Twitter Sentiment Analysis for Indian Election 2019

Abhishek Bhagwat
College of Engineering, Northeastern University
bhagwat.a@husky.neu.edu

Abstract

The 2019 Indian general election is scheduled to be held in 7 phases from 11 April 2019 to 19 May 2019 millions of voters will cast their votes during the elections, the largest democratic exercise in human history. Twitter is one of the most popular social media platforms that is used to express views regarding eclectic topics like sports, movies, politics, etc. Social Media has become one of the key platforms for politicians and political parties to promote themselves in the lead up to the elections. The aim of this project is to do sentiment analysis of the Twitter network in the months leading up to the election and build a model that will be able to interpret political subject matter. The analysis will be based on the politicians and the parties involved in the elections. This analysis can be used to acquire knowledge of the sentiments involved in the elections and predict the outcome of the upcoming elections. The models built are a Bidirectional Recurrent Neural Network and Glove word embedding model. The models are optimized using varying neural architectures and hyperparameter tuning. Kernel Regularization methods are used to address overfitting to get the best performance.

Index Terms— *Bidirectional Recurrent Neural Network, GloVe, Word embeddings, Sentiment Analysis*

Introduction

Extracting the public opinion from social media text provides a challenging and rich context to explore computational models of natural language processing, modeling a complex phenomenon such as an election is neither a simple nor easy task. Twitter is one of the most popular Social Network and it is widely used by politicians and political parties to promote themselves. Many people expressed their feelings for each candidate on different social networks including Twitter, a popular micro-blogging site. Each micro-blog is referred to as a "Tweet" and can be no more than one-hundred and forty characters long. Many tweets also include a label for other Twitter users they are referencing (e.g. @username), and a "hashtag" that usually indicates the topic of the tweet (e.g. #election2019). Tweets convey the sentiments of the users and it is important. Twitter users can express their sentiment of a candidate using a hashtag. Since the election is largely dominated by two parties, a hashtag that is used to convey positive sentiment for one candidate might express negative sentiment for the other candidate and vice versa. The two main parties that we will consider are the BJP and the Indian National Congress. For our work, we extract the tweets using hashtags. The hashtags will identify the tweets to a party or leader. Tweets for both the parties will be collected and labeled as positive or negative depending on the sentiment score. The models that we will be implementing are Bidirectional RNN and Word embedding model (Glove). The trained model will then be used to predict the sentiment label for both the parties. Depending on the outcome we can infer that the party with the greatest number of positive tweets is likely to do well in the elections.

Methodology

In this paper discusses two methods to do binary classification of the tweets. Bidirectional LSTM and a Model with pre-trained word embedding Glove. This work consists of the following steps:

1. Creating the tweets dataset
2. Data Cleaning and Exploratory Analysis
3. Model Building
4. Observations
5. Conclusion

Data Source

The data for sentiment analysis if obtained from Twitter. For this work, we collected tweets from January 2019 through April 2019 that contained the keywords like "BJP", "Congress", "Congress", "Modi". The Twitter API called Tweepy is used to download tweets that contain the hashtags that we want. The tweets for both the parties are stored in different CSV files. The Textblob library is used to obtain the sentiment score for the tweets. The sentiment scores are in the range of -1 to 1. The tweets with a score greater than 0 were labeled positive and the remaining were labeled negative. At this stage, the data was imbalanced as most of the tweets were negative for both the parties. If we create a model using this data, the model would have a high bias towards the majority category. To address this problem, we use a resampling technique. Additional data for both the

parties was extracted from Twitter. Only the positive tweets were selected and added to the data of both the parties. Thus, data for both the parties was balanced. The data from both the parties was merged to create a single dataset. This dataset is used as a final dataset for our analysis.

Data Cleaning and Exploratory

The preprocessing of the text data is an essential step as it makes the raw text ready for mining, i.e., it becomes easier to extract information from the text and apply machine learning algorithms to it. The objective of this step is to clean noise those are less relevant to find the sentiment of tweets such as punctuation, emoticons, special characters, numbers, Twitter handles and terms which don't carry much weightage in context to the text. The clean data is then used to build visualizations. Visualizations help in understanding the significance of the data by placing it in a visual context. Bar plots were generated for both the parties to understand the distribution of the positive and negative sentiments amongst the parties. It was observed that for both the parties the count of tweets conveying negative sentiment was more than the count of positive tweets. The other type of visualization that was used was a Word Cloud. A word cloud is a popular visualization of words typically associated with Internet keywords and text data. They are most commonly used to highlight popular or trending terms based on frequency of use and prominence. A word cloud is a beautiful, informative image that communicates much in a single glance. The word cloud for BJP had keywords like "Arvind Kejriwal", "Delhi", "Defeat BJP" indicating an overall negative sentiment. The word cloud for Congress had keywords like "Amethi", "Rahul Gandhi" indicating a neutral to positive sentiment for the party.

Models

1. Glove Word Embedding model

The first model uses a pre-trained word embedding GloVe. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. GloVe can be used to find relations between words like synonyms, company - product relations, zip codes, and cities etc. It is also used by the spaCy model to build semantic word embeddings/feature vectors while computing the top list words that match with distance measures such as Cosine Similarity and Euclidean distance approach. It was also used as the word representation framework for the online and offline systems designed to detect psychological distress in patient interviews. The Euclidean distance (or cosine similarity) between two-word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Fig 1 shows how the words that are related are present in a nearby cluster.

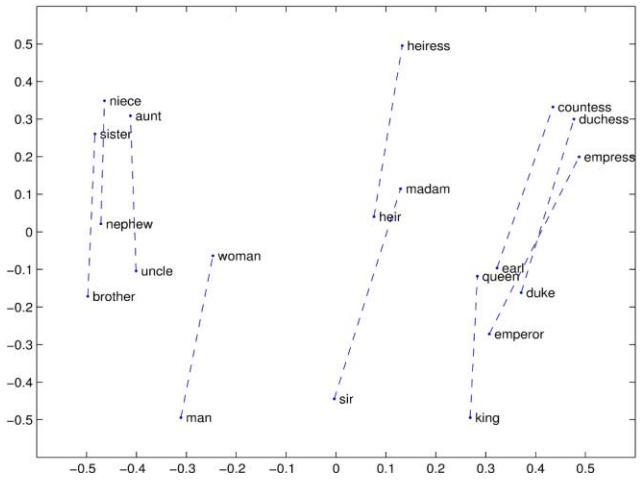

Fig 1: Glove Word-embeddings

The GloVe algorithm consists of following steps:

1. Collect word co-occurrence statistics in a form of word co-occurrence matrix XX. Each element XijXij of such matrix represents how often word *i* appears in context of word *j*. Usually we scan our corpus in the following manner: for each term we look for context terms within some area defined by a *window size* before the term and a *window size* after the term. Also, we give less weight for more distant words, usually using this formula:

   decay=1/offsetdecay=1/offset

2. Define soft constraints for each word pair:

   wTiwj+bi+bj=log(Xij)wiTwj+bi+bj=log(Xij)

   Here wiwi - vector for the main word, wjwj - vector for the context word, bibi, bjbj are scalar biases for the main and context words.

3. Define a cost function

   J=∑i=1V∑j=1Vf(Xij)(wTiwj+bi+bj−logXij)2J=∑i=1V∑j=1Vf(Xij)(wiTwj+bi+bj−log⁡Xij)2

   Here ff is a weighting function which help us to prevent learning only from extremely common

word pairs. The GloVe authors choose the following function:

f(Xij)={(Xijxmax)α1if Xij<XMAXotherwise

Word Embedding is done for the experiment with the pre-trained word vector Glove. Glove version used: 100-dimensional. GloVe embeddings of 400k words computed on a 2014 dump of English Wikipedia Training is performed on an aggregated global word-word co-occurrence matrix, giving us a vector space with meaningful substructures. Words are mapped to GloVe embeddings. Co-occurrence count matrix is created for the above mappings. Embedding layer is created with the help of embedding matrix created and set as the input layer for model. The model consists of 5 dense layers. The activation function used were RELU and Sigmoid for the output layer. Hyper-parameters are fine-tuned to obtain the best model.

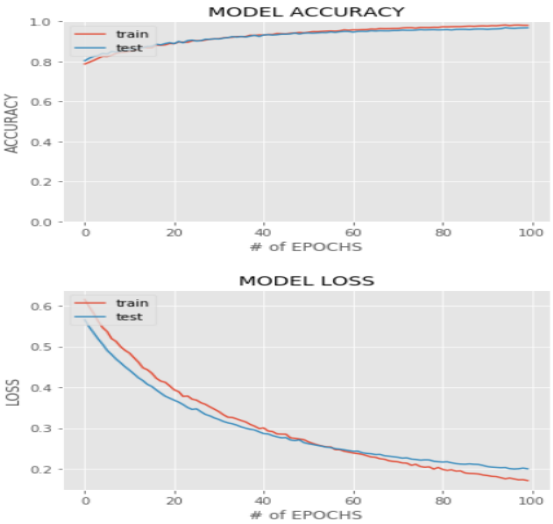

Fig 2: Model Accuracy and Loss (GloVe)

The model was initially overfitting. Therefore, L2 regularization and drop-out were used to reduce overfitting. The model was trained for 100 epochs. The best model gave an accuracy of 97.01 percent and loss of 0.20 for the test data.

2. Bidirectional RNN model

Bidirectional Recurrent Neural Networks (BRNN) connect two hidden layers of opposite directions to the same output. With this form of generative deep learning, the output layer can get information from past (backwards) and future (forward) states simultaneously. Invented in 1997 by Schuster and Paliwal, BRNNs were introduced to increase the amount of input information available to the network. For example, multilayer perceptron (MLPs) and time delay neural network (TDNNs) have limitations on the input data flexibility, as they require their input data to be fixed. Standard recurrent neural network (RNNs) also have restrictions as the future input information cannot be reached from the current state. On the contrary, BRNNs do not require their input data to be fixed. Moreover, their future input information is reachable from the current state.

BRNN are especially useful when the context of the input is needed. For example, in handwriting recognition, the performance can be enhanced by knowledge of the letters located before and after the current letter.

The principle of BRNN is to split the neurons of a regular RNN into two directions, one for positive time direction (forward states), and another for negative time direction (backward states). Those two states' output are not connected to inputs of the opposite direction states. The general structure of RNN and BRNN can be depicted in the right diagram. By using two-time directions, input information from the past and future of the current time frame can be used unlike standard RNN which requires the delays for including future information.

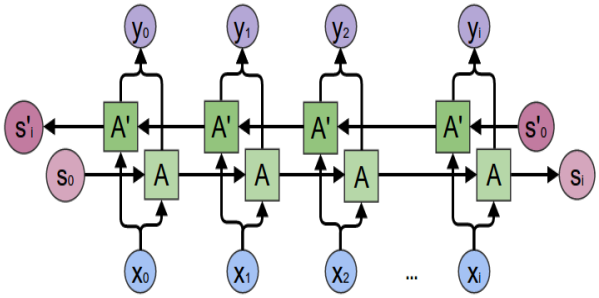

Fig 3: Bidirectional RNN

BRNNs can be trained using similar algorithms to RNNs, because the two directional neurons do not have any interactions. However, when back-propagation is applied, additional processes are needed because updating input and output layers cannot be done at once. General procedures for training are as follows: For forward pass, forward states and backward states are passed first, then output neurons are passed. For backward pass, output neurons are passed first, then forward states and backward states are passed next. After forward and backward passes are done, the weights are updated.

The model architecture created consists of 3 layers. An embedding layer. A single bidirectional LSTM layer and a dense layer. The model was trained for 100 epochs. The model accuracy was 98% and the loss was 0.20 for test data.

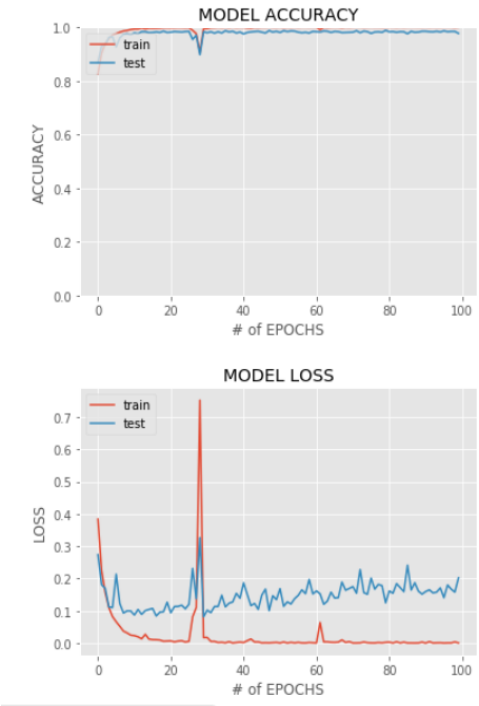

Fig 4: Model Accuracy and Loss (Bidirectional RNN)

Observations

The models were compared based on the Test loss and Test Accuracy. The Bidirectional RNN performed slightly better than the GloVe model. The RNN despite its simple architecture performed better than the Glove model. The following table shows the performance of both the models.

Model	Test Loss	Test Accuracy
GloVe Word Embedding Model	0.2	97.01
Bidirectional RNN	0.2	98

Fig 5: Model Performance Summary

Conclusion

The best model that was selected was the Bidirectional RNN as it gave the most accuracy for the experiment. A test dataset was created by following the same procedure that was used to create the training data. The test data consists of two files. The first file contains tweets related to the party BJP and the second file contains tweets related to the Indian National Congress. The number of tweets in both the files are the same. The tweets are preprocessed in the same way the tweets for the training data are processed. The model is used to make predictions on the tweets contained in both the files. The positive tweets for each file are calculated. The party with the greatest number of positive tweets can be considers the most likely to win the elections. According to the model's prediction the party BJP had more positive tweets. The model predicted that BJP has 660 positive tweets out of 2000 tweets in comparison to Congresses 416 positive tweets out of 2000. For the analysis it can be inferred that the BJP would not have a clean majority like they had in the 2014 elections. However, they are still the most likely to win the 2019 elections but with a lesser margin.

Future Work

Predicting an election is not an easy task. For this project only, a small sample of twitter data was considered for the analysis. It is difficult to give an estimate based on the limited amount of information we had access to. For future work, we can start by increasing the size of our dataset. In addition to Twitter, data can also be obtained from websites like Facebook, News websites. Apart from these we can try different models like Bidirectional RNN with attention mechanism. We can implement BERT which is currently the state of the art for solving various Natural Language Processing problems.

Acknowledgement

I would like to thank Professor Nik Brown for his guidance and valuable inputs throughout this project.

References

[1] Sepp Hochreiter and Jurgen Schmidhuber, "Long short- ¨ term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," Signal Processing, IEEE Transactions on, v ol. 45, no. 11, pp. 2673–2681, 1997.

[3] Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Global Vectors for Word Representation

[4] Apoorv Agarwal Boyi Xie Ilia Vovsha Owen Rambow Re becca Passonneau Sentiment Analysis of Twitter Data

[5] Alex Graves and Jurgen Schmidhuber, "Framewise ¨ phoneme classification with bidirectional LSTM and other neural network architectures," Neural Networks, vol. 18, no. 5, pp. 602–610, 2005