Twitter Sentimental Analysis for Predicting Election Result using LSTM Neural Network

Dipak Gaikar[1], Ganesh Sapare[2], Akanksha Vishwakarma[3], Apurva Parkar[4]

[1] *Asst. Professor, Dept. of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra*
[2] *B.E. student, Dept. of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra*
[3] *B.E. student, Dept. of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra*
[4] *B.E. student, Dept. of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra*

---***---

Abstract - *In recent years, social media has emerged as a powerful widespread technology and caused a huge impact on the public debate and communication in the society. More recently, micro-blogging services (e.g., twitter) and social network sites are presumed to have the potential for increasing political participation. Among the most popular social media sites, Twitter serves as an ideal platform for users to share not only information in general but also political opinions publicly through their networks, political institutions have also begun to use this media for the purpose of entering into direct dialogs with citizens and encouraging more political discussions. So we build a model that can analyze these data and extract sentiment that can help us determine the outcome of the election. The process consists methods such as extraction of tweets from twitter using API, data cleaning to get exact data, training the LSTM (Long Short Term Memory) classifier using labelled dataset and testing it to perform sentimental analysis for classification and then representation of result. Further, a comparison is made among the candidates over the type of sentiment by table and bar graph.*

Key Words: machine learning, twitter, social media, prediction, recurrent neural networks, sentiment analysis, embedding, keras

1. INTRODUCTION

Over the last decade with the arrival of social media, the efforts to determine people's point of view over a particular event or a topic have garnered a wide research interest in natural language processing (NLP) and thus introduced "sentiment analysis." [1] Many social networking websites and micro blogging websites in today's world has become the biggest web destinations for people to communicate with each other, to express their perspective about products or movies, share their daily life experience and present their opinion about real time and upcoming events, such as sports or movies, etc.

Analysis of public's sentiment requires a huge data. For achieving a large, diverse dataset of current public opinion or sentiments, Twitter could be used as a valuable resource that lets the users to send and read small text messages called "Tweets" [1]. Basically twitter allows users to post brief and quick real-time updates regarding various activities like sharing, forwarding and replying messages quickly which allows the quick spread of news or information. The wide use of hash tags also makes it easy to search for tweets dealing with a specific subject, thus making it quite a convenient way to gather data.

Using sentiment analysis for predicting an election's result is practically challenging to train a classification model to perform sentiment analysis on tweet streams for a dynamic event such as an election [7]. Implementing this model have certain key challenges such as changes in the topics of conversation and the people about whom social media posts express opinions. [5] For doing this, we first created a LSTM classifier (positive versus negative versus neutral) which is a modified version of RNN (Recurrent Neural Network) for analyzing opinions about different election candidates as expressed in the tweets. We then train our model for each candidate separately. The inspiration for this separation comes from our observation that the same tweet on an issue can be positive for one candidate while negative for another. In fact, a tweet's sentiment is candidate- dependent. To train the model we used over a 15,000 labelled tweets which are labelled as positive, negative and neutral. In the project. using twitter API, we extracted 40000 real time tweets from Jan 2019 to Mar 2019 relating to names of Indian political parties such as #BJP, #Congress, #TMC, #BSP, #Chowkidar, #BJP4India, # ChowkidarChorHai, etc.

The rest of the paper is organized as follows: Section II presents the objectives for implementing the system. Section III provides a related work on sentiment analysis lately. Section IV explains the proposed model. Section V describes the methodology to determine the sentiments associated with different candidates. Section VI is for visualization and results and finally, section VII is based around conclusions.

2. OBJECTIVES

- To predict the popularity of a candidate of a political party and therefore extrapolate their chances of winning the election by utilizing sentimental analysis of social media data.

- Identify the class to which the tweet belongs based on the sentiment of the tweet.
- Perform an efficient pre-processing unit to make data suitable for analysis.
- Obtain the labelled dataset to train the classifier model.
- Create a LSTM model to perform classification process by training the model with training data to fit and test to model to evaluate the test data.
- Evaluate the results obtained using bar graphs and pie charts and thus comparing it with the result of actual election.

3. RELATED WORK

In [1] they performed data set creation by first collecting data using twitter streaming API and stored them in CSV file, then pre-processing is done to remove special characters and URLs, followed by data labelling which is done manually using hash-tag labelling and then using VADER tool which is lexicon and rule based sentiment analysis tool. Here [2] they obtained the live twitter feeds using twitter streaming API tool and processed the data to remove URLs, #tags and special characters. Negation handling is applied to differentiate the meaning of words and sentiment classification is done using various approaches such as sentiWordNet, Naïve Bayes, HMM and ensemble approach. In [3] they fetched raw tweets in Hindi language using twitter archiver. Pre-processing of tweets is done and algorithms are applied to calculate the polarity of tweets, then sentimental analysis & prediction using NB, SVM and dictionary based. Here [4] they propose a unified CNN-RNN framework for multi-label image classification. The CNN-RNN framework learns a joint image- label embedding to characterize the semantic label dependency as well as the image-label relevance. For which they conclude the experimental results on public benchmark datasets demonstrate of system achieves better performance than the state-of-the-art multi-label classification models. In [5] they use the multi-task learning framework to jointly learn across multiple related tasks. The recurrent neural network has three different mechanisms of sharing information to model text with task-specific and shared layers. The entire network is trained jointly on all these tasks. Experiments on four benchmark text classification tasks show that the proposed models can improve the performance of a task with the help of other related tasks. In [6] they Proposed pattern based approach for sentiment quantification in twitter. They defined two metrics to measure the correctness of sentiment detection and proved that sentiment quantification can be more meaningful task than the regular multi-class classification.

4. PROPOSED MODEL

Proposed architecture for sentiment classification is shown in Figure 1. The system deals with the tweets extraction and sentiment classification. It consists of following modules.

1. Data collection

2. Preprocessing

3. Train the classifier

4. Sentiment Classification

5. Data visualization

4.1 Data collection

Accessing tweets from Twitter is the primary requirement for building a dataset to get processed and extract information. [8] Twitter allows its users to retrieve real time streaming tweets by using twitter API. We propose to use the python library Tweepy which has the API to extract the tweets through authenticating connection with Twitter server. While collecting tweets we filter out retweets. The tweets are then stored in csv file for further processing.

4.2 Data Preprocessing

The data extracted from twitter contains lot of special characters and unnecessary data which we not require [1]. If data is not processed beforehand, it could affect the accuracy as well as performance of the network down the lane. So it is very important to process this data before training. We need to get rid of all the links, URLs and @ tags. Pre-processing also includes removal of stop words from the text to make analysis easier.

4.3 Train the classifier

To train the classifier model we will be using a labelled dataset in which every single tweet is labelled as positive or negative based on sentiment [5]. The dataset is input to the LSTM model in which the text input is embedded and operations are performed to fit the model. One of the most common problem in training deep neural network is over-fitting, because the model performs exceptionally well on training data but poorly on test data. This happens as our classifier algorithm tries to fit every data point in the input even if they represent some randomly sampled noise. Once the model is trained we can determine its accuracy by testing it to classify the sentiments of test data.

4.4 Sentiment Classification

Sentiment Analysis is the most common text classification technique that analyses an incoming message and tells whether the underlying sentiment is positive, negative or neutral. After the model is trained we use it to classify the dataset which is extracted from twitter and stored in a csv file. Sentiment values are assigned to words that describe the positive, negative and neutral attitude of the speaker.

4.5 Data visualization

The final step of this process is to take in the classified tweets and generate pie chart, group bar graph and table to visualize the results. The most frequent words in the dataset can be used to generate word cloud.

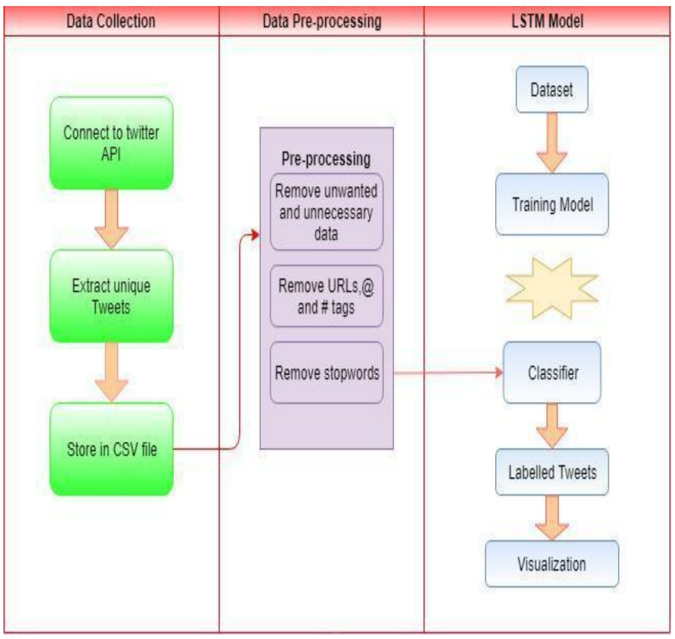

Fig -1: Proposed model

5. METHODOLOGY

We will use Recurrent Neural Networks, and in particular LSTMs, to perform sentimental analysis. The model will use a labelled dataset for training and will classify the test data which is collected from twitter. The test data contains the tweets which is pre-processed and then converted into vectors. These vectors is given as input to classifier.

5.1 Dataset Description

The dataset consists of 15 thousand tweets which are classified as positive, negative and neutral tweets. The positive tweets are labelled as 1, negative tweets are labelled as -1 and neutral tweets as 0. Dataset are structured in a way that each line contains only one tweet. Moreover, all tweets in the provided dataset have already been pre-processed so that all words (tokens) are separated by a single whitespace and the smileys (labels) have been removed. All the links and unwanted data is also removed. Thus, all the tweets are stored in csv file with UTF-8 encoding.

5.2 Word Embedding

Twitter tweets contain a lot of relevant data to generate insights, which can also be used to generate political data. These tweets cannot be processed as it is, because it is difficult for machine learning algorithms to process string inputs. So we need to transform them into numerical or vector format. Word embedding are in fact a class of techniques where individual words are represented as real-valued vectors in a vector space. Each word is mapped to a particular vector and the vector values are trained in a way that resembles a neural network. keras offers an embedding layer that can be used for processing text data. This layer is initially assigned with random weights and will learn an embedding for all of the words in the training data set. Also this format can be used to find semantic relationships. Embedding layer is a flexible layer that can be either used to learn a word embedding that can be saved and used in another model later or we can include it in a deep learning model where learning takes place with the model.

Three important arguments need to be specified for using keras Embedding which are input dimensions, which gives the size of vocabulary in our text data, output dimensions which give the size of vector space in where our words will be embedded, and input length, that is length of input sequences.
For e.g.:

eval = Embedding (150, 30, input length=45)

5.3 Neural Network

Neural network models have been demonstrated to be capable of achieving significant performance in text classification. [5] LSTM is a simple recurrent neural network. RNN analyzes the sentences word by word and stores the semantics of all the previous sentences in a fixed-sized hidden layer. The advantage of RNN is the ability to better store the information. LSTM's are improvement over RNN that can learn and remember long-term dependencies.

5.3.1 Recurrent Neural Network (RNN)

Recurrent Neural Networks understands each word based on the understanding of previous words. They are networks with loops in them, allowing information to persist. In Recurrent Neural Network the output from previous step is given as an input to the current step. [6] In traditional neural networks, all the inputs and outputs are independent of each other, but sometimes it is required to predict the next word of a sentence, for that previous words are required and hence there is a need to remember the previous words. To overcome the drawback, RNN is used which has solved this issue with the help of a Hidden Layer. The important feature of RNN is it has hidden state, which remembers some information about a sequence.

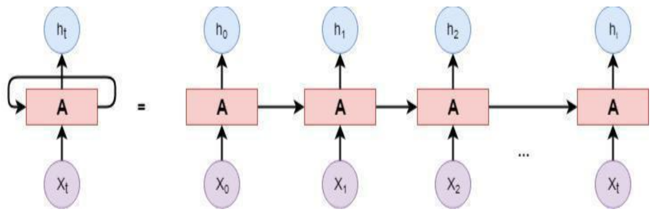

Fig -2: Recurrent Neural Network (RNN)

In figure 2, a chunk of neural network, A, looks at some input x_t and outputs a value h_t. The information is to be passed along the loop from one step of the network to the next [8]. A recurrent neural network can be viewed as multiple copies of the same network, each passing a message to a successor. This chain-like nature reveals that recurrent neural networks are related to sequences and lists. They form natural architecture of neural network to use for such data. RNN has got an incredible success in solving a variety of problems: speech recognition, language modelling, translation, image captioning, etc.

A usual RNN has a short-term memory, so it can't remember past data for a long period of time [8]. For large datasets where we have long texts we can't use RNN. This problem can be solved with the help of LSTM.

5.3.2 Long short term memory (LSTM)

Long Short-Term Memory networks are an extension for recurrent neural networks, which extends their memory. It is very necessary to learn from important experiences that have very long time gaps in between. They are capable of learning long-term dependencies. LSTM's work extremely well on a large variety of problems, and therefore are now widely used. They are specially designed to avoid the long-term dependency problem. To remember information for long periods of time is practically their default behavior.

LSTMs have same chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way [4]. The figure 3 below shows the internal structure of each cell in LSTM.

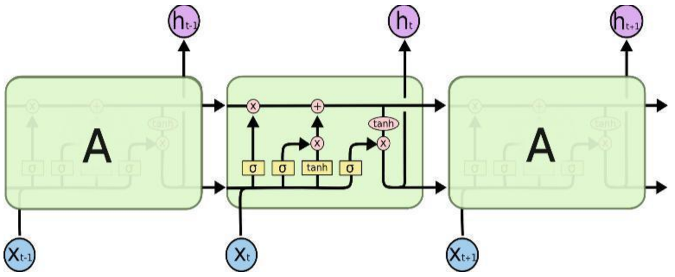

Fig -3: Long Short Term Memory (LSTM)

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram [4]. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means "let nothing through," while a value of one means "let everything through!"

LSTM's enable RNN's to remember their inputs over a long period of time [8]. This is because LSTM's contain their information in a memory that is much like the memory of a computer because the LSTM can read, write and delete information from its memory.

An LSTM has three of these gates, to protect and control the cell state [4]. The first step in LSTM is to decide what information is going to be thrown away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It is represented as σ. It then looks at h_{t-1} (output of previous layer) and x_t (input), and outputs a number between 0 and 1 for each number in the cell state C_{t-1} as shown in eq (1). 1 represents "completely keep this" while a 0 represents "completely get rid of this."

$$C_t = tanh\,(w_c \cdot [h_{t-1}, x_t] + b_C) \qquad (1)$$

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, C_t that could be added to the state. In the next step, we'll combine these two to create an update to the state. Following are the equations:

$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f) \qquad (2)$$

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \qquad (3)$$

It now updates the old cell state, C_{t-1}, into the new cell state C_t. the previous steps already decided what to do. It then multiplies the old state by f_t, forgetting the things it decided to forget earlier. Then we add i_t*C_t as shown in eq (2) and (3). This is the new candidate values, scaled by how much we decided to update each state value.

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

In the case of the language model, this is where we'd actually drop the information about the old subject's information and add the new information using O_t and h_t eq (4) and (5) as we decided in the previous steps. The equations are:

$$O_t = \sigma(w_o.[h_t - 1, x_t] + b_o) \quad (4)$$
$$h_t = O_t * \tanh(C_t) \quad (5)$$

6. RESULTS

Once the classification of tweets is done it is stored in a csv file labelled as positive, negative and neutral. These data can be visualized using bar graphs, pie charts, tables, word cloud, etc. There are various methods that are available in python and its libraries. The number of tweets which are extracted for a particular party may not be the same as for another party. Also the number of positive, negative and neutral tweets may be different for all the candidates, so we cannot directly conclude the winner based on the number of positive tweets, since the data set count may be biased towards a particular candidate.

Consider a scenario where 10,000 tweets for Congress are mined out of which only 3,000 are positive and 7,000 tweets are mined for BJP out of which 4,000 are positive then direct comparison of positive tweets would yield incorrect results since the percentage of positive tweets for BJP is much higher. We have to use the ratio of number of positive to the total count, for which we have considered total count of minimum 1000 tweets. The table 1 shows the positive to total count ratio.

Table -1: PvT ratios of political parties

Party	Positive	Negative	Neutral	Total	Pos/Total ratio
BJP	4216	2781	1083	8080	0.52
INC	3233	3492	1357	8082	0.40
BSP	2461	3792	1564	7817	0.31
TMC	2692	3582	1497	7771	0.34

The total positive, negative and neutral tweets are compared and plotted using bar graph as in figure 4. It shows the total number sentiments for each type for different parties [7]. It will help us to decide the overall score of sentiments for a particular party.

Tweets for the various parties such as BJP, Congress, BSP and TMC are considered. Here it shows that for BJP out of 8080 tweets 4216 are positive whereas for Congress 3233 tweets are positive out of 8082. Similarly BSP and TMC have 2461 and 2692 positive tweets out of 7817 and 7771 respectively. Here BJP has maximum positive to total count ratio.

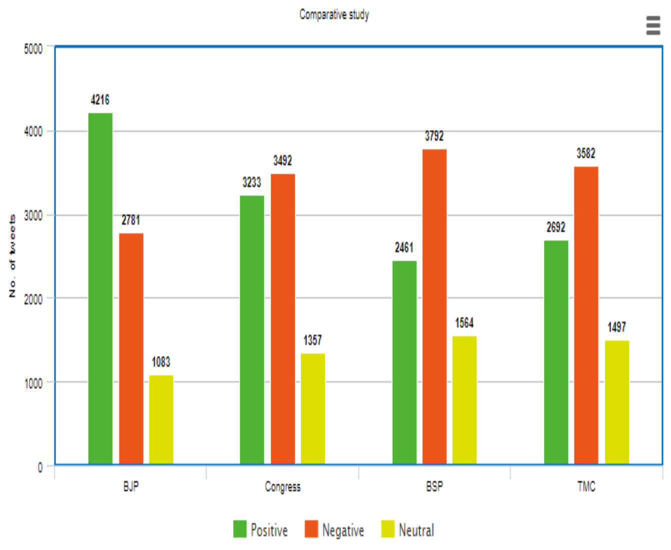

Fig -4: Tweets comparison using bar graphs

Similarly, we can also plot our data using pie chart which will give us the complete idea of how much percentage of tweets are positive and how much are negative for the corresponding party. The pie charts in figure 5 and 6 below shows the percentage of positive, negative and neutral tweets for congress and BJP.

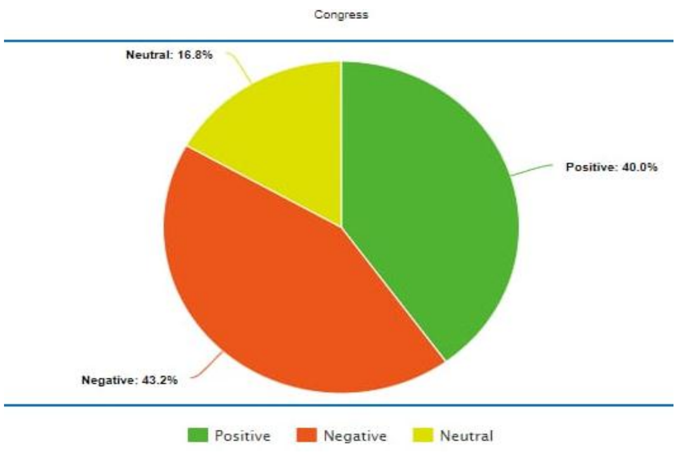

Fig -5: Congress tweets

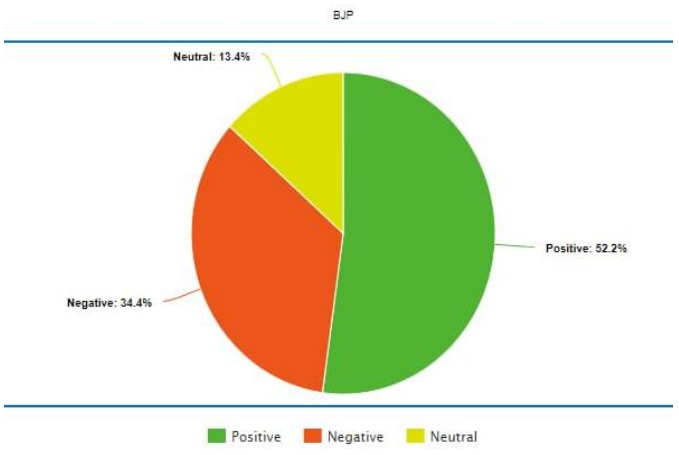

Fig -6: BJP tweets

This pie chart tells us the tweets extracted for BJP has more positive tweets percentage than other parties.

The figure 7 shows below is the word cloud which signifies the frequency of text or the importance of words.

bjp
congress
will
bjp manifesto
manifesto
sankalp
bjpjumlamanifesto
bjpsankalppatr2019
bjpmanifesto
promise
candidate
nation
parties
article
say
amp
modi
india
live
govt
vote
bharat
release
year
sushma swaraj
cm
pvt
secretary
secretaries
crore
residences
raids
sankalppatra

Fig -7: Word cloud

6.1 Comparison with news channel survey

As another factor we have compared our results with the online survey conducted by ABP-C voters and India Today Survey for lok sabha elections 2019 .The following figure 8 shows the opinion poll

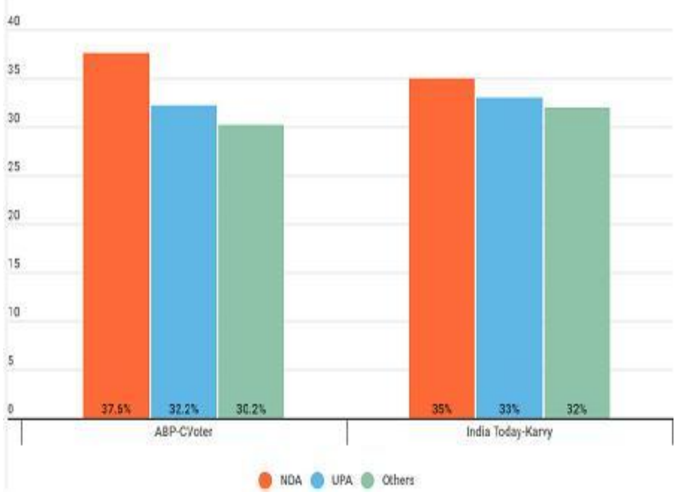

Fig -8: ABP-C voters and India Today online survey

7. CONCLUSION AND FUTURE SCOPE

In this paper, we performed LSTM based sentiment analyser which classifies the tweets based on its sentiment value. The tweets considered are for Indian elections 2019. Tweets corresponding to a particular party is extracted using twitter API which are then pre-processed to remove unwanted and irrelevant data. The LSTM model is built by training the classifier using labelled dataset. Thus classification is done on the extracted tweets to label them as per sentiments, which enables us to present the comparison between the top parties for general elections 2019. Later the obtained results are represented using bar graphs and pie charts. Then we compared political sentiment towards different parties by plotting graphs using results of sentiment analysis on extracted twitter data. In the future, a multi lingual based sentiment classification can be built to acquire tweets in different languages and perform analysis. Also there could be many other prospective areas to conduct this research in, including the data from other big social media sites like Facebook to increase the size of the data set.

REFERENCES

[1] Ramteke, Jyoti, Samarth Shah, Darshan Godhia, and Aadil Shaikh. "Election result prediction using Twitter sentiment analysis." In 2016 international conference on inventive computation technologies (ICICT), vol. 1, pp. 1-5. IEEE, 2016.

[2] Jose, Rincy, and Varghese S. Chooralil. "Prediction of election result by enhanced sentiment analysis on twitter data using classifier ensemble Approach." In 2016 international conference on data mining and advanced computing (SAPIENCE), pp. 64-67. IEEE, 2016.

[3] Sharma, Parul, and Teng-Sheng Moh. "Prediction of Indian election using sentiment analysis on Hindi Twitter." In 2016 IEEE International Conference on Big Data (Big Data), pp. 1966-1971. IEEE, 2016.

[4] Wang, Jiang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. "Cnn-rnn: A unified framework for multi-label image classification." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2285-2294. 2016.

[5] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang, "Recurrent Neural Network for Text Classification with Multi-Task Learning." Twenty-Fifth International Joint Conference on Artificial Intelligence, pp.2873-2879, 2016.

[6] Bouazizi, Mondher, and Tomoaki Ohtsuki. "Sentiment analysis in twitter: From classification to quantification of sentiments within tweets." In 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1-6. IEEE, 2016.

[7] Wang, Hao, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. "A system for real-time twitter sentiment analysis of 2012 us presidential election cycle." In Proceedings of the ACL 2012 System Demonstrations, pp. 115-120. Association for Computational Linguistics, 2012.

[8] http://colah.github.io/posts/2015-08Understanding-LSTM

[9] http://towardsdatascience.com/another-twitter-posentiment-analysis