive/tweet/model.h5")
print("Loaded model from disk")
lemma = nlp.WordNetLemmatizer()
tokenizer = TweetTokenizer(reduce_len=True)
max_review_length = 50
def clean_input(tweet):
   tweet = re.sub(r'http\S+', '', tweet) #remove links
   tweet = re.sub("[^a-zA-Z]", " ", tweet) #remove all characters except
letters
   tweet = tweet.lower() #convert all words to lowercase
   tweet = nltk.word_tokenize(tweet) #split sentences into word
   tweet = [word for word in tweet if not word in STOPWORDS] #remove
the stopwords
   tweet = [lemma.lemmatize(word) for word in tweet] #identify the
correct form of the word in the dictionary
   tweet = " ".join(tweet)
   return tweet
```

```
def convert_input(tweet):
   tweet = clean_input(tweet)
   # tokenizing
   tweet = tokenizer.tokenize(tweet)
   # converting tweet tokens to freq dist ranks
   terms = [term for term, count in fdist.most_common(top_words)]
   encoded_tweet = [terms.index(term) if term in terms else 0 for term
in tweet]
   tweet_arr = np.array([encoded_tweet])
   padded_tweet          =          sequence.pad_sequences(tweet_arr,
maxlen=max_review_length)
   return padded_tweet

def predict_tweet(tweet):
   tweet = convert_input(tweet)
   prediction = loaded_model.predict(tweet)
keys = dict(consumer_key="yuNQZSADRNqRiDj0U3oOWEaE9",
         consumer_secret="XXXXXXXXXXXXXX",
         access_token=”XXXXXXXXXXXX”,
         access_token_secret=”XXXXXXXXXXXXX”
         )

class Tweet:
   def __init__(self, user_handle):
      """
      :param user_handle: twitter username without '@' symbol
      :return: class method
      """
      self._consumer_key = keys['consumer_key']
      self._consumer_secret = keys['consumer_secret']
      self._access_token = keys['access_token']
      self._access_token_secret = keys['access_token_secret']

      # configure OAUTH
      self.auth         =         tweepy.OAuthHandler(self._consumer_key,
self._consumer_secret)
      self.auth.set_access_token(self._access_token,
self._access_token_secret)

      # set up tweepy client
      self.api = tweepy.API(
         self.auth,
            wait_on_rate_limit=True,
            wait_on_rate_limit_notify=True,
            timeout=60,
            compression=True
        )

#Networking Module

consumer_key = "yuNQZSADRNqRiDj0U3oOWEaE9"
consumer_secret                                                =
"OuJVoxDFBTy7wiePcEW0d0RkuHSaQwr5niBTWAEMdASnTWcOrX"
access_token                     =                     '1168231254308712449-
XuhViryRpYsrhYuhpETXImWFTHViis'
access_token_secret                                            =
'ZphiLvqrYIZpYsjGLeoD8lDDxU4ZgXC2wQT1WXpwJlykU'


class Frontier():
    """ Objects of this class keep track of the Breadth-First Search
    Frontier as it is expanded.
    Moreover, this implements the probabilistic expansion of nodes on
the
    perimeter.
    """

    def __init__(self, src, expander, get_out_degree, lookup):
        self.internal = set()
        self.perimeter = set()
        self.distribution = {}

        self.perimeter.add(src)

        # self.distribution is a map of distances to the nodes that lie at that
        # distance from the source.
        self.distribution[0] = [src]

        # some metadata to be able to trace back the path from a node
to
        # the source node
        self.metadata = {}
        self.metadata[src] = {"distance": 0, "parent": None}

        # Our expander makes a set out of the iterable returned
        self.expander = lambda u: set(expander(u))

        # get_out_degree is a function for obtaining the out-degree of a
node.
        # It is useful because the way we compute out-degrees is different
for
        # the BFS for the source and the destination nodes.
        self.get_out_degree = get_out_degree

        self.lookup = lookup

    def expand_perimeter(self):
        """This is going to implement the probabilistic algorithm for
        expanding the BFS territory.
        We pick the node with maximum outdegree at a specific
        distance. The distance is picked from an exponential probability
        distribution that favors BFS expansion, i.e. lower distances are
        more likely to be picked and the probability of a distance
        decreases exponentially with the distance value.
        """
        distances = self.distribution.keys()
        probabilities = []

        # Calculate probabilities for the distances
        for d in distances:
            probabilities.append(self._pdf(d))

        # Sample a distance and then a node to expand
        d                  =                  np.random.choice(a=list(distances),
p=self._normalize(probabilities))

        # I think this is getting more difficult by not taking in the api object
        # as one of its inputs.
        nodes_at_d = self.lookup(self.distribution[d])
        u = max(nodes_at_d, key=self.get_out_degree).id

        self.internal.add(u)
        self.perimeter.remove(u)

        d = self.metadata[u]["distance"]
        self.distribution[d].remove(u)

        # Do not keep empty lists
        if self.distribution[d] == []:
            del self.distribution[d]

        # let's move the frontier forward at the node 'u'
        new_nodes        =        self.expander(u).difference(self.internal,
self.perimeter)
        # print(".", end="")
        sys.stdout.flush()

        # Keep track of distance of a node from src and its parent
        d = self.metadata[u]["distance"]
        for n in new_nodes:
            self.metadata[n] = {"distance": d + 1, "parent": u}
            try:
                self.distribution[d + 1].append(n)
            except KeyError:
                self.distribution[d + 1] = []

        self.perimeter.update(new_nodes)

        return set(new_nodes)

    def is_on_perimeter(self, user_id):
        """ Tells whether user_id is in on the perimeter of this bfs frontier.
        """
        return (user_id in self.perimeter)

    def covered_all(self):
        """ True if we have covered all the graph, i.e.
        The whole graph is now internal. There remain no nodes on
        the periphery.
        """
        return bool(self.perimeter)

    def _pdf(self, d):
        """ Compute the probability of selecting a distance d.
        p(d) = exp(alpha * d).
        We   normalize   values   returned   by   this   function   later   with
_normalize()
        """
        alpha = -2
        return math.exp(alpha * d)
    def _normalize(self, probs):
        """ Normalize probability values to bring them in [0, 1] and to make
        their sum equal to 1.
        """
        probs = np.array(probs)
        probs = probs / probs.sum()
        return probs

    def get_parent(self, n):
        """ Returns the parent for node n.
        Will throw KeyError when we haven't seen the node before.
        But in this application, we don't expect that to happen. So, if it
        happens, something is really messed up.
        """
        return self.metadata[n]["parent"]

    def get_distance(self, n):
        """" Returns the distance of node `n` from the source.
        """
        return self.metadata[n]["distance"]


def grouper(iterable, n, fillvalue=None):
    "Collect data into fixed-length chunks or blocks"
    args = [iter(iterable)] * n
    return zip_longest(*args, fillvalue=fillvalue)


def safe_lookup_users(api, ids):
    """ Handles looking up users more than 100.
    """
    users = []
    for batch_ids in grouper(ids, 100):
        ids = filter(lambda n: n != None, list(batch_ids))
        try :
          users.extend(api.lookup_users(user_ids=ids))
        except tweepy.TweepError as ex:
          if ex.reason == "Not authorized.":
            print("A Particular Twitter User was not authorized for access,
skipping")
    return users
```

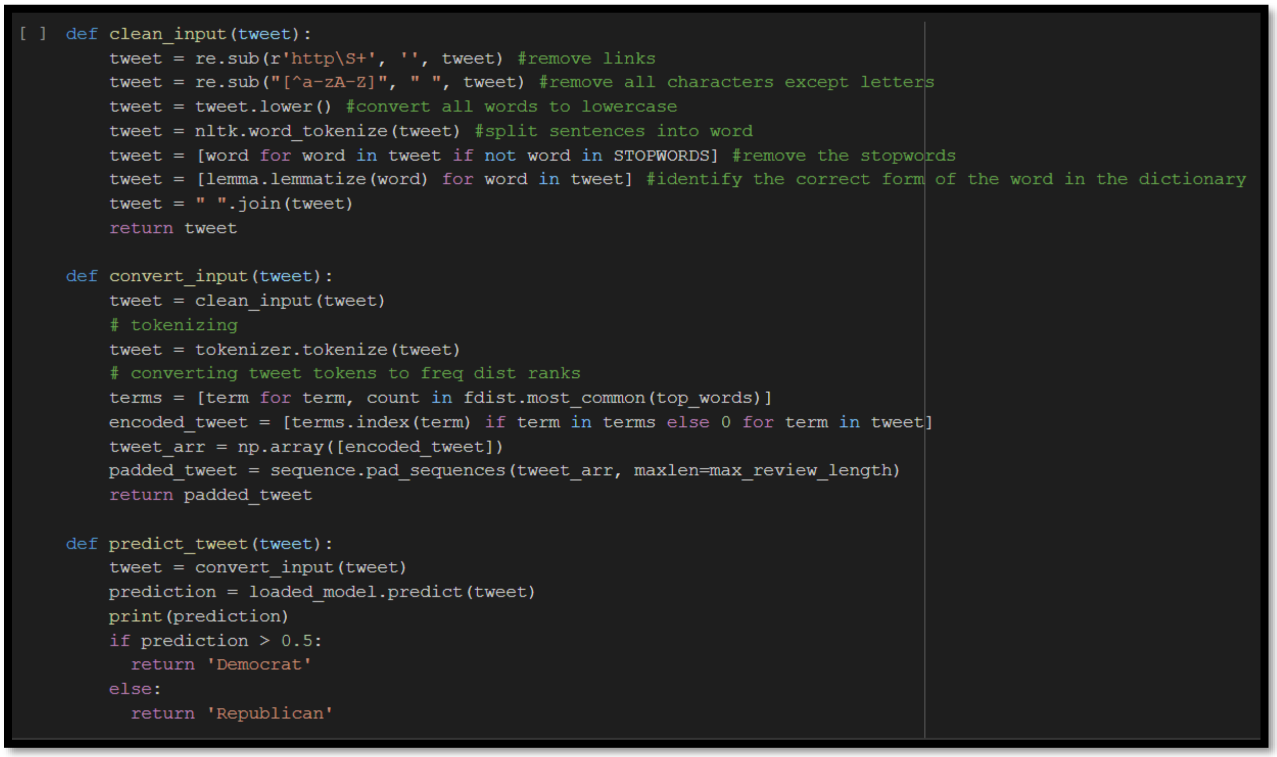
```
#if __name__ == "__main__":
def networkprediction (inputhandle):
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)

    source = inputhandle

    dest_dem_usrs = ['BernieSanders', 'AOC', 'HillaryClinton']
    dest_rep_usrs = ['realDonaldTrump', 'VP', 'GOP']

    api = tweepy.API(auth, wait_on_rate_limit=True,
            wait_on_rate_limit_notify=True,
            timeout=60,
            compression=True)

    try:
        # Get user ids from the user handles
        src_user = api.get_user(source)

        dem = {}
        rep = {}

        for party in (dest_dem_usrs, dest_rep_usrs):
            if party == dest_dem_usrs:
                party_flag = 'dem'
            else:
                party_flag = 'rep'
            for destination in party:
                dest_user = api.get_user(destination)
                if source == destination :
                  separation = 0
                else :
                  src_frontier = Frontier(src_user.id, api.friends_ids
                                        , lambda n: n.friends_count
                                        , lambda ids: safe_lookup_users(api, ids))
                  dest_frontier = Frontier(dest_user.id, api.followers_ids
                                         , lambda n: n.followers_count
                                         , lambda ids: safe_lookup_users(api, ids))
                  while                 src_frontier.covered_all()                  or
dest_frontier.covered_all():
                    # Expand the source node's frontier first
                    nodes = src_frontier.expand_perimeter()
```

```
cols    =    ['screen_name',    'tweet_score',    't_party',    'retweet_score',
'rt_party',
       'favtweet_score', 'ft_party', 'weighted_score', 'Y_pred', 'Y_true']
df = pd.DataFrame(columns=cols)
for _, row in testcases.iterrows():
  print(row['screen_name'])
  finalres = get_final_predictions(row['screen_name'])
  df = df.append({'screen_name': row['screen_name'],
                'tweet_score': finalres[0],
                't_party': ['democrat' if finalres[1] > 0.5 else 'republican'][0],
                'retweet_score': finalres[2],
                'rt_party': ['democrat' if finalres[3] > 0.5 else 'republican'][0],
                'favtweet_score':finalres[4],
                'ft_party': ['democrat' if finalres[5] > 0.5 else 'republican'][0],
                'weighted_score':finalres[6],
                'Y_pred':finalres[7],
                'Y_true':row['Y_true']
                },
               ignore_index=True)
df.to_csv("/content/drive/My Drive/tweet/TestResults.csv")
```

APPENDIX B

SCREENSHOTS

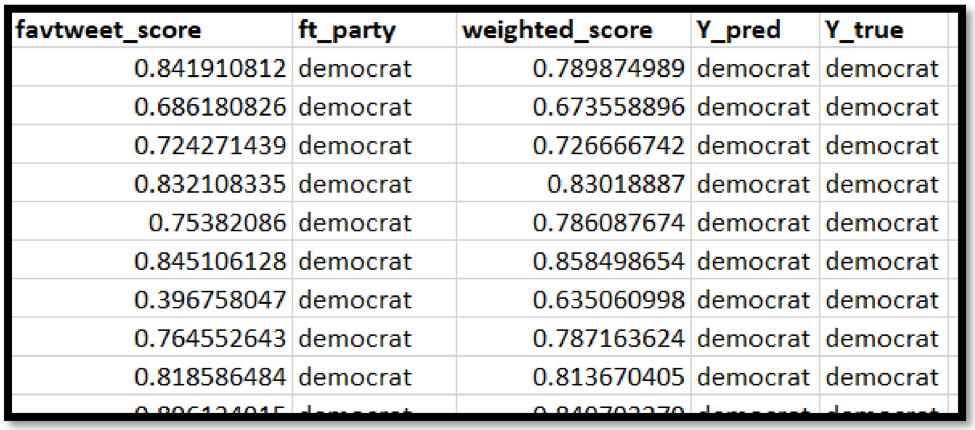
B.1 Tweet Analysis Module



```
[ ]  def clean_input(tweet):
         tweet = re.sub(r'http\S+', '', tweet) #remove links
         tweet = re.sub("[^a-zA-Z]", " ", tweet) #remove all characters except letters
         tweet = tweet.lower() #convert all words to lowercase
         tweet = nltk.word_tokenize(tweet) #split sentences into word
         tweet = [word for word in tweet if not word in STOPWORDS] #remove the stopwords
         tweet = [lemma.lemmatize(word) for word in tweet] #identify the correct form of the word in the dictionary
         tweet = " ".join(tweet)
         return tweet

     def convert_input(tweet):
         tweet = clean_input(tweet)
         # tokenizing
         tweet = tokenizer.tokenize(tweet)
         # converting tweet tokens to freq dist ranks
         terms = [term for term, count in fdist.most_common(top_words)]
         encoded_tweet = [terms.index(term) if term in terms else 0 for term in tweet]
         tweet_arr = np.array([encoded_tweet])
         padded_tweet = sequence.pad_sequences(tweet_arr, maxlen=max_review_length)
         return padded_tweet

     def predict_tweet(tweet):
         tweet = convert_input(tweet)
         prediction = loaded_model.predict(tweet)
         print(prediction)
         if prediction > 0.5:
           return 'Democrat'
         else:
           return 'Republican'
```

Figure B.1: Tweet Analysis Module

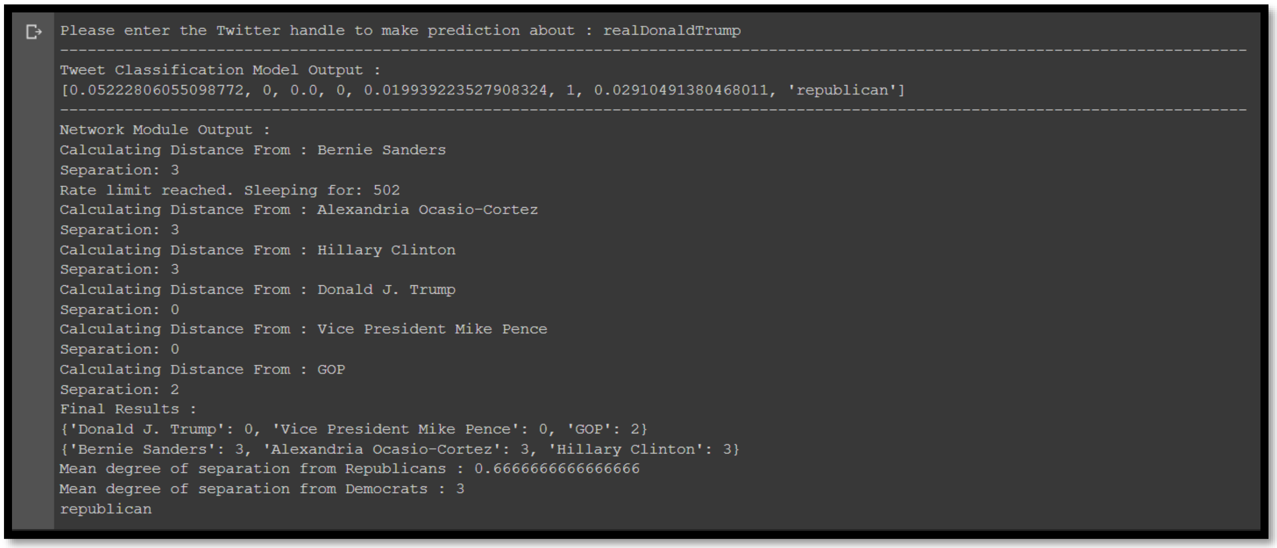
B.2 Prediction Values



favtweet_score	ft_party	weighted_score	Y_pred	Y_true
0.841910812	democrat	0.789874989	democrat	democrat
0.686180826	democrat	0.673558896	democrat	democrat
0.724271439	democrat	0.726666742	democrat	democrat
0.832108335	democrat	0.83018887	democrat	democrat
0.75382086	democrat	0.786087674	democrat	democrat
0.845106128	democrat	0.858498654	democrat	democrat
0.396758047	democrat	0.635060998	democrat	democrat
0.764552643	democrat	0.787163624	democrat	democrat
0.818586484	democrat	0.813670405	democrat	democrat

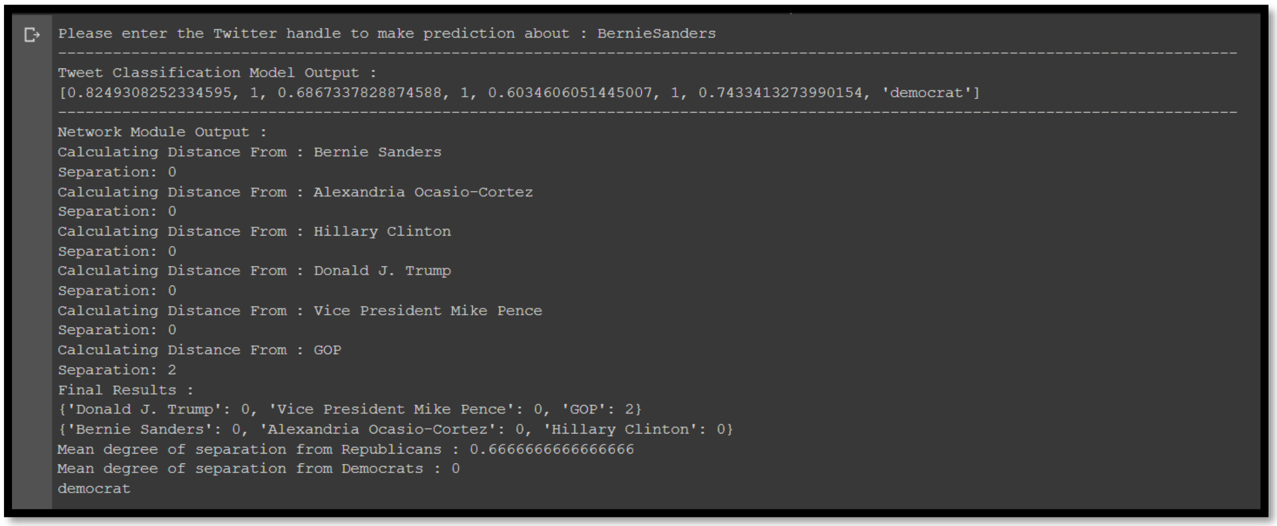
Figure B.2: Final Prediction Values from the Ensemble Classifier.

B.3 Runtime Screenshots



```
Please enter the Twitter handle to make prediction about : realDonaldTrump
--------------------------------------------------------------------------------------------------------------------------------
Tweet Classification Model Output :
[0.05222806055098772, 0, 0.0, 0, 0.019939223527908324, 1, 0.02910491380468011, 'republican']
--------------------------------------------------------------------------------------------------------------------------------
Network Module Output :
Calculating Distance From : Bernie Sanders
Separation: 3
Rate limit reached. Sleeping for: 502
Calculating Distance From : Alexandria Ocasio-Cortez
Separation: 3
Calculating Distance From : Hillary Clinton
Separation: 3
Calculating Distance From : Donald J. Trump
Separation: 0
Calculating Distance From : Vice President Mike Pence
Separation: 0
Calculating Distance From : GOP
Separation: 2
Final Results :
{'Donald J. Trump': 0, 'Vice President Mike Pence': 0, 'GOP': 2}
{'Bernie Sanders': 3, 'Alexandria Ocasio-Cortez': 3, 'Hillary Clinton': 3}
Mean degree of separation from Republicans : 0.6666666666666666
Mean degree of separation from Democrats : 3
republican
```

Figure B.3 : Output of the system for input handle “@realDonaldTrump”.



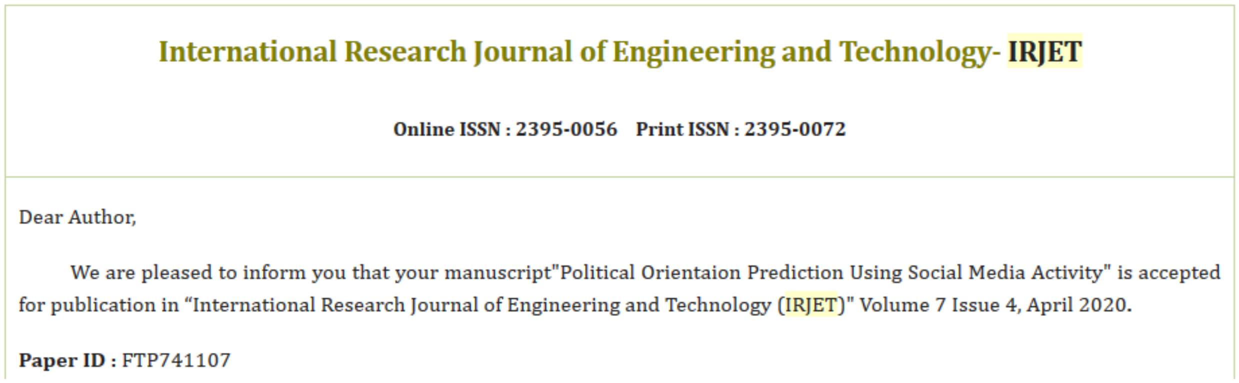
```
Please enter the Twitter handle to make prediction about : BernieSanders
--------------------------------------------------------------------------------------------------------------------------------
Tweet Classification Model Output :
[0.8249308252334595, 1, 0.6867337828874588, 1, 0.6034606051445007, 1, 0.7433413273990154, 'democrat']
--------------------------------------------------------------------------------------------------------------------------------
Network Module Output :
Calculating Distance From : Bernie Sanders
Separation: 0
Calculating Distance From : Alexandria Ocasio-Cortez
Separation: 0
Calculating Distance From : Hillary Clinton
Separation: 0
Calculating Distance From : Donald J. Trump
Separation: 0
Calculating Distance From : Vice President Mike Pence
Separation: 0
Calculating Distance From : GOP
Separation: 2
Final Results :
{'Donald J. Trump': 0, 'Vice President Mike Pence': 0, 'GOP': 2}
{'Bernie Sanders': 0, 'Alexandria Ocasio-Cortez': 0, 'Hillary Clinton': 0}
Mean degree of separation from Republicans : 0.6666666666666666
Mean degree of separation from Democrats : 0
democrat
```

Figure B.4 : Output of the system for input handle “@BernieSanders”.

APPENDIX C

PUBLICATION

C.1 Acceptance Letter



International Research Journal of Engineering and Technology- IRJET

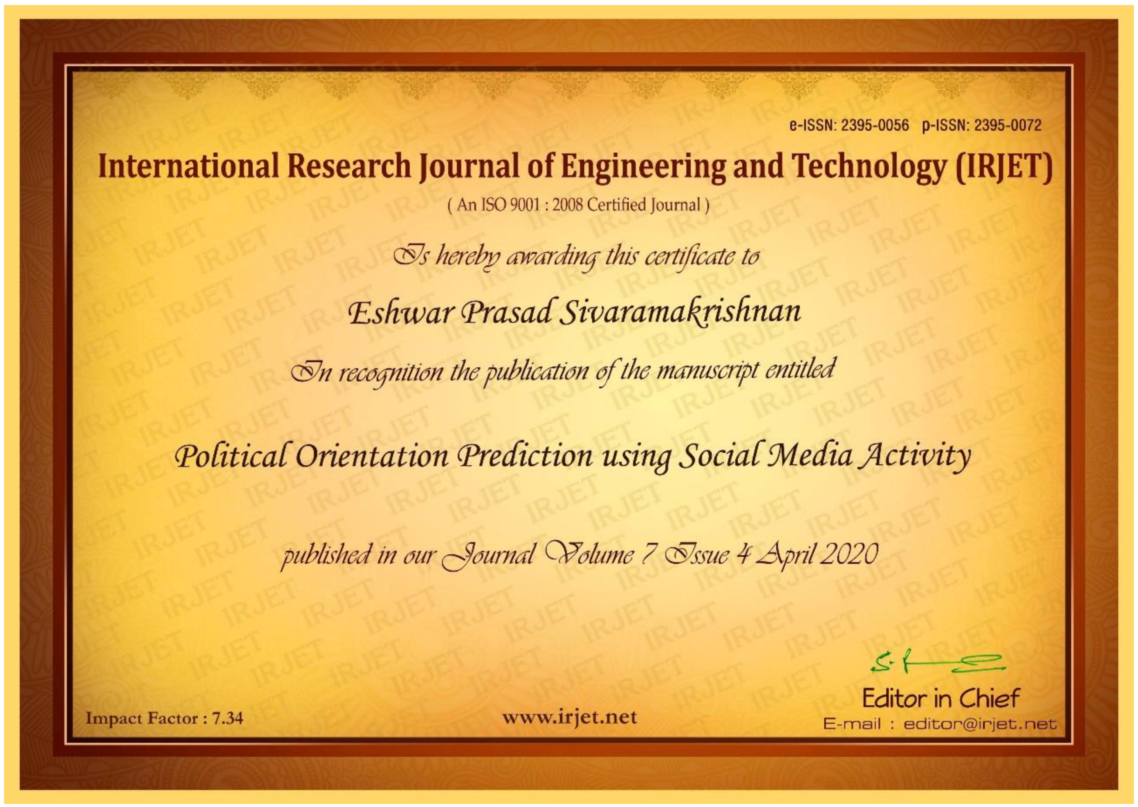
Online ISSN : 2395-0056 Print ISSN : 2395-0072

Dear Author,

We are pleased to inform you that your manuscript"Political Orientaion Prediction Using Social Media Activity" is accepted for publication in "International Research Journal of Engineering and Technology (IRJET)" Volume 7 Issue 4, April 2020.

Paper ID : FTP741107

C.2 Certificates



e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Eshwar Prasad Sivaramakrishnan

In recognition the publication of the manuscript entitled

Political Orientation Prediction using Social Media Activity

published in our Journal Volume 7 Issue 4 April 2020

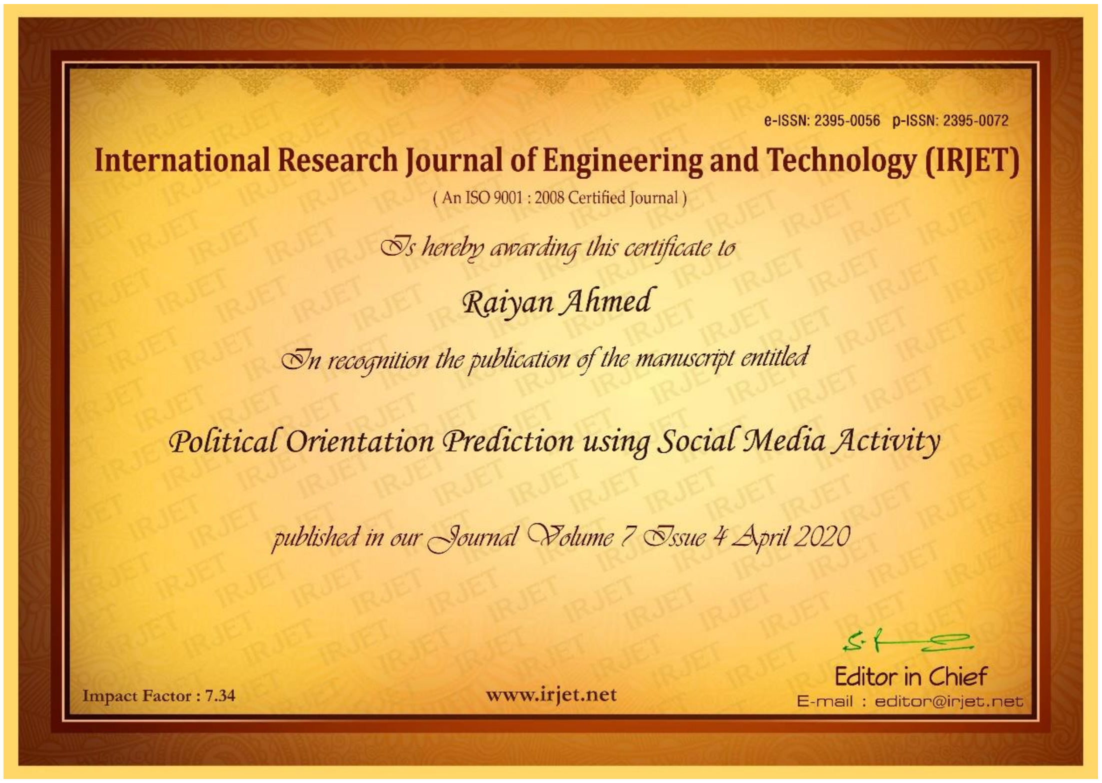
Impact Factor : 7.34

www.irjet.net

[illegible]

Editor in Chief

E-mail : editor@irjet.net


e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Raiyan Ahmed

In recognition the publication of the manuscript entitled

Political Orientation Prediction using Social Media Activity

published in our Journal Volume 7 Issue 4 April 2020

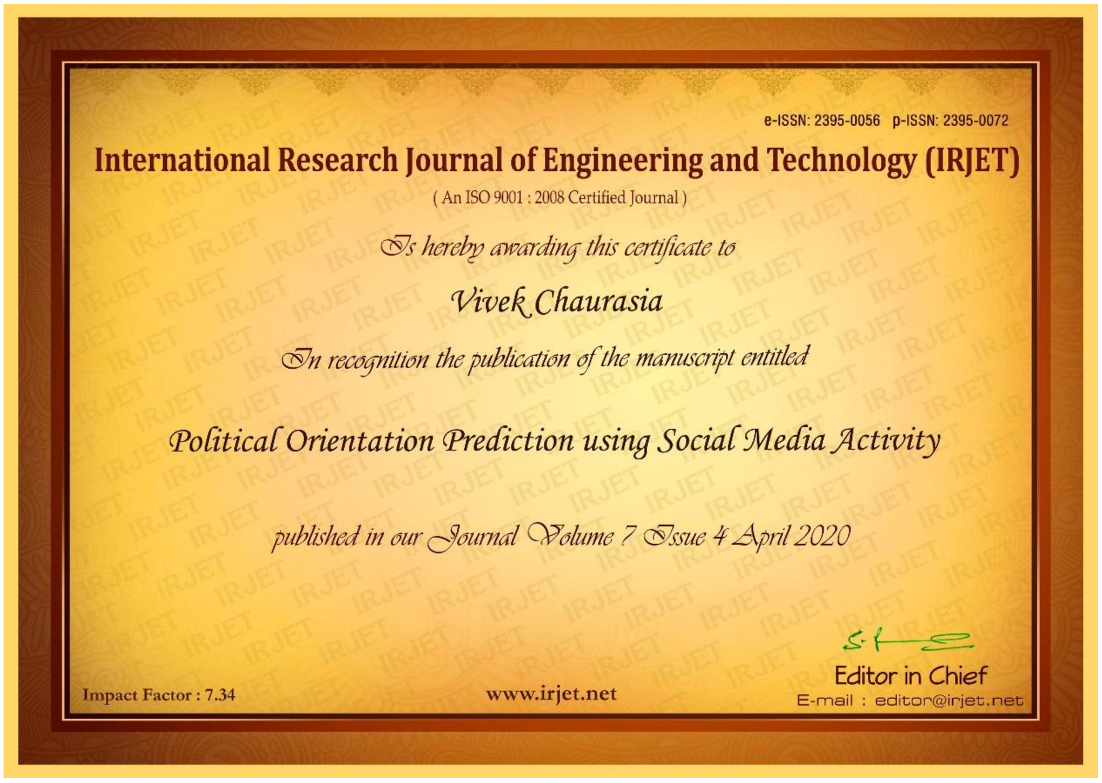
[illegible]

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 7.34

www.irjet.net


e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Vivek Chaurasia

In recognition the publication of the manuscript entitled

Political Orientation Prediction using Social Media Activity

published in our Journal Volume 7 Issue 4 April 2020

[illegible]

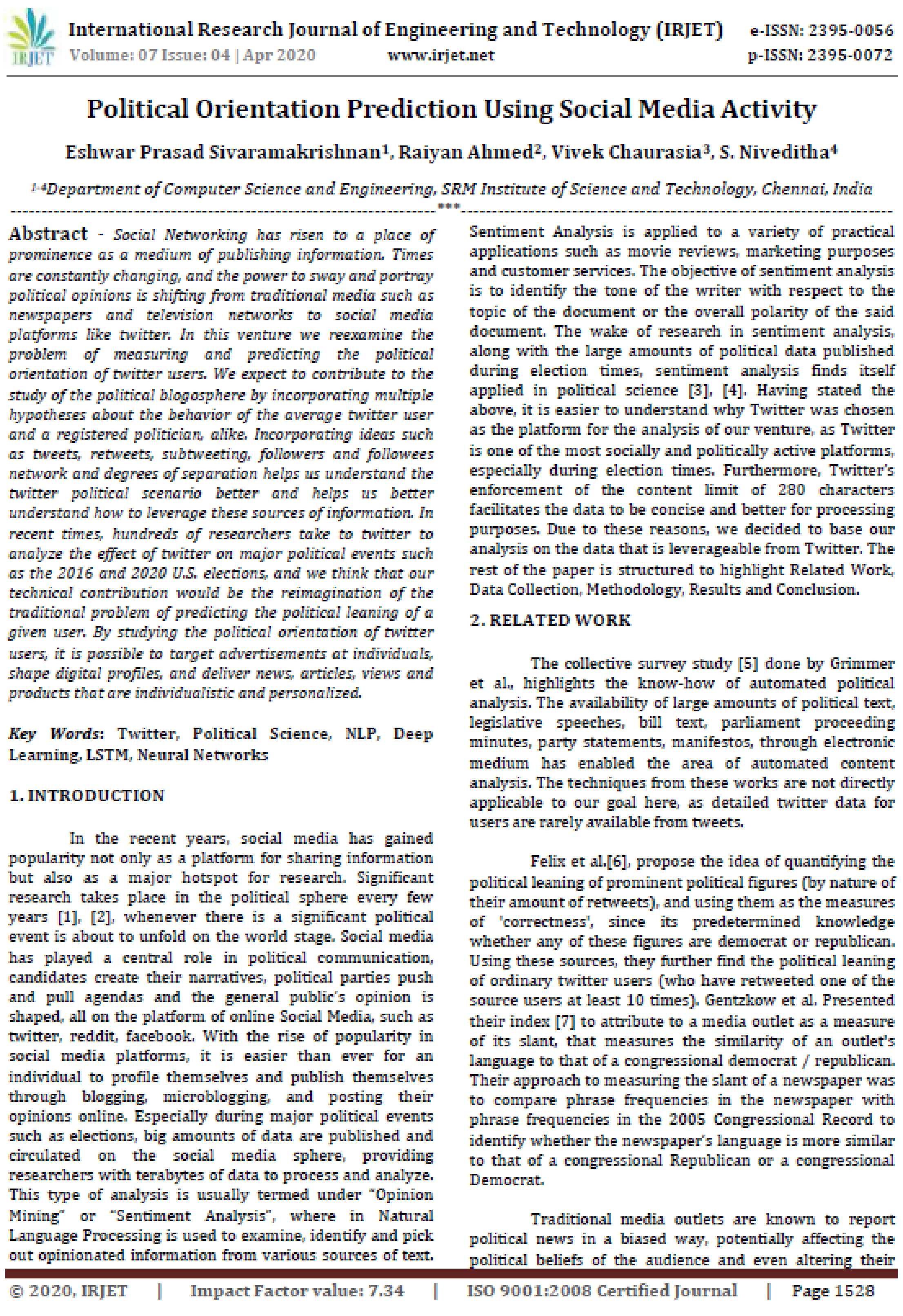
Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 7.34

www.irjet.net

C.2 Book of Abstract



International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

Political Orientation Prediction Using Social Media Activity

Eshwar Prasad Sivaramakrishnan[1], Raiyan Ahmed[2], Vivek Chaurasia[3], S. Niveditha[4]

[1-4]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India

***

Abstract - *Social Networking has risen to a place of prominence as a medium of publishing information. Times are constantly changing, and the power to sway and portray political opinions is shifting from traditional media such as newspapers and television networks to social media platforms like twitter. In this venture we reexamine the problem of measuring and predicting the political orientation of twitter users. We expect to contribute to the study of the political blogosphere by incorporating multiple hypotheses about the behavior of the average twitter user and a registered politician, alike. Incorporating ideas such as tweets, retweets, subtweeting, followers and followees network and degrees of separation helps us understand the twitter political scenario better and helps us better understand how to leverage these sources of information. In recent times, hundreds of researchers take to twitter to analyze the effect of twitter on major political events such as the 2016 and 2020 U.S. elections, and we think that our technical contribution would be the reimagination of the traditional problem of predicting the political leaning of a given user. By studying the political orientation of twitter users, it is possible to target advertisements at individuals, shape digital profiles, and deliver news, articles, views and products that are individualistic and personalized.*

Key Words: Twitter, Political Science, NLP, Deep Learning, LSTM, Neural Networks

1. INTRODUCTION

In the recent years, social media has gained popularity not only as a platform for sharing information but also as a major hotspot for research. Significant research takes place in the political sphere every few years [1], [2], whenever there is a significant political event is about to unfold on the world stage. Social media has played a central role in political communication, candidates create their narratives, political parties push and pull agendas and the general public's opinion is shaped, all on the platform of online Social Media, such as twitter, reddit, facebook. With the rise of popularity in social media platforms, it is easier than ever for an individual to profile themselves and publish themselves through blogging, microblogging, and posting their opinions online. Especially during major political events such as elections, big amounts of data are published and circulated on the social media sphere, providing researchers with terabytes of data to process and analyze. This type of analysis is usually termed under "Opinion Mining" or "Sentiment Analysis", where in Natural Language Processing is used to examine, identify and pick out opinionated information from various sources of text. Sentiment Analysis is applied to a variety of practical applications such as movie reviews, marketing purposes and customer services. The objective of sentiment analysis is to identify the tone of the writer with respect to the topic of the document or the overall polarity of the said document. The wake of research in sentiment analysis, along with the large amounts of political data published during election times, sentiment analysis finds itself applied in political science [3], [4]. Having stated the above, it is easier to understand why Twitter was chosen as the platform for the analysis of our venture, as Twitter is one of the most socially and politically active platforms, especially during election times. Furthermore, Twitter's enforcement of the content limit of 280 characters facilitates the data to be concise and better for processing purposes. Due to these reasons, we decided to base our analysis on the data that is leverageable from Twitter. The rest of the paper is structured to highlight Related Work, Data Collection, Methodology, Results and Conclusion.

2. RELATED WORK

The collective survey study [5] done by Grimmer et al., highlights the know-how of automated political analysis. The availability of large amounts of political text, legislative speeches, bill text, parliament proceeding minutes, party statements, manifestos, through electronic medium has enabled the area of automated content analysis. The techniques from these works are not directly applicable to our goal here, as detailed twitter data for users are rarely available from tweets.

Felix et al.[6], propose the idea of quantifying the political leaning of prominent political figures (by nature of their amount of retweets), and using them as the measures of 'correctness', since its predetermined knowledge whether any of these figures are democrat or republican. Using these sources, they further find the political leaning of ordinary twitter users (who have retweeted one of the source users at least 10 times). Gentzkow et al. Presented their index [7] to attribute to a media outlet as a measure of its slant, that measures the similarity of an outlet's language to that of a congressional democrat / republican. Their approach to measuring the slant of a newspaper was to compare phrase frequencies in the newspaper with phrase frequencies in the 2005 Congressional Record to identify whether the newspaper's language is more similar to that of a congressional Republican or a congressional Democrat.

Traditional media outlets are known to report political news in a biased way, potentially affecting the political beliefs of the audience and even altering their

REFERENCES

1. Bakliwal, A., Foster, J., Puil, J.V., O’brien, R., Tounsi, L., & Hughes, M. (2013). Sentiment Analysis of Political Tweets: Towards an Accurate Classifier.

2. Rao, Adithya & Spasojevic, Nemanja. (2016). Actionable and Political Text Classification using Word Embeddings and LSTM.

3. B. Pang, L. Lee, and S. Vaithyanathan, ”Thumbs up?: Sentiment classification using machine learning techniques” in Proc. ACL Conf. Empirical Methods Natural Lang. Process. (EMNLP), vol. 10, 2002, pp. 7986.

4. B. Pang and L. Lee, Opinion Mining and Sentiment Analysis, vol. 2, nos. 12. Boston, MA, USA: Now, 2008.

5. J. Grimmer and B. M. Stewart, “Text as data: The promise and pitfalls of automatic content analysis methods for political texts,” Political Analysis, 2013.

6. F. M. F. Wong, C. W. Tan, S. Sen and M. Chiang, ”Quantifying Political Leaning from Tweets, Retweets, and Retweeters,” in IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 8, pp. 2158-2172, 1 Aug. 2016

7. Gentzkow, M. and Shapiro, J.M. (2010), What Drives Media Slant? Evidence From U.S. Daily Newspapers. Econometrica, 78: 35-71.

8. AN, J.; CHA, M.; GUMMADI, K.; CROWCROFT, J.; QUERCIA, D.. Visualizing Media Bias through Twitter. International AAAI Conference on Web and Social Media, North America, may. 2012.

9. Sounman Hong, Sun Hyoung Kim, Political polarization on twitter: Implications for the use of social media in digital governments, Government Information Quarterly, Volume 33, Issue 4, 2016, Pages 777-782.

10. Twitter Character count, available online at: https://developer.twitter.com/en.html, accessed on March 22, 2020.

11. Luca Buccoliero, Elena Bellio, Giulia Crestini & Alessandra Arkoudas (2020) Twitter and politics: Evidence from the US presidential elections 2016, Journal of Marketing Communications, 26:1, 88-114.

12. Twitter Developer Document, available online at : https://developer.twitter.com/en/docs, accessed on March 22, 2020.

13. NLTK Documentation available online at https://www.nltk.org/index.html, accessed on March 22, 2020.

14. Keras Documentation available online at https://keras.io/, accessed on March 22, 2020.

15. Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, Fuad E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing, Volume 234, 2017, Pages 11-26

16. A. Graves, M. Liwicki, S. Fern´andez, R. Bertolami, H. Bunke and J. Schmidhuber, ”A Novel Connectionist System for Unconstrained Handwriting Recognition,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, pp. 855-868, May 2009.

17. Mikolov, Tomas & Sutskever, Ilya & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems. 26.

18. Kim, Yoon. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. 10.3115/v1/D14-1181.

19. Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.

20. He, Yulan; Saif, Hassan; Wei, Zhongyu and Wong, Kam-Fai (2012). Quantising opinions for political tweets analysis. In: LREC 2012, Eighth International Conference on Language Resources and Evaluation, 21-27 May 2012, Istanbul, Turkey.

21. De Castro, R., Grossman, J.W. Famous trails to Paul Erd˝os. The Mathematical Intelligencer 21, 51–53 (1999).

22. Oakland University, The Erdos Number Project, Available online at http://www.oakland.edu/enp/, accessed on March 22, 2020.

23. Flajolet, Philippe; Martin, G. Nigel (1985).Probabilistic counting algorithms for data base applications. Journal of Computer and System Sciences. 31 (2): 182–209.

24. Bakhshandeh, Reza & Samadi, Mehdi & Azimifar, Zohreh & Schaeffer, Jonathan. (2011). Degrees of Separation in Social Networks.. Proceedings of the 4th Annual Symposium on Combinatorial Search, SoCS 2011.

25. Pew Research Center, US Politics & Policy, National Politics on Twitter: Small Share of U.S. Adults Produce Majority of Tweets, Available online at https://www.pewresearch.org/, accessed on March 22, 2020.

26. Bethany A. Conway, Kate Kenski, Di Wang, The Rise of Twitter in the Political Campaign: Searching for Intermedia Agenda-Setting Effects in the Presidential Primary, Journal of Computer-Mediated Communication, Volume 20, Issue 4, 1 July 2015, Pages 363–380.

27. Wang Z., Wu Z., Wang R., Ren Y. (2015) Twitter Sarcasm Detection Exploiting a Context-Based Model. In: Wang J. et al. (eds) Web Information Systems Engineering – WISE 2015. WISE 2015. Lecture Notes in Computer Science, vol 9418.

Project report draft1

ORIGINALITY REPORT

6%	6%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Felix Ming Fai Wong, Chee Wei Tan, Soumya Sen, Mung Chiang. "Quantifying Political Leaning from Tweets, Retweets, and Retweeters", IEEE Transactions on Knowledge and Data Engineering, 2016 Publication	1%
2	Submitted to Imperial College of Science, Technology and Medicine Student Paper	1%
3	qfrd.pure.elsevier.com Internet Source	1%
4	dl6.globalstf.org Internet Source	1%

5	www.scribd.com Internet Source	1%
6	epdf.tips Internet Source	1%
7	www.iaeme.com Internet Source	1%
8	machinelearningmastery.com Internet Source	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		