Opinion mining on newspaper headlines using SVM and NLP

Chaudhary Jashubhai Rameshbhai, Joy Paulose
Dept. of Computer Science, Christ University, India

Article Info

Article history:
Received Jan 4, 2018
Revised Jul 23, 2018
Accepted Dec 15, 2018

Keywords:
Newspaper
Sentiment analysis
Opinion mining
NLTK
Stanford coreNLPm
SVM
SGDClassifier
Tf-idf
CountVectorizer

ABSTRACT

Opinion Mining also known as Sentiment Analysis, is a technique or procedure which uses Natural Language processing (NLP) to classify the outcome from text. There are various NLP tools available which are used for processing text data. Multiple research have been done in opinion mining for online blogs, Twitter, Facebook etc. This paper proposes a new opinion mining technique using Support Vector Machine (SVM) and NLP tools on newspaper headlines. Relative words are generated using Stanford CoreNLP, which is passed to SVM using count vectorizer. On comparing three models using confusion matrix, results indicate that Tf-idf and Linear SVM provides better accuracy for smaller dataset. While for larger dataset, SGD and linear SVM model outperform other models.

Corresponding Author:
Chaudhary Jashubhai Rameshbhai,
Department of Computer Science,
Christ University Hosur Road, Bangalore, Karnataka, India 560029.
Phone: +91 7405497405
Email: chaudhary.rameshbhai@cs.christuniversity.in

1. INTRODUCTION

Opinion Mining or Sentiment Analysis is a task to analyze opinions or sentiments from textual data. It is useful in analyzing NLP applications. With the development of applications like network public opinion analysis, the demand on sentiment analysis and opinion mining is growing. In today's world, utmost people are using internet and social media platforms, to share their views. These analysis are available in various forms on the internet, like reviews about product, Facebook post as feedback, Twitter feeds, blogs etc. At present, news play a dynamic role in developing a person's visions and opinions related to any product, political party or company. News article published in the newspaper or shared on the web can sometimes create negative or positive impacts on the society on large scale. As per Dor (1), most of the people judge the news contents directly by scanning only the news headlines relatively than going through the complete story. Hence, minor headlines can also impact on large scale. In this paper, Opinion mining is performed based on just the headlines without going through whole articles. The proposed method begins with data collection and preprocessing. Data are collected from different news source using Python newspaper package. Review for news headlines are assigned either +1 or -1 manually based on sentiments. In order to perform SVM to build a classification model, news headline data are fetched and processed in CoreNLP (2). CoreNLP returns set of relative words which are imported into count vectorizer (3) to generate matrix.

This paper is organized as follows: Section 2 provides overview of related works on opinion mining. Section 3 contains elaborate explanation of the proposed method. Section 4 discusses the experimental results of three models. Section 5 concludes and provides future scope of this proposed method.

2. RELATED WORK

Agarwal et al. (4) proposed a method containing two algorithms. First algorithm is used for data pre-processing while other to detect polarity value of a word. Natural Language Processing Tool (NLTK) and SentiWordNet are employed to build the proposed method. NLTK is a Python library for word tokenization , POS (Part of Speech) - Tagging , Lemmatization and Stemming. SentiWordNet (5) lexicon, a method extended from WordNet is specifically designed for Sentiment Analysis. Each word is assigned with positive or negative numerical scores. Negative words are denoted with negative score, while positive words are assigned with positive score. Output of NLTK is fed to SentiWordNet in order to assign numerical score for each word and compute the final sentiment score, which is sum of all the numerical scores. If the final value is greater or equal to 0, the headline is classified as positive or negative headline.

Yang et al. (6) proposed a hybrid model for analyzing sentiments of textual data for a single domain. It is implemented domain wise due to increase in complexity upon segregation. Single classification model is used to segregate the responses as positive, negative and neutral. The model is a combination of multiple single classification method which provides more efficient classification.

Rana and Singh (7) compared two machine learning algorithms Linear SVM and Naive Bayes for Sentiment Analysis. Review on movies are used as dataset containing 1000 samples. Proter Stemmer is employed to preprocess dataset. While Rapid Miner tool is used to generate model, Linear SVM and Naive Bayes are used as classifier. Precision, recall and accuracy of both the models are calculated. From the result, it is observed that the Linear SVM gives better result than Naive Bayes.

Bakshi et al. (8) proposed an approach to classify positive, negative and neutral tweets of Twitter. It is focused on a single company Samsung Electronics Ltd. Data is fed and processed to clean the data. The algorithm is applied to analyze the sentiment of tweets and segregate into different categories.

Aroju et al. (9) proposed a method to perform Opinion Mining on three different newspapers related to similar news. SVM and Naive Bayes are used for opinion mining. Around 105 news headlines are collected from three different sources (35 headlines from The Hindu, The Times of India and Deccan Chronicle Newspaper). Data is processed using POS - tagger and stemming. Weka tool is used to implement the method. For experimental results, F-score, Precision and Recall are calculated. From the result it is evident that The Hindu newspaper contains more positive news than The Times of India and Deccan Chronicle.

Hasan et al. (10) proposed an algorithm which uses Naive Bayes to perform opinion mining. Data are reviewed in English from e-commerce website Amazon. The reviews are also translated to Bangla using Google Translator. Opinion mining is calculated for reviews in both Bangla and English. The Bangla dataset translated from English contains noise. The reviews in Bangla as training data are fed into Naive Bayes to build classifier excluding noisy words.

Akkineni et al. (11) proposed a method of classifying opinion, opinions are classified based on subject of opinions and objective of having such opinion, this method helps to classify whether sentence is a fact or opinion. Approach adopted for classifying in this paper range are heuristic approach where results within a realistic time-frame. They are likely to produce the results themselves but are mostly used with the optimized algorithms, Discourse Structure which focuses on the given text that just communicates a message, and linking it to how that message constructs a social reality or view of the world,key word analysis which classifies text by affect categories based on the presence of unambiguous affect words such as happy, sad, afraid, and bored, Concept analysis which concentrates on semantic analysis of text through the use of web ontologies or semantic networks.The conceptual and affective information associated with natural language opinions are aggregated.

Arora et al. (12) proposed Cross_BOMEST, a cross domain sentimental classification. Existing method BOMEST, it retrieves +ve words from a content, followed by determination of +ve word with assistance of Ms Word Introp. In order to escalate the polarity it replaces all it's synonym. Moreover, it helps in blending two different domains and detect self-sufficient words. The proposed method is test and implemented on Amazon dataset. Total of 1500 product reviews are randomly selected for both +ve and -ve polarity. Out of which 1000 are used for training and remaining to test the classification model. As a result, when applying on cross domain precision, accuracy of 92% is achieved. For single domain, precision and recall of BOMEST is improved by 16% and 7%. Thus, Cross_BOMEST improves the precision and accuracy by 5% when compared to other existing techniques.

Susanti et al. (13) employs Multinomial Naïve Bayes Tree which is combination of Multinomial Naïve Bayes and Decision Tree. The technique is used in data mining for classification of raw data. Multinomial Naïve Bayes method is used specifically to address frequency calculation in the text of the sentence or document. Documents used in this study are comments of Twitter users on the GSM telecommunications provider in Indonesia.[] This paper used the method to categorize customers sentiment opinion towards telecommunication providers in Indonesia. Sentiment analysis only consists of positive, negative and neutral class. Decision Tree is generated with this method and roots in the feature ”aktif”, where probability of feature ”aktif” belongs in positive class. In result and analysis, it is indicated that the highest accuracy of classification using Multinomial Naïve Bayes Tree (MNBTree) method is 16.26% when using 145 features. Furthermore, the Multinomial Naïve Bayes (MNB) yields the highest accuracy of 73,15% by using all dataset of 1665 features.

In this type of research selection of appropriate feature is one of the challenges. Many researchers are using decision tree and n-gram approach for feature selection and Supervised Machine learning technique for model building. The tedious job in this type of research is data preprocessing. Most of the researchers are using NLP tools for data preprocessing (14), (15), (16), (17), (18). In this paper, n-gram and coreNLP is used for feature selection and Linear SVM are used for model building.

3. PROPOSED WORK

Many algorithms are available for finding sentiment from text, but they are performed on large text dataset like movie reviews, product reviews etc. Finding opinions from news headlines is also possible but the accuracy of existing algorithm is not satisfactory. This paper tries to improve the accuracy of existing algorithm (4) using different approach. The proposed method is distributed into three processes.

As shown in Figure 1, Process I is regarding data collection and pre-processing. Process II is the core fragment to build classifier and Process III tests the classification model on test data. These processes are discussed in detail further.

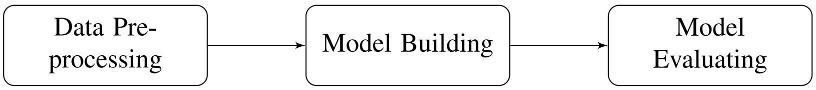


Figure 1. Process diagram

3.1. Data Pre-processing And Model Building

In this process, data pre-processing and model building are implemented. Figure 2 depicts the basic flowchart of the data preprocessing and model building. Stop words are removed from news headlines followed by converting uppercase texts to lowercase. The semi-processed headlines are fed to coreNLP. The output of coreNLP with sentiment scores are set as input for process II. The input is received from process I and converted into unigram and bi-gram representation. Model A is generated from the unigram and bi-gram representation. Model A employs Linear SVM. Data representation is further converted into Tf-idf resulting in Model B and C. Model B and C employ Linear SVM. However, unlike model B, model C uses SGD classifer to train the data.

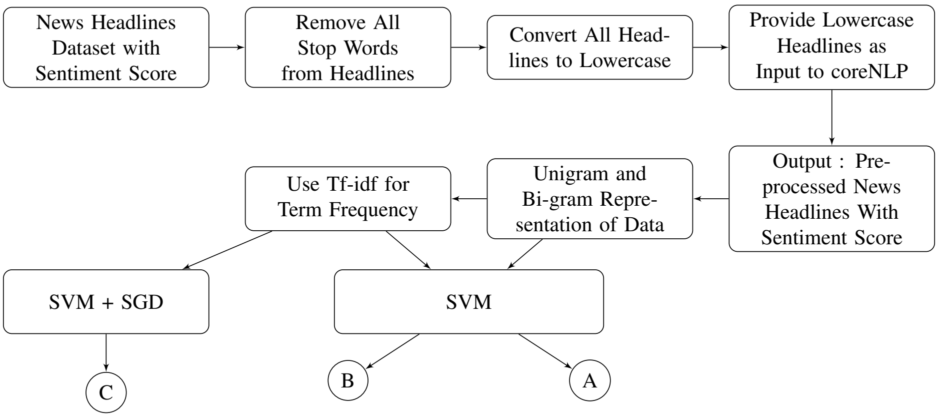
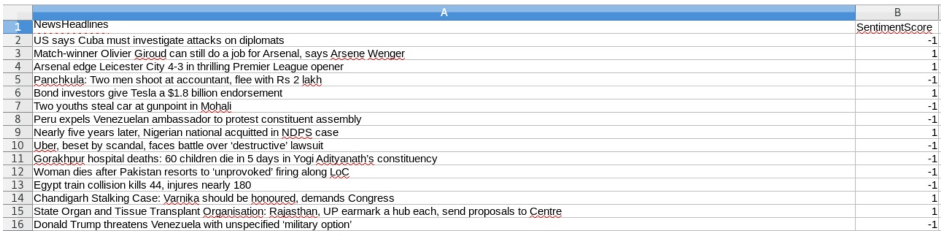

News Headlines Dataset with Sentiment Score
Remove All Stop Words from Headlines
Convert All Headlines to Lowercase
Provide Lowercase Headlines as Input to coreNLP
Output : Pre-processed News Headlines With Sentiment Score
Unigram and Bi-gram Representation of Data
Use Tf-idf for Term Frequency
SVM + SGD
SVM
C
B
A

Figure 2. Flow diagram of data pre-processing and model building

Data are collected from the website http://www.indianexpress.com during August 2017. Figure 3 depicts sample unprocessed data. 1472 news headlines are collected and manually classified as either +1 or -1. For positive news, sentiment score is +1, while for negative news is -1. Subsequently, after allocating the sentiment score, all the stop words from headlines such as: is, the, are etc are eliminated. NLTK STOP WORDS is used to perform the task of removing stop words. The algorithm is case sensitive for which all the headlines are converted to lowercase. The data is fed to Stanford CoreNLP to generate the dependency parser. Standford CoreNLP dependency parser (19) checks the grammatical construction of a sentence, which establishes relation between ”ROOT” word and altering words. To understand how coreNLP works consider news headlines: ”Two killed in car bomb in Iraq Kirkuk”. Figure 4 depicts parsing of sample data using dependency parser. ”ROOT” word of the headline is returned and relation between each word of the headline. From the sample data, killed is returned as ”ROOT” word and relation between all the words. Figure 5 shows parsing of sample headline using dependency parser and converting the output in the form of string array. String array consists of ”ROOT” word and relative words. The process is applied on all the data to generate array of strings with ”ROOT” word and relative words. Figure 6 is statistical representation of frequency of words in data. Figure 7 depicts sample data with sentiment score. The processed data are input for process II.



	A	B
1	NewsHeadlines	SentimentScore
2	US says Cuba must investigate attacks on diplomats	-1
3	Match-winner Olivier Giroud can still do a job for Arsenal, says Arsene Wenger	1
4	Arsenal edge Leicester City 4-3 in thrilling Premier League opener	1
5	Panchkula: Two men shoot at accountant, flee with Rs 2 lakh	-1
6	Bond investors give Tesla a $1.8 billion endorsement	1
7	Two youths steal car at gunpoint in Mohali	-1
8	Peru expels Venezuelan ambassador to protest constituent assembly	-1
9	Nearly five years later, Nigerian national acquitted in NDPS case	1
10	Uber, beset by scandal, faces battle over ‘destructive’ lawsuit	-1
11	Gorakhpur hospital deaths: 60 children die in 5 days in Yogi Adityanath’s constituency	-1
12	Woman dies after Pakistan resorts to ‘unprovoked’ firing along LoC	-1
13	Egypt train collision kills 44, injures nearly 180	-1
14	Chandigarh Stalking Case: Varnika should be honoured, demands Congress	1
15	State Organ and Tissue Transplant Organisation: Rajasthan, UP earmark a hub each, send proposals to Centre	1
16	Donald Trump threatens Venezuela with unspecified ‘military option’	-1

Figure 3. Dataset screenshot

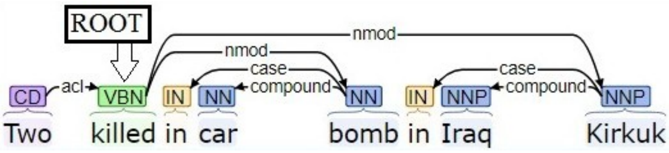
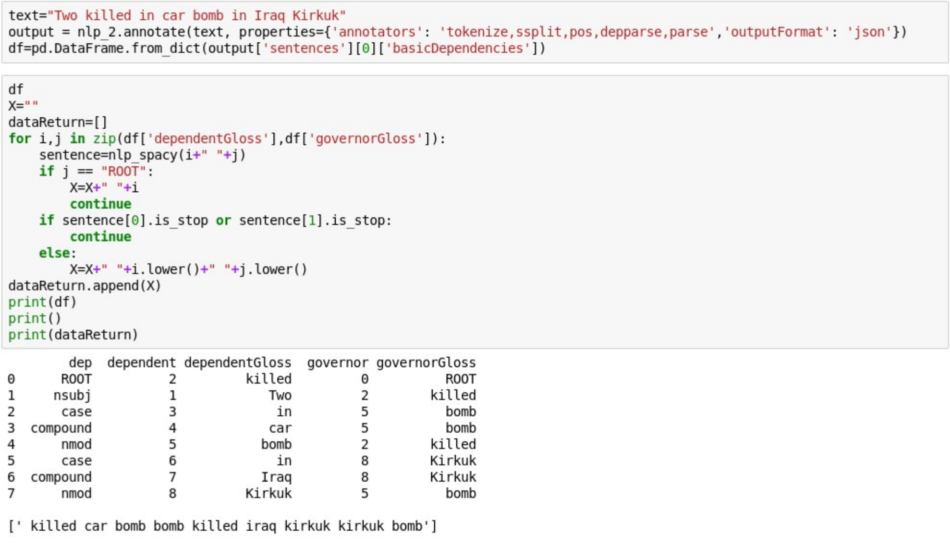


Figure 4. Dependency parser working diagram



```
text="Two killed in car bomb in Iraq Kirkuk"
output = nlp_2.annotate(text, properties={'annotators': 'tokenize,ssplit,pos,depparse,parse','outputFormat': 'json'})
df=pd.DataFrame.from_dict(output['sentences'][0]['basicDependencies'])
```

```
df
X=""
dataReturn=[]
for i,j in zip(df['dependentGloss'],df['governorGloss']):
    sentence=nlp_spacy(i+" "+j)
    if j == "ROOT":
        X=X+" "+i
        continue
    if sentence[0].is_stop or sentence[1].is_stop:
        continue
    else:
        X=X+" "+i.lower()+" "+j.lower()
dataReturn.append(X)
print(df)
print()
print(dataReturn)
```

```
        dep  dependent dependentGloss  governor governorGloss
0      ROOT          2         killed         0          ROOT
1     nsubj          1            Two         2        killed
2      case          3             in         5          bomb
3  compound          4            car         5          bomb
4      nmod          5           bomb         2        killed
5      case          6             in         8        Kirkuk
6  compound          7           Iraq         8        Kirkuk
7      nmod          8         Kirkuk         5          bomb

[' killed car bomb bomb killed iraq kirkuk kirkuk bomb']
```

Figure 5. Generating pandas dataframe using coreNLP

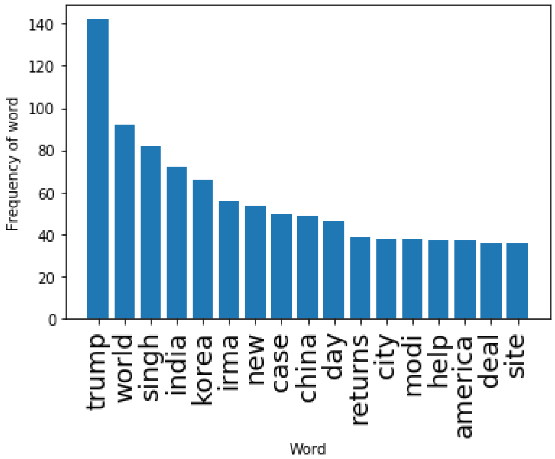
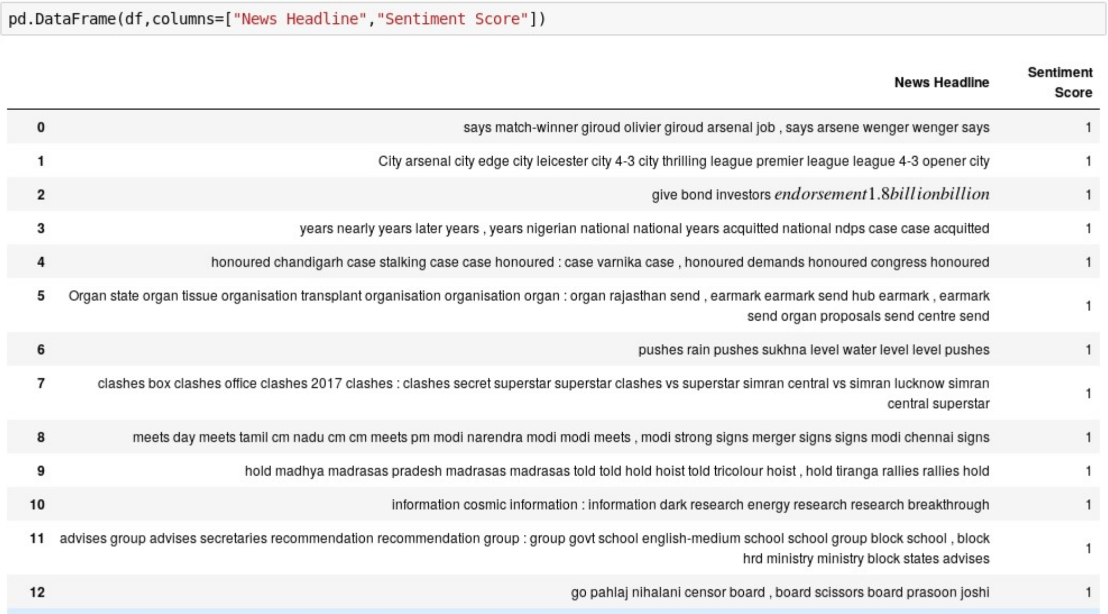


Figure 6. Most frequently occurred words in dataset



```
pd.DataFrame(df,columns=["News Headline","Sentiment Score"])
```

	News Headline	Sentiment Score
0	says match-winner giroud olivier giroud arsenal job , says arsene wenger wenger says	1
1	City arsenal city edge city leicester city 4-3 city thrilling league premier league league 4-3 opener city	1
2	give bond investors $endorsement1.8billionbillion$	1
3	years nearly years later years , years nigerian national national years acquitted national ndps case case acquitted	1
4	honoured chandigarh case stalking case case honoured : case varnika case , honoured demands honoured congress honoured	1
5	Organ state organ tissue organisation transplant organisation organisation organ : organ rajasthan send , earmark earmark send hub earmark , earmark send organ proposals send centre send	1
6	pushes rain pushes sukhna level water level level pushes	1
7	clashes box clashes office clashes 2017 clashes : clashes secret superstar superstar clashes vs superstar simran central vs simran lucknow simran central superstar	1
8	meets day meets tamil cm nadu cm cm meets pm modi narendra modi modi meets , modi strong signs merger signs signs modi chennai signs	1
9	hold madhya madrasas pradesh madrasas madrasas told told hold hoist told tricolour hoist , hold tiranga rallies rallies hold	1
10	information cosmic information : information dark research energy research research breakthrough	1
11	advises group advises secretaries recommendation recommendation group : group govt school english-medium school school group block school , block hrd ministry ministry block states advises	1
12	go pahlaj nihalani censor board , board scissors board prasoon joshi	1

Figure 7. Sample dataset after applying coreNLP

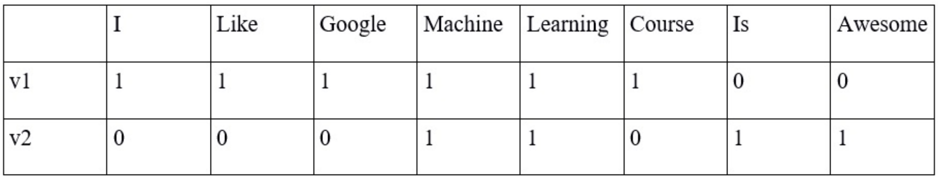
3.2. Unigram and Bi-gram Representation of Data

Before building the model, the raw data is converted from string to numerical values. In machine learning, Text Analysis is a key application area. Most of the algorithms accept numerical data with fixed size rather than text data of varying size. A collective approach uses a document-term vector where individual document is encrypted as a discrete vector that sums occurrences for each word in the vocabulary it contains (3). For example, consider two one-sentence documents:

D1: ”I like Google Machine Learning course”

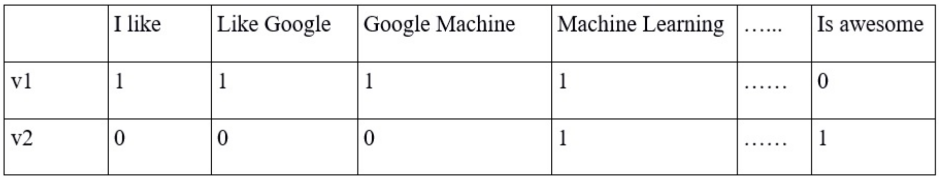
D2: ”Machine Learning is awesome”

The vocabulary V = { I, like, Google, Machine, Learning, course, is, awesome } and two documents can be encoded as v1 and v2. Figure 8 and 9 show the representation of given sentences in unigram and bi-gram model. The bi-gram model refines data representation where occurrences are determined by a sequence of two words rather than individually.



	I	Like	Google	Machine	Learning	Course	Is	Awesome
v1	1	1	1	1	1	1	0	0
v2	0	0	0	1	1	0	1	1

Figure 8. Unigram representation of given vocabulary



	I like	Like Google	Google Machine	Machine Learning	…...	Is awesome
v1	1	1	1	1	……	0
v2	0	0	0	1	……	1

Figure 9. Bi-gram representation of given vocabulary

Figure 10 shows snipped code to generate unigram and bi-gram representation of given data. Data is an array of 4 news headlines. Here, $CountVectorizer()$ is used to convert string data into numeric values. To generate unigram model, pass argument $ngram\_range = (1, 1)$ and $ngram\_range = (1, 2)$ for bi-gram. In figure, there are two matrices generated using pandas library. In matrix, columns represents unique words (31 for unigram and 61 for bi-gram) and rows represents 4 news headlines. If the word exists in particular news headlines the value for that particular feature will be 1 and if the word exists twice then the value for that feature will be 2. Value depends on frequancy of word in headline. This method is used when a model is built in Process II.

```
data = [
'Coal Burying Goa: What the toxic train leaves in its wake',
'Coal Burying Goa: Lives touched by coal',
'Coal burying Goa: Danger ahead, new coal corridor is coming up',
'Goa mining: Supreme Court issues notices to Centre, state government'
]
clf=CountVectorizer(ngram_range=(1,1))
df=clf.fit_transform(data).toarray()
pd.DataFrame(df)
        0       1       2       3       ...
0       0       1       0       0       ...
1       0       1       1       0       ...
2       1       1       0       0       ...
3       0       0       0       1       ...
4 rows * 31 columns
clf=CountVectorizer(ngram_range=(1,2))
df=clf.fit_transform(data).toarray()
pd.DataFrame(df)
        0       1       2       3       ...
0       0       1       0       0       ...
1       0       1       1       0       ...
2       1       1       0       0       ...
3       0       0       0       1       ...
4 rows * 61 columns
```

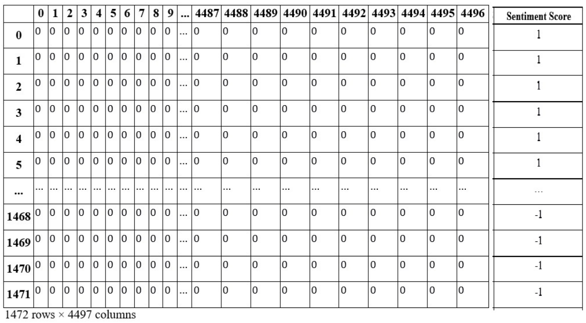
Figure 10. Snipped python code to generate Unigram and Bi-gram from given string data.

3.3. Model A. Linear SVM

In this process, Linear SVM is used to build the model and the data that has been used to build the model is Numeric. The Description for total dataset is shown in Table 1. The Matrices in Figure 11 and 12 have been generated by using unigram and bi-gram, column shows the number of words in headlines and row represents number of headlines. When the value in Matrix is 0, it means that particular word does not exist in the headlines. In unigram, number of feature depends on total unique words in dataset.

Table 1. Dataset Description

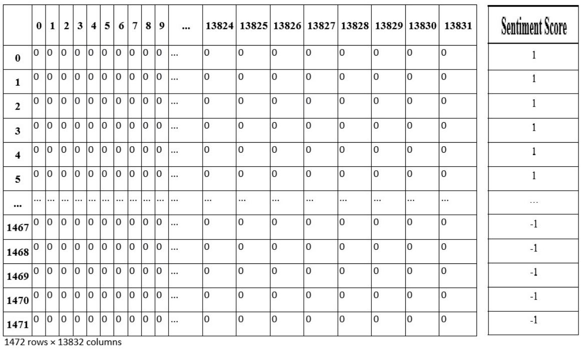
	Total Sample	Total Feature
Unigram	1472	4497
Bi-gram	1472	13832



	0	1	2	3	4	5	6	7	8	9	...	4487	4488	4489	4490	4491	4492	4493	4494	4495	4496	Sentiment Score
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1468	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	-1
1469	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	-1
1470	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	-1
1471	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	-1

1472 rows × 4497 columns

Figure 11. Dataset representation in Unigram model.



	0	1	2	3	4	5	6	7	8	9	...	13824	13825	13826	13827	13828	13829	13830	13831	Sentiment Score
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1467	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	-1
1468	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	-1
1469	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	-1
1470	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	-1
1471	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	-1

1472 rows × 13832 columns

Figure 12. Dataset representation in Bi-gram model.

Table 1 shows total number of features and samples in unigram and bi-gram. Total sample size is total number of news headlines and total feature size is number of unique words in dataset. Here, total sample dataset size is same for both models. But in unigram model total number of feature is 4497 and 13832 for bi-gram model. For building this model, 80% data are considered for training set and 20% are considered for evaluating the model. Here, kernel is linear because we have two class labels, so SVM generates linear hyper plane which will separate words. It separates all negative news headline words and positive news headline words.

3.4. Model B. Tf-idf and Linear SVM

Linear SVM is used in this model building and the dataset is converted into document frequency using Tf-idf. Tf is Term-frequency while Tf-idf (3) is Term-frequency time's inverse document-frequency. It is used to classify the documents. The main aim of Tf-idf is to calculate the importance of a word in any given headline with respect to overall occurrence of that word in the dataset. The importance of a word is high if it is frequent in the headline, but less frequent in overall headline. Tf-idf can calculated as follows (3):

$$tfidf(t, d) = tf(t, d) * idf(t) \tag{1}$$

Where tf(t,d) is term frequency in particular headline, the term occurred number of times in particular headline and is multiplied with idf(t).

$$idf(t) = 1 + log\frac{1+n_d}{1+df(d,t)} \tag{2}$$

Where n_d is the total number of headlines, and df(d,t) is the number of headlines that contain term t. The resulting Tf-idf vectors are then normalized by the Euclidean norm:

$$v_{norm} = \frac{v}{||v||^2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + .. + v_n^2}} \tag{3}$$

```
data_counts = [[3, 0, 1],
               [2, 0, 0],
               [3, 0, 0],
               [4, 0, 0],
               [3, 2, 0],
               [3, 0, 2]]
```

For example, Tf-idf is computed for the first term in the first document in the data_counts array as follows:

To calculate Tf-idf of first term in document:

Total No. of Documents : $n_{d,term1} = 6$

Total No of Documents which contain this term 1 :

$$df(d,t)_{term1} = 6$$

idf is for term 1 :

$$idf(d,t)_{term1} = 1 + log\tfrac{n_d}{df(d,t)} = log\tfrac{6}{6} + 1 = 1$$

$$tfidf_{term1} = tf_{term1} * idf_{term1} = 3 * 1 = 3$$

Similarly for other two terms:

$$tfidf_{term2} = 0 * (\tfrac{6}{1} + 1) = 0$$

$$tfidf_{term3} = 0 * (\tfrac{6}{1} + 1) = 2.0986$$

Represent Tf-idf in vector:

$$tfidf_{raw} = [3, 0, 2.0986]$$

After applying Euclidean norm:

$$\tfrac{[3,0,2.0986]}{\sqrt{(3^2+0^2+2.0986^2)}} = [0.819, 0, 0.573]$$

In idf(t), to avoid zero divisions ”smooth_idf=True” adds ”1” to the numerator and denominator.After modification in equation the first two term value will be same but in term3 value changes to 1.8473:

$$tfidf_{term3} = 1 * log\frac{7}{3} + 1 = 1.8473$$

$$\frac{[3,0,1.8473]}{\sqrt{(3^2+0^2+1.8473^2)}} = [0.8515, 0, 0.5243]$$

```
Similarly by calculating every value in data_counts array the final
output will be:

Tfidf = TfidfTransformer()
X=Tfidf.fit_transform(data_counts)
X.toarray()
output of is:
array([
[ 0.8515,0.0000,0.5243],
[ 1.0000,0.0000,0.0000],
[ 1.0000,0.0000,0.0000],
[ 1.0000,0.0000,0.0000],
[ 0.5542,0.8323,0.0000],
[ 0.6303,0.0000,0.7763]
])
```

The total number of sample data size and feature data size will remain same for building this model. Here, the frequency of each word is changed according to Tf-idf. Unlike the previous model words with less frequency in headlines will have lower values. It means words with lesser frequency will have lesser impact on model. For training, this model is using 80% data and for testing it is using 20% data.

3.5. Model C. Stochastic Gradient Descent (SGD) Classifier

The SGD is used to train the data for Linear SVM. SGD (3) is a discriminative learning of linear classifiers like SVM and Logistic Regression which is simple and very efficient. SGD has been effectively implemented in numeric data machine learning problems majorly involved in text categorization and NLP. Data provided is in sparse, the classifiers in SGD efficiently scales to the problem with more than 10^5 training samples and with more than 10^5 attributes. The major advantage of SGD is that it can handle large dataset. Here, in this research problem small size of dataset has been used but this approach can be extended this research for up to 10^5 features.

4. MODEL EVALUATING

Result of three different models are compared. The confusion matrix has been used as a performance metric. Table 2 describes the structure of confusion matrix. The formula for accuracy of model is shown in eq. 4. Table 3, 4 and 5 are confusion matrices of Model A, B and C and Table 6 shows the accuracy score of three models. This table implies that the bi-gram will give more accurate result than unigram. However, in unigram model, number of feature is less than bi-gram model, due to which time in building the model in unigram is less than bi-gram. Here, the accuracy of Model B is higher than Model A because it is trained with Tf-idf. With the increase in feature size (>20000), Model A and B will not provide feasible solutions. To overcome such issues Model C is introduced in this paper and it is trained using SGD, it supports up to 10^5 features (3) for building a model. Thus Model C can be used when the feature size is high, otherwise Model B works well when the feature size is less.

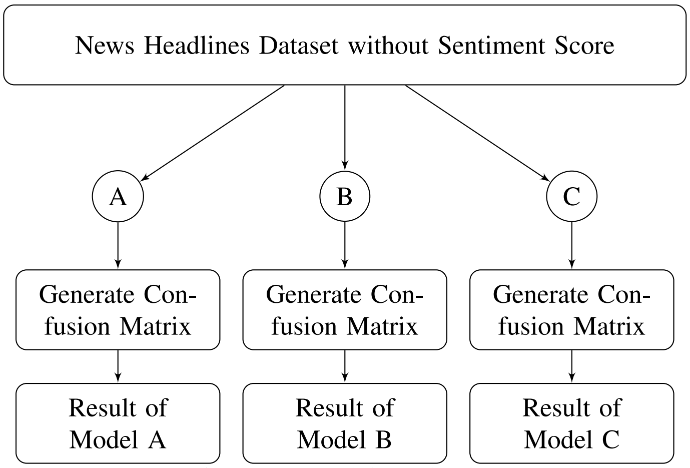

News Headlines Dataset without Sentiment Score
A
B
C
Generate Confusion Matrix
Generate Confusion Matrix
Generate Confusion Matrix
Result of Model A
Result of Model B
Result of Model C

Figure 13. Models evaluation

Table 2. Confusion Matrix

		PREDICTED	
		TRUE	FALSE
ACTUAL	TRUE	TP	FN
	FALSE	FP	TN

$$Accuracy\ Score = \frac{TP + TN}{TP + TN + FP + FN} * 100\ \% \tag{4}$$

Table 3. Model A (Linear SVM) Confusion Matrix

Unigram Model			
		PREDICTED	
		TRUE	FALSE
ACTUAL	TRUE	223	29
	FALSE	9	34
Bi-gram Model			
		TRUE	FALSE
ACTUAL	TRUE	228	27
	FALSE	4	36

Table 4. Model B (Linear SVM + Tf-idf) Confusion Matrix

Unigram Model			
		PREDICTED	
		TRUE	FALSE
ACTUAL	TRUE	227	22
	FALSE	5	41
Bi-gram Model			
		TRUE	FALSE
ACTUAL	TRUE	228	21
	FALSE	4	42

Table 5. Model C (SGD) Confusion Matrix

Unigram Model			
		PREDICTED	
		TRUE	FALSE
ACTUAL	TRUE	218	34
	FALSE	14	29
Bi-gram Model			
		TRUE	FALSE
ACTUAL	TRUE	226	29
	FALSE	6	34

Table 6. Accuracy Comparison between Different Models:

	Model A (Linear SVM)	Model B (Tf-idf + Linear SVM)	Model C (SGD)
Unigram	87.11 %	90.84 %	83.72 %
Bi-gram	89.49 %	91.52 %	88.13 %

5. CONCLUSION AND FUTURE SCOPE

Opinion mining is a vast research domain which can be implemented using Neural Networks, NLP etc. The proposed method is implemented and tested on National/International news headlines of many different regions. The proposed method outperforms existing model. However, the proposed method can not handle sarcasm.

As a future work, based on data accessibility, the proposed method can be implemented on social media data like Twitter, Facebook etc. Also in public sector, the proposed method can be used for the government to analyze impact of a policy in particular region. The proposed method can be altered to handle sarcasm in process I and II in order to increase the accuracy of the model.

REFERENCES

[1] D. Dor, "On newspaper headlines as relevance optimizers," *Journal of Pragmatics*, vol. 35, no. 5, pp. 695 – 721, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378216602001340

[2] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] A. Agarwal, V. Sharma, G. Sikka, and R. Dhir, "Opinion mining of news headlines using sentiwordnet," in *Colossal Data Analysis and Networking (CDAN), Symposium on.* IEEE, 2016, pp. 1–5.

[5] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining." in *LREC*, vol. 10, 2010, pp. 2200–2204.

[6] K. Yang, Y. Cai, D. Huang, J. Li, Z. Zhou, and X. Lei, "An effective hybrid model for opinion mining and sentiment analysis," in *Big Data and Smart Computing (BigComp), 2017 IEEE International Conference on.* IEEE, 2017, pp. 465–466.

[7] S. Rana and A. Singh, "Comparative analysis of sentiment orientation using svm and naive bayes techniques," in *Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on.* IEEE, 2016, pp. 106–111.

[8] R. K. Bakshi, N. Kaur, R. Kaur, and G. Kaur, "Opinion mining and sentiment analysis," in *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on*. IEEE, 2016, pp. 452–455.

[9] S. Aroju, V. Bhargavi, S. Namburi *et al.*, "Opinion mining on indian newspaper quotations," in *Signal Processing and Integrated Networks (SPIN), 2016 3rd International Conference on*. IEEE, 2016, pp. 749–752.

[10] K. A. Hasan, M. S. Sabuj, and Z. Afrin, "Opinion mining using naïve bayes," in *Electrical and Computer Engineering (WIECON-ECE), 2015 IEEE International WIE Conference on*. IEEE, 2015, pp. 511–514.

[11] H. Akkineni, P. Lakshmi, and B. V. Babu, "Online crowds opinion-mining it to analyze current trend: A review," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 5, no. 5, pp. 1180–1187, 2015.

[12] P. Arora, D. Virmani, and P. Kulkarni, "An approach for big data to evolve the auspicious information from cross-domains," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 2, pp. 967–974, 2017.

[13] A. R. Susanti, T. Djatna, and W. A. Kusuma, "Twitter's sentiment analysis on gsm services using multinomial naïve bayes." *Telkomnika*, vol. 15, no. 3, 2017.

[14] J. Jotheeswaran and Y. Kumaraswamy, "Opinion mining using decision tree based feature selection through manhattan hierarchical cluster measure." *Journal of Theoretical & Applied Information Technology*, vol. 58, no. 1, 2013.

[15] C. Bhadane, H. Dalal, and H. Doshi, "Sentiment analysis: measuring opinions," *Procedia Computer Science*, vol. 45, pp. 808–814, 2015.

[16] J. Kaur and S. Vashisht, "Analysis and indentifying variation in human emotion through data mining," *International Journal of Computer Technology and Applications*, vol. 3, no. 6, p. 1963, 2012.

[17] I. Smeureanu and C. Bucur, "Applying supervised opinion mining techniques on online user reviews," *Informatica economica*, vol. 16, no. 2, p. 81, 2012.

[18] K. Gao, H. Xu, and J. Wang, "A rule-based approach to emotion cause detection for chinese micro-blogs," *Expert Systems with Applications*, vol. 42, no. 9, pp. 4517–4528, 2015.

[19] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 740–750.