

# **SENTIMENT ANALYSIS ON TWITTER DATA**

## **A PROJECT REPORT**

*Submitted by*

**UJJWAL JAIN [Reg No: RA1411003010646]  
MAYANK BANSAL [Reg No: RA1411003010694]**

*Under the guidance of*

**Dr. Saad Yunus Sait**

(Research Associate Professor, Department of Computer Science & Engineering)

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE ENGINEERING**

of

**FACULTY OF ENGINEERING AND TECHNOLOGY**



S.R.M. Nagar, Kattankulathur, Kancheepuram District

**April 2018**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

(Under Section 3 of UGC Act, 1956)

**BONAFIDE CERTIFICATE**

Certified that this project report titled “**SENTIMENT ANALYSIS ON TWITTER DATA**” is the bonafide of work done by “ **UJJWAL JAIN [Reg No: RA1411003010646], MAYANK BANSAL [Reg No: RA1411003010694]**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

Dr. Saad Yunus Sait  
**GUIDE**  
Research Associate Professor  
Dept. of Computer Science & Engineering

Signature of the Internal Examiner

**SIGNATURE**

Dr.B.Amutha  
**HEAD OF THE DEPARTMENT**  
Dept. of Computer Science Engineering

Signature of the External Examiner

## **ABSTRACT**

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.

In this project, we are performing sentiment analysis on the twitter data. Assessment and analysis of sentiment of twitter data, we get a very good approach to screen the public's sentiments towards any business, product, service, etc. In this project, we discuss the different approaches of sentiment analysis on twitter data.

One approach is based on three sentiments like positive, negative or neutral. The twitter data is pre-processed into set of words. They are then compared to the dictionary of words which states them as positive, negative or neutral. The sentiment is then predicted using the score of the tweet. If the score is positive, the tweet is considered positive and if the score is negative, the tweet is considered to be negative. On the other hand this approach just counts the number of positive and negative words and makes a conclusion based on their difference.

The other approach in this project combines sentiment analysis with machine learning. Logistic regression model is used for classification of the data. The data is used to train a model which predicts the probability of the positiveness of the data. Doc2vec package is used for text analysis. The approach plots the graph and calculates the area under the curve. The text is constructed using document term matrix. The DTM is weighted using the term frequency-inverse document frequency. The project has a pretty good accuracy. It can be used for sentiment analysis in product reviews.

## **ACKNOWLEDGEMENTS**

We owe a great many thanks to the following people who helped and support us in the preparation of this project. We would like to convey our gratitude to our director Dr.C.MUTHAMIZHCELVAN, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Chennai.

We would like to express our deep sense of gratitude to Dr. B.AMUTHA, Professor and Head of Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, for giving us opportunity to take up this project. We would also like to thank our Project Coordinator Dr. M.PUSHPALATHA, Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, for her valuable inputs during the project reviews.

We take immense pleasure in conveying our thanks to our panel head Dr. M.Murali, Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai for accepting the project and providing his valuable suggestions. We would also like to thank our guide Dr. Saad Yunus Sait, Research Associate Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai for his continuous support and cooperation.

We would like to express our deepest gratitude to our guide, Dr. Saad Yunus Sait, Research Associate Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, for his immense support and cooperation at every stage of the project completion. Several other people also contributed immensely to the project.

We sincerely acknowledge all their effort, which were indispensable to our project.

**Ujjwal Jain**  
**Mayank Bansal**

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iv</b>
<b>LIST OF FIGURES</b>	<b>ix</b>
<b>ABBREVIATIONS</b>	<b>ix</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Sentiment Analysis . . . . .	1
1.2 Twitter . . . . .	4
1.2.1 Properties of a tweet . . . . .	4
1.2.2 Structure of a tweet . . . . .	4
1.2.3 Pre-processing of data . . . . .	5
1.3 Machine Learning . . . . .	5
1.3.1 Supervised Machine Learning algorithms . . . . .	5
1.3.2 Unsupervised Machine Learning algorithms . . . . .	6
1.3.3 Semi-supervised Machine Learning algorithms . . . . .	6
<b>2 LITERATURE SURVEY</b>	<b>7</b>
2.1 Twitter as corpus in opinion Mining . . . . .	7
2.1.1 Linguistic Analysis approach . . . . .	7
2.1.2 Drawbacks of the linguistic approach . . . . .	7
2.2 Feature-based analysis . . . . .	8
2.2.1 Introduction of feature representation . . . . .	8
2.3 TERMINOLOGY USED . . . . .	9
2.3.1 POS-Tagger . . . . .	9
2.3.2 Naive Bayes Classifier . . . . .	9
2.3.3 DTM (Document Term Matrix) . . . . .	10

2.3.4	GENERALIZATION . . . . .	11
2.3.5	UNDER FITTING . . . . .	11
2.3.6	OVERFITTING . . . . .	12
2.3.7	TF-IDF: . . . . .	15
2.3.8	N-GRAMS . . . . .	17
<b>3</b>	<b>SENTIMENT ANALYSIS APPROACHES</b>	<b>18</b>
3.1	PRE-PROCESSING OF DATA: . . . . .	19
3.2	ALGORITHM . . . . .	20
3.3	CHALLENGES . . . . .	23
3.3.1	Problem with this approach . . . . .	23
3.4	DRAWBACKS . . . . .	24
3.5	NEW IMPROVEMENT: . . . . .	24
3.6	CONCLUSION: . . . . .	25
<b>4</b>	<b>SENTIMENTAL ANALYSIS WITH MACHINE LEARNING</b>	<b>26</b>
4.1	BAG OF WORDS . . . . .	26
4.2	LINEAR REGRESSION MODEL FOR MACHINE LEARNING: .	28
4.3	LOGISTIC REGRESSION MODEL . . . . .	29
<b>5</b>	<b>PROJECT DESCRIPTION</b>	<b>31</b>
5.1	Loading and Preparing Data . . . . .	32
5.2	Preparing a Corpus . . . . .	32
5.3	Model training . . . . .	33
5.4	Fetching tweets . . . . .	34
5.5	Plotting graph . . . . .	34
<b>6</b>	<b>IMPLEMENTATION OF THE PROJECT</b>	<b>35</b>
<b>7</b>	<b>PACKAGES USED IN R</b>	<b>41</b>
7.1	twitterR . . . . .	41
7.2	TIDYVERSE . . . . .	41
7.3	PURRRLYR . . . . .	42

7.4	TEXT2VEC . . . . .	42
<b>8</b>	<b>EXPERIMENTATION AND RESULTS</b>	<b>45</b>
8.1	STEPS FOLLOWED . . . . .	45
8.2	RESULT . . . . .	48
8.2.1	RESULT FOR #justiceForAsifa . . . . .	49
8.2.2	RESULT FOR #unnao . . . . .	50
<b>9</b>	<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	<b>51</b>

## LIST OF FIGURES

1.1	Image showing Application of sentiment analysis . . . . .	3
1.2	Lexical analysis . . . . .	4
2.1	Table image showing internal representation of dtm . . . . .	10
2.2	figure showing underfitting between two variable y and x . . . . .	11
2.3	figure showing overfitting between variable y and x . . . . .	12
2.4	Image showing TF weight . . . . .	16
2.5	Image showing idf weight . . . . .	17
3.1	Architecture Diagram . . . . .	18
3.2	Flow Diagram of Proposed System . . . . .	19
3.3	Graph for the algorithm . . . . .	20
3.4	PI chart showing sentiment score percentage . . . . .	22
3.5	sentiment analysis using positive score . . . . .	25
4.1	Graphical representation of bag of words . . . . .	28
4.2	Graphical representation of linear regression . . . . .	29
6.1	code snippet for loading data set and pre-processing . . . . .	35
6.2	Code Snippet For Tokenizing the tweet . . . . .	36
6.3	code snippet for creating Document Term Matrix . . . . .	37
6.4	code snippet for training the model and prediction . . . . .	38
6.5	code snippet for fetching tweet . . . . .	39
6.6	code snippet for showing result . . . . .	40
8.1	Architecture Diagram . . . . .	46
8.2	figure showing AUC versus Lambda . . . . .	47
8.3	probability of positiveness . . . . .	48
8.4	image showing probability of positiveness for #justiceForAsifa . .	49
8.5	image showing probability of positiveness for unnao . . . . .	50



## **ABBREVIATIONS**

<b>DTM</b>	Document Term Matrix
<b>TFIDF</b>	Term Frequency - Inverse Domain Frequency
<b>CV</b>	Cross validation

# CHAPTER 1

## INTRODUCTION

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.

In this project, we are performing sentiment analysis on the twitter data. Assessment and analysis of sentiment of twitter data, we get a very good approach to screen the public's sentiments towards any business, product, service, etc. In this project, we discuss the different approaches of sentiment analysis on twitter data.

One approach is based on three sentiments like positive, negative or neutral. The twitter data is pre-processed into set of words. They are then compared to the dictionary of words which states them as positive, negative or neutral. The sentiment is then predicted using the score of the tweet. If the score is positive, the tweet is considered positive and if the score is negative, the tweet is considered to be negative. On the other hand this approach just counts the number of positive and negative words and makes a conclusion based on their difference.

The other approach in this project combines sentiment analysis with machine learning. Logistic regression model is used for classification of the data. The data is used to train a model which predicts the probability of the positiveness of the data. Doc2vec package is used for text analysis. The approach plots the graph and calculates the area under the curve. The text is constructed using document term matrix. The DTM is weighted using the term frequency-inverse document frequency. The project has a pretty good accuracy. It can be used for sentiment analysis in product reviews.

### 1.1 Sentiment Analysis

Sentiment analysis is a method of computationally analyzing and predicting sentiments, opinion from text, image and any form of data. It is a process of identifying and categorizing sentiments and opinions from a written piece of text. Sentiment Analysis tries to

determine the writers attitude towards a particular topic whether it is positive, negative or neutral.

Sentiment is all about the writers feeling which are attitude, opinions, emotions, etc they are subjective impressions but not facts. Sentiment Analysis is generally a binary classification of opinions like for/against, like/dislike, good/bad, etc.

It is very useful in areas of social media, finance applications. Data is processed and made more accurate in order to modify the predefined algorithms. To identify the usage of algorithms, some addition of classifiers makes it more generous and accurate one.

Many companies use this approach to grow their business; as they will have a huge help if they know where the market is heading; shortcomings may be addressed in order to make life better for the community. Enterprises look for these methodologies in order to grow their business in the outside world. These algorithms indicate as to how people react on social media and to how the demands of these people can be easily met, which is better from the prospective of both the company and the people.

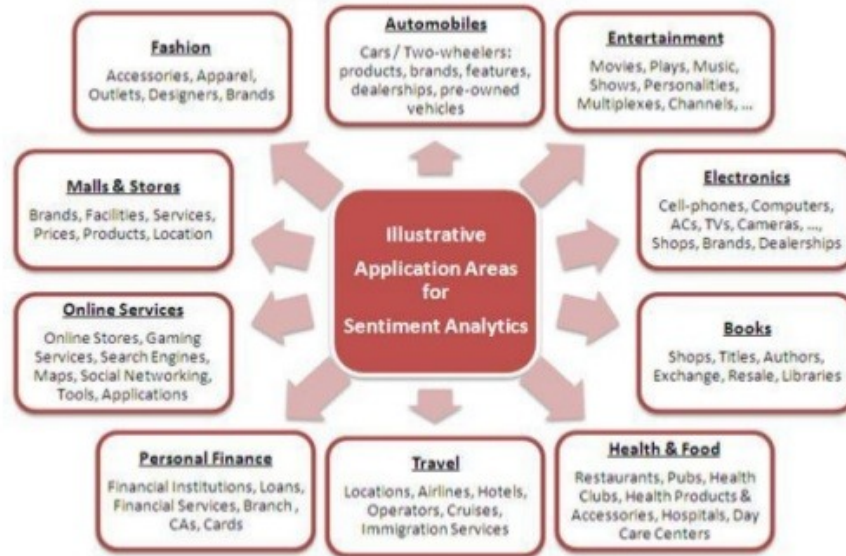
This approach has keywords like it can divide words in a brief set of three domains of words i.e. positive, negative or neutral. Positive words are those which define a positive emoticon or emotion towards a words given like win whereas negative words define a negative emotion or emoticon towards a given words like restricted. Neutral words are those which neither have a positive emotion attached to them nor a negative emoticon i.e. they are perfectly neutral towards any emotion.

In computer science, sentiment analysis is also known as emotion Artificial Intelligence. Sentiment Analysis means the use of natural language processing, text analysis, computational linguistics and bio-metrics to gather the sentiment of the text and survey based upon the extracted information. Sentimental analysis is a process of identifying, mining and to learn affected state. It includes the usage of algorithms of machine learning, text analysis etc.

Major use of sentiment analysis is in finding the attitude of the customer by analyzing customer surveys, product reviews, stock market analysis, online and social media and health care applications to extract information and provide a opinion of people like positive, negative or neutral towards a topic.

Basic sentiment analysis is classifying the polarity of a given text, document, sentence, feature/aspect level.

## Sentiment Analysis Application Areas

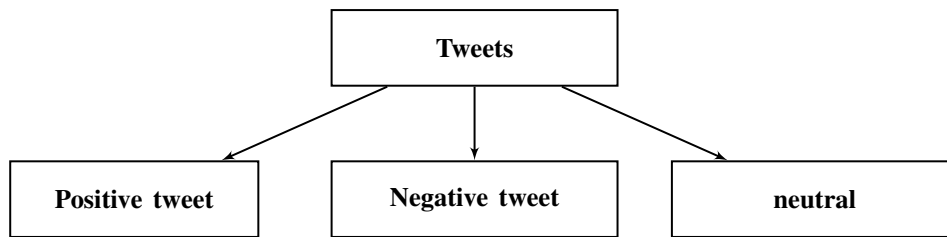


**Figure 1.1:** Image showing Application of sentiment analysis

Sentiment analysis is basically based on three approaches : knowledge based approach and statistical based approach. In knowledge based, we compare the text with a set of words which are classified as positive, negative or neutral.

In statistical approach, one of many machine learning techniques are used to train a model with a large amount of data and predict based upon that. It is classification of the polarity of a given text in the document, sentence or phrase. The goal is to determine whether the expressed opinion in the text is positive, negative or neutral.

Sentiment analysis problem can be solved using either lexical analysis or feature vector extraction. In lexical analysis, the tweets are classified as positive or negative and neutral based on the number of positive words and number of negative words and their overall score as positive or negative and neutral as shown in Figure 1.2 whereas in feature vector problem, Doc2Vec computes a feature vector for every document in the corpus.



**Figure 1.2:** Lexical analysis

## **1.2 Twitter**

Twitter is a micro-blogging website where people share their messages in the form of tweets. Users post and interact with messages known as tweets. It is an online social networking website. People share information through it.

In twitter, an user creates an account and follows the people of whom he likes to see the tweets. On the other hand, users can follow the above user if they like his set of tweets.

### **1.2.1 Properties of a tweet**

The properties of a tweet determines the characteristic of the sentence. It shows emotions of users and determines the length of tweet.

1. twitter.com is a popular micro-blogging website.
2. Each tweet is 140 characters in length.
3. Tweets are frequently used to express a tweeter's emotion on a particular subject.
4. There are firms which poll twitter for analyzing sentiment on a particular topic.
5. The challenge is to gather all such relevant data, detect and summarize the overall sentiment on a topic.

### **1.2.2 Structure of a tweet**

Structure of the tweet shows how the tweet is constructed. It also express the meaning of each annotations in the tweet.

1. 140 characters - spelling errors, acronyms, emoticons, etc.
2. @ symbol refers to a target twitter user.
3. hashtags can refer to topics.

### **1.2.3 Pre-processing of data**

Pre-processing is basically means the cleaning of tweet, or in other words it means removing unwanted words or acronym which does not take part in sentiment of the tweet.

1. Emoticons are replaced with their labels :) = positive :( = negative
2. 170 such emoticons
3. Acronyms are translated. 'lol' to laughing out loud.
4. 5184 such acronyms
5. URLs are replaced with ||U|| tag and targets with ||T|| tag
6. All types of negations like no, n't, never are replaced by NOT
7. Replace repeated characters by 3 characters.

## **1.3 Machine Learning**

Machine learning is a technique which makes a computer act without being explicitly programmed. Machine Learning has given us automated cars, speech recognition, natural language processing and information about the human genes. Machine learning is a field of computer science which enables a computer to perform a task trained on a large amount of data without being purely programmed. Machine learning is a part of Artificial Intelligence. It has evolved from pattern recognition techniques and computational learning theory. Machine Learning is an application of Artificial Intelligence which provides systems the ability to automatically learn and improve from experience. The main aim is to allow a system act without any human intervention or actions and adjust accordingly.

There are three main types of machine learning:

### **1.3.1 Supervised Machine Learning algorithms**

In supervised machine learning algorithms the computer is trained from the past data and applied to the new data using specific examples to predict future events. From the

analysis of training data-set, the algorithm produces a function which helps to predict the output values.

### **1.3.2 Unsupervised Machine Learning algorithms**

In unsupervised machine learning algorithms performs by interacting with the environment by producing actions and discover the errors later.

### **1.3.3 Semi-supervised Machine Learning algorithms**

It is a hybrid of supervised and unsupervised learning. They use both structured and unstructured data to train the machine.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 Twitter as corpus in opinion Mining**

With population of online journals and informal organizations and miniaturized scale blogging, assessment mining and slant examination turned into a field of enthusiasm for some scientists. An extremely wonderful review of the current work was exhibited in (Alexander Pak, Patrick Paroubek) in their study, They gathered a corpus of 300000 content posts from twitter and performed semantic investigation on the corpus and fabricate an opinion classifier that uses the gathered corpus as a preparation information.

##### **2.1.1 Linguistic Analysis approach**

In their exploration work they have introduced a technique for a programmed gathering of a corpus that can be utilized to prepare an assessment classifier . They utilized Tree-Tagger for POS-labeling and watched the distinction in appropriations among positive, negative and unbiased sets. from the perceptions they presumed that creators utilize syntactic structures to depict feelings or state realities.

They have utilized the gathered corpus to prepare an assessment classifier. Their classifier can decide positive, negative and unbiased notions of reports. The classifier depends on the multinomial Naive Bayes classifier that utilizes N-gram also, POS-labels as highlights.

##### **2.1.2 Drawbacks of the linguistic approach**

They play out an alternate arrangement undertaking however: subjective versus objective. For subjective information they gather the tweets finishing with emojis in a similar way as Go et al. (2009). For target information they creep twitter records of well known



daily papers like New York Times, Washington Posts and so on. They report that POS and bigrams both push (in opposition to comes about displayed by Go et al. (2009)). Both these approaches, in any case, are essentially in view of ngram models. Also, the information they use for preparing and testing is gathered via look questions and is along these lines one-sided.

## **2.2 Feature-based analysis**

Sentiment analysis has been dealt with as a Natural Language Processing undertaking at numerous levels of granularity. a decent research done by (Apoorv Agarwal Boyi Xie Ilia Vovsha Owen Rambow Rebecca Passonneau) on highlights that accomplish a noteworthy increase over a unigram benchmark and they investigate an alternate technique for information portrayal and report change over the unigram models.

### **2.2.1 Introduction of feature representation**

They have utilize already proposed best in class unigram demonstrate as their gauge and report an in general pick up of more than 4 percent for two characterization assignments: a twofold, positive versus negative and a 3-way positive versus negative versus impartial. they exhibited a complete set of examinations for both these errands on physically explained information that is an irregular example of stream of tweets. they researched two sorts of models: tree bit and highlight based models and illustrate that both these models outflank the unigram gauge. For their element based approach, they do highlight investigation which uncovers that the most critical highlights are those that join the earlier extremity of words and their parts-of-discourse labels. they likely presume that assessment examination for Twitter information is not that unique in relation to assessment examination for other types.

## 2.3 TERMINOLOGY USED

There are many different methods and technique used through out project. Below are the given terminology used in this project and their brief description.

### 2.3.1 POS-Tagger

A part-of-speech (PoS) tagger is a software tool that labels words as one of several categories to identify the word's function in a given language. In the English language, words fall into one of eight or nine parts of speech. Part-of-speech categories include noun, verb, article, adjective, preposition, pronoun, adverb, conjunction and interjection.

PoS taggers use algorithms to label terms in text bodies. These taggers make more complex categories than those defined as basic PoS, with tags such as "noun-plural" or even more complex labels. Part-of-speech categorization is taught to school-age children in English grammar, where children perform basic PoS tagging as part of their education.

PoS taggers categorize terms in PoS types by their relational position in a phrase, relationship with nearby terms and by the word's definition. PoS taggers fall into those that use stochastic methods, those based on probability and those which are rule-based. One of the first PoS taggers developed was the E. Brill tagger, a rule-based tagging tool. E. Brill is still commonly used today. Other tools that perform PoS tagging include Stanford Log-linear Part-Of-Speech Tagger, Tree Tagger, and Microsoft's POS Tagger. Part-of-speech tagging is also referred to as word category disambiguation or grammatical tagging. PoS tagging is used in natural language processing (NLP) and natural language understanding (NLU).

### 2.3.2 Naive Bayes Classifier

Naive Bayes is a simple technique for constructing classifiers, models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such

classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

### 2.3.3 DTM (Document Term Matrix)

A document term matrix or term document matrix is a model to represent a list words in a document in a matrix form. It is mathematical representation in the form of a matrix which depicts the frequency of occurrence of the words in a document. In a document-term matrix, rows correspond to the documents in the collection whereas columns point towards the terms. The value taken by the matrix is decided by various methods. One of the scheme is term frequency-inverse document frequency (tf-idf). These are the methods used in natural language processing techniques. DTM is a term used widely in deep learning.

The Figure 2.1 below shows the internal representation of dtm in memory. As we can

```
> inspect(tdm[20:30, 1:30])
<<TermDocumentMatrix (terms: 11, documents: 30)>>
Non-/sparse entries: 1/329
Sparsity           : 100%
Maximal term length: 7
Weighting          : term frequency (tf)
```

	Docs																													
Terms	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
abs	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
absb	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
absenc	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
absolut	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
absorb	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
abu	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
abus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
abut	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
academi	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
acceler	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
accept	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	

**Figure 2.1:** Table image showing internal representation of dtm

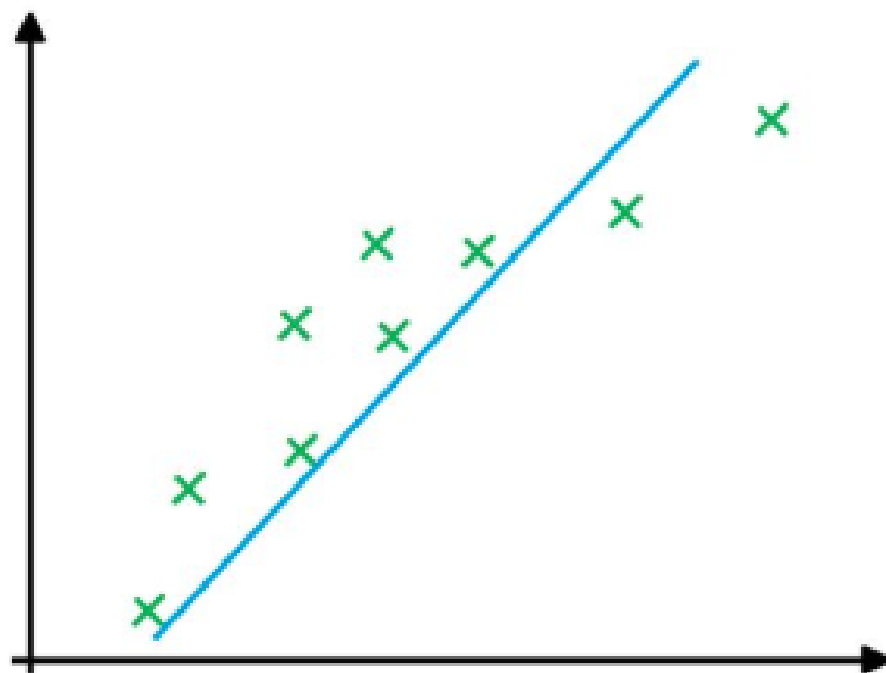
see the words are represented as document and each column represents as docs and if the docs are present in some collection then it is represented as 1 and if it is not present then it is represented as 0.

### 2.3.4 GENERALIZATION

We should not test a machine learning model on a data-set which we used to train the model, since it will not give any indication of how well our model performs on unknown data. The ability to perform well on unknown data is called generalization.

### 2.3.5 UNDER FITTING

Underfitting is the problem when our model can not take up the nodes in the data set. For example suppose we have one variable  $x$  and another variable  $y$ .  $y$  is linearly dependent on  $x$  this is a linear model but it does not perform well on our data, so we add more features to cover the whole training data set.



**Figure 2.2:** figure showing underfitting between two variable  $y$  and  $x$

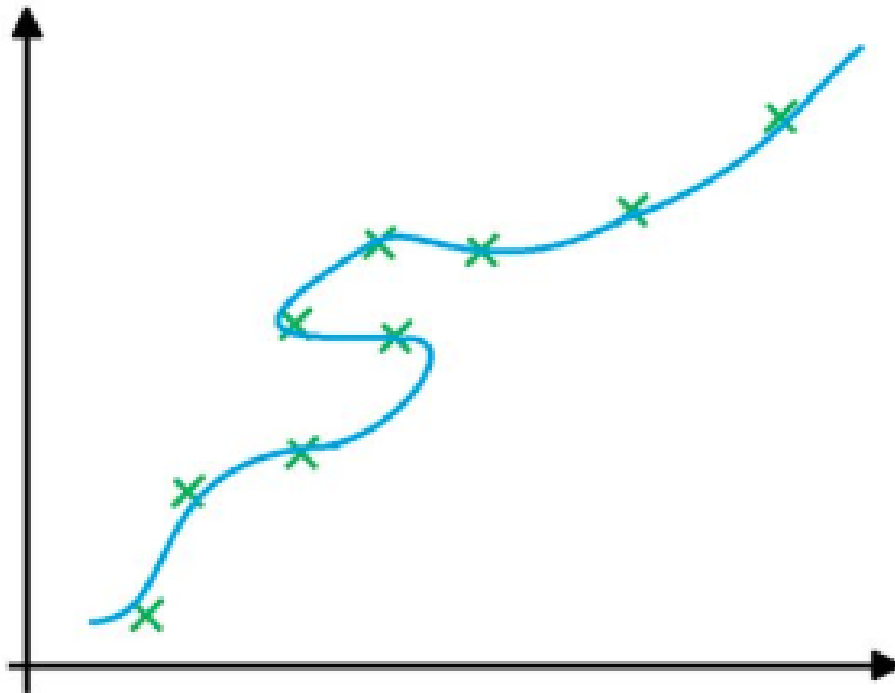
The figure 2.2 shows the linear functions curve between two variables one is dependent variable  $y$  and another independent variable  $x$ . As we can see the curve does not cover all the points in  $x$ - $y$  plane, this is called underfitting.

### 2.3.6 OVERFITTING

When a model performs well on training data but does not perform accurately on test data, we say that it has overfit the training data or that the model is overfitting. It happens because the model learns the extra noise from the model which does not constitute in the actual accuracy of the model. If we keep adding more features we'll get a function that is more and more complex and that covers almost every data point in training data set. This is an example of overfitting. In this case, we are performing polynomial fitting

$$y = w_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d.$$

Even though the curve goes through all the points, it will not perform well on unknown data



**Figure 2.3:** figure showing overfitting between variable y and x

The figure 2.3 shows the relationship between two variable y and x, y is dependent and x is independent, from the graph we can see the function curve covers all the point but it is specific to the training data set, This is called overfitting. overfitting can be prevented

with many different methods as given below.

1. k-fold cross validation
2. regularization

## **K-FOLD CROSS VALIDATION**

Cross Validation is an extremely valuable strategy for evaluating the execution of machine learning models. It helps in knowing how the machine learning model would sum up to a unknown data set. We can utilize this strategy to gauge how exact the forecasts our model will give in practice.

When we are given a machine learning issue, we will be given two kind of data sets, known information (training data set) and obscure information (test data set). By utilizing cross approval, we would be "testing" our machine learning model in the "training" phase to check for overfitting and to get a thought regarding how our machine learning model will perform in generalize area, which is the test data given in the problem.

To do cross validation on the given data set first we have to divide the whole data set in two parts.

1. Training data set
2. testing data set

We first train our machine learning model on the training data set and will test the accuracy of the model on the testing data set

we will get to know how accurate our machine learning model predictions are when we compare the model predictions on the validation set and the actual labels of the data points in the validation set.

For reducing the variance, k rounds of cross validation are performed by using different training sets and testing sets. The results from all the rounds are averaged to estimate the accuracy of the machine learning model. K-fold cross validation Steps

1. First partition the whole data set in k parts like  $f_1, f_2 \dots f_k$  here f represent fold.

2. We will use the f1 as testing data set and f2 to fk will be used as training data set
3. Now we will use cross validation training set for training our model and calculate the accuracy of the model by validating against the testing data set
4. We will estimate the accuracy by averaging the result obtained in all the k cases validation

## REGULARIZATION

Many machine learning algorithms face the problem of overfitting. Which means the learned model does not perform accurate on test data. This occurs as increasing training effort we start to model every node in the training data. One way to avoid this is to use simpler models. Since many learning methods learns weights of a model.

Regularization in machine learning is a technique to solve the problem of over fitting. Overfitting of a model can cause the less accurate results bcoz we use training data-set and our model works well on training data-set and will not work accurate on unknown data-set.

This is the method we will talk about and try to derive the formula for the same. Basically, it presents a cost term for acquiring more highlights with the goal work. Consequently, it tries to push the coefficients for some, factors to zero and subsequently lessen cost term.

They are different type of regularization technique used to solve the problem of overfitting.

1. L1 Regularization
2. L2 Regularization

A regression model that uses L1 regularization technique is called Lasso Regression and model which uses L2 is called Ridge Regression. A linear regression model is a function consisting of one dependent variable y and many independent variable x.

$$Y = a_1 + a_2x_1 + a_3x_3$$

In this equation,  $a_1$ ,  $a_2$ ,  $a_3$  are the coefficients and  $x_1$ ,  $x_2$ ,  $x_3$  .. are the independent variables. Given a data containing x and y, we estimate  $a_1$ ,  $a_2$ ,  $a_3$  based on an objective

function. For a linear regression the objective function is given in figure 2.1

$$\min_f |Y_i - f(x_i)|^2$$

Now this function will not hold to be true to optimize if number of independent variable increases too much, in order to avoid that we put another penalize term in our objective function to find the estimate of co-efficient of the independent terms.

$$\min_{f \in H} \sum_{i=1}^n |Y_i - f(X_i)|^2 + \lambda ||f||_H^2$$

Term in the equation is the sum of squares of the coefficients multiplied by the param lambda. Lambda = 0 means a over-fit scenario and Lambda = Infinity means the problem can now be solved for just single mean estimation. Optimizing Lambda is the task we have to solve, looking at the trade-off between the prediction accuracy of training sample and prediction accuracy of the testing sample.

### 2.3.7 TF-IDF:

TF-IDF is a numerical statistic used in information retrieval. TFIDF is abbreviation for Term frequency-inverse document frequency. It is a numerical statistic which determines the importance of a word in a document in a collection or corpus. It performs as a determining and weighting factor in text analysis, searches to gather information and user modeling.

The tf-idf value is directly proportional to the number of times a word appears in a document and is set offset by the increase in frequency of total number of words in the document.

This technique helps to adjust the fact that some words may appear more frequently in the documents.

Tf-idf is a dependable statistic to weigh the importance of a word in a document, 85% of text based recommender system use tf-idf as a scheme for text analysis. It is one the most popular term weighting techniques. Simplest method for ranking is just the summation of tf-idf for each term query in the document. There are many ranking functions available for ranking the document in search.



Different types of tf-idf technique is used by search engines to rank whether a document is relevant to the user or not. TF-IDF is also used for text summarization and text classification.

For finding the tf-idf weight, two different terms need to be calculated, the term frequency is the first term which is calculated by the frequency of the word in the document divided by the total number of words in the document.

The second term is the Inverse Document Frequency which means logarithm of the number of total document in the collection divided by the number of document where the term appears.

### TF:

Term Frequency is a term which measures how frequently a term occurs in a document. As the length of the documents is different, it may be possible that a term would appear more times in long documents than shorter ones. Therefore the term frequency is divided by the document length as a way of normalization:

$TF(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document}).$

weighting scheme	TF weight
binary	0, 1
raw count	$f_{t,d}$
term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

**Figure 2.4:** Image showing TF weight

## IDF:

Inverse Document Frequency is a term which measures the importance of a term. We consider all terms equally important while computing the term frequency. However it is known that terms, such as is, of, and that, appear a lot of times but have little importance. Therefore we weigh down the frequent terms while scale up the rare ones, by computing the following.

$\text{IDF}(t) = \log_e(\text{Total number of documents} / \text{Number of documents with term } t \text{ in it})$ .

weighting scheme	IDF weight ( $n_t =  \{d \in D : t \in d\} $ )
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log \left( 1 + \frac{N}{n_t} \right)$
inverse document frequency max	$\log \left( \frac{\max_{t' \in d} n_{t'}}{1 + n_t} \right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

**Figure 2.5:** Image showing idf weight

### 2.3.8 N-GRAMS

A n-gram is just a succession of tokens. With regards to computational phonetics, these tokens are normally words, however they can be characters or subsets of characters. The n just alludes to the quantity of tokens. On the off chance that we are checking words, the string "Tomorrow it will rain" is a 4-gram. This 4-gram contains the 3-grams Tomorrow it will and it will rain. A unigram is a solitary token, e.g., Tomorrow. These tokens require not be words or characters. They might be DNA codes, double numbers, or possibly any sort of consecutive information possible. They are ordinarily used to catch factual data from a few informational index, and are irrationally successful in doing as such, regularly beating apparently more unpredictable methodologies.

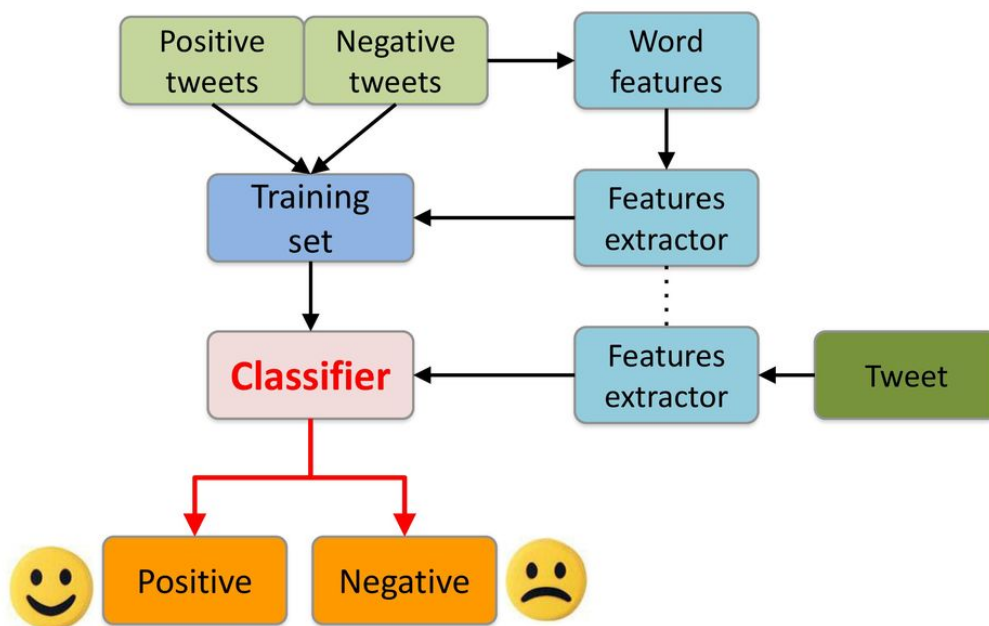
## CHAPTER 3

### SENTIMENT ANALYSIS APPROACHES

The first approach to sentiment analysis was a very basic approach. It was carried out by Pang and Lee. They used a very simplistic approach towards sentimental analysis. The initial approach was based on a list of words. It was also called dictionary based approach or lexicon based approach.

In this approach, the system has a collection of words which are classified as positive,

### Sentiment Analysis Architecture



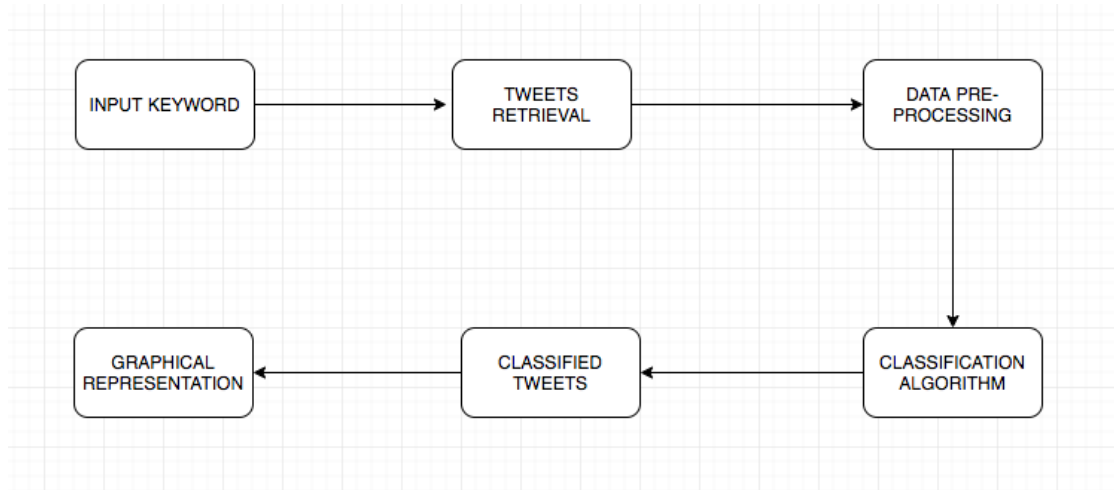
Vishal Kharde and Sheetal Sonawane (2016), "Sentiment Analysis of Twitter Data: A Survey of Techniques," International Journal of Computer Applications, Vol 139, No. 11, 2016, pp.5-15

27

**Figure 3.1:** Architecture Diagram

negative or neutral. The collections include words like happy, sad, like, not, etc. This is a knowledge based design of algorithm. In the above method of sentiment analysis, we first gather the required amount of tweets.

The approach is as follows:-



**Figure 3.2:** Flow Diagram of Proposed System

1. Tokenize Tweet Downloader-This software helps us to tokenize the tweets and download a particular number of tweets.
2. Download the tweets using Twitter API-The tweets are then downloaded using the twitter API. There are some problems with twitter API as only last 7-8 days tweets can be downloaded. We cannot process the historical trends of the tweets. This is a drawback of this approach.
3. Pre-processing of non-English Tweets-People express their emotions in many different ways. People use different languages to express their opinion. The tweets need to be pre-processed and converted into english text as we are performing the algorithm on a list of words.
4. Remove URL, Target Mentions, Hash tags, Numbers-As we are performing the method on a bag of words, we need to process a large amount of text. People mention different url's, target mentions, Hash tags , Numbers, etc. They are removed while pre-processing of the tweets.
5. Replace Negative Mentions-The repeated Sequence of Characters is replaced. For example, coooooooooool by cool.

### 3.1 PRE-PROCESSING OF DATA:

1. Emoticons are replaced with their labels-The emoticons are replaced using the below mentioned labels. They are classified based on their polarity. There are 170 such emoticons. :) = positive :( = negative
2. Acronyms are translated to the respective text. 'lol' is translated to laughing out loud. 'ootd' is translated to outfit of the day. There are 5184 such acronyms.
3. URLs are replaced with ||U|| tag and targets with ||T|| tag.

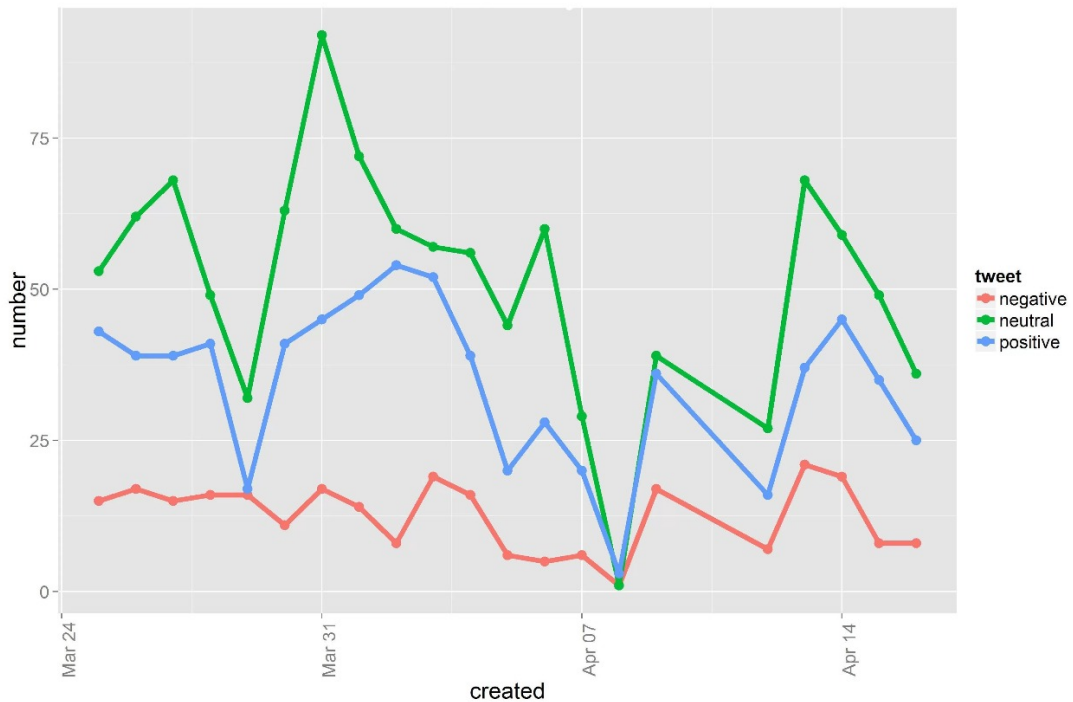
4. All types of negations like no, n't, never are replaced by NOT. The word doesn't is replaced by does not
5. Replace repeated sequence of characters by 3 characters.
6. Remove the different non-english words.

## 3.2 ALGORITHM

In the following algorithm, firstly the keyword is taken as input for which the sentiment analysis has to be done. The tweets are first downloaded and retrieved. The number of tweets are then pre-processed. The pre-processing of data is done by following the above set of rules.

The tweets are then classified according to the classification algorithm. The tweets are now a collection of words. They are now compared to the dictionary of words created by us or a set of words downloaded from the internet. We can add our own required set of words.

Positive words are marked as one, neutral words are marked as zero and negative words are marked as minus one



**Figure 3.3:** Graph for the algorithm

1. Positivecount = Number of positive words.
2. Negativecount = Number of negative words.
3. Ncount = Number of positive words –Number of negative words.

The figure 3.3 shows the graph of number of tweets versus the date, Blue line represent the neutral tweet and green line represent the positive and red line represent the negative tweets.

---

```
score.sentiment = function(sentences, pos.words, neg.words,
  .progress='none')
{
  require(plyr)
  require(stringr)
  list=lapply(sentences, function(sentence, pos.words,
    neg.words)
  {
    sentence = gsub('[:punct:]', ' ', sentence)
    sentence = gsub('[:cntrl:]', '', sentence)
    sentence = gsub('\\d+', '', sentence) #removes decimal
      number
    sentence = gsub('\\n', '', sentence) #removes new lines
    sentence = tolower(sentence)
    word.list = str_split(sentence, '\\s+')
    words = unlist(word.list) #changes a list to character
      vector
    pos.matches = match(words, pos.words)
    neg.matches = match(words, neg.words)
    pos.matches = !is.na(pos.matches)
    neg.matches = !is.na(neg.matches)
    pp = sum(pos.matches)
    nn = sum(neg.matches)
    score = sum(pos.matches) - sum(neg.matches)
    list1 = c(score, pp, nn)
    return (list1)
  })
}
```

```

}, pos.words, neg.words)
score_new = lapply(list, '[', 1)
pp1 = lapply(list, '[', 2)
nn1 = lapply(list, '[', 3)

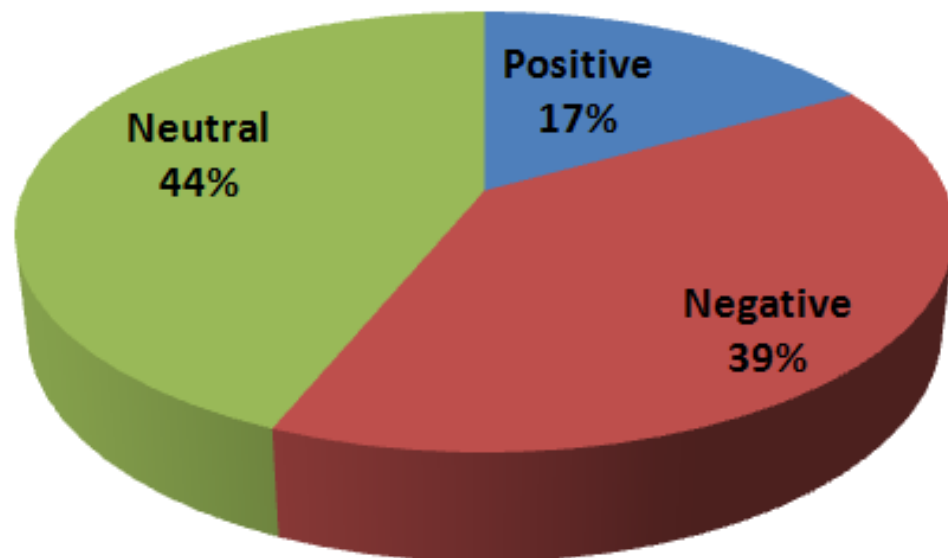
scores.df = data.frame(score = score_new, text=sentences)
positive.df = data.frame(Positive = pp1, text=sentences)
negative.df = data.frame(Negative = nn1, text=sentences)

list_df = list(scores.df, positive.df, negative.df)
return(list_df)
}

```

---

If the Ncount is supposed to be positive, the overall sentiment of the tweet is considered as positive. If the Ncount is supposed to be zero, the overall sentiment of the tweet is considered as neutral. If the Ncount is supposed to be negative, the overall sentiment of the tweet is considered as negative.



**Figure 3.4:** PI chart showing sentiment score percentage

## 3.3 CHALLENGES

1. Tweets are highly unstructured and also non-grammatical. It is very hard to make proper English from the tweets as people do not write them grammatically.
2. Out of Vocabulary Words-Some of the words may be out of vocabulary or our dictionary. We may not be able to predict the exact sentiment without processing of the above words.
3. Lexical Variation-People are of different regions and different areas. They use different languages in their tweets. Pre-processing of data does not take into consideration the following set of languages.
4. Extensive usage of acronyms like asap, lol, idk-A very high percentage of tweets is filled with acronyms.
5. Extensive use of emoticons-Extensive use of emoticons is done which makes it hard to process.

### 3.3.1 Problem with this approach

The caveats of sentiment analysis:

Sentence1: I want a burger so bad.

Sentence2: I had a burger. It was so bad.

The first sentence states that the user wants a burger so badly. The second sentence states that the user just had a burger and it was bad in taste. If we perform sentiment analysis on both the sentences using approach 1, it shows negative as both the sentences contain a negative word.

Both the sentences have different sentiment using the same word. Here our sentiment analysis fails as it is not able to display the correct sentiment. This is a major loophole in sentiment analysis.

This example proves that we cannot depend on sentiment analysis as it is not 100% accurate. Now, different feature based analysis is being performed. More possible machine learning models can be made which represent twitter data in a better way. This is important because we pre-process the data and delete a lot of information.

We need to develop good techniques to pre-process the data.



### 3.4 DRAWBACKS

Following are the drawbacks of knowledge based approach. Knowledge based approach is 50% accurate.

1. computes the number of positive and negative words and makes a conclusion based on their difference.
2. when using a simple vocabularies approach for a phrase 'not bad 'we'll get a negative estimation.
3. results comparing to tweets analysis because they usually include lots of misspelling.

### 3.5 NEW IMPROVEMENT:

In knowledge based approach, we have a limited number of words. The higher the number of words in the dictionary, the better the approach is and approximately it is nearer to the actual sentiment.

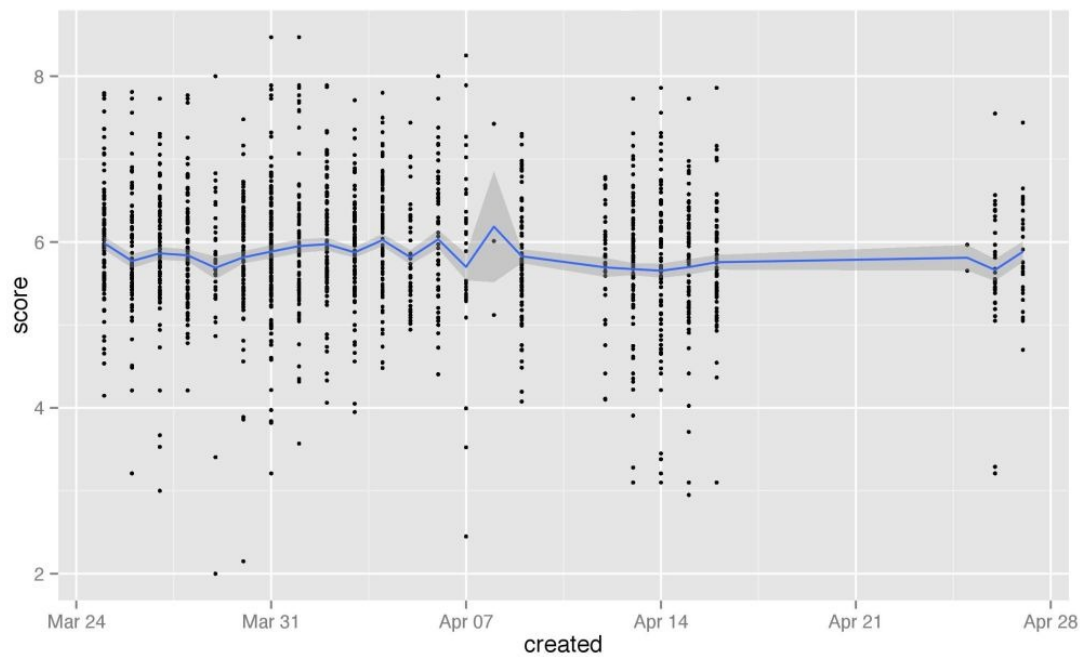
In new approach, we use the knowledge based approach to a higher level and approximate value. Also, we recover tweets from the past to have historical data about the tweets and predict approximate sentiments.

In this approach, we maintain a new dictionary of words which has certain values attached to it. For example, "perfect" is given 6 points whereas "good" is rated as 4 points. This will give more correctness towards the prediction of sentiments.

If a tweet contains a single "good" and three words as "bad" it should have more negative impact than positive impact.

In this approach we used the following algorithm:

1. We mark all the words as a rating from 0 to 6.
2. Then we take the average rating of a tweet.
3.  $\text{Average} = \frac{\text{Total rating of all words}}{\text{number of words}}$



**Figure 3.5:** sentiment analysis using positive score

### 3.6 CONCLUSION:

In the above approach, we will have better approximation of the sentiment of the tweet. Overall the amount of negative words will have less influence on the amount of positive words.

## **CHAPTER 4**

### **SENTIMENTAL ANALYSIS WITH MACHINE LEARNING**

The most advanced approach to sentimental analysis is based on the statistical methods. It is a hybrid approach of knowledge based and machine learning algorithms. Statistical approach are based on parts of machine learning like semantic analysis, support vector machines, logistic regression model, "bag of words", etc.

In statistical approach, the model is trained with a large amount of data and predefined examples. Then the input is taken from the user to predict the following output.

Statistical approach-In statistical approach or evolutionary approach, the information is classified based upon some statistical method which is known as the genetic algorithm.

In our project we are using the logistic regression model as our base model.

In our model, we are working on two packages doc2vec and text2vec. They are very useful approaches in text analysis.

The problem with the old approach was that it calculated the number of positive and negative words and made a result based on the difference between them. Also some phrases like "not bad" are simple but would give a negative conclusion based upon the count.

Semantic Analysis of twitter data: In semantic analysis, the analysis done based on semantics of the text, phrases are made meaningful, synonyms are included in the vocabulary.

#### **4.1 BAG OF WORDS**

The Bag-of-Words model is a model used for simplifying representation of text in natural language processing and information retrieval. It is also known as vector space model. In this model, a text such as a sentence or document is represented as a bag or a

multiset of its words without considering the grammar and even the sequence of words. The bag of words model keeps the multiplicity of the words. This model is also used for image processing considering the image as set of text.

The following models a text document using bag-of-words. Here are two simple text documents:

1. Ujjwal likes to watch movies. Gauri likes movies too.
2. Ujjwal also likes to watch cricket games.

Based on these two text documents, a list constructed as follows for each document:

{ Ujjwal, likes, to, watch, movies, Gauri,likes,movies, too. }

{ Ujjwal, also, likes, to, watch, cricket, games. }

Representing each bag-of-words as:

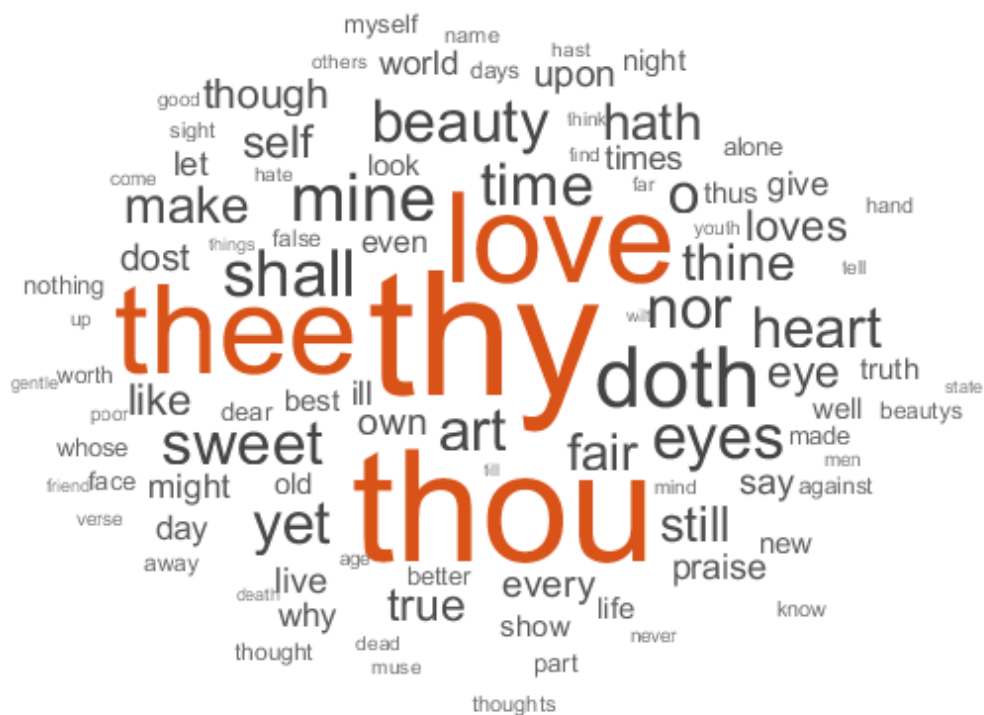
BoW1={ "Ujjwal":1,"likes":2,"to":1,"watch":1,"movies":2,"Gauri":1,"too":1 };

BoW2={ "Ujjwal":1,"also":1,"likes":1,"to":1,"watch":1,"cricket":1,"games":1 };

Each key is the word, and each value is the number of occurrences of that word in the given text document. The order of elements is free. For example.

BoW1 = { "too":1,"Ujjwal":1,"movies":2,"Gauri":1,"watch":1,"likes":2,"to":1 }

The Bag-of-words model is mainly used as a tool of feature generation. After transforming the text into a "bag of words", we can calculate various measures to characterize the text. The most common type of characteristics, or features calculated from the Bag-of-words model is term frequency, namely, the number of times a term appears in the text.



## 4.2 LINEAR REGRESSION MODEL FOR MACHINE LEARNING:

Linear regression model is most easy and understanding algorithm in statistical and machine learning. Machine learning is a field in computer science which predicts the output based upon the training data and minimizing the error of making a model and make the most approximate predictions based on the fact of explain ability. It is a statistical algorithm as well as machine learning algorithm. Linear regression model is based on the linear relationship between input variables and a single output variable. The output variable ( $y$ ) is calculated from the given input variables ( $x$ ).

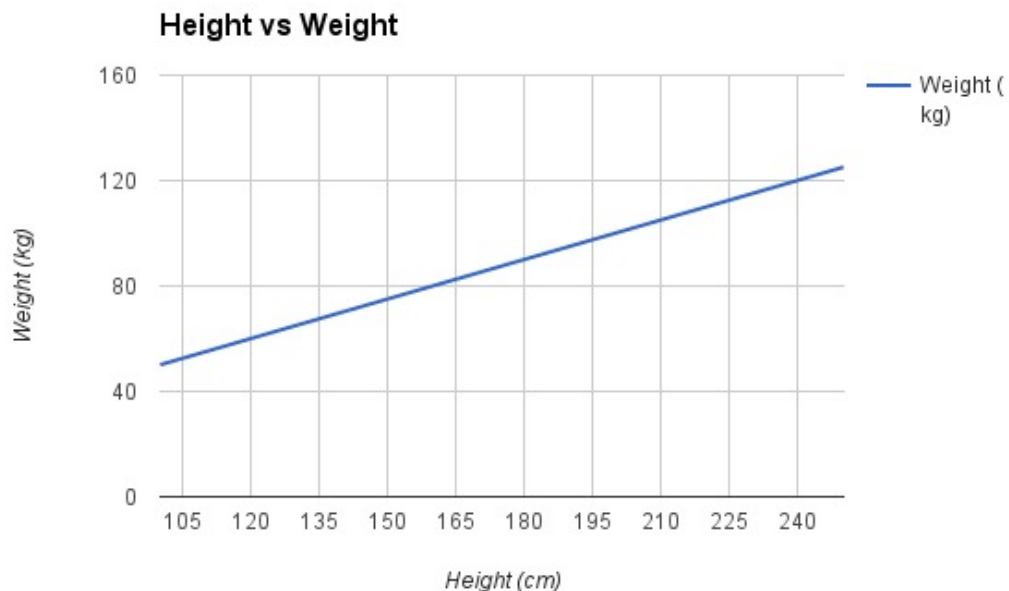
$$\mathbf{y} = \mathbf{a} + \mathbf{bX}$$

Where Y=output variable;

X=input variable;

a, b=coefficients.

For example, if a and b are 4 and 3, the value of x is 10; From the above formula, we get the value of Y as 34.



**Figure 4.2:** Graphical representation of linear regression

### 4.3 LOGISTIC REGRESSION MODEL

In statistics, logistic regression or logistic model or logit model is a model where the dependent variable is binary or categorized. In our case the output variable is binary dependent variable means the output will be binary like positive-negative, like-dislike, 0 or 1, good-bad, pass-fail, win-lose, alive-dead, etc.

Case where the output variable has more than two outcomes, it is known as multinomial regression model or multinomial logistic regression model. The binary logistic model is used for predicting binary outcomes based on one or more input variables or features. Logistic regression is widely used in machine learning, medical purposes and social science. Logistic regression may be used to predict the probability of a given disease. Logistic regression can be binomial, multinomial or ordinal. Binomial regression deals with model whose dependent variable has binary outcome 0 or 1 whereas multinomial

regression deals with a model whose dependent variable has multiple outcomes. Ordinal logistic regression is a model where dependent variables has an ordered outcome.

In logistic regression, we have one or two dependent variables which are predicted with the help of two more independent variables.

The logistic model is defined on the basis of standard logistic function. The logistic model is very helpful because it takes all real values  $t \in \mathbb{R}$  and provides an output which is between 0 and 1. The logistic function  $f(t)$  is as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

let us assume  $t$  is a linear function of a single explanatory variable  $x$  (the case where  $t$  is a linear combination of multiple explanatory variables is treated similarly) we can then express  $t$  as follows:

$$t = \beta_0 + \beta_1 x$$

and the logistic function can now be written as:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

# CHAPTER 5

## PROJECT DESCRIPTION

We have integrated machine learning algorithms with sentiment analysis. This helps us overcome the drawbacks of the old knowledge based approach. The knowledge based approach counts the number of positive and negative words and makes a prediction on the basis of their difference. In the following approach, we use logistic regression model to classify the data.

In this project we will be describing about the logistic regression model and we will use lasso regression for regularization. Logistic regression is another widely-used model when the response is categorical. If there are two possible outcomes, we use the binomial distribution, else we use the multinomial.

In simple terms, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. In our case, we will use it to determine if a text has a positive, negative, or neutral mood. Imagine for example that we apply it to entries in Twitter about the hashtag Windows10. We will be able to determine how people feels about the new version of Microsoft operating system. Of course this is not a big deal applied to an individual piece of text. However our model will show its benefits when automatically processing large amounts of text very quickly, or processing a large number of entries.

So in sentiment analysis process, there are a couple of stages more or less differentiated. The first is about natural language processing, and the second about training a model. The first stage is in charge of processing text in a way that, when the model is ready to be trained, the variables considered as the input for the model are known.

The model itself is in charge of learning how to determine the sentiment of a piece of text based on these variables. For the model part, logistic regression model is used. They are the most powerful methods in terms of accuracy and are simple enough to be interpreted in their results. Logistic regression model is a machine learning algorithm which gives categorical outcomes based on inputs like 0 or 1.

Finally we will need some data to train our model. For this we will use data from the



website [analyticsvidhya.com](https://analyticsvidhya.com). As described here, it is a sentiment classification task. Every document (a line in the data file) is a sentence extracted from social media. The files contain:

1. 1.6 million tweets
2. Training: 80% of the data is used for training the model, roughly 1.3 million tweets.
3. Testing: 20% of the data is used for testing the model. The model is validated using K-fold cross validation(5-fold).

## 5.1 Loading and Preparing Data

As usual, we will first download our datasets locally, and then we will load them into data frames in R. In R, we use `read.csv` to read CSV files into `data.frame` variables. Although the R function `read.csv` can work with URLs, `https` is a problem for R in many cases, so you need to use a package like `Rcurl` to get around it. Moreover, from the [analyticsvidhya](https://analyticsvidhya.com) page description we know that the file is tab-separated, there is not header, and we need to disable quoting since some sentences include quotes and that will stop file parsing at some point.

Now we have our data in data frames. We have 1.3 million tweets for the training data and 0.3 million tweets for the test data. The tweets are in a column named `Text` and the sentiment tag (just for training data) in a column named `Sentiment`. We get a glimpse at how tags are distributed. In R we can use `table` to see the tags. That is, we have data more or less evenly distributed.

## 5.2 Preparing a Corpus

In linguistics, a corpus or text corpus is a large and structured set of texts (nowadays usually electronically stored and processed). They are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules within a specific language territory. In our particular case, we are talking about the collection of text fragments that we want to classify in either positive or negative sentiment. Working

with text corpus involves using natural language processing techniques. R is capable of performing really powerful transformation with textual data. However doc2vec package has been used to perform the experiment. The doc2vec package is a very good package for text analysis which converts the text in a document into a vector or data frame. The words are weighted using term frequency-inverse document frequency in this approach. The simplification process mostly includes removing punctuation, lowercasing, removing stop-words, and reducing words to its lexical roots. The tweets are processed and a corpus is created. The important words are extracted and established as input variables for our classifier.

### 5.3 Model training

To training the model, `cv.glmnet` function is used, this function takes one variable `x` which is dtm tf-idf model. Other parameter is passed as `nfolds` which is k-fold cross validation and `nfolds` represent the value of `k`. Value of `alpha` defines what type of regression would be used in `glmnet` function as it provides mix of ridge and lasso regression, if `alpha = 1` means pure lasso and if `alpha = 0` means pure ridge regression. `type.measure` is used to minimize the error, Currently five options, not all available for all models. The default is `type.measure="deviance"`, which uses squared-error for gaussian models (a.k.a `type.measure="mse"` there), deviance for logistic and poisson regression, and partial-likelihood for the Cox model. `type.measure="class"` applies to binomial and multinomial logistic regression only, and gives misclassification error. `type.measure="auc"` is for two-class logistic regression only, and gives area under the ROC curve. `type.measure="mse"` or `type.measure="mae"` (mean absolute error) can be used by all models except the "cox"; they measure the deviation from the fitted mean to the response. `predict` function is used to predict the sentiment on testing data set. the final model is named as `glmnet_classifier`.

## 5.4 Fetching tweets

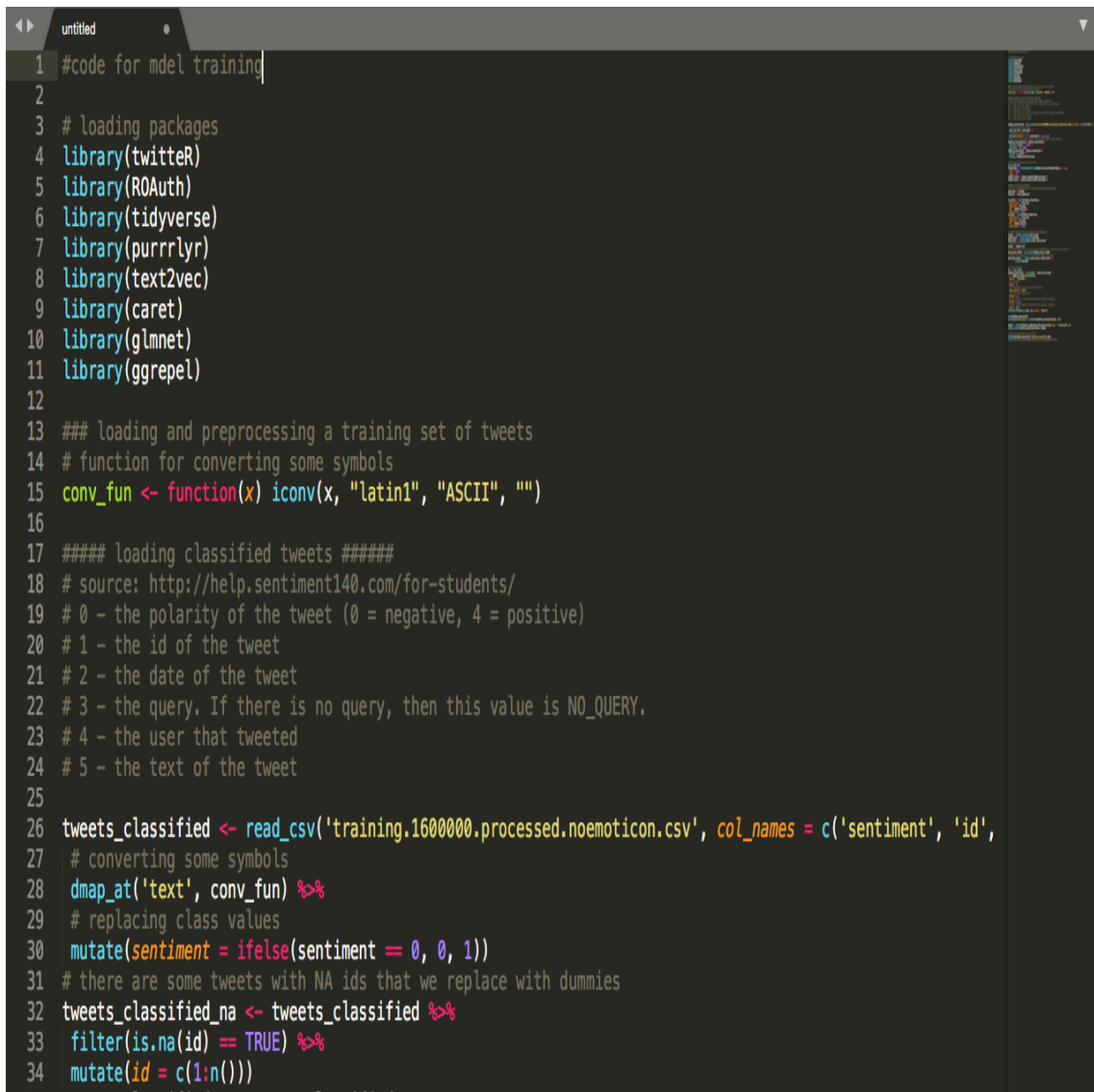
Next step is to fetch the tweet from twitter and process them to do prepare a corpus same as we did in the process to train the model. to fetch the tweet we need the twitter api access, for that we need to sign up on twitter developer portal and register for the api. We get four access credentials which are used to connect to twitter api. The function `searchTwitter` is used to fetch the api from twitter, this function takes one input term and the number of terms we want to query. The next step is to tokenize the tweet and then process them for removing unwanted terms. Finally we create a Document term matrix using the tokenizer. after this we need to fit this dtm to tf-idf matrix which is done with the help of function from `doc2vec` library named `fit_transform`. Then we predict the sentiment by using the `predict` function which takes three argument, first is the Trained model which we created earlier and second is the input matrix of the data on which we make prediction, the third is the response type which means what type of output we want. `type=response` means we want numerical value, finally we store the result by adding one more column to the existing data-frame, which is basically the probability value of the sentiment being positive or negative for the tweets that we fetched.

## 5.5 Plotting graph

This step is to show the graphical representation of the result that we got from the last step. To generate the graph we use a library called `ggplot` which is extremely famous for plotting graph in R. we first defined the color vector for differentiating between positive line and negative lines. `ggplot` function takes many argument according to the requirement of the graph, we have passed the color vector and then we pass the separator to label the y axis with four value form 0 to 1. `geom_hline` argument is passed to create the horizontal lines to differentiate between positive and negative tweets. `ggtitle` argument put the main heading label of the graph.

## CHAPTER 6

### IMPLEMENTATION OF THE PROJECT



```
1 #code for mdel training
2
3 # loading packages
4 library(twitter)
5 library(ROAuth)
6 library(tidyverse)
7 library(purrrlyr)
8 library(text2vec)
9 library(caret)
10 library(glmnet)
11 library(ggrepel)
12
13 ### loading and preprocessing a training set of tweets
14 # function for converting some symbols
15 conv_fun <- function(x) iconv(x, "latin1", "ASCII", "")
16
17 ##### loading classified tweets #####
18 # source: http://help.sentiment140.com/for-students/
19 # 0 - the polarity of the tweet (0 = negative, 4 = positive)
20 # 1 - the id of the tweet
21 # 2 - the date of the tweet
22 # 3 - the query. If there is no query, then this value is NO_QUERY.
23 # 4 - the user that tweeted
24 # 5 - the text of the tweet
25
26 tweets_classified <- read_csv('training.1600000.processed.noemoticon.csv', col_names = c('sentiment', 'id',
27 # converting some symbols
28   dmap_at('text', conv_fun) %>%
29   # replacing class values
30   mutate(sentiment = ifelse(sentiment == 0, 0, 1))
31 # there are some tweets with NA ids that we replace with dummies
32 tweets_classified_na <- tweets_classified %>%
33   filter(is.na(id) == TRUE) %>%
34   mutate(id = c(1:n()))
```

**Figure 6.1:** code snippet for loading data set and pre-processing

Refer to figure 5.1 is the code for initial loading of different library required by R to perform different operations on data. the con\_fun is used to convert all the latin character present in the tweets to ascii. the read\_csv function is used to load the csv file which contains the 1.6m tweets, and the mutate function is used convert the sentiment value of each column to either 0 or 1 or because in current data set tweets are classified, 0 as

negative and 4 as positive , but in our algorithm we want 0 as negative and 1 as positive.

```
32 tweets_classified_na <- tweets_classified %>%
33   filter(is.na(id) == TRUE) %>%
34   mutate(id = c(1:n()))
35 tweets_classified <- tweets_classified %>%
36   filter(!is.na(id)) %>%
37   rbind(., tweets_classified_na)
38
39 # data splitting on train and test
40 set.seed(2340)
41 trainIndex <- createDataPartition(tweets_classified$sentiment, p = 0.8,
42   list = FALSE,
43   times = 1)
44 tweets_train <- tweets_classified[trainIndex, ]
45 tweets_test <- tweets_classified[-trainIndex, ]
46
47 ##### Vectorization #####
48 # define preprocessing function and tokenization function
49 prep_fun <- tolower
50 tok_fun <- word_tokenizer
51
52 it_train <- itoken(tweets_train$text,
53   preprocessor = prep_fun,
54   tokenizer = tok_fun,
55   ids = tweets_train$id,
56   progressbar = TRUE)
57 it_test <- itoken(tweets_test$text,
58   preprocessor = prep_fun,
59   tokenizer = tok_fun,
60   ids = tweets_test$id,
61   progressbar = TRUE)
62
63 # creating vocabulary and document-term matrix
64 vocab <- create_vocabulary(it_train)
```

**Figure 6.2:** Code Snippet For Tokenizing the tweet

Refer to figure 5.2 shows how the data partition in two data set, one is training data set and another as testing data set. 80% of the data is training set and 20% is testing set. itoken function is used to tokenize all the tweets for training set and testing set. it\_train represent tokenizer for training data set and it\_test represent the tokenizer for testing data set.

```

65 vectorizer <- vocab_vectorizer(vocab)
66 dtm_train <- create_dtm(it_train, vectorizer)
67 # define tf-idf model
68 tfidf <- Tfidf$new()
69 # fit the model to the train data and transform it with the fitted model
70 dtm_train_tfidf <- fit_transform(dtm_train, tfidf)
71 # apply pre-trained tf-idf transformation to test data
72 dtm_test_tfidf <- create_dtm(it_test, vectorizer) %>%
73   transform(tfidf)
74
75 # train the model
76 t1 <- Sys.time()
77 glmnet_classifier <- cv.glmnet(x = dtm_train_tfidf,
78   y = tweets_train[['sentiment']],
79   family = 'binomial',
80   # L1 penalty
81   alpha = 1,
82   # interested in the area under ROC curve
83   type.measure = "auc",
84   # 5-fold cross-validation
85   nfolds = 5,
86   # high value is less accurate, but has faster training
87   thresh = 1e-3,
88   # again lower number of iterations for faster training
89   maxit = 1e3)
90 print(difftime(Sys.time(), t1, units = 'mins'))
91
92 plot(glmnet_classifier)
93 print(paste("max AUC =", round(max(glmnet_classifier$cvm), 4)))
94
95 preds <- predict(glmnet_classifier, dtm_test_tfidf, type = 'response')[,1]

```

**Figure 6.3:** code snippet for creating Document Term Matrix

Refer to figure 5.3 shows how the document term matrix is formed from the tokenizer and vectorizer, vectorizer is made from the vocabulary and vocabulary is made from from the tokenizer alone. vocabulary is basically is a collection of words.

```

76 t1 <- Sys.time()
77 glmnet_classifier <- cv.glmnet(x = dtm_train_tfidf,
78 y = tweets_train[['sentiment']],
79 family = 'binomial',
80 # L1 penalty
81 alpha = 1,
82 # interested in the area under ROC curve
83 type.measure = "auc",
84 # 5-fold cross-validation
85 nfolds = 5,
86 # high value is less accurate, but has faster training
87 thresh = 1e-3,
88 # again lower number of iterations for faster training
89 maxit = 1e3)
90 print(difftime(Sys.time(), t1, units = 'mins'))
91
92 plot(glmnet_classifier)
93 print(paste("max AUC =", round(max(glmnet_classifier$cvm), 4)))
94
95 preds <- predict(glmnet_classifier, dtm_test_tfidf, type = 'response')[,1]
96 auc(as.numeric(tweets_test$sentiment), preds)
97
98 # save the model for future using
99 saveRDS(glmnet_classifier, 'glmnet_classifier.RDS')
100 #####
101

```

**Figure 6.4:** code snippet for training the model and prediction

Refer to figure 5.4 shows the training of the model, `cv.glmnet` function is used to train the model, this function takes one variable `x` which is `dtm` tf-idf model. Other parameter is passed as `nfolds` which is k-fold cross validation and `nfolds` represent the value of `k`. Value of `alpha` defines what type of regression would be used in `glmnet` function as it provides mix of ridge and lasso regression, if `alpha = 1` means pure lasso and if `alpha = 0` means pure ridge regression. `type.measure` is used to minimize the error, Currently five options, not all available for all models. The default is `type.measure = "deviance"`, which uses squared-error for gaussian models (a.k.a `type.measure = "mse"` there), deviance for logistic and poisson regression, and partial-likelihood for the Cox model. `type.measure = "class"` applies to binomial and multinomial logistic regression only, and gives misclassification error. `type.measure = "auc"` is for two-class logistic regression

only, and gives area under the ROC curve. `type.measure="mse"` or `type.measure="mae"` (mean absolute error) can be used by all models except the "cox"; they measure the deviation from the fitted mean to the response. `predict` function is used to predict the sentiment on testing data set. the final model is named as `glmnet_classifier`.

```

1  ### fetching tweets ###
2  download.file(url = "http://curl.haxx.se/ca/cacert.pem",
3  destfile = "cacert.pem")
4  setup_twitter_oauth('8iPM25VsTV1nGGh9KRgyftTq1', # api key
5  '0F7svBTg0g4pPg04gkC1f3Fei9L5puox92NtW2MnUdPSM5wJmQ', # api secret
6  '797372905940078592-ylKsE17KvRjLV1L205tjYt3HgRq5woJ', # access token
7  'Z1EJGVDlTWhg8zuXLSx8bY7m05Nc105RaRm8grL0sqLSp' # access token secret
8  )
9
10 df_tweets <- twListToDF(searchTwitter('#bleedblue', n = 150, lang = 'en')) %>%
11 # converting some symbols
12 dmap_at('text', conv_fun)
13
14 # preprocessing and tokenization
15 it_tweets <- itoken(df_tweets$text,
16 preprocessor = prep_fun,
17 tokenizer = tok_fun,
18 ids = df_tweets$id,
19 progressbar = TRUE)
20
21 # creating vocabulary and document-term matrix
22 dtm_tweets <- create_dtm(it_tweets, vectorizer)
23
24 # transforming data with tf-idf
25 dtm_tweets_tfidf <- fit_transform(dtm_tweets, tfidf)
26
27 # loading classification model
28 glmnet_classifier <- readRDS('glmnet_classifier.RDS')
29
30 # predict probabilities of positiveness
31 preds_tweets <- predict(glmnet_classifier, dtm_tweets_tfidf, type = 'response')[,1]
32
33 # adding rates to initial dataset
34 df_tweets$sentiment <- preds_tweets

```

**Figure 6.5:** code snippet for fetching tweet

Refer to figure 5.5 shows the process of fetching the tweets from tweeter and processing them to convert into a dataframe and then making the tokenizer. final prediction is done on the fetched tweet from tweeter by using the same model we trained earlier with function named `predict`, finally the `preds_tweets` contains the predicted sentiment for the tweets fetched from twitter.



```

1
2 # color palette
3 cols <- c("#ce472e", "#f05336", "#ffd73e", "#e673a", "#4ab04a")
4
5 set.seed(932)
6 samp_ind <- sample(c(1:nrow(df_tweets)), nrow(df_tweets) * 0.1) # 10% for labeling
7
8 # plotting
9 ggplot(df_tweets, aes(x = created, y = sentiment, color = sentiment)) +
10 theme_minimal() +
11 scale_color_gradientn(colors = cols, limits = c(0, 1),
12 breaks = seq(0, 1, by = 1/4),
13 labels = c("0", round(1/4*1, 1), round(1/4*2, 1), round(1/4*3, 1), round(1/4*4, 1)),
14 guide = guide_colourbar(ticks = T, nbin = 50, barheight = .5, label = T, barwidth = 10)) +
15 geom_point(aes(color = sentiment), alpha = 0.8) +
16 geom_hline(yintercept = 0.65, color = "#4ab04a", size = 1.5, alpha = 0.6, linetype = "longdash") +
17 geom_hline(yintercept = 0.35, color = "#f05336", size = 1.5, alpha = 0.6, linetype = "longdash") +
18 geom_smooth(size = 1.2, alpha = 0.2) +
19 geom_label_repel(data = df_tweets[samp_ind, ],
20 aes(label = round(sentiment, 2)),
21 fontface = 'bold',
22 size = 2.5,
23 max.iter = 100) +
24 theme(legend.position = 'bottom',
25 legend.direction = "horizontal",
26 panel.grid.major = element_blank(),
27 panel.grid.minor = element_blank(),
28 plot.title = element_text(size = 20, face = "bold", vjust = 2, color = 'black', lineheight = 0.8),
29 axis.title.x = element_text(size = 16),
30 axis.title.y = element_text(size = 16),
31 axis.text.y = element_text(size = 8, face = "bold", color = 'black'),
32 axis.text.x = element_text(size = 8, face = "bold", color = 'black')) +
33 ggtitle("Tweets Sentiment rate (probability of positiveness)")
34

```

**Figure 6.6:** code snippet for showing result

Refer to figure 5.6 shows the making of the final graph using the library function ggplot which provided a rich experience to shows graph by passing different colors and label for each axis and defining the fonts for titles.

# CHAPTER 7

## PACKAGES USED IN R

### 7.1 **twitterR**

Twitter is a R package used to get access to the twitter API. It loads all the functionality of twitter API. It is biased towards API calls so as to do better data analysis as compared to daily representation.

Some of the important functions of this package are:

1. Retweets-Function to work with retweets.
2. taskStatus- A function to send a twitter DM after completion of a task.
3. timelines-Function to view twitter timelines.
4. `get_latest_tweet_id`-a function to retrieve recent tweet id from a database
5. `search_twitter_and_store`-A function to store searched tweets to a database

### 7.2 **TIDYVERSE**

Tidyverse is a set of packages which work in harmony because they need to share a lot data representations and API design. This package is designed for ease of the user as it installs more than one required packages in a single step.

Some of the important functions of this package are:

1. `Tidyverse_conflicts`: Conflicts between tidyverse and other packages.
2. `Tidyverse_packages`: List all the packages in tidyverse.
3. `Tidyverse_update`: Update tidyverse packages.

## 7.3 PURRRLYR

Purrrlyr is a R package that lies between purrr package and dplyr package. It was removed from the purrr package to make it lighter and it has been replaced by many solutions in tidyverse.

Some of the important functions of this package are:

1. `Slice_rows`- slices a data frame into groups of two rows
2. `Dmap`-it maps the columns of a data frame.`%>%`-it is the pipe operator.
3. `By_row`- applies a function to each row of the data frame.
4. `By_slices`-applies a function to the slices of a data frame.

## 7.4 TEXT2VEC

Text2vec is a R package which is a modern text mining framework for R. Quick and memory-accommodating tools for content vectorization, theme displaying (LDA, LSA), word embeddings (GloVe), similitudes. This bundle gives a source-freethinker spilling API, which enables scientists to perform examination of accumulations of records which are bigger than accessible RAM. All center capacities are parallelized to profit by multicore machines.

Text2vec is an R package which provides a very easy and efficient framework with a concise API for text analysis, text mining and natural language processing.

Objectives which we meant to accomplish because of improvement of text2vec:

1. Compact - uncover as few functions as could be expected under the circumstances
2. Predictable - uncover bound together interfaces, no compelling reason to investigate new interface for each assignment

3. Adaptable - permit to effectively understand complex assignments
4. Quick - expand productivity per single string, straightforwardly scale to various strings on multicore machines
5. Memory productive - utilize streams and iterators, not keep information in RAM if conceivable

The center functionality right now incorporates:

1. Quick content vectorization on self-assertive n-grams, utilizing vocabulary or highlight hashing.
2. GloVe word embeddings
3. Topic displaying with:
  - (a) Latent Dirichlet Allocation
  - (b) Latent Semantic Analysis
4. Likenesses/separates between 2 frameworks
  - (a) Cosine
  - (b) Jaccard
  - (c) Loose Word Mover's Distance
  - (d) Euclidean

This package is effective in light of the fact that it is precisely composed in C++, which additionally implies that text2vec is memory friendly. A few sections, (for example, GloVe) are completely parallelized utilizing the fantastic RcppParallel package. This implies the word embeddings are processed in parallel on OS X, Linux, Windows, and even Solaris (x86) with no extra tuning or traps.

Other emarrassingly parallel errands, (for example, vectorization) can utilize any parallel backend which bolsters foreach bundle. They can accomplish close direct versatility with number of accessible centers.

At last, a streaming API implies that clients don't need to stack every one of the information into RAM.

Some of the important functions of this package are:

1. `Check_analogy_accuracy`-checks accuracy of word embeddings on the analogy task

2. `Create_dtm`-Document-term matrix construction
3. `LatentSemanticAnalysis`-Latent Semantic Analysis model
4. `RelaxedWordMoversDistance`-Creates model which can be used for calculation of relaxed word movers distance.
5. `Perplexity`-perplexity of a topic model
6. `Prepare_analogy_questions`-prepares the list of analogy questions.
7. `Similarities`-computes pairwise similarity matrix.

# CHAPTER 8

## EXPERIMENTATION AND RESULTS

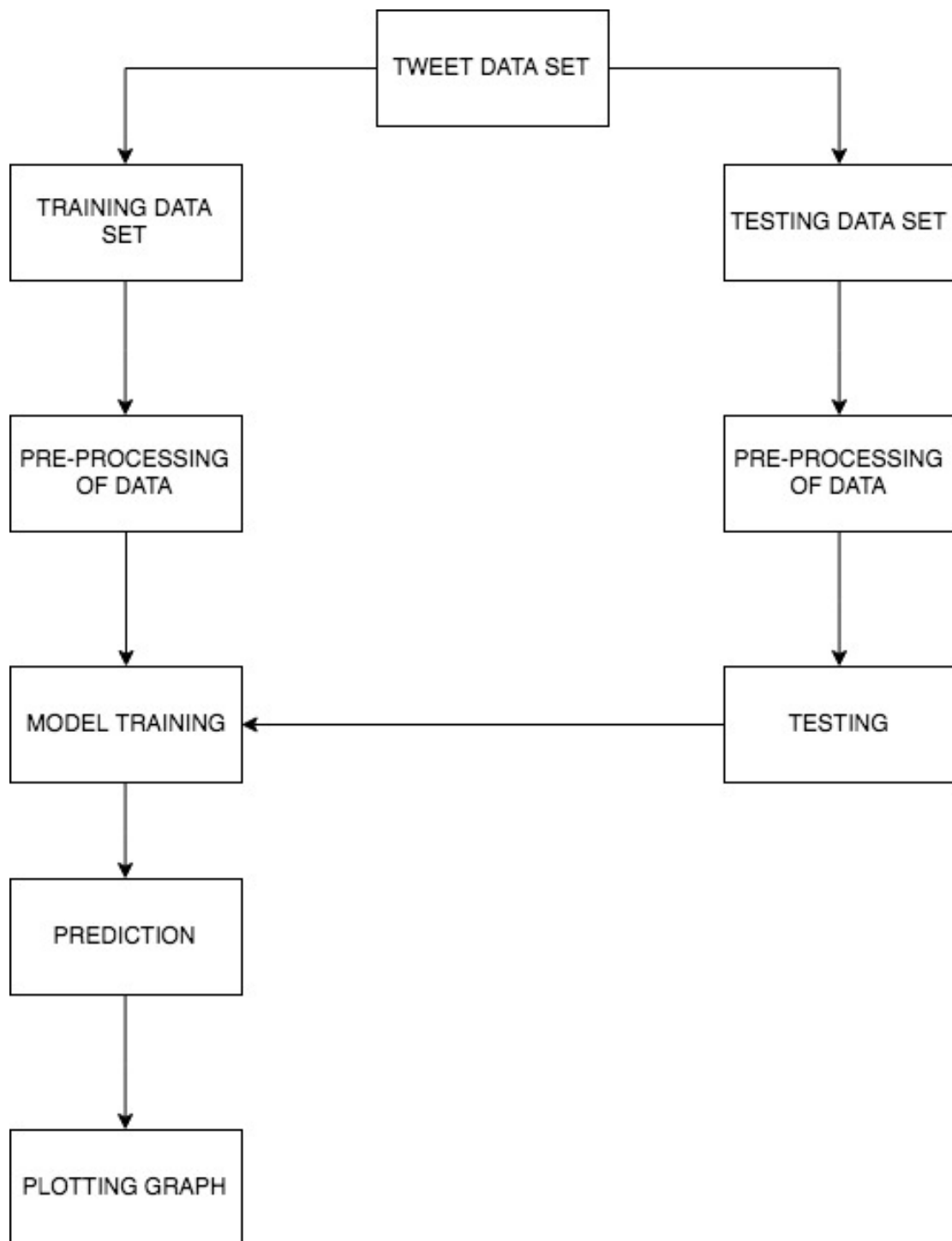
In our project, we have made an improvement from the last approaches towards sentiment analysis. In this project, we have been helped by two packages from R. They are doc2vec and text2vec. These packages convert a sentence or document into a vector of words. These are mainly used for text analysis and we have used them for sentiment analysis.

We had to move from the old approach as it had some drawbacks. The old approach used to make conclusion by finding the difference between the number of positive words and negative words. Also phrase level analysis could not be done in the old approach. Sentences like ‘not bad’ had negative conclusion irrespective of actual sentiment.

In the project, firstly we use a dataset of words to train the model. The dataset consists of words from 1.6 million tweets. We have taken a large number of tweets so that approximation and correctness of prediction is very high.

### 8.1 STEPS FOLLOWED

1. We shall use tweets for our project because they are easy to get them from Twitter API. We create an app on <https://dev.twitter.com> (My apps menu) and find an API Key, API secret, Access Token and Access Token Secret on Keys and Access Tokens menu tab.
2. We download 1.6 million tweets from a dataset. Doc2vec package from R helps us to convert the document to vector of words for context analysis.
3. We have classified the tweets as negative to positive (0 to 1). There is no neutral tweet as we are talking about the probability of positiveness of the tweet. If the probability is between 0.35 and 0.70 we consider them as the neutral tweets.
4. We use 80 % of the above dataset to train the model. We use R code to train the model using DTM Document Term Matrix which is a part of Vocabulary Based vectorization.
5. We use tf-idf approach for text pre-processing. Tf-Idf is a function known as term frequency- inverse domain frequency. Training the model can take more than one



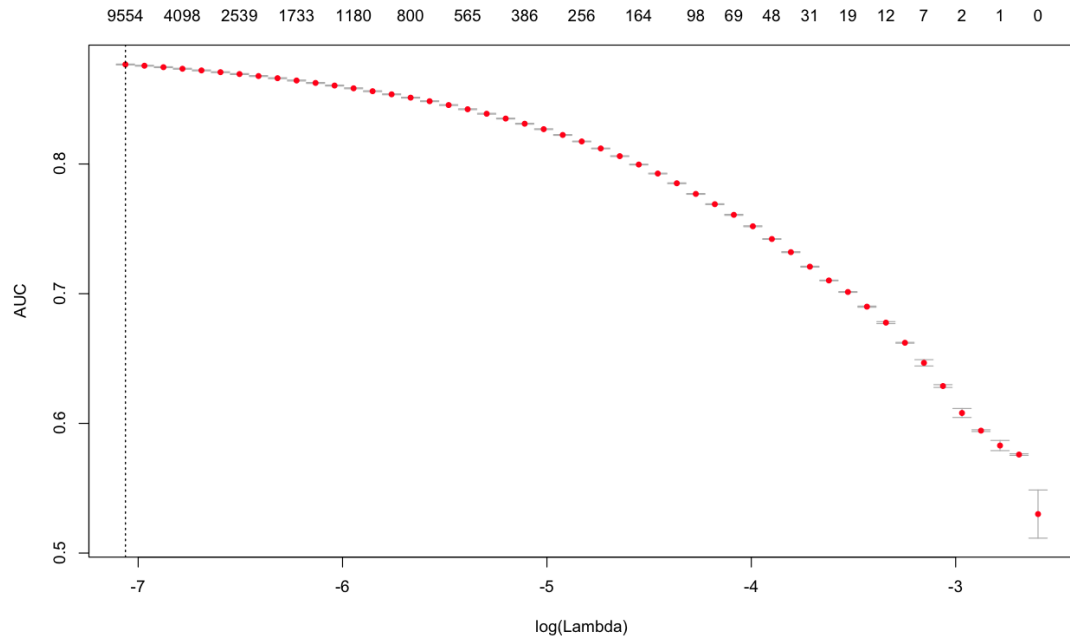
**Figure 8.1:** Architecture Diagram

hour because it depends on the configuration of the system the project is being done. For our convenience we have trained the model before itself.

6. Before predicting the output, we need to cross validate the model whether it is predicting the correct output. We use K-fold cross validation technique to validate the model.
7. We check AUC (are under the curve) for the correctness of the dataset. In our

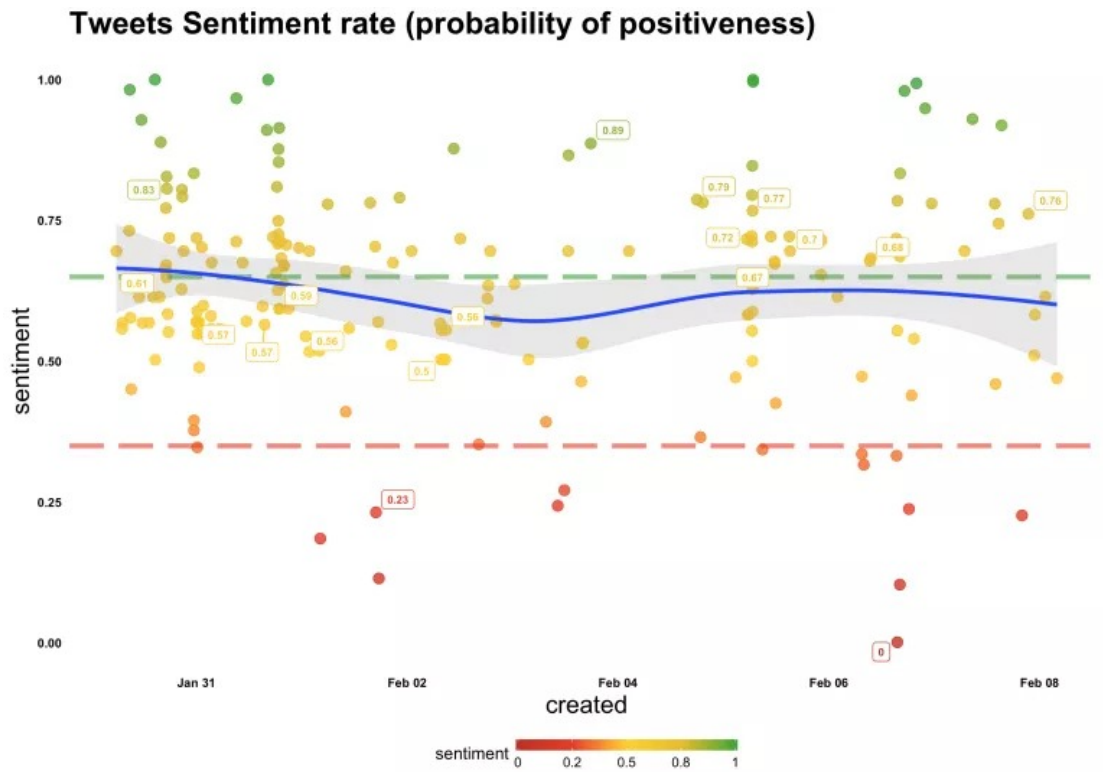
dataset we have 0.876 and 0.875 auc under the train and test dataset.

8. After training and validating the model, we repeat the same process by fetching tweets from the twitter API. The tweets are now pre-processed using the above method and classified according to the algorithm.



**Figure 8.2:** figure showing AUC versus Lambda





**Figure 8.3:** probability of positiveness

Table 8.1: Table showing auc on train data and test data

Value 1	Value 2
data	auc
trainig data	0.875
testing data	0.874

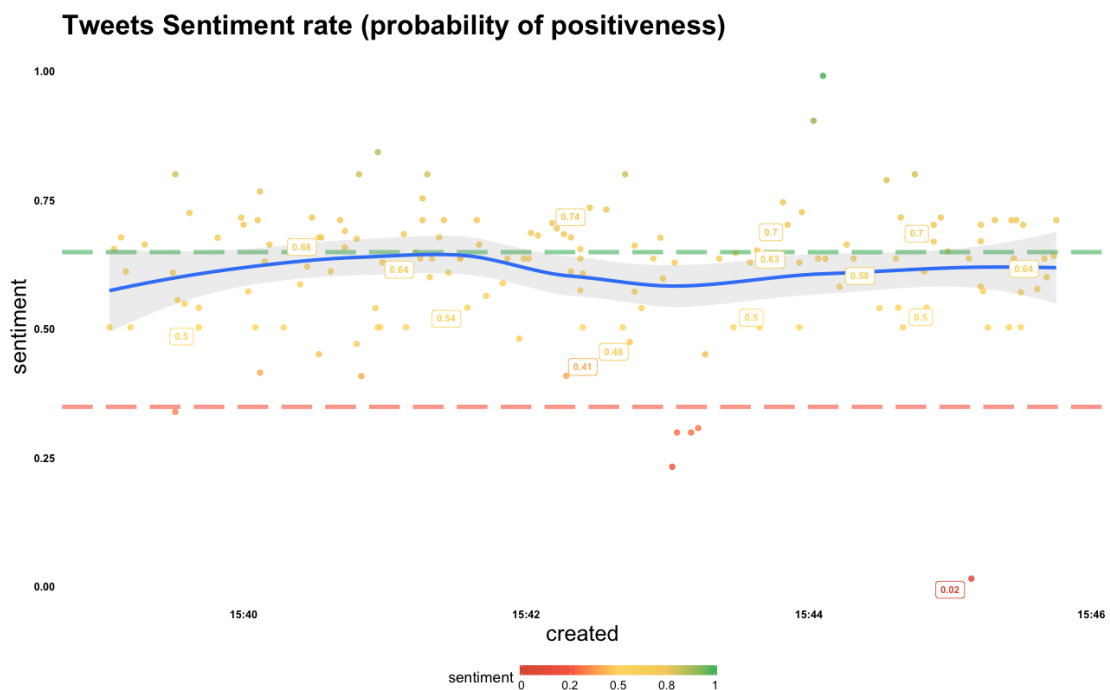
## 8.2 RESULT

In the above graph, the red line depicts the negative boundary of tweets and the green line depicts the positive boundary of tweets, and blue line depicts the logistic curve which almost cover all the tweets. As we can see most of the tweets are on the positive side, we can term the set of tweets as positive.

### 8.2.1 RESULT FOR #justiceForAsifa

In the case study, we are carrying out sentiment analysis on a very sensitive topic. Few days back, a 8 year old girl was brutally gang raped and murdered in kathua, a small village in Kashmir, India. The event took a very ugly turn as it was turned into a political agenda. Many people from the political world supported the rapists and divided the topic on the basis of religion and communal rights.

the graph given below shows the sentiment of people via their tweets with the hashtag #JusticeForAsifa.

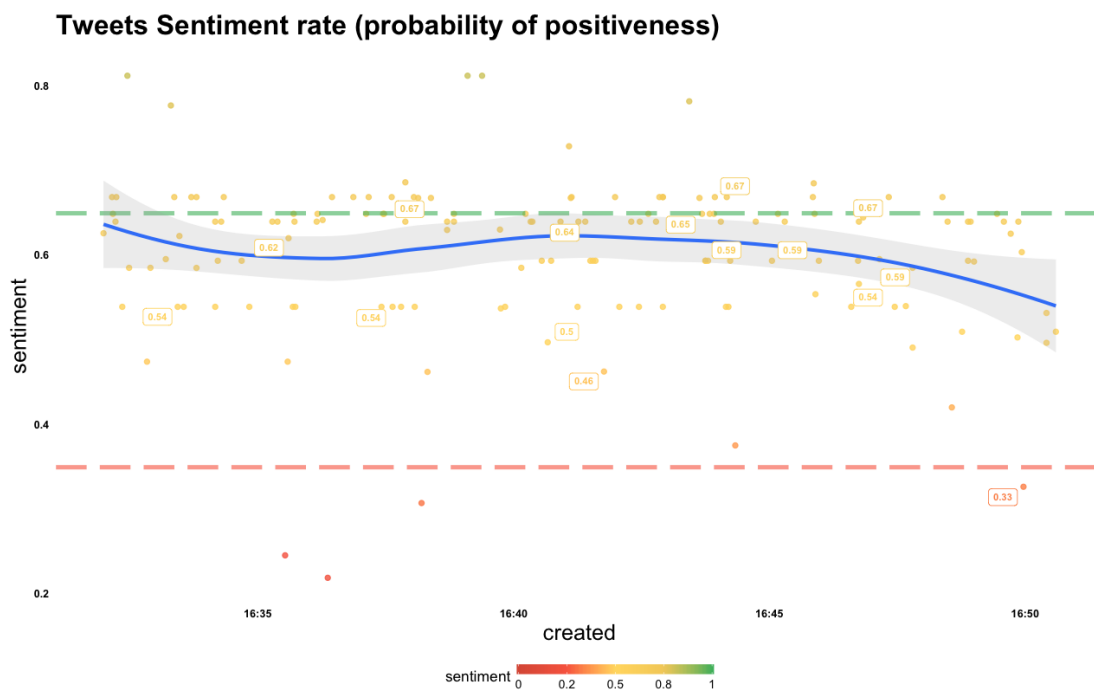


**Figure 8.4:** image showing probability of positiveness for #justiceForAsifa

## 8.2.2 RESULT FOR #unnao

The Unnao case refers to the alleged act to a 17 year old girl on 4 June 2017. The Unnao case and the Kathua case received national attention during the same period, leading to joint protests seeking justice for both victims.

the graph given below shows the the probability of sentiment being positive which people being expressed through twitter.



**Figure 8.5:** image showing probability of positiveness for unnao

from the graph we can say people around the nation has been showing signs of positivity about the girl.

## **CHAPTER 9**

### **CONCLUSION AND FUTURE ENHANCEMENT**

In our project we have performed sentiment analysis on twitter data. we have concluded that sentiment analysis can be done on product reviews and hospital reviews. we have explored the sentiment analysis on twitter and it can be also done on different type of text and documents like product launch in a company or service provided to customer, we can get valuable feedback with sentiment analysis. During this project we learned and explored the area of sentiment analysis and had a good knowledge of machine learning.

The period of getting important bits of knowledge from online networking information has now touched base with the progress in innovation. The contextual investigation gives you a look at the energy of Contextual Semantic Search. It's the ideal opportunity for your association to move past general conclusion and tally based measurements.

Organizations have been utilizing the energy of information of late, however to get the most profound of the data, you need to use the energy of AI, Deep learning and savvy classifiers like Contextual Semantic Search and Sentiment Analysis.

This project offers element based opinion investigation and in addition angle based (or highlight based) assessment examination. At the substance level, this project recognizes conclusions toward different kinds of elements, for example, individuals, associations, brands, and items, opinions expressed on twitter.

At the perspective level, the project gives better grained assessment examination coordinated toward various parts of elements, for instance, the cost of an item, the new arrangement of a nation, the battle of a presidential hopeful, and so on.

Along these lines, we pinpoint what the conclusion is and correctly what the opinion is tied in with, making assessment investigation significantly more useful and valuable to its clients. We will explore even richer linguistic analysis, for example, parsing, seman-

tic analysis and topic modeling.

Until now we have been dealing with a large amount of text and data but people usually express different sentiment with the same set of words, current model can not predict the exact emotion.

People belong to different geographical location around the globe and people tend to speak in their native language, we are just processing English language. In future we plan to include regional language in our model. In future we plan to consider the emotion which also contribute to the overall sentiment. This project shows that we cannot depend on sentiment analysis as it is upto 90% accurate. Now, different feature based analysis is being performed. More possible machine learning models can be made which represent twitter data in a better way.

This is important because we pre-process the data and delete a lot of information. We need to develop good techniques to pre-process the data, and we will try to incorporate the data from other social media channel and use them to model different topic and get the more accurate result.

There are some areas where with more research, this project can be improved to give a better accuracy and efficiency. Some of those points are as follows:-

1. With more research and resources, one can find an algorithm to combine the results of both Sentimental Analysis as well as Machine Learning Algorithms into one single algorithm which will give accurate results for the sentiments about the user opinion about the product.
2. Improve the accuracy of the prediction of machine learning algorithm such that the results predicted for the future (sentiments and opinions) are very similar to the obtained data.

## REFERENCES

1. Bagheri, H. and Islam, M. J. (2017). “Twitter sentiment analysis.
2. Baqapuri, A. I. (2015). “Twitter sentiment analysis.” *arXiv preprint arXiv:1509.04219*.
3. Bollen, J., Mao, H., and Zeng, X. (2011). “Twitter mood predicts the stock market.” *Journal of computational science*, 2(1), 1–8.
4. Ghiassi, M., Skinner, J., and Zimbra, D. (2013). “Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network.” *Expert Systems with applications*, 40(16), 6266–6282.
5. Gokulakrishnan, B., Priyathan, P., Ragavan, T., Prasath, N., and Perera, A. (2012). “Opinion mining and sentiment analysis on a twitter data stream.” *Advances in ICT for emerging regions (ICTer), 2012 International Conference on*, IEEE, 182–188.
6. Huang, L. and Bhayani, R. (2009). “Twitter sentiment analysis.
7. Jose, A. K., Bhatia, N., and Krishna, S. (2010). “Twitter sentiment analysis.” *Seminar Report, National Institute of Technology Calicut*.
8. Kanakaraj, M. and Guddeti, R. M. R. (2015). “Performance analysis of ensemble methods on twitter sentiment analysis using nlp techniques.” *Semantic Computing (ICSC), 2015 IEEE International Conference on*, IEEE, 169–170.
9. Khan, A. Z., Atique, M., and Thakare, V. (2015). “Combining lexicon-based and learning-based methods for twitter sentiment analysis.” *International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCE)*, 89.
10. Kouloumpis, E., Wilson, T., and Moore, J. D. (2011). “Twitter sentiment analysis: The good the bad and the omg!” *Icwsn*, 11(538-541), 164.
11. Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., and Stoyanov, V. (2016). “Semeval-2016 task 4: Sentiment analysis in twitter.” *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 1–18.
12. Neethu, M. and Rajasree, R. (2013). “Sentiment analysis in twitter using machine learning techniques.” *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on*, IEEE, 1–5.
13. Saif, H., He, Y., and Alani, H. (2012a). “Alleviating data sparsity for twitter sentiment analysis.” *CEUR Workshop Proceedings (CEUR-WS. org)*.
14. Saif, H., He, Y., and Alani, H. (2012b). “Semantic sentiment analysis of twitter.” *International semantic web conference*, Springer, 508–524.
15. Taboada, M., Brooke, J., Tofiloski, M., Voll, K., and Stede, M. (2011). “Lexicon-based methods for sentiment analysis.” *Computational linguistics*, 37(2), 267–307.

16. Tang, D., Qin, B., and Liu, T. (2015). "Deep learning for sentiment analysis: successful approaches and future challenges." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(6), 292–303.
17. Tatum, J. and Sanchez, J. T. (2013). "Twitter sentiment analysis." *CS29 Machine Learning course at Stanford University*.
18. Vinodhini, G. and Chandrasekaran, R. (2012). "Sentiment analysis and opinion mining: a survey." *International Journal*, 2(6), 282–292.
19. Wang, H., Can, D., Kazemzadeh, A., Bar, F., and Narayanan, S. (2012). "A system for real-time twitter sentiment analysis of 2012 us presidential election cycle." *Proceedings of the ACL 2012 System Demonstrations*, Association for Computational Linguistics, 115–120.

Kouloumpis et al. (2011) Jose et al. (2010) Neethu and Rajasree (2013) Taboada et al. (2011) Vinodhini and Chandrasekaran (2012) Kanakaraj and Guddeti (2015) Bollen et al. (2011) Saif et al. (2012b) Khan et al. (2015) Tang et al. (2015) Gokulakrishnan et al. (2012) Gokulakrishnan et al. (2012) Wang et al. (2012) Bagheri and Islam (2017) ? Huang and Bhayani (2009) Saif et al. (2012a) Tatum and Sanchez (2013) Baqapuri (2015) Nakov et al. (2016) Ghiassi et al. (2013)