User Sentimental Analysis from Comments

Report submitted to the SASTRA Deemed to be University as the requirement for the course

BITCIT701R03: BIG DATA ANALYTICS PROJECT

Submitted by

Sai Kiruthika.S -121015085

DECEMBER 2020

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

User Sentimental Analysis from Comments

Thesis submitted to the SASTRA Deemed to be University in partial fulfillment of the requirements for the award of the degree of

B. Tech Information Technology

Submitted by

Sai Kiruthika.S

(Reg no:121015085)

DECEMBER 2020

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA-613 401

SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 OF THE UGC ACT, 1956)
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA-613 401

BONAFIDE CERTIFICATE

This is to certify that the project titled "**User Sentimental Analysis From Comments**" submitted in partial fulfillment of the requirements for the award of the degree of B.Tech Information Technology to the SASTRA Deemed to be University, is a bonafide record of the work done by **Miss.Sai Kiruthika.S**(Reg. No. 121015085) during the final semester of the academic year 2020-21, in the **School of Computing**, under the supervision of **Dr. ErankiLN.Kiran,** Senior Assistant Professor, SASTRA Deemed University. This project work has not formed the basis for the award of any degree, diploma, associate ship, fellowship, or other similar title to any candidate of any University.

Signature of Project Supervisor :

Name with Affiliation :

Date :

Project *Viva Voice* to be held on -

Examiner 1 **Examiner 2**

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA-613 401

DECLARATION

I declare that the project titled "**User Sentimental Analysis From Comments"** submitted by me is an original work done by me under the guidance of **Prof. Dr. ErankiLN.Kiran,** SASTRA Deemed University during the final semester of the academic year 2020-21, in the **School of Computer Science and Engineering**. The work is original and wherever I have used materials from other sources, I have given due credit and cited them in the text of the project report. This project has not formed the basis for the award of any degree, diploma, associate-ship, fellowship or other similar title to any candidate of any University.

Signature of the candidate : [illegible]

Name of the candidate : **Sai Kiruthika.S**

Date :

TABLE OF CONTENTS

LIST OF FIGURES

LISTOF TABLES

ABBREVIATIONS

ALGA	Adaptive Lexicon Learning using Genetic Algorithm
BOW	Bag Of Words
NB	Naive Bays
RAM	Random Access Memory
DTM	Digital Transaction Management

ABSTRACT

Automatic classification of sentiment is significant for many applications such as estimation mining, opinion summarization, contextual advertising, and market investigation. Typically, sentiment classification has been modelled as the problem of preparation a binary classifier using reviews annotated for positive or negative sentiment. However, sentiment is articulated differently in diverse domains, and annotating corpora for every potential domain of interest is costly. Applying a sentiment classifier trained using labelled data for a meticulous domain to classify sentiment of user reviews on a different domain often results in poor performance because words that occur in the train (source) domain might not appear in the test (target) domain. We propose a method to overcome this problem in cross-domain sentiment classification.

1. First, we produce an attitude sensitive distributional thesaurus using labelled data for the source domains and unlabelled data for both source and target domains. Sentiment sensitivity is achieve in the vocabulary by include text height sentiment labels in the condition vectors worn as the foundation for measure the distributional comparison between words.

2. Next, we use the created thesaurus to expand feature vectors during train and test times in a binary classifier. The proposed technique considerably outperforms frequent baselines and proceeds results that are alike with previously planned cross-domain sentiment classification methods on a benchmark data set containing Amazon user reviews for dissimilar types of products. We conduct an extensive empirical analysis of the proposed method on single and multisource domain adaptation, unsupervised and supervised domain adaptation, and frequent similarity measures for creating the sentiment sensitive thesaurus. Moreover, our comparisons against the SentiWordNet, a lexical resource for word polarity, show that the created sentiment-sensitive thesaurus precisely captures words that express similar sentiments.

Sentiment analysis is the process of identifying opinions expressed in text. It aims to analyze the users' sentiment towards particular topics, products, and other subjects. The aim of this study is to present the parallel version of a method of polarity classification of sentiment for big data. In the original method, named ALGA, a genetic algorithm is incorporated to generate lexicons. Since the fitness calculation of that method is its most time consuming part, especially in the big data, in this research, a new parallel method is presented to efficiently calculate the fitness of ALGA. The experiments are conducted on four datasets in terms of running time, speedup and time complexity. Results show that the proposed method achieves better runtimes than the sequential ALGA when the datasets are big. Therefore, in spite of the sequential ALGA, this method can employ the strength of genetic algorithm for searching the landscapes of big data mining problems.

CHAPTER 1

INTRODUCTION

1.1 SENTIMENTAL ANALYSIS AND THEIR USES:

Nowadays, vast amounts of data are produced every day. It is a very challenging task to derive usable information from data. One of the main outlets of data is social media, on which, millions of people express their views on various subjects, such as products, people and organizations. It is important for companies, marketing experts, and researchers to automatically and accurately extract the opinions and sentiments of people. Sentiment analysis or opinion mining is a subfield of text mining that deals with identifying the opinions in text. Text mining faces significant challenges. The data in text mining are unstructured, and it is difficult to deal with them from an algorithmic point of view. Text mining includes various tasks, such as text summarization, document retrieval, text categorization, document clustering, language identification, identifying key phrases, and sentiment analysis (opinion mining). Of these tasks, the problem considered here is sentiment analysis. The text in social media may be objective or subjective.

Unwanted messages are always a challenge to the users of any kind of mail or message delivery system, whether it is normal post or electronic mails. Email spam is a growing threat to the users of any electronic messaging or mail system. Many different kinds of measures have been taken to rectify this issue but there always seems to have a unaddressed problem. Spam filters which use algorithms are used to filter the messages and to remove spam. However no matter how evolved a spam detection system spammers always finds a way to bypass it. Accuracy in spam detection poses a big challenge to those who develop spam filters and related algorithms. Mails mistaken as spam get moved in to the spam folder there by increasing the risk missing important mails. The primary driving force or the objective of this project is to allow the filtering of spam in a more accurate level .This is done by narrowing the search and identification of spam messages by using the basic weapon of spam filtering called keywords in a more advanced way i.e. to concatenate the various keywords to form new sentences or string collections that would identify spam and also use the synonyms of these key words to unmask the masked spam.

1.2 TASKS INVOLVING SENTIMENTAL ANALYSIS:

In the former case, it is constructed of facts and does not contain the personal view of the writer. However, in the latter case, the text includes the personal feelings, which may be positive or negative. The problem of sentiment analysis involves different tasks, as follows:

(i) Subjectivity classification: in this case, the problem is to classify the text into subjective and objective classes. For example, the text "Samsung just released a new phone" is an objective sentence, since it does not contain any subjective views, but the sentence "This new phone is amazing" contains a personal view, and hence, is subjective. This is a subjective view, since another person may not find it "amazing".

(ii) Polarity classification: in this task, the objective is the classification of subjective text into positive and negative classes. The text "This camera is not bad" can be considered a positive text, while "Do not ever buy such a bad product" is negative.

(iii) Strength/Intensity classification: Here, the task is to find the intensity of an opinion. The sentence "This is the best phone ever built" is more positive than "I think it is a good phone", and its intensity is higher. Both sentences are considered positive in the polarity classification.

(iv) Emotion classification: in this task, sometimes called emotion mining, the emotion of the writer of text is considered as its label. This label can have values such as "happiness", "sadness", "anger" and "fear".

For sentiment analysis of texts, a sentiment lexicon could be incorporated. A sentiment lexicon is a list of words and phrases that are either divided into different groups, such as positive and negative, or sentiment scores are assigned to them. Some of the well-known lexicons are Bing Liu's lexicon, AFINN, SentiWordNet 3.0, SentiStrength, and Sentiment140. These lexicons have thousands, to hundreds of thousands of words and phrases. As an example, in the AFINN lexicon, the words have a score from -5 to +5. The score of the word “bad” is -3, and the score for the word “breath-taking” is +5. There are methods to use these lexicons. One of the key methods is to use meta-level features.

For each record, the number/sum of scores of positive words and the number/sum of scores of negative words based on a lexicon are extracted as features. Then, using these features, the classification is accomplished. However, calculating the meta-level features for records is a time consuming task, especially when the number of records is high, or the size of lexicon is large.

A method, named ALGA has been proposed for generating dynamic lexicons based on text. This method is based on the genetic algorithm, and addresses an optimization problem about generating optimized lexicons for polarity classification of micro blogs. The aim of this research is to tackle a limitation of ALGA, when used for big data. ALGA results are shown to be significantly good. However, it has a main drawback when run on massive datasets, and ``in this case, it can be very time-consuming. In this paper, we addressed this problem by replacing the fitness function of ALGA with a new fitness function that is based on the Map Reduce programming model. The records in datasets have also been replicated to make them massive. The proposed method in this paper is called ALGA-Big.

1.3. SOFTWARE SPECIFICATION:

Following are the software used:

1. **R Studio:** R studio was used because it is easy to use and visualize.

1.4 R PACKAGES USED

1.NLPK:

The natural Language Toolkit is a package used for Text Analysis**.**

2.TOKENIZERS:

It is a package used for converting Human Readable text to Machine Readable text.

3.SpaCy:

It is a package used for Production Use and provides a User friendly Api.

4.NORMALIZATION:

It is a package used for normalizing the data for better visualizing

5.MONKEYLEARNR:

It is a package which makes Sentimental analysis in R easy and simply Straightforward

6. NTLK CORPUS:

It is used to read a large number of Lexical databases

CHAPTER 2

OBJECTIVE

- collecting data
- pre-processing the data
- text vectorizations: TFIDF
- sentiment analysis with various techniques
- deploy the model with lexicon and naive bays algorithm

CHAPTER 3

METHODOLOGY

This method calculates accuracy of sentiment analysis with Lexicon and Naive Bays algorithm.

3.1 Model 1 – Lexicon algorithm

Training a classifier from one or more domains (source domains) and applying the qualified classifier on a different domain (target domain). A sentiment sensitive thesaurus that aligns different words that articulate the same sentiment in diverse domains.

In the proposed a cross-domain sentiment classifier using a by design extracted sentiment sensitive thesaurus. To trounce the feature mismatch problem in cross-domain sentiment classification, we use labelled data from numerous source domains and unlabelled data from source and aim domains to calculate the relatedness of features and construct a sentiment sensitive thesaurus. Then use the shaped thesaurus to augment feature vectors during train and test times for a binary classifier.

Input Splits:

An input to a Map Reduce in Big Data job is divided into fixed-size pieces called input splits Input split is a chunk of the input that is consumed by a single map

```
# get the sentiment from the first text:
tokens %>%
  inner_join(get_sentiments("bing")) %>% # pull out only sentiment words
  count(sentiment) %>% # count the # of positive & negative words
  spread(sentiment, n, fill = 0) %>% # made data wide rather than narrow
  mutate(sentiment = positive - negative) # # of positive words - # of nega
tive owrds
```

So this text has 117 negative polarity words and 240 positive polarity words. This means that there are 123 more positive than negative words in this text. Now that we know how to get the sentiment for a given text, let's write a function to do this more quickly and easily and then apply that function to every text in our dataset.

Mapping

This is the very first phase in the execution of map-reduce program. In this phase data in each split is passed to a mapping function to produce output values. In our example, a job of mapping phase is to count a number of occurrences of each word from input splits (more details about input-split is given below) and prepare a list in the form of <word, frequency>

```
GetSentiment <- function(file){
    # get the file
    fileName <- glue("../input/", file, sep = "")
    # get rid of any sneaky trailing spaces
    fileName <- trimws(fileName)

    # read in the new file
    fileText <- glue(read_file(fileName))
    # remove any dollar signs (they're special characters in R)
    fileText <- gsub("\\$", "", fileText)

    # tokenize
    tokens <- data_frame(text = fileText) %>% unnest_tokens(word, text)

    # get the sentiment from the first text:
    sentiment <- tokens %>%
      inner_join(get_sentiments("bing")) %>% # pull out only sentiment words
      count(sentiment) %>% # count the # of positive & negative words
      spread(sentiment, n, fill = 0) %>% # made data wide rather than narrow
      mutate(sentiment = positive - negative) %>% # # of positive words - # of
 negative owrds
      mutate(file = file) %>% # add the name of our file
      mutate(year = as.numeric(str_match(file, "\\d{4}"))) %>% # add the year
      mutate(president = str_match(file, "(.*?)_")[2]) # add president

    # return our sentiment dataframe
    return(sentiment)
}
```

Shuffling

This phase consumes the output of mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubbed together along with their respective frequency.

```
sentiments <- data_frame()
```

```
# get the sentiments for each file in our datset
for(i in files){
    sentiments <- rbind(sentiments, GetSentiment(i))
}

# disambiguate Bush Sr. and George W. Bush
# correct president in applicable rows
bushSr <- sentiments %>%
  filter(president == "Bush") %>% # get rows where the president is named "
Bush"...
  filter(year < 2000) %>% # ...and the year is before 200
  mutate(president = "Bush Sr.") # and change "Bush" to "Bush Sr."

# remove incorrect rows
sentiments <- anti_join(sentiments, sentiments[sentiments$president == "Bus
h" & sentiments$year < 2000, ])

# add corrected rows to data_frame
sentiments <- full_join(sentiments, bushSr)

# summerize the sentiment measures
summary(sentiments)
```

Reducing

In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. In short, this phase summarizes the complete dataset.

```
fileToTokens <- function(file){
    # get the file

fileName <- glue("../input/", file, sep = "")
    # get rid of any sneaky trailing spaces
    fileName <- trimws(fileName)


# read in the new file
    fileText <- glue(read_file(fileName))
    # remove any dollar signs (they're special characters in R)
    fileText <- gsub("\\$", "", fileText)

    # tokenize
    tokens <- data_frame(text = fileText) %>% unnest_tokens(word, text)
}
```

Analysing the model

We have observed the accuracy of our model is around 70% - 80%.

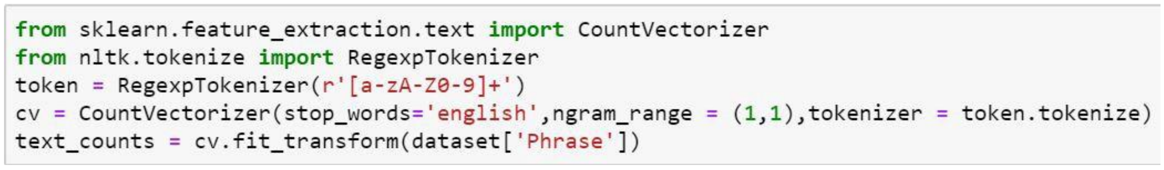
3.2 Model 2 - Naive bayes algorithm

To proceed further with the sentiment analysis we need to do text classification. We can use ‘bag of words (BOW)’ model for the analysis. In laymen terms, BOW model converts text in the form of numbers which can then be used in an algorithm for analysis.

Specifically, BOW model is used for feature extraction in text data. It returns a vector with all the words and the number of times each word is repeated. It is known as BOW because it is only concerned with the number of times a word is repeated rather than the order of words.

Now, we will generate DTM using CountVectorizer module of sci-kit-learn (figure 3). As discussed above we will use:

- tokenizer = Overrides the string tokenization step, we generate tokenizer from NLTK’s Regex tokenizer (by default: None)
- lowercase = True (no need to use, as it is set True by default)
- stop words = ‘english’ (by default None is used, to improve the result we can provide a custom made list of stop words)
- ngram_range = (1,1) (by default its (1,1) i.e strictly monograms will be used, (2,2) only bigrams while (1,2) uses both)



```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(stop_words='english',ngram_range = (1,1),tokenizer = token.tokenize)
text_counts = cv.fit_transform(dataset['Phrase'])
```

Figure 3.1: Using CountVectorizer to prepare the ‘bag of words’

We will now split the data for training and testing to check how well our model has performed (figure 3). Also, we will randomize the data in case our data includes all the positive first and then all negative or some other kind of bias. We will use:

scikit_learn's train_test_split() for splitting the text_count (which contains our X) and dataset['Sentiment'] (this contains Y).

Now we have the training and testing data. We should start the analysis. Our analysis (as most of the ML analysis) will be in 5 steps(a mnemonic to remember them is **DC-FEM** remember as District of Columbia Fire and Emergency Medical service):

- Defining the model
- Compiling the model
- Fitting the model
- Evaluating the model
- Making predictions with the model

1. Defining the model

We will use one of the **Naive Bayes (NB)** classifier for defining the model. Specifically, we will use **Multinomial-INB classifier**. It tells us to use NB classifier. Let us take a detour to learn more about the NB model.

Naive Bayes Model

This model applies Bayes theorem with a Naive assumption of no relationship between different features. According to Bayes theorem:

Posterior = likelihood * proposition/evidence

or

P(A|B) = P(B|A) * P(A)/P(B)

Naive Bayes Model works particularly well with text classification and spam filtering.

Advantages of working with NB algorithm are:

- Requires a small amount of training data to learn the parameters
- Can be trained relatively fast compared to sophisticated models

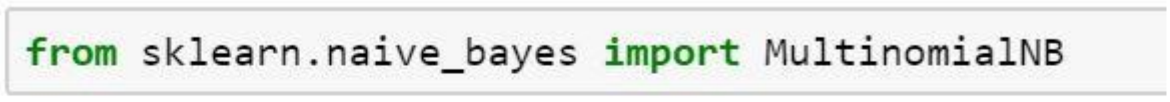
The main disadvantage of NB Algorithm is:

- It’s a decent classifier but a bad estimator
- It works well with discrete values but won’t work with continuous values (can’t be used in a regression)

The first two steps of defining and compiling the model are reduced to identifying and importing the model from sklearn (as sklearn gives as precompiled models).

2. Compiling the model

Since we are using sklearn’s modules and classes we just need to import the precompiled classes.

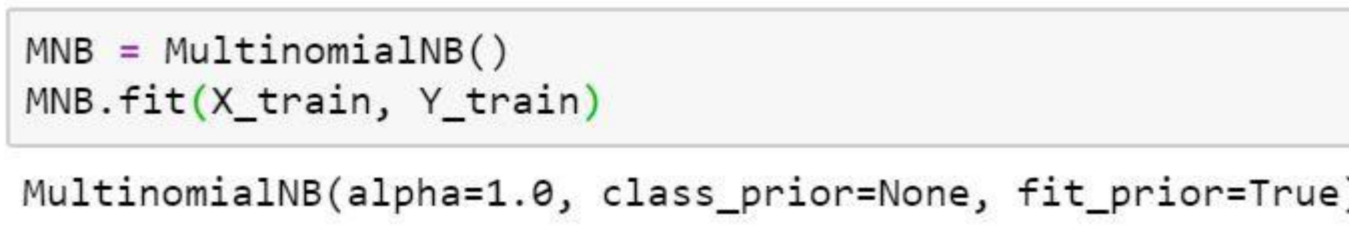


```
from sklearn.naive_bayes import MultinomialNB
```

Figure 3.2: Importing Multinomial Naive Bayes model from sklearn library

3. Fitting the model

In this step, we generate our model-fitting our dataset in the MultinomialNB. In order to look for the arguments which can be passed while fitting the model, it's advised to check the sklearn webpage of the module underuse.



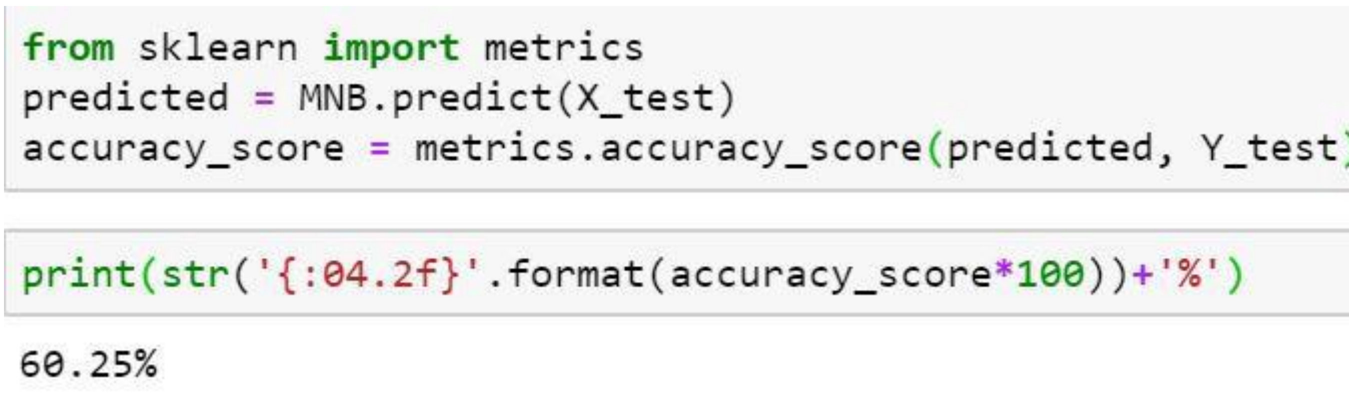
```
MNB = MultinomialNB()
MNB.fit(X_train, Y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True
```

Figure 3.3: Fitting the data in Multinomial Naive Bayes.

4. Evaluating the model

Here we quantify the quality of our model.



```
from sklearn import metrics
predicted = MNB.predict(X_test)
accuracy_score = metrics.accuracy_score(predicted, Y_test
```

```
print(str('{:04.2f}'.format(accuracy_score*100))+'%')
```

```
60.25%
```

Figure 3.4: Using sklearn’s metrics to evaluate the model

Analysing the model

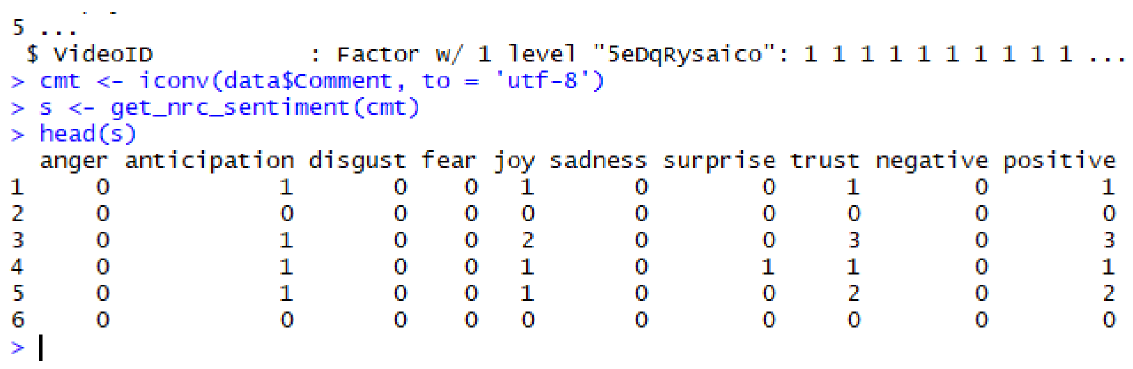
We have observed the accuracy of our model is around 60%.

CHAPTER 4

RESULTS

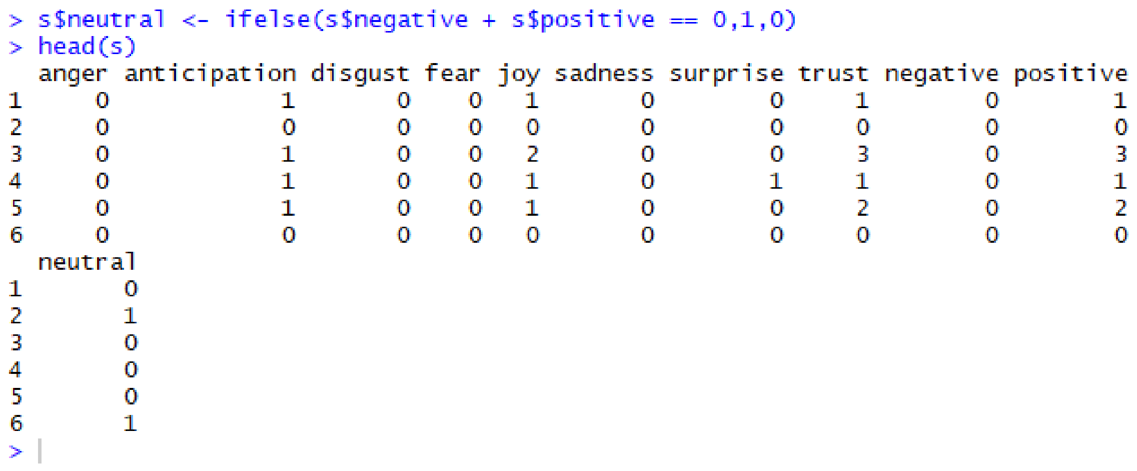
Our proposed method find the most suitable algorithm for sentimental analysis. Here Naive bayes outperformed lexicon algorithm.

Training Datasets are:



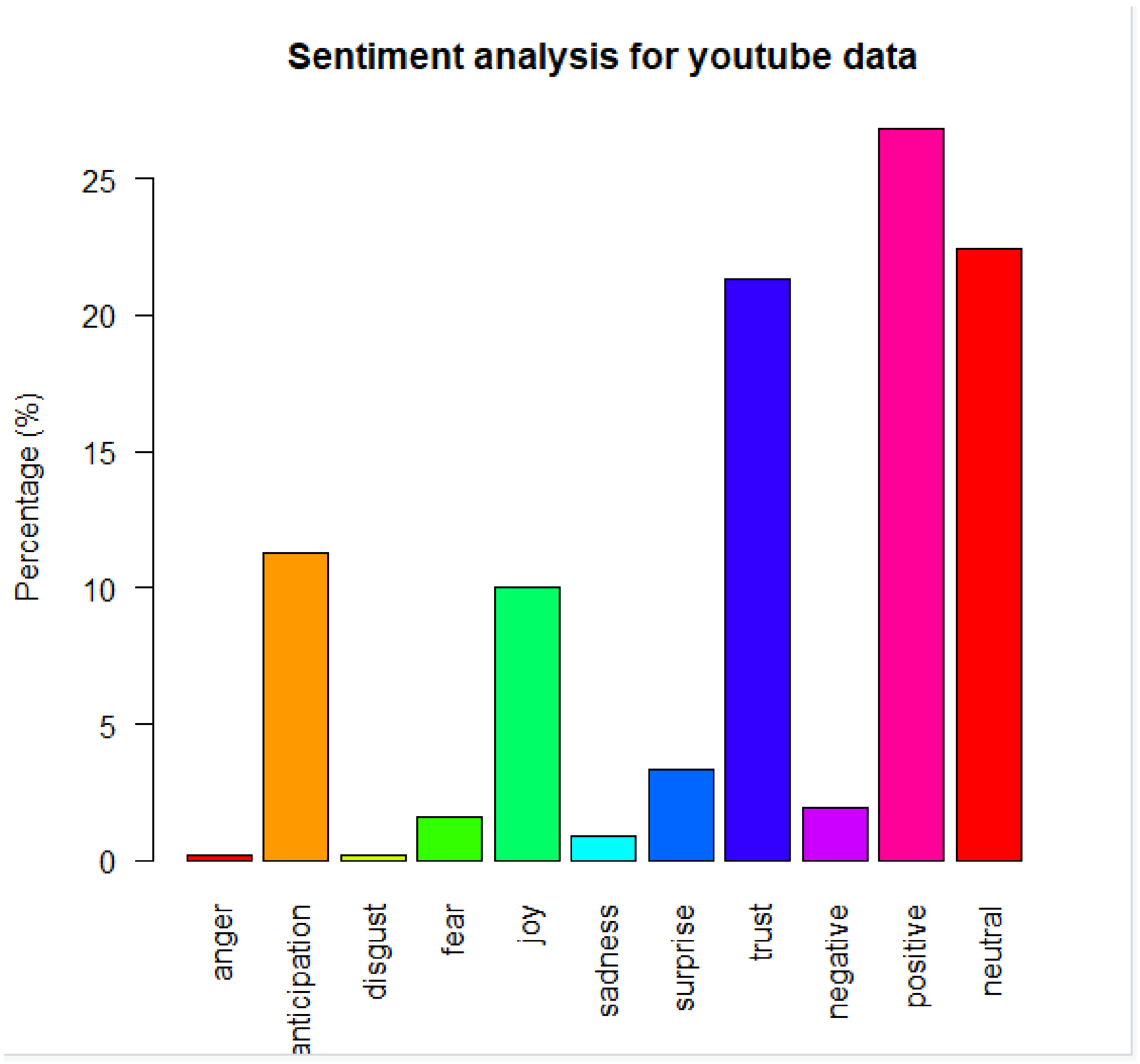
```
5 ...
 $ videoID          : Factor w/ 1 level "5eDqRysaico": 1 1 1 1 1 1 1 1 1 1 ...
> cmt <- iconv(data$Comment, to = 'utf-8')
> s <- get_nrc_sentiment(cmt)
> head(s)
  anger anticipation disgust fear joy sadness surprise trust negative positive
1     0            1       0    0   1       0        0     1        0        1
2     0            0       0    0   0       0        0     0        0        0
3     0            1       0    0   2       0        0     3        0        3
4     0            1       0    0   1       0        1     1        0        1
5     0            1       0    0   1       0        0     2        0        2
6     0            0       0    0   0       0        0     0        0        0
> 
```

Need to neutralize the dataset fields for sentiment analysis:



```
> s$neutral <- ifelse(s$negative + s$positive == 0,1,0)
> head(s)
  anger anticipation disgust fear joy sadness surprise trust negative positive
1     0            1       0    0   1       0        0     1        0        1
2     0            0       0    0   0       0        0     0        0        0
3     0            1       0    0   2       0        0     3        0        3
4     0            1       0    0   1       0        1     1        0        1
5     0            1       0    0   1       0        0     2        0        2
6     0            0       0    0   0       0        0     0        0        0
  neutral
1       0
2       1
3       0
4       0
5       0
6       1
> 
```

Using barplot function we can draw the analysis results:



CHAPTER 5

CONCLUSION AND FURTHER RESEARCH

In the proposed system we calculated accuracy of sentiment analysis for YouTube data. In Naive bayes we created BOW with CountVectorizer which counts the occurrence of the word in the text. More number of times a word occurs it becomes more important for classification. The Naive Bayes algorithm results over sixty per-cent accuracy of sentiment analysis while lexicon algorithm return higher then (eighty per-cent) naive bayes.

REFERENCES

[1] P. N. Tan, Introduction to data mining, Pearson Education India, 2006.

[2] B. Liu, Bing, Sentiment analysis and opinion mining, Synthesis lectures on human language technologies 5.1 (2012): 1-167.

[3] M. Giatsoglou, M. G. Vozalis, K. Diamantaras, A. Vakali, G. Sarigiannidis, and K. C. Chatzisavvas, Sentiment analysis leveraging emotions and word embedding, Expert Systems with Applications, 69 (2017), 214-224.

[4] N. Li, and D. D. Wu, Using text mining and sentiment analysis for online forums hotspot detection and forecast, Decision support systems 48, no. 2 (2010): 354-368.

[5] I. H. Witten, Text Mining, The Practical Handbook of Internet Computing, Chapman and Hall/CRC, 2004.

[6] F. Nielsen, A new anew: evaluation of a word list for sentiment analysis in micro blogs, in Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big Things Come in Small Packages, Heraklion, Crete, Greece, 2011.

[7] S. Baccianella, A. Esuli, F. Sebastiani, Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining, in Proceedings of the Seventh International Conference on Language Resources and Evaluation, Valletta, Malta, (2010): 2200-2204.

[8] M. Thelwall, K. Buckley, G. Paltoglou, Sentiment strength detection for the social web, Journal of the American Society for Information Science and Technology 63.1 (2012): 163-173.

[9] F. Bravo-Marquez, M. Mendoza, B. Poblete, Meta-level sentiment models for big social data analysis, Knowledge-Based Systems 69 (2014): 86-99.

[10] H. Keshavarz, and M. S. Abadeh, ALGA: Adaptive lexicon learning using genetic algorithm for sentiment analysis of microblogs, Knowledge-Based Systems 122 (2017): 1-16.

[11] J. Dean, and S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51.1.

ANNEXURE

CODE IMPLEMENTATION

```
library(SocialMediaLab)

# Storingapikey value in apiKey

apiKey<- "AXXXXXxXXXXXXXXXXXXXXXXXx"

# Storing result of Authentication func with apiKey in key

key<- AuthenticateWithYoutubeAPI(apiKey)

# Storing video Id in video

# For multiple IDs

# video<- c('id1', 'id2')

video<- c('JwvM4Fiha7E','zsaff26T4Wg')

# Data collected from the youtube videos are stored in ytdata

ytdata<- CollectDataYoutube(video, key, writeToFile = FALSE)

str(ytdata)

# writing the data to a csvfile in the current directory

write.csv(ytdata, file = '~ sai.csv', row.names = F)

# reading the dat from csv file and storing the value in data

data<- read.csv(file.choose(), header = T)

str(data)

# Data frame of 'User' and 'ReplyToAnotherUser' attributs are stored in y

y <- data.frame(data$User, data$ReplyToAnotherUser)
```

```
library(igraph)

# graph values of data.frameandare stored in net

net<- graph.data.frame(y, directed=T)

# simplified data of net is stored in simpNet

simpNet<- simplify(net)

# Vetices of simpNet

V(simpNet)

# Edge of simpNet

E(simpNet)

# Name from V(simpNet) is stored as V(simpNet)$lable

V(simpNet)$label <- V(simpNet)$name00

# Degree of simpNet is stored as V(simpNet)$degree

V(simpNet)$degree <- degree(simpNet)


# Histograph

# Histographfunc was called with simpNet's degree as a data arg, color, title, Y axis lable,
X axis lable

hist(V(simpNet)$degree,

col = 'green',

main = 'Histogram of the Node Degree',

ylab = 'Frequency',

xlab = 'Degree of Vertices')

#sentimental analysis
```

```
library(syuzhet)

#read data file

data<- read.csv(file.choose(), header = T)

str(data)

comments<- iconv(data$Comment, to = 'utf-8')



s <- get_nrc_sentiment(comments)

head(s)

s$neutral<- ifelse(s$negative+s$positive==0, 1, 0)

head(s)

comments[4]

#bar plot

barplot(100*colSums(s)/sum(s),

las = 2,

col = rainbow(10),

ylab = 'Percentage',

main = 'Sentiment Scores For Youtube Comments')
```