Introduction to Machine Learning (CS 412)
Project Final Report
Stock Market Prediction Using Natural Language Processing

Arnav Dahal
(UIN: 650314561)
Hassan Pasha
(UIN: 650378806)

Manoj Kumar Gunasekaran
(UIN: 651297651)
Niharika Balachandra
(UIN: 668084671)

Sathvik Raju
(UIN: 650241857)
Yi-Huan Chen
(UIN: 661060755)

Abstract

We used Machine learning techniques to evaluate past data pertaining to the stock market and world affairs of the corresponding time period, in order to make predictions in stock trends. We built a model that will be able to buy and sell stock based on profitable prediction, without any human interactions. The model uses Natural Language Processing (NLP) to make smart “decisions” based on current affairs, article, etc. With NLP and the basic rule of probability, our goal is to increases the accuracy of the stock predictions.

Introduction

Natural Language Processing is a technique used by a computer to understand and manipulate natural languages. By natural languages, we mean all human derived languages. Natural language processing or NLP for short is used to analyze text and let machines derive meaning from the input. This human-computer interaction allows us to come up with many different applications to bring man and machine as one. For example, on google, if we use google translation, that is NLP and so is speech recognition. In our project, we make use of some established NLP techniques to evaluate past data pertaining to the stock market and world affairs of the corresponding time period, in order to make predictions in stock trends.

In order to proceed with this objective, we needed to understand what Sentimental Analysis is. Sentimental Analysis is an analytical method that the computer uses to understand a natural language and deduce if the message is positive, neutral or negative. In our case, Sentimental analysis refers to the deduction of the news headlines if they increase the stock or reduce it. By doing so, we end up with the ‘emotional’ status of the data which is what sentimental analysis gives its user.

Data Collection and Wrangling

We have used the Combined_News_DJIA.csv dataset (courtesy Aaron7sun, Kaggle.com) .The Combined_News_DJIA.csv dataset spans from 2008 to 2016. We extended the dataset to include additional data. This additional data is collected from the Guardian’s Restful News API for the 2000-2008 period. We take the 25 most popular headlines for each given day in this period. In addition, we also pull the Dow Jones Index (DJI) of Yahoo Finance’s website for the 2000- 2008 period to compare the influence of the data. There are two channels of data provided in this dataset:

1. News data that has historical news headlines from Reddit World News Channel. Only the top 25 headlines are considered for a single date.

2. Stock data for Dow Jones Industrial Average (DJIA) over the corresponding time range is used to label the data. The stock data is compiled from Yahoo Finance.

Note: The headlines for each data acts as the explanatory data that causes the stock price to either rise (labeled 1) or to fall (labelled 0). We have the top 25 headlines for one single date arranged as one row of the extracted data set.

Since our goal is to predict the tendency of the stock of a specific company, the data that lead the stock's price of the next day to decline or stay the same are labelled "0", while the data that lead the price of the next day to rise are labelled "1". We compare the data between the two pulled data set and then merge them together to get the more accurate prediction.

With the raw data, we cannot proceed much further until we manipulate the data to suit our analysis and convert the data into vectors that are much easier to work on. For this, we use Word2Vec. This is a group of related models used to create word embeddings. Word embeddings are sets of language modeling and feature learning techniques in NLP where words or phrases from the vocabulary are mapped to vectors of real numbers. These vectors make up the training and test sets. English is really easy – see all those spaces? That makes it really easy to tokenize – in other words, to determine what's a word. So we just use a simple set of rules for English tokenization.

This raw data is manipulated using python. We first split the data into lists of words but these lists are flooded with HTML tags and punctuations. We cleaned up the data and removed all HTML tags and punctuations. Then we moved forward with removing stop words. Stop words are words that do not contribute to the meaning or sentiment of the data such as 'the', 'is', 'and', etc. We have also converted all the letters to lowercase to make a more even data set to play with.

Workflow

With the training data set, we got to convert them into numeric representation for machine learning. For this, we use 'Bag of Words' model. The Bag of Words model learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. These values are the feature vectors that are derived from the model.

The thing is we cannot stop at just using the Bag of Words model as this generates feature vectors that only give importance to the number of occurrences of words rather than where they occur and with what words they accompany. To get past this, we use the n-gram model or the skip gram model. Now, with this model, we can store the order of words in the way they occur in the data. The number of words stored in a single order depends on the value on n. Say n=2, calls for a bigram model which stores sets of 2 words in order.

We use Natural Language Processing (NLP) to interpret and construct the data set. The data are composed of a row of sentences. In order to reduce the complexity, the stop words such as "a", "and", "the" have been cleaned. In addition, we came across the N-gram model which helps predict the next set of words in an n- worded text or speech. The Google's Word2Vec deep learning method are also provided to focus on the sentiment of the words by means of the bag of words concept. This method is suitable for us because it doesn't need labels in order to create meaningful representations. If there are enough training data, it would produces word vector with intriguing characteristics so that we

could analyze the relationship between the words that have similar meanings.

With all this done, we have our manipulated data vectors ready to be trained and tested. We have split the extracted dataset in the ratio of 4:1. 80% of the extracted data will be the training data and 20% of the extracted data will be the test data. We work with 4 models in this project to train our data.

- Naive Bayes
- Random Forest
- Logistic Regression
- Support Vector Machines

Model Selection

This section describes the method we use to construct the model. There are four main model to analyze the data: Naïve Bayes, Random Forest (RF), Support Vector Machine (SVM), and Logistic Regression. We use the scikit-learn toolbox in Python to program.

1. Naïve Bayes:

This model provides a family of probabilistic classifiers that are based on the Bayes theorem with strong independence characteristics within its feature vectors.

$$P(A/B) = P(B/A)P(A)\ /P(B)$$

where *A* and *B* are events and P(B) ≠ 0.

- *P*(*A*) and *P*(*B*) are the probabilities of observing *A* and *B* without regard to each other.
- *P*(*A* | *B*), a conditional probability, is the probability of observing event *A* given that *B* is true.

P(*B* | *A*) is the probability of observing event *B* given that *A* is true.

Naïve Bayes is common to use in bag of words. Since we have 25 features (25 top headlines) in each data set (a given day), the step is as below:

Training: Estimate $P(Y|X_{1:K})$ for all $X_{1:K}$

Testing: Predict $Y = argmax_Y\ P(Y|X_{1:K})$

where k=25, and

$$P(Y|X_{1:K}) = \frac{P(X_{1:K}|Y)P(Y)}{P(X_{1:K})} = \frac{P(X_{1:K}|Y)P(Y)}{\sum_{Y'} P(X_{1:K}|Y')P(Y')}$$

the relationship between each word can be classified.

2. Random Forest:

Random Forest consists of a collection or ensemble of simple tree predictors, each capable of producing a response when presented with a set of predictor values.The RF algorithm grows n decision trees as the weak classifier, each provides different kind of classification, and then merge all the trees into a forest. Unlike decision tree or K-NN method, RF don't have to take the cross validation. The step is as follow:

(1) Grow many trees. Each trees has m input data, which m is a constant chosen randomly from K element in our data set and $m << K$ during the trees grow.

(2) At each node test the value of features $X_{1:K}$, divide the data into two leafs (Y=1 if the price increases and Y=0 if the price decreases).

(3) Each tree grows to the large extent possible. No pruning for the trees.

(4) Estimate the error for the prediction $h_i(x)$ of each tree as the tree grows entirely, that

is, the out-of-bag error (oob) sample of the i^{th} tree.

(5) Merge all the trees into a forest and estimate the oob of the whole forest by the oob of each tree in the forest. The final prediction is

$$H = \frac{1}{K}\sum_{i=1}^{K}\alpha_i h_i(x)$$

therefore, according to the vote from each tree, we can get the classification prediction of our data set. Furthermore, the accuracy will be $1 - oob$

3. Support Vector Machine:

A SVM is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples.

SVM is a margin-based classification method. It discriminates the data by a separating hyperplane and it's margin, i.e.

$$maximize\ M$$

subject to,

$$y_i(\beta_0 + \sum_{j=1}^{K}\beta_j x_{ij}) \le M \quad \forall i = 1, 2, ..., n$$

where,

$$\sum_{j=1}^{K}\beta_j^2 = 1$$

since we have only 2 classes and 25 features in each data, we can get low testing error if the training error is also small. Therefore it is suitable for classifying our data set. In addition, the kernel method is also taken into our SVM model to sparse the data which cannot be distinguished in original space.

4. Logistic Regression:

It is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

Logistic Regression is widely used in classification and data analysis. It is appropriate when the response take only one of two values, that is, there are only two classes for the dependent variables. Therefore it is also appropriate to our data set. In HW3 we also use this method to identify whether the mail is spam or not. Here we can use as the similar way. The probability can be calculated as below:

$$P(Y = 0|x, w) = \frac{1}{1+exp(w\cdot x)} = \frac{1}{1+exp(w_0+\sum_{j=1}^{K} w_j x_j)}$$

$$P(Y = 1|x, w) = \frac{exp(w\cdot x)}{1+exp(w\cdot x)} = \frac{exp(w_0+\sum_{j=1}^{K} w_j x_j)}{1+exp(w_0+\sum_{j=1}^{K} w_j x_j)}$$

$$P(Y = 1|x, w) = 1 - P(Y = 0|x, w)$$

and the weight w can be modified for each iteration as:

$$w_j \leftarrow w_j + \eta\frac{dl(w)}{dw_j}$$

$$l(w) = ln\prod_{i=1}^{K}[P(y_i|x_i, w)] - \tfrac{1}{2}\lambda||w||^b$$

$$l(w) = \sum_{i=1}^{K}[y_i(w\cdot x_i) - ln(1 + exp(w\cdot x_i))] - \tfrac{1}{2}\lambda||w||^b$$

where η is the learning rate and $\frac{1}{2}\lambda||w||^b$ term is used to avoid overfitting (decrease the fluctuation of each iteration of the regression). Upon this we are able to connect the relation between the specific words that appear frequently, and identify what tendency the headline (our data set) has, so that can predict the price of the stock in the next day.

After applying the models, we were left with testing the output data and evaluating the comparisons of the results which we will be covering in the results section.

Environment

We used Jupyter Notebook which is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. We perform data cleaning and transformation, statistical modeling and machine learning in this environment.

Code Snippets

Logistic Regression Model:

1-Gram Model Prediction Accuracy

Predicted	**0**	**1**
Actual		
0	149	37
1	30	162

In [11]:

```
print(basictrain.shape)
print (classification_report(test["Label"], predictions))
print (accuracy_score(test["Label"], predictions))
```

```
(3975, 46002)
             precision    recall  f1-score   support

          0       0.83      0.80      0.82       186
          1       0.81      0.84      0.83       192

avg / total       0.82      0.82      0.82       378
0.822751322751
```

Bi-Gram Model Prediction Accuracy

Predicted	**0**	**1**
Actual		
0	159	27
1	27	165

In [17]:

```
print(basictrain2.shape)
print(classification_report(test["Label"], predictions2))
print (accuracy_score(test["Label"], predictions2))
```

```
(3975, 584289)
             precision    recall  f1-score   support

          0       0.85      0.85      0.85       186
          1       0.86      0.86      0.86       192

avg / total       0.86      0.86      0.86       378
0.857142857143
```

Tri-Gram Model Prediction Accuracy

Predicted	**0**	**1**
Actual		
0	142	44
1	12	180

In [18]:

```
print(basictrain2.shape)
print(classification_report(test["Label"], predictions3))
print (accuracy_score(test["Label"], predictions3))
```

```
(3975, 969254)
             precision    recall  f1-score   support

          0       0.92      0.76      0.84       186
          1       0.80      0.94      0.87       192

avg / total       0.86      0.85      0.85       378
0.851851851852
```

Random Forests Model:

1-Gram Model Prediction Accuracy

Predicted	0	1
Actual		
0	144	42
1	16	176

In [12]:

```
print(basictrain.shape)
print (classification_report(test["Label"], predictions))
print (accuracy_score(test["Label"], predictions))
```

(3975, 46002)

	precision	recall	f1-score	support
0	0.90	0.77	0.83	186
1	0.81	0.92	0.86	192
avg / total	0.85	0.85	0.85	378

0.846560846561

Bi-Gram Model Prediction Accuracy

Predicted	0	1
Actual		
0	143	43
1	10	182

In [16]:

```
print(basictrain2.shape)
print(classification_report(test["Label"], predictions2))
print (accuracy_score(test["Label"], predictions2))
```

(3975, 584289)

	precision	recall	f1-score	support
0	0.93	0.77	0.84	186
1	0.81	0.95	0.87	192
avg / total	0.87	0.86	0.86	378

0.859788359788

Tri-Gram Model Prediction Accuracy

Predicted	0	1
Actual		
0	130	56
1	0	192

In [20]:

```
print(basictrain3.shape)
print(classification_report(test["Label"], predictions3))
print (accuracy_score(test["Label"], predictions3))
```

(3975, 969254)

	precision	recall	f1-score	support
0	1.00	0.70	0.82	186
1	0.77	1.00	0.87	192
avg / total	0.89	0.85	0.85	378

0.851851851852

Linear SVM Model:

1-Gram model Prediction Accuracy

Predicted	0	1
Actual		
0	151	35
1	32	160

In [12]:

```
print(basictrain.shape)
print (classification_report(test["Label"], predictions))
print (accuracy_score(test["Label"], predictions))
```

(3975, 46002)

	precision	recall	f1-score	support
0	0.83	0.81	0.82	186
1	0.82	0.83	0.83	192
avg / total	0.82	0.82	0.82	378

0.822751322751

Bi-Gram Model Prediction Accuracy

Predicted	**0**	**1**
Actual		
0	160	26
1	32	160

In [16]:

```
print(basictrain2.shape)
print(classification_report(test["Label"], predictions2))
print (accuracy_score(test["Label"], predictions2))
```

(3975, 584289)

	precision	recall	f1-score	support
0	0.83	0.86	0.85	186
1	0.86	0.83	0.85	192
avg / total	0.85	0.85	0.85	378

0.846560846561

Tri-Gram Model Prediction Accuracy

Predicted	**0**	**1**
Actual		
0	145	41
1	17	175

In [20]:

```
print(basictrain3.shape)
print(classification_report(test["Label"], predictions3))
print (accuracy_score(test["Label"], predictions3))
```

(3975, 969254)

	precision	recall	f1-score	support
0	0.90	0.78	0.83	186
1	0.81	0.91	0.86	192
avg / total	0.85	0.85	0.85	378

0.846560846561

SVM (Gaussian Kernel) Model:

1-Gram Model Prediction Accuracy

Predicted	**0**	**1**
Actual		
0	130	56
1	0	192

In [11]:

```
print(basictrain.shape)
print (classification_report(test["Label"], predictions))
print (accuracy_score(test["Label"], predictions))
```

(3975, 46002)

	precision	recall	f1-score	support
0	1.00	0.70	0.82	186
1	0.77	1.00	0.87	192
avg / total	0.89	0.85	0.85	378

0.851851851852

Bi-Gram Model Prediction Accuracy

Predicted	**0**	**1**
Actual		
0	130	56
1	0	192

In [15]:

```
print(basictrain2.shape)
print(classification_report(test["Label"], predictions2))
print (accuracy_score(test["Label"], predictions2))
```

(3975, 584289)

	precision	recall	f1-score	support
0	1.00	0.70	0.82	186
1	0.77	1.00	0.87	192
avg / total	0.89	0.85	0.85	378

0.851851851852

Tri-Gram Model Prediction Accuracy

Predicted	**0**	**1**
Actual		
0	120	66
1	0	192

In [19]:

```
print(basictrain3.shape)
print(classification_report(test["Label"], predictions3))
print (accuracy_score(test["Label"], predictions3))
```

(3975, 969254)

	precision	recall	f1-score	support
0	1.00	0.65	0.78	186
1	0.74	1.00	0.85	192
avg / total	0.87	0.83	0.82	378

0.825396825397

Naïve Bayes Model:

1-Gram Model Prediction Accuracy

Predicted	**0**	**1**
Actual		
0	155	31
1	37	155

In [11]:

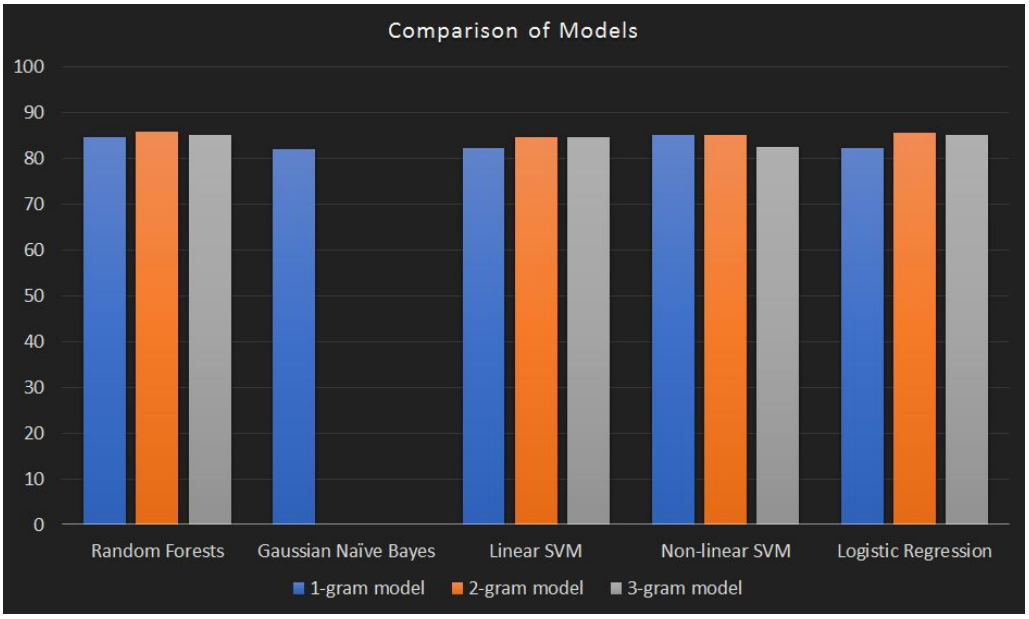
```
print(basictrain.shape)
print (classification_report(test["Label"], predictions))
print (accuracy_score(test["Label"], predictions))
```

(3975, 46002)

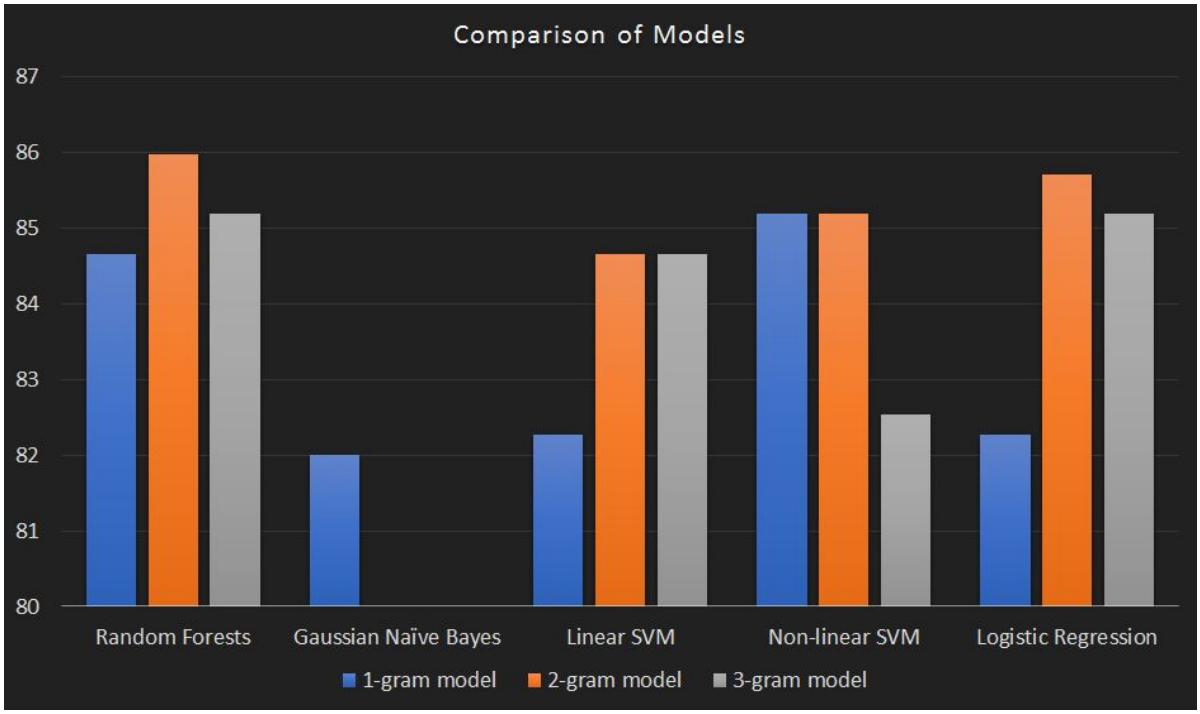
	precision	recall	f1-score	support
0	0.81	0.83	0.82	186
1	0.83	0.81	0.82	192
avg / total	0.82	0.82	0.82	378

0.820105820106

Comparison of Model Performance using Accuracy of Prediction



Random forests had highest accuracy on the a Bi-gram model as shown in the chart. The prediction accuracy was 85.97%. Using Natural Language Processing techniques, we were able to accurately predict the stock market trends 85% of the time.



Conclusion

Social Media can sometimes be deceiving when delivering the right frame of speech. Here we have used twitter feed and news articles around the web that has influenced the stock market of a company. Through this project, it helped us understand the basics of Natural Language Processing. Even though you can't bet your money on the stock from this project, this work can be treated as solid understanding of the basics of Natural Language Processing. Using the same model for different text data is also feasible. It was interesting to know about how to go from text data to vectors of numbers and applying Machine learning techniques that can help to influence the stock market of a company. It helped us a gain wider sense of the power of NLP in various applications. From reading about machine learning models in class to implement them with real data and observe the performance of a model, tuning the parameters, performing exploratory data analysis set a great learning curve for future projects.

We also went from using an available data set to scraping our own data which made this project a little more interesting and challenging. To get more insights on which model to use and how to construct them, we learnt by reading through research papers and usage of scikit learn to build our models. As any project that does not take a straight path towards completion we hit certain roadblocks while implementing this project.

Road block 1:

As any machine learning task is concerned the data in consideration was restrictive and had few data cleaning to be done. Firstly the data consisted of a character "'b" appended to text in multiple ways and was a little challenging to remove it across the entire dataset.

Road block 2:

It was also challenging to find more data. The dataset available to use was from 2008 to 2016. We had to scrape it from another source (Yahoo News) completely and put in the format that we wanted to work for our Machine learning model.It was a challenging task to scrape it and wrangle it to the data set that we wanted to (25 top headlines and labels associated with them)

Road block 3:

While we put the dataset for our SVM model, there were quite a few errors it kept throwing and one of them being NaN values and the number of cores we used to train our model. We used all the 4 cores to run our algorithm in parallel for faster execution.

Future work

This project leaves room for future work and ways to accomplish them:

1. The number of features used in the data set can be expanded. Right now we have gathered the top 25 News headlines, It is important to have more features that help the model learn better.
2. We are looking at the stock of one company. We can expand it to work for multiple companies at once and we can also include real time- time series analysis.
3. Perform multi class classification for various parameters of stock trading.

Appendix: Code

All the source files used in this project can be found at:
https://github.com/niharikabalachandra/Stock-Market-Prediction-Using-Natural-Language-Processing

References

1. F. Xu and V. Keelj, "Collective Sentiment Mining of Microblogs in 24-Hour Stock Price Movement Prediction," *2014 IEEE 16th Conference on Business Informatics*, Geneva, 2014, pp. 60-67. doi: 10.1109/CBI.2014.37

2. L. Bing, K. C. C. Chan and C. Ou, "Public Sentiment Analysis in Twitter Data for Prediction of a Company's Stock Price Movements," *2014 IEEE 11th International Conference on e-Business Engineering*, Guangzhou, 2014, pp. 232-239. doi: 10.1109/ICEBE.2014.47

3. D. Rao, F. Deng, Z. Jiang and G. Zhao, "Qualitative Stock Market Predicting with Common Knowledge Based Nature Language Processing: A Unified View and Procedure," *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, Hangzhou, 2015, pp. 381-384. doi: 10.1109/IHMSC.2015.114

4. Z. Jiang, P. Chen and X. Pan, "Announcement Based Stock Prediction," *2016 International Symposium on Computer, Consumer and Control (IS3C)*, Xi'an, 2016, pp. 428-431. doi: 10.1109/IS3C.2016.114

5. W. Bouachir, A. Torabi, G. A. Bilodeau and P. Blais, "A bag of words approach for semantic segmentation of monitored scenes," *2016 International Symposium on Signal, Image, Video and Communications (ISIVC)*, Tunis, 2016, pp. 88-93. doi: 10.1109/ISIVC.2016.7893967

6. D. Sehgal and A. K. Agarwal, "Sentiment analysis of big data applications using Twitter Data with the help of HADOOP framework," *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*, Moradabad, 2016, pp. 251-255. doi: 10.1109/SYSMART.2016.7894530

7. R. Zhao; K. Mao, "Fuzzy Bag-of-Words Model for Document Representation," in *IEEE Transactions on Fuzzy Systems* , vol.PP, no.99, pp.1-1 doi: 10.1109/TFUZZ.2017.2690222

8. V. U. Thompson, C. Panchev and M. Oakes, "Performance evaluation of similarity measures on similar and dissimilar text retrieval," *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, Lisbon, 2015, pp. 577-584

9. C. Sreejith, M. Indu and P. C. R. Raj, "N-gram based algorithm for distinguishing between Hindi and Sanskrit texts," *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, Tiruchengode, 2013, pp. 1-4. doi: 10.1109/ICCCNT.2013.6726777

10. M. Kaya, G. Fidan and I. H. Toroslu, "Sentiment Analysis of Turkish Political News," *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, Macau, 2012, pp. 174-180. doi: 10.1109/WI-IAT.2012.115