MSc In Information Technology

Customer Satisfaction Monitoring with Sentiment Analysis Based on Twitter Feeds in Telecom Domain

Prepared by

A.P.L.D. Sachith Chamara

(158752P)

Supervised by

Mr. S.C. Premaratne

Faculty of Information Technology

University of Moratuwa

2018

MSc In Information Technology

Customer Satisfaction Monitoring with Sentiment Analysis Based on Twitter Feeds in Telecom Domain

Prepared by

A.P.L.D. Sachith Chamara

(158752P)

Supervised by

Mr. S.C. Premaratne

Faculty of Information Technology

University of Moratuwa

2018

Declaration

I declare that this thesis/dissertation does not incorporate without acknowledgementany material, previously submitted for a Degree or Diploma in any University or otherinstitute of higher learning and to the best of my knowledge and belief it does notcontain any material previously published or written by another person except where the acknowledgement is made in the text.

A.P.L.D. Sachith Chamara
(158752P)
2018/05

I have supervised and accepted this thesis for the submission of the degree.

Mr. S.C. Premaratne
(Main Supervisor)
2018/05

Acknowledgements

I would like to take this opportunity to thank my supervisor, Mr. S.C. Premaratne for all his dedicated support and guidance throughout this research. His guidance and inspiration are the key factors that enabled the successful completion of this research.
I would like to express my sincere thanks to Mr. Ian Ramsden (F IDM) for providing an excellent introduction to this research area.His valuable discussions and comments were also effective stimuli to this research work.
My deepest thanks go to my family for their assistance, encouragement and unwavering belief in me. Finally, I would like to thank all the staff of University of Moratuwa who helped me directly or indirectly to carry out this research work.

Abstract

With this increased competition among telecom service providers, it has become more difficult to retain the existing customers, but when the number of customers reaches its peak, finding and securing new customers become increasingly difficult and costly. Therefore, it would be better to prioritize the retention of the existing customers, than trying to win new ones.

Customer reviews can be recognized as fruitful information sources for monitoring and enhancing customer satisfaction levels as they convey the real voices of actual customers expressing relatively unambiguous opinions.

This research is aimed at mining and measure customer satisfaction toward Telecom Service based on reviews and feedbacks from Twitter. This research is mainly focus on one of the largest mobile operator in Sri Lanka and the analysis has been done only for English language.

Tweets were classified into three classes as Positive, Negative and Neutral with the use of four dictionaries (Lexicon, SentiWordNet, Slangs& Emoticons). The framework was built based on six steps and it shows that Lexicon performs well on the dataset better than SentiWordNet. After fine-tuning lexicon and stop words dictionary and integrating with Slangs dictionary, positive classification shows 91.98% accuracy without Emoticon dictionary while for negative classification, the accuracy is 82.27% with Emoticons dictionary.

Keywords: Twitter Feeds,Telecom Industry, Sentiment Analysis, Lexicon

Table ofContents

List of Figures

List of Tables

Chapter 1

1Introduction

1.1 Prolegomena

The main purpose of this research is to design and develop a model for customer satisfaction of Telecom Service based on reviews and feedbacks from Twitter. In addition to this purpose, this chapter provides all the other objectives of this research work and then briefly explains the background and the motivation factors to this research. It also defines the problem statement and the research purpose.

The aim of this research is to design and develop a framework for monitoring customer satisfaction of Telecom Service based on reviews and feedbacks from social media (Only Twitter feeds written in English will be considered in this research work). This model will do a sentiment analysis and will come up with a novel approach based on Lexicon based approach.

1.2Problem Statement

When the number of customers reaches its peak, finding and securing new customers become increasingly difficult and costly. Therefore, it would be better to prioritize the retention of the existing customers, than trying to win new ones. With the rapid growth of mobile services, enhancement of customer satisfaction has become as a core issue. Customer reviews can be recognized as fruitful information sources for monitoring and enhancing customer satisfaction levels as they convey the real voices of actual customers expressing relatively unambiguous opinions.

Therefore, developing a model, which can be used to measure the customer satisfaction, would be very important for the telecom industry especially for their survival. In addition to that, by analyzing the satisfaction, the telecom operator would be able to identify which services needs to be improved further in order to satisfy the existing customer.

1.3Aims &Objectives

This research is aimed at mining and measure customer satisfaction toward Telecom Service based on reviews and feedbacks from Twitter. This research is mainly focus on one of the largest mobile operator in Sri Lanka and the analysis has been done only for English language.The main objectives of this research work have been given below.

1. Review of the existing customer satisfaction monitoring frameworkbased

onanalyzing reviews and feedbacks from Twitter.

2. In-depth study of technologies used for sentimental analysis.
3. Design and development a framework for a novel approach for sentimental analysis on Twitter feeds.
4. Evaluate the novel solution.

1.4 Background and Motivation

The telecommunication industry in Sri Lanka is emerging in the competitive, business world increasing the number of service providers. According to the financial analysis (2017 June) done by The Telecommunication Regulatory Commission of Sri Lanka, there were 28,113,153number of cellular mobile telephone subscriptions as at June 2017. The cellular subscription growth is shown in Figure 1.1 [1].

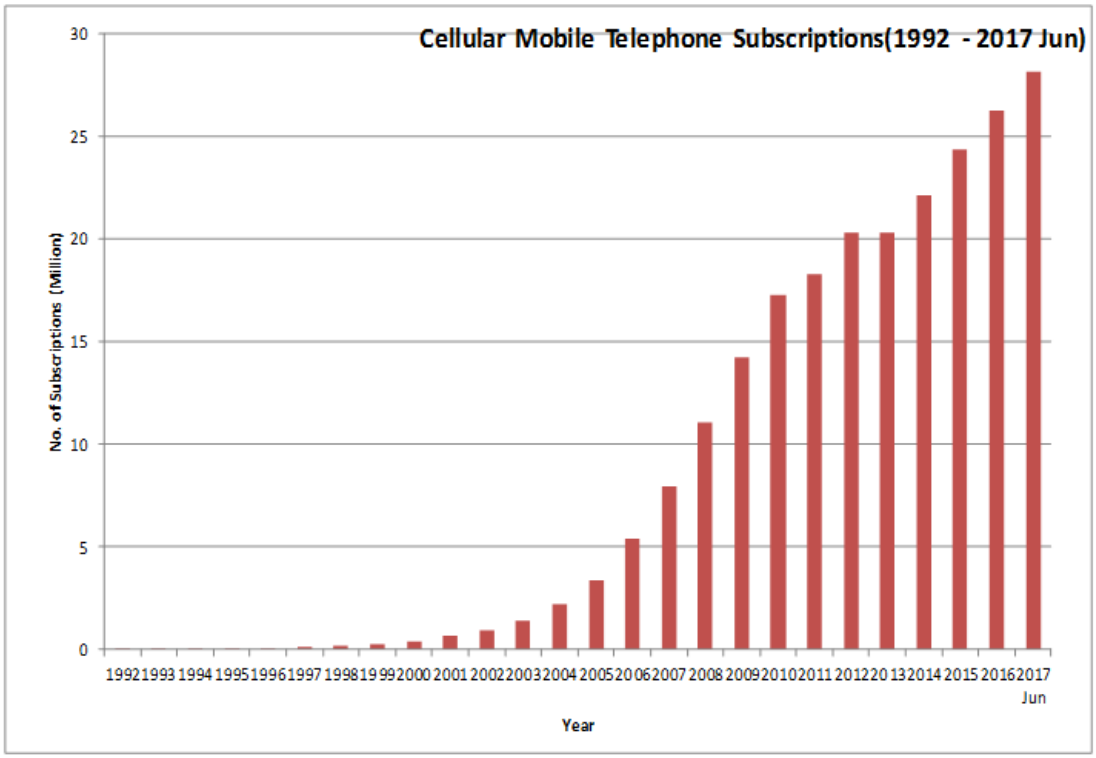

Figure 1.1: Cellular Mobile Telephone Subscriptions [1]

With this increased competition among telecom service providers, it has become more difficult to retain the existing customers. New companies make efforts to concentrate on acquiring new customers, while the matured ones try to focus on retention of the

existing customers. Defending and protecting the customer base should be the number one job in this challenging economic time period. Most companies are unprepared for the arrival of the storm and therefore companies allocate some resources from the top of the funnel demand creation to customer retention [2].

Customer satisfaction is not an easy task as the expectations of the customers are difficult to measureand businesses come to know that after the product has already been delivered [3]. Despitethe difficulty and cost of measuring it, customer satisfaction remains a major concern of businesses because it isconsidered as an important tool for securing a competitive advantage [4].

Customer sentiment analysis is a method of processing information, generally in text format and often from social media sources, in order to determine customer opinions and responses. By analyzing this data, organizations are allowed to assess whether customer reaction to a new product was positive or negative, or whether owners of a product are experiencing major technical difficulties. Analysis of aggregated data over time provides insights into trends, while analysis of individual cases in near real time lets companies to address and resolve customer issues quickly [5].

Text analysis is a complex process based on statistical and linguistic analyses and is used for many different applications, including fraud detection and analysis of scientific or intelligence data. Many of the social media streams are filled with slang, abbreviations and sarcasm, all of which are difficult for analytical tools to process. Depending on the application and the software tool, users of customer sentiment solutions have varying degrees of success [5].

1.5Problem in brief

When the number of customers reaches its peak, finding and securing new customers become increasingly difficult and costly. Therefore, it would be better to prioritize the retention of the existing customers, than trying to win new ones. With the rapid growth of mobile services, enhancement of customer satisfaction has become as a core issue and hence developing a model, which can be used to measure the customer satisfaction, would be very important for the telecom industry especially for their survival. In addition to that, by analyzing the satisfaction, the telecom operator would be able to identify which services needs to be improved further in order to satisfy the existing customer.

1.6Proposed Solution

The purpose of this research is to design and develop a model for customer satisfaction of Telecom Service based on reviews and feedbacks from Twitter. This research is aimed at mining tweets toward one of the largest mobile operator in Sri Lanka.

Apart from that, sentimental analysis will be done only for English language.

1.7Structure of the thesis

The documentation of this thesis is outlined in the following way. Chapter 1 provides the background to this research and briefly describes the context of this research. It also defines the problem statement and the purpose of this research. Chapter 2 provides a survey of literature and explains the main phases involved in building a model. Chapter 3 specifies the technology adopted for this research work while fourth chapter provides the details on the research approaches. The analysis and design part are specified in Chapter 5. The next Chapter explains on the implementation phase.Last Chapter provides a discussion on the proposed methodologywith an evaluation of the approachesand it also provides some insight into the future research work.

Chapter 2

2 Literature Review

2.1 Introduction

The current chapter consists of six main sections which describe the major phasesinvolved in developing the customer satisfaction monitoring framework. These steps have been identified based on the literature survey done focusing on mining customer reviews on social media. Each step will be explained in detail with the reference of the previous research work.

2.2 Customer Satisfaction Monitoring Framework

After doing a research on existing approaches for mining reviews of different domains [6][7][8], it has been identified that the following steps will be mainly involved in sentimental analysis.

1. Goal Setting
2. Text Preprocessing
3. Parsing the content
4. Text Refinement
5. Analyzing and Scoring
6. Finalize and validate the model

2.3 Goal Setting

This determines the sentiment analysis goal and the scope for the text content [6]. For example, sentiment analysis goal can be set to a specific domain and therefore only the reviews under that domain will be considered.As well as the goal setting, scope for the text content is also important. It would be difficult to do the analysis for a larger scope and therefore it is better to define the scope before starting the research. For example, the scope for the text content can be defined for a specific language.

2.4 Text Preprocessing

Twitter feeds have to be collected for the selected telecom operator. Then the collected feeds have to be loaded to the processing system (the system, technique to be used for the analysis) and unwanted words from the text have to be deleted. Also, the emotional symbols that people use in texts have to be organized into words.

Removing suffixes from words reduce number of words, tohave exactly matching stems, to save memory space andtime. Forexample, as shown in Figure 2.1, the words material, materially, materialize etc.,all can bestemmed to the word “material” [9].

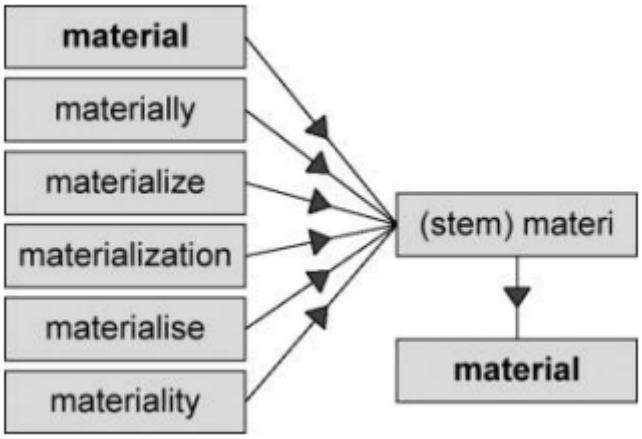

Figure 2.1: Stemming Process [9]

The stemming process is done using variousalgorithms. “M.F. PortersAlgorithm" has been widely adopted and extended so that it has becomethe standard approach to word conflation for information retrieval in a wide range of languages [10].

2.5 Parsing the content

This involves segmenting the words based on their polarity, tagging the parts of speech used (adjective, noun, etc.) and identifying the terms [6].

For example, there are three reviews (documents) of a book as provided below.

Document 1: I am an avid fan of this sport book. I love this book.

Document 2: This book is a must for athletes and sport.

Document 3: This book tells how to command the sport.

Parsing this document collection (reviews) generates the following term-by-document matrix as shown in below figure2.2 [11].

Term/Document	Document 1	Document 2	Document 3
the	0	0	1
I	2	0	0
am	1	0	0
avid	1	0	0
fan	1	0	0
this	2	1	1
book	2	1	1
athletes	0	1	0
sportsmen	0	1	0
sport	1	0	1
command	0	0	1
tells	0	0	1
for	0	1	0
how	0	0	1
love	1	0	0
an	1	0	0
of	1	0	0
is	0	1	0
a	0	1	0
must	0	1	0
and	0	1	0
to	0	0	1

Figure 2.2: Term-By-Document Matrix [11]

2.6 Text Refinement

This will ensure the correct analysis by finding the stop words and synonyms, etc. [6]. Most frequently used words in English are useless in text mining and such words are called as stop words. Stop words arelanguage specific functional words which carry noinformation. It may be of the following types such aspronouns, prepositions, conjunctions [9].

2.7 Analyzing and Scoring

2.7.1 Analyzing

With the exponential increase in the internetusage, people prefer to express and share informationon different topics. Due to ever increasingexistence of these emotions, opinions, views, feedbacksand suggestions on the web, it has become necessary toexplore, analyze and organize this information forbetter decision making by subsequent users [12].Sentiment analysis or opinion mining is the computational

study of people'sopinions, appraisals, attitudes, and emotions toward entities, individuals,issues, events, topics and their attributes [13].

2.7.1.aDifferent Levels of Sentiment Analysis

Sentiment analysis has been investigated mainly at three levels as provided below.

Document level Sentiment Analysis

This is the simplest form of classification. The wholedocument of opinionated text is considered as basicunit of information. It is assumed that document ishaving opinion about single object only (film, book orhotel). This approach is not suitable if documentcontains opinions about different objects as in forumsand blogs. Classification for full document is done aspositive or negative. Irrelevant sentences need to beeliminated before processing. There are twoapproaches to do classification [12].

1. Supervised machine learning approach

Given the training data, the system classifiesthe document by using one of the common classification algorithms such as Support VectorMachine, Naïve Bayes, K Nearest Neighbours, andMaximum Entropy etc.

2. Unsupervised machine learning approach

In unsupervised approach, SentimentOrientation (SO) of opinion words in document isdetermined. If the SO of these words is positive, thenthe document is classified as positive otherwisenegative.

Sentence Level Sentiment Analysis

Sentence level sentiment analysis is the most fine-grainedanalysis of the document. In this, polarity iscalculated for each sentence as each sentence isconsidered as separate unit and each sentence canhave different opinion. Sentence level sentimentanalysis has two tasks [12].

1. Subjectivity Classification

A sentence can be either subjective sentence orobjective sentence. Objective sentence contains thefacts. It has no judgment or opinion about the objector entity while subjective sentence has opinions.

2. Sentiment Classification

Sentence can be classified as positive, negative orneutral depending upon the opinion words present init.

Entity and Aspect level

Aspect level is the opinion mining and summarization based on feature. The classification concerns byidentifying and extracting product features from the source data. This type is used when we need sentiments aboutdesired aspect/feature in a review [14].

2.7.1.b Sentiment Classification Techniques

Mainly approaches are classified into two categories namelylexicon based approach and machine learning based approach (Refer Figure 2.3).

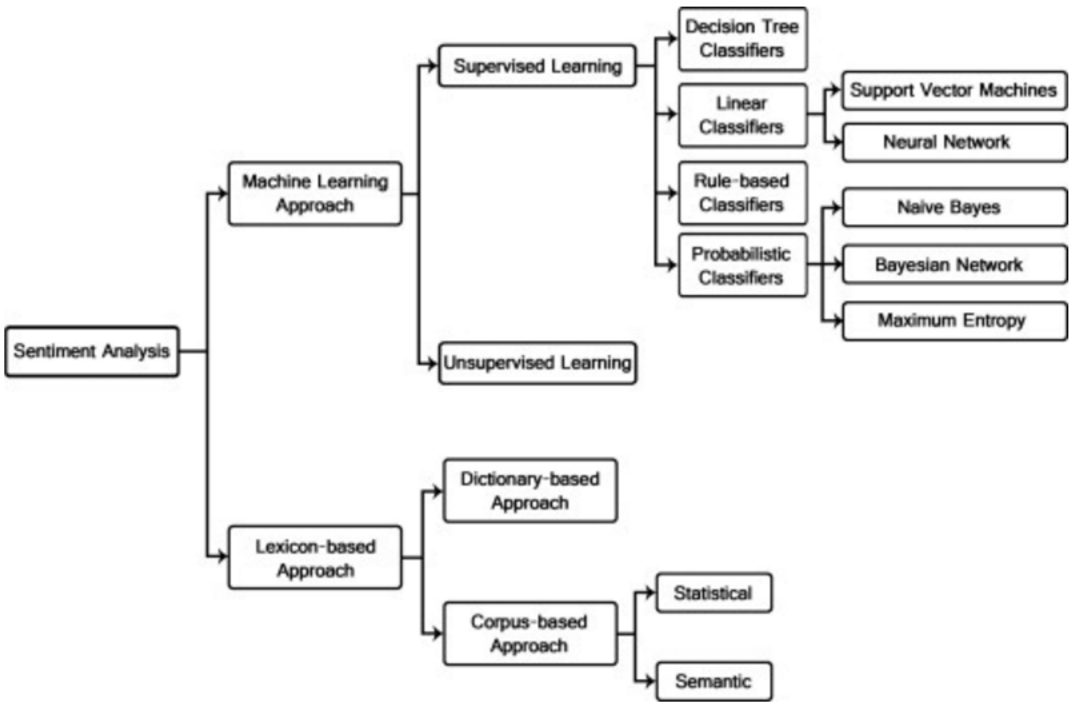

Figure 2.3: Sentiment Classification Techniques [15]

Machine learning is further divided into two category namelysupervised and unsupervised learning. Supervised classificationalgorithms are probabilistic classifier, linear classifier, decisiontree and rule based classifier. Supervised learning technique isbased on labeled dataset which is provided as input to train themodel and this model

is applied to test data to generate output.Sentiment classification in machine learning consists of twosteps. First one is to extract feature and store in feature vectorand second one is to train feature vector by using classificationalgorithms [16].

Lexicon based approach is further divided into two categories namely dictionary based and corpus based approach. Indictionary based approach, sentiment is identified usingsynonym and antonym from lexical dictionary like WordNet. Incorpus based approach, it identifies opinion words byconsidering word list. Corpus based approach furthermoreclassified as statistical and semantic approach. In statisticalapproach, co-occurrences of words are calculated to identifysentiment. In semantic approach, terms are represented insemantic space to discover relation between terms [17].

2.7.2 Scoring

Scoring is the process in which the intensity of the sentiment is analyzed. Finally,each customer feeds will be categorized into one of the below labels [6].

- Positive
- Negative
- Neutral

An example for scoring is shown in Figure 2.4.First it identifies the sentiments bearing phrases from the data and score accordingly.

SCORE	TEXT
2	You're awesome and I love you
-5	I hate and hate and hate. So angry. Die!
4	Impressed and amazed: you are peerless in your achievement of unparalleled mediocrity.

Figure 2.4: Sample score for sentiment [6]

2.8 Finalize and validate the model

The performance of sentiment analysis is calculated by usinghelp of confusion matrix which is generated when algorithm isimplemented on dataset. Various performance measures areused that are Precision, Recall, F-measure and Accuracy [16].

In classification problems, the primary source of performance measurementsis a coincidence matrix (a.k.a. classification matrix or a contingencytable). The Figure shows a coincidence matrix for a two-class classificationproblem (Refer Figure 2.5).

The equations of most commonly used metrics that can be calculatedfrom the coincidence matrix is also given below [18].

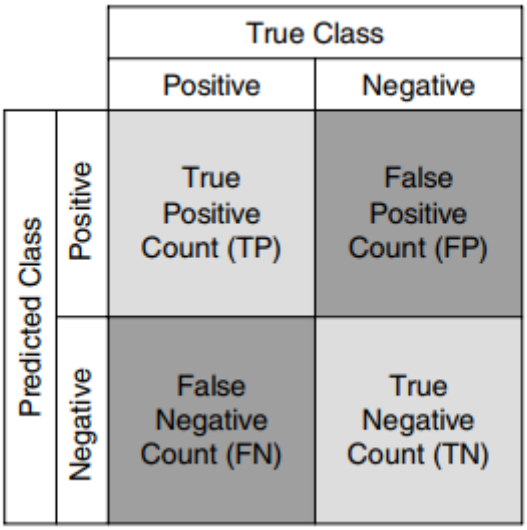

Figure 2.5: Confusion matrix for binary classification [18]

The FP, FN, TP and TN concepts may be described as:

False positives (FP): examples predicted as positive,which are from the negative class.

False negatives (FN): examples predicted as negative,whose true class is positive.

True positives (TP): examples correctly predicted as pertainingto the positive class.

True negatives (TN): examples correctly predicted as belongingto the negative class.

2.8.1 The true positive rate (Hit rate or Recall) of aclassifier

This is estimated by dividing the correctly classified positives (thetrue positive count) by the total positive count. Large recallvalue means few positive cases misclassified as a negative[19]. This is also known as Sensitivity.

$$Sensitivity = \frac{TP}{TP + FN}$$

2.8.2 The false positive rate of aclassifier (FPR)

FPR or false alarm ratio measures the cases classified as positive incorrectly [19].

$$False\ Positive\ Rate = \frac{FP}{TN + FP}$$

2.8.3Precision

This is also called as positive predicted value and it measures the correctness of the model. Higher precisionindicates less FP[19].

$$Precision = \frac{TP}{TP + FP}$$

2.8.4F-Score

F-score or F1-measure is the harmonic mean of precision and recall [19].

$$F1\,Score = \frac{2TP}{2TP + FP + FN}$$

2.9 Summary

This chapter explains on main steps which can be mainly identified in developing the model of customer satisfaction monitoring based on twitter feeds.This also includes an explanation of different sentiment classification techniques including machine learning and lexicon based approach. Apart from that, it briefs on how to score the model and to measure and evaluate the model performance with the use of confusion matrix.

Chapter 3

3 Technology

3.1 Introduction

For this research work, lexicon based method is used. Application of a lexicon involves calculating the sentiment from the semantic orientation of tokenized word or phrases that can be found in a text. In this approach a dictionary of positive and negative words is required and each of the words will get assigned with a neutral, positive or negative sentiment value. In this research work, the following four dictionaries will be considered in the classification.

- Lexicon
- SentiWordNet
- Slangs
- Emoticons

First it will select the best lexicon resource and then consider the different combination of dictionaries in order to come up with a better solution. In each step use Lexicon-based method where the semantic orientation of the document is calculated by summing the semantic orientation of the words and phrases in the document.

Apart from that, in this research work, tweets were extracted using Rapid Miner. Main framework is developed and evaluated on R and only for using SentiWordNet, python is used due to some compatibility issues come across with operating system.

3.2 Machine Learning Vs Lexicon Based Method

Supervised methodsrequire large amounts of labeled training data thatare very expensive whereas acquisition of unlabeled data is easy. The main limitation of supervisedlearning is that it generally requires large expertannotatedtraining corpora to be created from scratch,specifically for the application at hand, and may failwhen training data are insufficient [21].For this research scenario, it will be very difficult to gather a better training data set.

3.3 Limitations of Lexicon Based Method

The opinion words that are included in thedictionary are very important for the lexicon basedapproach. If the dictionary contains less words orthorough, one risks the chance of over or underanalyzing the results, leading to a decrease inperformance.

Another significant challenge to thisapproach is that the polarity of many words is domainand context dependent. For example, ‘funny movie’is positive in movie domain and ‘funny taste’ isnegative in food domain. Such words are associatedwith sentiment in a particular domain. Currentsentiment lexicons do not capture such domain andcontext sensitivities of sentiment expressions.Without a comprehensive lexicon, the sentimentanalysis results will suffer. The lexicon-basedapproach can result in low recall for sentimentanalysis [21].

3.4 Summary

Lexicon based approaches often reply on sentiment lexicons for sentiment analysis and hence training from labeled instances does not require. For this research scenario, it will be very difficult to gather a better training data set and therefore this approach has been selected and applied in this framework. Apart from that, if the overall model to be performed well, the training dataset should be in better quality. If its quality is not good, the accuracy of the model will also be affected badly. In such a situation, it is better to go with Lexicon approach in which the accuracy of the model will be mainly based on the dictionaries used.

Chapter 4

4 Approach

4.1 Introduction

Development of a methodology which is capable of accurately monitoring the customer satisfaction could prevent significant loss of revenue. This chapter discusses the methodology which has guided the main activities of thisresearch. This also includes the inputs and outputs of this analysis and then provides the details of the high-level process, which uses four dictionaries.

The following six steps will be followed up in this research work and will select the best combination of dictionaries accordingly.

Step 01: After pre-processing and refinement of Twitter Feeds, it will select the best lexicon resource (Lexicon vs SentiWordNet).

Step 02: Performance measures were checked after fine-tuning the selected lexicon resource.

Step 03: Performance measures were checked after fine-tuning Stop Words Dictionary.

Step 04: Check whether the Hybrid Approach (Using both Lexicon & SentiWordNet) will perform well.

Step 05: Performance measures were checked with Lexicon dictionary after Slang Replacements.

Step 06: Performance measures were checked with Lexicon dictionary after Slang Replacements and also integrating with Emoticons.

4.2 Hypothesis

Main goal of this research is to aim at mining tweets toward one of the largest mobile operator in Sri Lanka.Apart from that, sentimental analysis will be done only for English language.

4.3 Input

Main input of this research is Twitter feeds which is extracted using Rapid Miner Tool and then these twitter feeds are then loaded to R for further processing. There were 1450 tweets which were collected randomly during 2nd February 2017 to 15th January 2018. From the collected tweets, only 740 tweets were filtered out as only English tweets were considered in this research work.

4.4 Output

Most of the researchers have focused on binary classification as multi class classification is quite difficult to evaluate. In this research work, Twitter feeds will be classified into threeclasses as negative, positive and neutral. Therefore, the model has to be evaluated as a 3X3 matrix and this has to be used three confusion matrices for each class in order to calculate the accuracy.

4.5 Process

This research work will carry out mainly based on four dictionaries (Lexicon, SentiWordNet, Slangs and Emoticons).First it will select the best lexicon resource and then consider the different combination of dictionaries in order to come up with a better solution. In each step use Lexicon-based method where the semantic orientation of the document is calculated by summing the SO of the words and phrases in the document.Before data is subjected to analyze, they will undergo a set of text preprocessing tasks.

Then all tweets are scored as positive, negative, or neutral and after that each step will be evaluated using following performance metrics.

1. Accuracy
2. Precision
3. Recall
4. F-Measure

4.6 Summary

This chapter describes the high-level solution approach of this research work. This provides an understanding on the input to this model and what will be the outcome of this model which is useful for the customer satisfaction identification. The next chapter will explain this process with more details.

Chapter 5

5 Analysis & Design

5.1 Introduction

In this chapter, the proposed solution will be shown using a high-level diagram and each module will be explained in detail under the later sections of this chapter. This chapter also provides the details of all the dictionaries used in this research work. In addition to that, it explains how to measure the accuracy of the three-dimensional classification using confusion matrix concept.

5.2 High-Level Solution Diagram

Proposed solution consists of four major modules as shown in Figure 5.1.

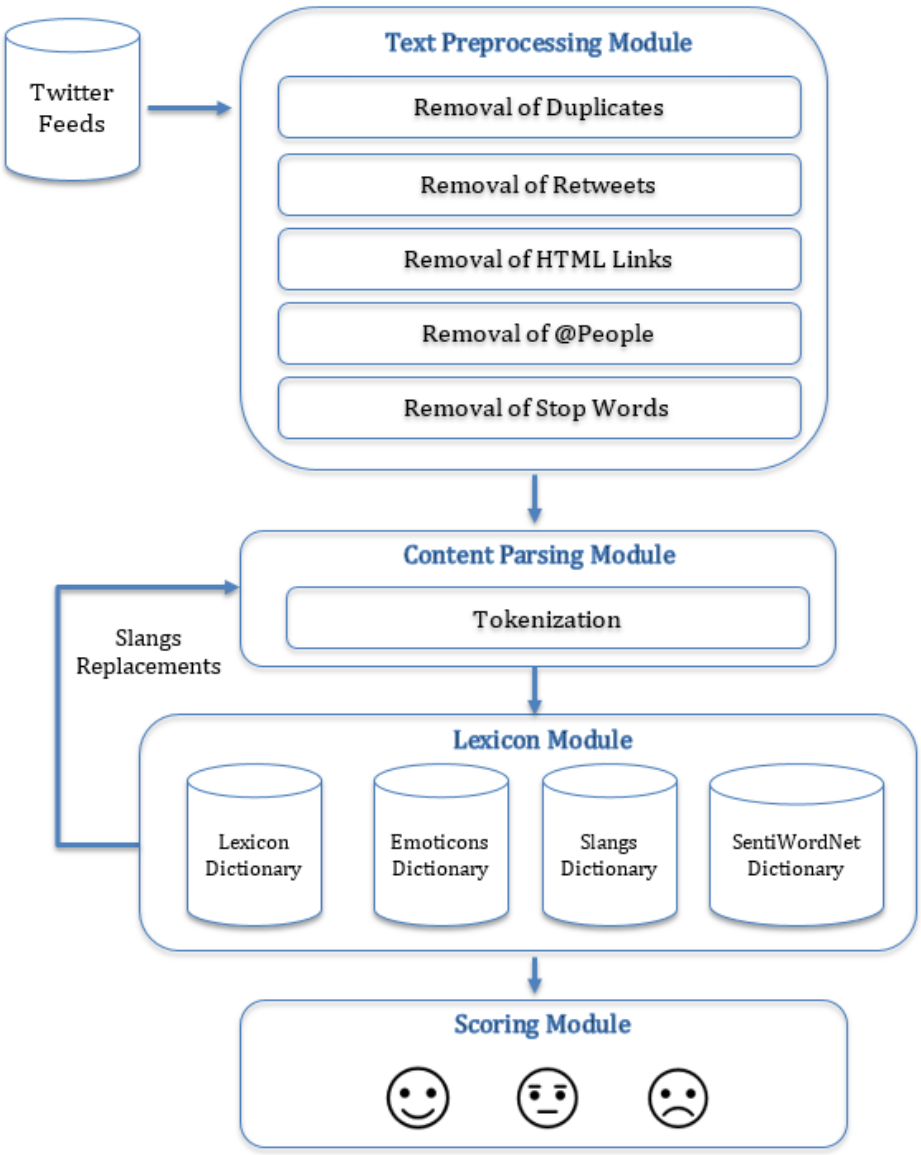

Figure 5.1: High Level Solution Diagram

5.2.1 Text Preprocessing Module

The data is extracted from Twitter feeds using Rapid Miner Tool. There were 1450 tweets which were collected randomly during 2nd February 2017 to 15th January 2018. From the collected tweets, only 740 tweets were filtered out as only English tweets were considered in this research work. These twitter feeds are then loaded to R using "fread" function which comes under "data.table" library.

Example:

```
x <- fread ("C:\\Users\\MscProject\\TwitterFeeds\\TestData01.txt", header = TRUE, select = c("Text","Rating"))
```

Then the collected tweets were then pre-processed using the following ways.

5.2.1.a Removal of Duplicates

Duplicate of any tweet are deleted from the data set."duplicated" function will remove the duplicated rows from the dataset. Removal of duplicate feeds will reduce the unnecessary processing time on the model.

Example:

```
x <- x[!duplicated(x), ]
```

5.2.1.bRemoval of Retweets

Retweets (reposting or forwarding a tweet which is posted by another user) entries are removed from the data set and this can be done using "gsub" function available with R.

Example:

```
x$Text <- gsub("RT @[a-z,A-Z]*:", "", x$Text)
```

5.2.1.cRemoval of HTML Links

HTML links are removed from the data set and for this, "rm_url" function can be used as mentioned on the below example.

Example:

```
x$Text <- rm_url(x$Text, pattern=pastex("@rm_twitter_url", "@rm_url"))
```

5.2.1.dRemoval of @people

@Users are removed from the data set. Same "gsub" function can also be used to remove @people from the tweets and an example is given below.

Example:

```
x$Text <- gsub("@\\w+", "", x$Text)
```

5.2.1.e Removal of Stop Words

Most frequently used words in English which are called as stop words are removed from the data set. They are language specific functional words ("a", "and", "but", "how", "or", and "what" etc.) which carry no information.

The following "rm_words" function can be used to remove stop words from the tweets.

```
rm_words <- function(string, words) {
stopifnot(is.character(string), is.character(words))
spltted<- strsplit(string, " ", fixed = TRUE) # fixed = TRUE for speedup
vapply(spltted, function(x) paste(x[!tolower(x) %in% words], collapse = " "), character(1))
}
```

Once created the above function, it can be called and applied on tweets with the Stop Words dictionary ("stopwords") which is provided with "tm" package. The below example extracts only the stop words related to English Dictionary by specifying "en" as this research is considering only English tweets.

Example:

```
x$Text <-  rm_words(x$Text, tm::stopwords("en"))
```

5.2.2 Content Parsing Module

5.2.2.aTokenization

The process of breaking a stream of text into words, phrases, symbols, or other meaningful elementsis called as tokens.The aim of the tokenization is the exploration of the words in a sentence. The list of tokens becomes input for further processing such as parsingor text mining.

5.2.3Lexicon Module

This module is based on the integration of different opinion lexicons and dictionary resources for sentiments. Following lexicons and dictionaries are used in this research work.

5.2.3.aLexicon Dictionary

This contains a list of positive and negative words. This research uses a dictionary of 1980 number of positive words and 4800 number of negative words.

5.2.3.bEmoticons Dictionary

This contains a list of emoticons. Emoticons are mainly face-basedand represent happy or sad feelings, although a wide range ofnon-facial variations also exist. For example, <3 represents a heartand expresses love or affection. In this research, 750 number of emoticons are used in the model [6].

5.2.3.cSentiWordNet

This is a lexical resource and an extension of WordNet [20].Thedictionary groups adjectives and nouns and other parts ofspeech in sets with similar meaning words which are called as‘synsets’(Synonym sets). SentiWordNet does the scoring ofthe words from the sets to indicate the sentiment as positive, negative or neutral [6].

5.2.3.dSlang Dictionary

People use slang words such as “OMG” and “LOL” to express their feelings and therefore,identifying slang sentiment words can be an advantage to accurately discovering sentiment hidden in tweets and customer reviews. For this analysis, 5400 number of slangs have been collected from the web and used for scoring in this research work.

5.2.4Scoring Module

Each dictionary contributes to the final score as mentioned in the below sections.

5.2.4.a Lexicon Score

Score is modified by assigning +1 for positive words and -1 for negative words in the lexicon dictionary.

5.2.4.b Emoticon Score

Score is assigned either as +1 for positive and -1 for negative emoticons as in the dictionary.

5.2.4.c SentiWordNet Score

This associates each Wordnet synset withthree numerical scores as positive, negative and objective. These scores range from 0.0 to 1.0 and sum of scoresfor each synset is 1.

5.2.4.d Slang Score

Slang dictionary supports to identify the slangs and then translate them in to meaningful words and compiled with their translation for scoring.

5.2.6Finalize and validate the model

In order to validate the approaches, all tweets are manually labeled as positive, negative, or neutral and hence the confusion matrix needs to be developed as a 3x3 matrix. As shown in Table 5.1, 5.2 and 5.3, three matrices will be generated for eight approaches.

5.2.6.aConfusion Matrix for Positive Reviews

	Predicted Class for **Positive** Reviews		
True Class for **Positive** Reviews		Positive	Other
	Positive	TP1	FN1
	Other	FP1	TN1

Table 5.1: Confusion Matrix for Positive Reviews

5.2.6.b Confusion Matrix for Negative Reviews

	Predicted Class for **Negative** Reviews		
True Class for **Negative** Reviews		Negative	Other
	Negative	TP2	FN2
	Other	FP2	TN2

Table 5.2: Confusion Matrix for Negative Reviews

5.2.6.c Confusion Matrix for Neutral Reviews

	Predicted Class for **Neutral** Reviews		
True Class for **Neutral** Reviews		Neutral	Other
	Neutral	TP3	FN3
	Other	FP3	TN3

Table 5.3: Confusion Matrix for Neutral Reviews

Then the validation of the model is done by analyzingthe following performance metrics for the above values of each confusion matrix.

1. Accuracy
2. Precision
3. Recall
4. F-Measure

5.3 Summary

This chapter provides an explanation on the design approach in each module and in the final section, it shows how to do the model validation with the use of confusion matrix and the relevant performance metrics. Since three class classification has to be facilitated in this research, three confusion matrixes have to be used in getting the accuracy of each class. The next chapter provides an explanation on the implementation process of this model.

Chapter 6

6 Implementation

6.1 Introduction

This chapter provides the details of the implementation phase of the proposed solution. Implementation was done based on six steps which uses different combinations of dictionaries or with fine-tuned dictionaries. Scoring process has been describedusing pseudocodes for each step.

6.2 Approach

The following steps were considered in order to identify the most accurate model for this analysis. Outcome of these steps are separately validated and compared.

Step No	Description
01	Select the best lexicon resource (Lexicon vs SentiWordNet) after pre-processing and refinement of Twitter Feeds
02	Feeds with Fine-tuned Lexicon Resource
03	Feeds with Fine-tuned Stop Words Dictionary
04	Hybrid Approach (Using both Lexicon & SentiWordNet)
05	Feeds with Lexicon with Slang Replacements
06	Feeds with Lexicon with Slang Replacements and Emoticons

Table 6.1: Main Approaches

6.3 Dataset

Twitter feeds from the selected telecom provider are extracted from Rapid Miner and it containstweets which were collected randomly during 2nd February 2017 to 15thJanuary 2018. There were 1450 tweets which were collected randomly during 2nd February 2017 to 15th January 2018. From the collected tweets, only 740 tweets were filtered out as only English tweets were considered in this research work.

6.4 Text Preprocessing Module

Extracted Twitter feeds are loaded to R for further processing.

- Install & Load required R packages

- Import Twitter Feeds
- Remove Duplicate Tweets
- Duplicate “Text” field to another column (To keep the original text for references)

The collected tweets were then pre-processed using the flowing steps.

- Removal of Retweets
- Removal of HTML Links
- Removal of @People
- Removal of Stop words

6.5 Content Parsing Module

Tweet comments were split into words before applying the Lexicon module. This process is called as text tokenization process.

6.6 Lexicon Module

In this module, the following four dictionaries were used.

- Lexicon Dictionary
- Emoticons Dictionary
- Slang Dictionary
- SentiWordNet

6.7 Scoring Module

Pseudocodes for sentiment scoring algorithms for each approach are provided below.

6.7.1 Algorithm for using Lexicon Dictionary only

This uses Lexicon dictionary only.

***Input**: Tweets*

***Output:** Sentiment Score*

Function_Score(tweet)

ptext = preprocessor(tweet)

tokens = tokenize(ptext)

***For** word in tokens*

*   **If** word found in Lexicon Dictionary**Then***

Score = Score + Lexicon Score

Else

Score = 0

End If

Next

If *Score > 0* ***Then***

Tweet = Positive

If *Score < 0* ***Then***

Tweet = Negative

If *Score = 0* ***Then***

Tweet = Neutral

End If

End Function

6.7.2 Algorithm for using SentiWordNet only

This uses SentiWordNet Dictionary only.

Input*: Tweets*

Output: *Sentiment Score*

Function_Score(tweet)

ptext = preprocessor(tweet)

tokens = tokenize(ptext)

For *word in tokens*

If*word found in SentiWordNet****Then***

Score = Score + SentiWordNet Score

Else

Score = 0

End If

Next

If *Score > 0* ***Then***

Tweet = Positive

If *Score < 0* ***Then***

Tweet = Negative

If** Score = 0 **Then

Tweet = Neutral

End If

End Function

6.7.3 Algorithm for using both Lexicon and SentiWordNet

This uses Lexicon & SentiWordNet Dictionaries.

***Input**: Tweets*

***Output:** Sentiment Score*

Function_Score(tweet)

ptext = preprocessor(tweet)

tokens = tokenize(ptext)

***For** word in tokens*

If** word found in Lexicon Dictionary**Then

Score = Score + Lexicon Score

Else If**word found in SentiWordNet**Then

Score = Score + SentiWordNet Score

Else

Score = 0

End If

Next

If** Score > 0 **Then

Tweet = Positive

If** Score < 0 **Then

Tweet = Negative

If** Score = 0 **Then

Tweet = Neutral

End If

End Function

6.7.4 Algorithm for using Lexicon with Slang Replacements

This uses Lexicon & Slang Dictionaries.

***Input**: Tweets*

Output: *Sentiment Score*

Function_Score(tweet)

ptext = preprocessor(tweet)

tokens = tokenize(ptext)

For *word in tokens*

If*word found in Slang Dictionary****Then***

Replace text with Slang Text

stext = SlangText

tokens = tokens + tokenize(stext)

If *word found in Lexicon Dictionary****Then***

Score = Score + Lexicon Score

Else

Score = 0

End If

End If

Next

If *Score > 0* ***Then***

Tweet = Positive

If *Score < 0* ***Then***

Tweet = Negative

If *Score = 0* ***Then***

Tweet = Neutral

End If

End Function

6.7.5 Algorithm for using Lexicon with Slang Replacements and Emoticons Dictionary

This uses Lexicon, Slang& Emoticon Dictionaries.

Input*: Tweets*

Output: *Sentiment Score*

Function_Score(tweet)

ptext = preprocessor(tweet)

tokens = tokenize(ptext)

For *word in tokens*

If*word found in Slang Dictionary****Then***

Replace text with Slang Text

stext = SlangText

tokens = tokens + tokenize(stext)

If *word found in Emoticons Dictionary****Then***

Score =Score + Emoticon Score

Else

If *word found in Lexicon Dictionary****Then***

Score = Score + Lexicon Score

Else

Score = 0

End If

End If

Next

If *Score > 0* ***Then***

Tweet = Positive

If *Score < 0* ***Then***

Tweet = Negative

If *Score = 0* ***Then***

Tweet = Neutral

End If

End Function

6.8 Summary

This chapter provides the implementation details and provide the pseudocodes for sentiment scoring of sixsteps. The next chapter will provide the results and evaluations of all these combinations.

Chapter 7

7 Discussion

7.1 Introduction

This chapter provides the details of the evaluation phase with the results of the selected performance metrics. This supports to get an understanding on to what extend accuracy is enhanced from each dictionary. Evaluation is mainly done based on the six steps mentioned in implementation chapter and four performance matrixes have been calculated to understand the performance of each step.

7.2 Results & Analysis

After preprocessing the twitter feeds, SentiWordNet and Lexicon dictionaries were applied on the feeds separately and classified the feeds as positive, negative or neutral. Table 7.1 shows the performance measures for the both approaches and as per that, Lexicon dictionary performs well on this classification.

Rating	Measure	Lexicon	SentiWordNet
Positive	Accuracy	85.19%	56.50%
	Precision	63.03%	28.09%
	Recall	87.72%	52.08%
	F_Measure	73.35%	36.50%
Negative	Accuracy	78.80%	66.75%
	Precision	84.96%	49.64%
	Recall	45.38%	52.65%
	F_Measure	59.16%	51.10%
Neutral	Accuracy	72.69%	54.00%
	Precision	65.75%	42.68%
	Recall	75.95%	20.35%
	F_Measure	70.49%	27.56%

Table 7.1: Lexicon vs SentiWordNet

Lexicon dictionary was fine-tuned further in order to align to Telecom domain and some of the modifications done were shown on Table 7.2.

Positive Words		Negative Words	
Removed	**Added**	**Removed**	**Added**
Portable	Fixed	Torrent	Deducted
Well	Solved	Static	Exceeded
Work			Overcharging
Like			
Support			

Table 7.2: Samples of Fine-tuned Words

After fine-tuning Lexicon Dictionary, the accuracy of Positive, Negative and Neutral Segments has been increased by 5.16%, 2.59% and 3.67% respectively (Refer Table 7.3).

Rating	Measure	Lexicon	Fine-tuned Lexicon	Increased %
Positive	**Accuracy**	85.19%	90.35%	5.16%
	Precision	63.03%	75.00%	11.97%
	Recall	87.72%	87.72%	0.00%
	F_Measure	73.35%	80.86%	7.51%
Negative	**Accuracy**	78.80%	81.39%	2.59%
	Precision	84.96%	89.44%	4.48%
	Recall	45.38%	51.00%	5.62%
	F_Measure	59.16%	64.97%	5.81%
Neutral	**Accuracy**	72.69%	76.36%	3.67%
	Precision	65.75%	68.02%	2.27%
	Recall	75.95%	84.81%	8.86%
	F_Measure	70.49%	75.49%	5.00%

Table 7.3: Lexicon vs Fine-tuned Lexicon

Same measures were extracted after removing “No” word from the Stop Words Dictionary and as per that (Refer Table 7.4), accuracy of Positive, Negative and Neutral classifications have been increased by 0.82%, 0.02% and 0.13% respectively.

Rating	Measure	Fine-tuned Lexicon	Fine-tuned Lexion with Stop Words Changes	Increased %
Positive	Accuracy	90.35%	91.17%	0.82%
	Precision	75.00%	77.60%	2.60%
	Recall	87.72%	87.13%	-0.59%
	F_Measure	80.86%	82.09%	1.23%
Negative	Accuracy	81.39%	81.41%	0.02%
	Precision	89.44%	87.91%	-1.53%
	Recall	51.00%	54.22%	3.22%
	F_Measure	64.97%	66.18%	1.21%
Neutral	Accuracy	76.36%	76.49%	0.13%
	Precision	68.02%	68.57%	0.55%
	Recall	84.81%	83.54%	-1.27%
	F_Measure	75.49%	75.32%	-0.17%

Table 7.4: Fine-tuned Lexicon vs Fine-tuned Lexicon with Stop Words Changes

All the measures taken based on fine-tuned Lexicon dictionary (With the changes in Stop Words Dictionary) were then compared with the measures taken after applying both Lexicon and SentiWordNet together. According to the result in Table 7.5, this hybrid approach has not been performed well on the feeds.

Rating	Measure	Lexicon	Lexicon + SentiWordNet
Positive	Accuracy	91.17%	64.81%
	Precision	77.60%	39.05%
	Recall	87.13%	91.81%
	F_Measure	82.09%	54.80%
Negative	Accuracy	81.41%	75.00%
	Precision	87.91%	62.08%
	Recall	54.22%	67.07%
	F_Measure	66.18%	64.48%
Neutral	Accuracy	76.49%	62.09%
	Precision	68.57%	78.46%
	Recall	83.54%	16.14%
	F_Measure	75.32%	26.77%

Table 7.5: Hybrid Approach (Both Lexicon and SentiWordNet)

After that some words in the twitter feeds were replaced based on Slang Dictionary and then Lexicon dictionary were applied on the feeds with replaced words. As per Table 7.6, it shows that there is an increase in accuracy for Positive, Negative and Neutral classifications by 0.81%, 0.59% and 1.09% respectively.

Rating	Measure	Lexicon	Lexicon with Slang Replacement	Increased %
Positive	Accuracy	91.17%	91.98%	0.81%
	Precision	77.60%	79.78%	2.18%
	Recall	87.13%	87.72%	0.59%
	F_Measure	82.09%	83.57%	1.48%
Negative	Accuracy	81.41%	82.00%	0.59%
	Precision	87.91%	84.21%	-3.70%
	Recall	54.22%	57.83%	3.61%
	F_Measure	66.18%	68.57%	2.39%
Neutral	Accuracy	76.49%	77.58%	1.09%
	Precision	68.57%	70.00%	1.43%
	Recall	83.54%	83.54%	0.00%
	F_Measure	75.32%	76.19%	0.87%

Table 7.6: Lexicon vs Lexicon with Slang Replacements

Feeds with Slangs replacements classified with Lexicon dictionary were then compared with the results of the same approach with Emoticons also. According to Table 7.7, it shows an increase in accuracy for Negative segment by 0.47% while decreasing the accuracy of Positive Class by 0.95%.

Rating	Measure	Lexicon with Slang Replacement	Lexicon with Slang Replacement + Emoticons	Increased %
Positive	Accuracy	91.98%	91.03%	-0.95%
	Precision	79.78%	74.65%	-5.13%
	Recall	87.72%	92.98%	5.26%
	F_Measure	83.57%	82.81%	-0.76%
Negative	Accuracy	82.00%	82.47%	0.47%
	Precision	84.21%	84.09%	-0.12%
	Recall	57.83%	59.44%	1.61%
	F_Measure	68.57%	69.65%	1.08%
Neutral	Accuracy	77.58%	77.58%	0.00%
	Precision	70.00%	71.76%	1.76%
	Recall	83.54%	78.80%	-4.74%
	F_Measure	76.19%	75.11%	-1.08%

Table 7.7: Lexicon with Slang Replacements and Emoticons

Figure 7.8 shows the accuracy comparison of Positive Class for Lexicon, Lexicon with Slang replacement and Lexicon with Slang replacement and Emoticons. According to that comparison, if positive comments need to be extracted, it is better to exclude Emoticon Dictionary as it shows the highest accuracy without that dictionary.

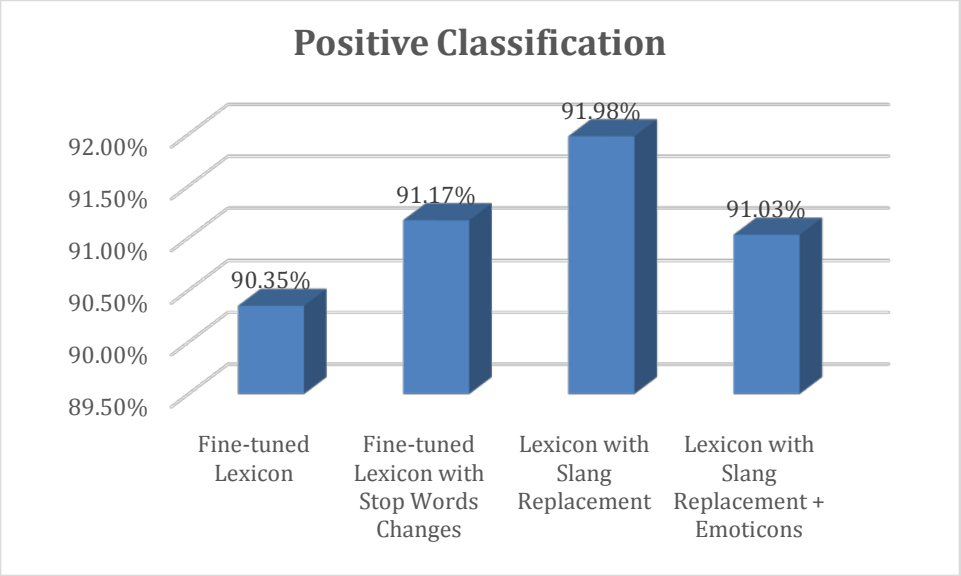

Figure 7.1: Comparison of Positive Classification Accuracy for each step

It would be advantageous to analyze the negative comments specially for getting an understanding on the products and service provided by the telecom operator. This kind of in depth analysis can be used for business decisions on products and service enhancements. As per Figure 7.9, if negative comments only are taken in to consideration, it would be better to go with Lexicon and Emoticon Dictionaries after replacing the slangs.

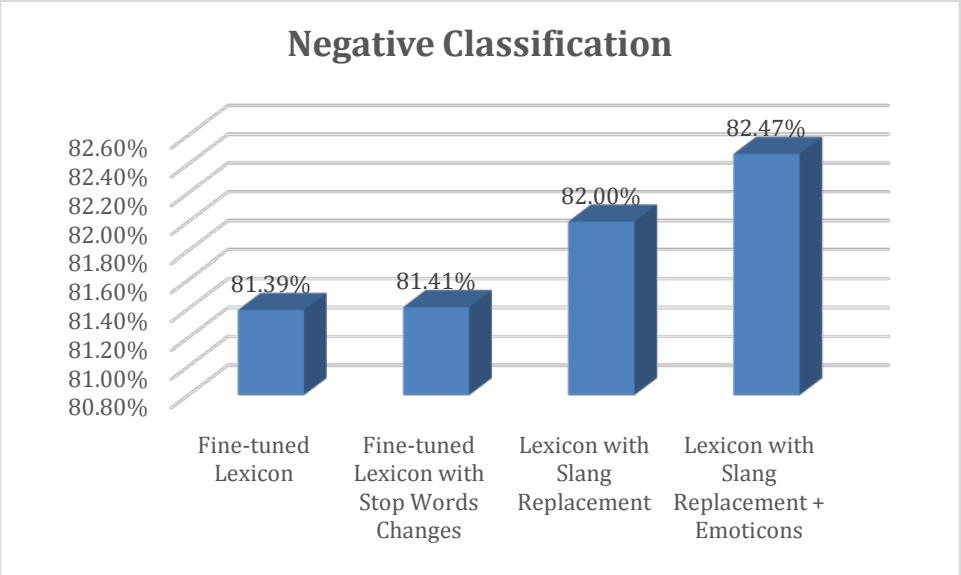

Figure 7.2: Comparison of Negative Classification Accuracy for each step

7.3 Model Validation with existing tools/APIs

There are some general sentiment analysis tools/APIs can be found in the internet and therefore it would be better to compare the accuracy level of these readily available tools/APIs with the proposed model. But most of these tools/APIs are provided an interface where only one comment at a time can be given to get the sentiment or it may give one sentiment score for all comments, not for each comment. This is one of the biggest limitation in these kinds of sentiment analysis tools/APIs. The accuracy of the proposed model has been validated with the results of two APIs as mentioned in the below sections.

7.3.1 Google Cloud Natural Language API

Google Cloud Platform provides Natural Language API and from that, users are able to provide their dataset and get the relevant sentiment score (Refer Figure 7.3). As per

Google definition, if the score is less than -0.25, it is considered as a negative text and if the score is between -0.25 to 0.25, it is taken as a neutral text. If the score is more than 0.25, then the text will be tagged as positive.

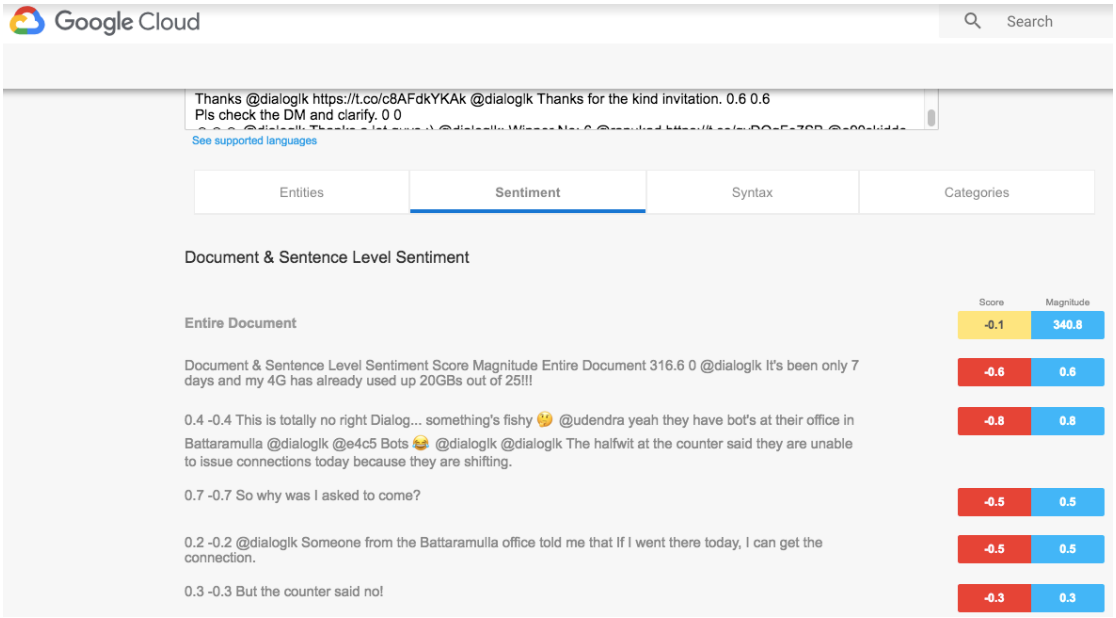

Figure 7.3: Snapshot of Sentiment Score on Google Cloud API

Using this API, three class classification were done on the same dataset which has been used in this research work and evaluate the accuracy of the classification with that API. As per Table 7.8, it shows that the model built for this research work has been shown higher accuracy level compared with the cloud natural language API.

Rating	Proposed Model	Google Cloud Natural Language API
Positive	91.98%	71.47%
Negative	82.47%	67.34%
Neutral	77.58%	62.78%

Table 7.8: Accuracy Comparison with Google Cloud API

7.3.2 tap.aylien.com API

tap.aylien.com has provided an interface to the users to do the sentiment analysis of the given dataset on their web site itself (Refer Figure 7.4).

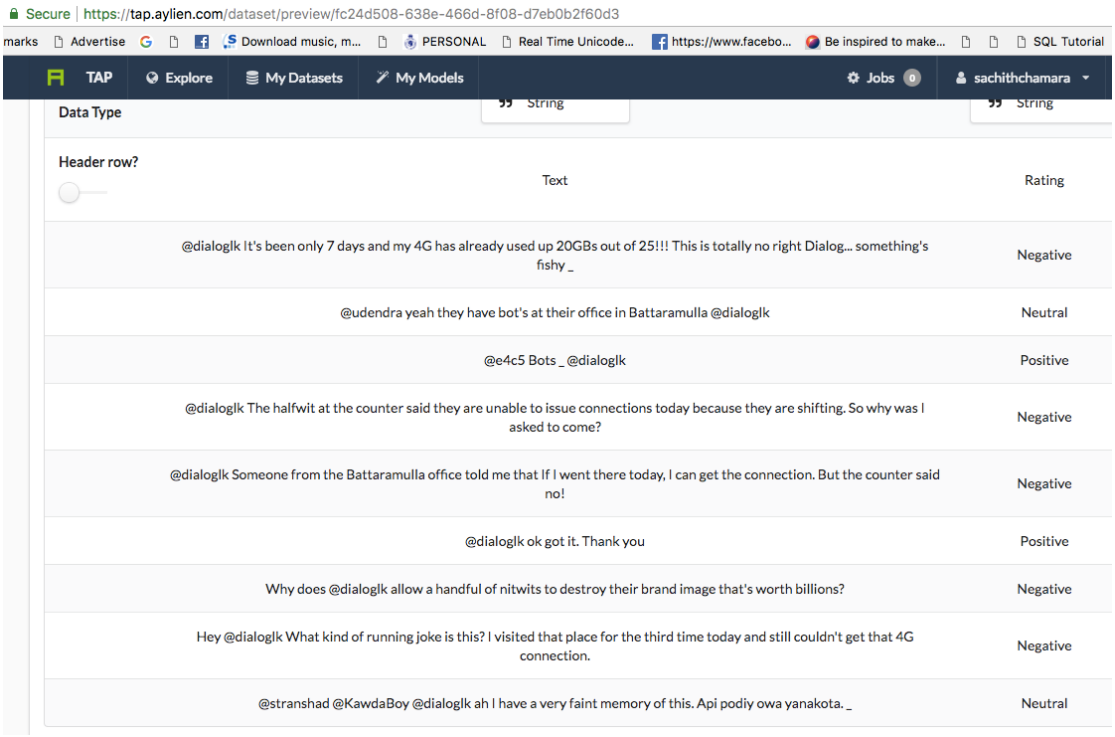

Figure 7.4: Snapshot of tap.aylien.com API

As per Table 7.9, it shows that the proposed model has been shown a higher accuracy level compared with the results of this API.

Rating	Proposed Model	tap.aylien.com API
Positive	91.98%	68.76%
Negative	82.47%	73.89%
Neutral	77.58%	71.40%

Table 7.9: Accuracy comparison with tap.aylien.com API

7.3Research Limitations

Some of the limitations faced during the research are mentioned below.

Twitter feeds extracted during some seasonal months (for example in Wesak Season), most of the comments were not worth for analyzing for customer satisfaction as they were about some decorations, travels etc. Therefore, this kind of tweets especially taken during seasonal months were removed from the dataset.

7.4 Suggestions for Further Research

Four dictionaries were involved in this research work and if they get fine-tuned further, the accuracy may be increased more. Spelling checker also can be used and see whether the accuracy can be enhanced further.

This research was done in order to analyze all tweets together, but if this framework can be enhanced to get the customer satisfaction for products, packages etc. and this would be a great support for the company to evaluate these individual items.

In this research work, it has only been considered the tweets tagged under English language, but there are many tweets which might be worth for analyzing, were written in Sinhala language. The framework developed can be further extended to read these tweets also.

The accuracy of this work can be compared and evaluated with the following approaches also.

- Using machine learning methods to train a model
- Using a hybrid approach (with both machine learning and lexicon dictionaries)

7.5 Summary

This chapter shows that for all three classes (Positives, Negative and Neutral) have been classified better with fine-tuned Lexicon and Stop Words Dictionary with Slang replacements. Also, it shows that hybrid approach (Lexicon and SentiWordNet together) has not performed well on the dataset.

If positive comments needto be extracted along, it is better to exclude Emoticon Dictionary as it shows the highest accuracy without that dictionary. If negative comments are taken in to consideration, it would be better to go with Lexicon and Emoticon Dictionaries after replacing the slangs. This chapter also discusses on the research limitations and the future research works also.

References

[1] Telecommunication Regulatory Commission of Sri Lanka (TRCSL), “Financial Analysis of the Telecom Sector,”June, 2017.

[2] Ed. King, *Staying In Droves: How to Win the Customer Retention Revolution*, 2009.

[3] Bamfo, A. “Exploring the relationship between customer satisfaction and loyalty in the mobiletelecommunication industry in Ghana,”*Indian Journal of Economics and Business*, vol. 8, no. 20, pp. 299-311, 2009.

[4] Metzler, K., & Hinterhuber, H. “How to make product development projects more successful byintegrating Kano's model of customer satisfaction into quality function deployment,”*Technovation*, vol. 18, no. 1, pp.25-38, 1998.

[5]Judith Lamont, P. “Customer sentiment analysis: A shift to customer service,” *KMWorld Magazine*, vol. 22, no. 2, pp.8, 2013.

[6] Manasee Godsay. “The Process of Sentiment Analysis: A Study,”*International Journal of Computer Applications*, vol. 126, no. 7, pp. 26-30, 2015.

[7] L. Almuqren and A. I. Cristea, “Twitter analysis to predict the satisfaction of telecom company customers,” in *Late-breaking Results, Demos, Doctoral Consortium, Workshops Proceedings and Creative Track of the 27th ACM Conference on Hypertext and Social Media*, 2016.

[8] W. Kasper and M. Vela, “Sentiment analysis for hotel reviews,”in*Proceedings of the Computational Linguistics-Applications Conference*, 2011.

[9]C. Ramasubramanian and R. Ramya, “Effective Pre-Processing Activities in Text Miningusing Improved Porter’s Stemming Algorithm,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 12, 2013.

[10] P. Willett, “The Porter stemming algorithm: then and now,” Program: electronic library and information systems, vol. 40, no. 3, pp.219-223, 2006.

[11] G. Chakraborty, M. Pagou and S. Garla, *Text Mining and Analysis: Practical Methods, Examples, and Case Studies Using SAS*. Cary, North Carolina, USA, 2013.

[12] S. Kolkur, G. Dantal and R. Mahe, “Study of Different Levels for Sentiment Analysis,” *International Journal of Current Engineering and Technology*,vol. 5, no. 2, 2015.

[13] B. Liu and L. Zhang, “A Survey of Opinion Mining and Sentiment Analysis,”in*Proceedings of Mining Text Data*, 2013.

[14] P. Patil and P. Yalagi, “Sentiment Analysis Levels and Techniques: A Survey,” *International Journal of Innovations in Engineering and Technology*, vol. 6, no. 4, pp. 523-528, 2016.

[15]W. Medhat, A. Hassan and H. Korashy, “Sentiment analysis algorithms and applications: A survey,” *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093-1113, 2014.

[16] V. B. Vaghela and B. M. Jadav, “Analysis of Various Sentiment ClassificationTechniques,” *International Journal of Computer Applications*, vol. 140, no. 3, pp. 22-27, 2016.

[17] M. W. Berry, A. Mohammed and B.W. Yap, “Soft Computing in Data Science,”in*Proceedings of Soft Computing in Data Science*,Putrajaya, Malaysia, 2015.

[18] D. L. Olson andD. Delen, “Performance Evaluation for Predictive Modeling,” in *Advanced Data Mining Techniques*, Springer-Verlag Berlin Heidelberg, pp. 137-139.

[19]F. M. Kundi, A. Khan, S. Ahmad, M. Z. Asghar, “Lexicon-Based Sentiment Analysis in the Social,” *Journal of Basic and Applied Scientific Research*, vol. 4, no. 6, pp. 238-248, 2014.

[20] A. Esuli ,F. Sebastiani, “Senti Word Net: A Publicly Available Lexical Resource for Opinion Mining,” In*Proceedings of International Conference on Language Resources and Evaluation (LREC)*, 2006, pp. 417-422.

[21] S. M. Vohira and J. B. Teraiya, “A Comparative Study of Sentiment Analysis Techniques,” *Journal of Information, Knowledge and Research in Computer Engineering*, vol. 2, no. 2, pp. 313-317, 2013.

Appendixes

Appendix A – R code for using only Lexicon Dictionary

```
#Install & Load Required R packages
#install.packages("data.table")
library(data.table)
library(qdapRegex)
library(plyr)
library(stringr)
library(qdap)


#Import Twitter Feeds
x <- fread ("C:\\Users\\Sachi\\Desktop\\MscProject\\TwitterFeeds\\TestData01.txt",
header = TRUE, select = c("Text","Rating"))

#Remove Duplicate tweets
x <- x[!duplicated(x), ]

#Duplicate "Text" field column
x$OriginalText = x$Text

#Lower all the letters
x$Text <- tolower(x$Text)

#Text Preprocessing
##Removal of Retweets
x$Text <- gsub("RT @[a-z,A-Z]*:", "", x$Text)

##Removal of HTML Links - Need qdapRegex Package to use rm_url
x$Text <- rm_url(x$Text, pattern=pastex("@rm_twitter_url", "@rm_url"))

##Removal of @People
x$Text <- gsub("@\\w+", "", x$Text)

##Removal of Special Characters ?& .
x$Text <- gsub("?", " ", x$Text, fixed = TRUE)
x$Text <- gsub(".", " ", x$Text, fixed = TRUE)
x$Text <- gsub("!", " ", x$Text, fixed = TRUE)
x$Text <- gsub("\"", " ", x$Text, fixed = TRUE)


#Text Refinement
#Remove Stop words
rm_words <- function(string, words) {
stopifnot(is.character(string), is.character(words))
spltted<- strsplit(string, " ", fixed = TRUE) # fixed = TRUE for speedup
vapply(spltted, function(x) paste(x[!tolower(x) %in% words], collapse = " "),
character(1))
```

```
}

x$Text <-  rm_words(x$Text, tm::stopwords("en"))


#Function for Positive & Negative Words match
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
require(plyr)
require(stringr)


scores = laply(sentences, function(sentence, pos.words, neg.words) {

   # convert to lower case:
sentence = tolower(sentence)

   # split into words. str_split is in the stringr package
   word.list = str_split(sentence, '\\s+')
   # sometimes a list() is one level of hierarchy too much
words = unlist(word.list)

   # compare our words to the dictionaries of positive & negative terms
   pos.matches = match(words, pos.words)
   neg.matches = match(words, neg.words)

   # match() returns the position of the matched term or NA
   # we just want a TRUE/FALSE:
   pos.matches = !is.na(pos.matches)
   neg.matches = !is.na(neg.matches)

   # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():
score = sum(pos.matches) - sum(neg.matches)

return(score)
  }, pos.words, neg.words, .progress=.progress )

  scores.df = data.frame(score=scores, text=sentences)
return(scores.df)
}

#Import Positive & Negative Words
pos_words <-
scan("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\Positive.txt",
what='character', comment.char=';')

neg_words <-
scan("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\Negative.txt",
what='character', comment.char=';')
```

```
#Add additional words to dictionaries
#neg_words = c(neg_words,'no')

#Score based on Positive & Negative words
result1 <- score.sentiment( x$Text, pos_words, neg_words)

#Calculate Lexicon Score
x$Lexicon = result1$score

#Calculate Total Score
x$TotalScore = (x$Lexicon)

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))

Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
```

```
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+P
erformance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step01_Lexicon = Performance_matrix
Performance_matrix_Step01_Lexicon
```

```
#Step 01 - Run Python Script
import pandas as pd

x = pd.read_csv('C:/Users/Sachi/Desktop/MscProject/TwitterFeeds/TestData01.txt',
sep='\t' , encoding='latin-1', skiprows=1, names = ["Text", "Rating"])

x = x.fillna('')

def sentiwordnet_python(doc):
import nltk
from nltk.corpus import sentiwordnet as swn
   #doc=   "Nice and friendly place with excellent food and friendly and helpful staff.
You need a car though. The children wants to go back! Playground and animals
entertained them and they felt like at home. I also recommend the dinner! Great value
for the price!"
sentences = nltk.sent_tokenize(doc)
stokens = [nltk.word_tokenize(sent) for sent in sentences]
taggedlist=[]
for stoken in stokens:
taggedlist.append(nltk.pos_tag(stoken))
wnl = nltk.WordNetLemmatizer()

   score_list=[]
for idx,taggedsent in enumerate(taggedlist):
      score_list.append([])
for idx2,t in enumerate(taggedsent):
newtag=''
lemmatized=wnl.lemmatize(t[0])
if t[1].startswith('NN'):
newtag='n'
elif t[1].startswith('JJ'):
newtag='a'
elif t[1].startswith('V'):
newtag='v'
elif t[1].startswith('R'):
newtag='r'
else:
newtag=''
if(newtag!=''):
synsets = list(swn.senti_synsets(lemmatized, newtag))
            #Getting average of all possible sentiments, as you requested
score=0
if(len(synsets)>0):
for syn in synsets:
score+=syn.pos_score()-syn.neg_score()
               score_list[idx].append(score/len(synsets))

   #print(score_list)
```

```
   sentence_sentiment=[]

for score_sent in score_list:
if len(score_sent)>0:
          sentence_sentiment.append(sum([word_score for word_score in
score_sent])/len(score_sent))
      #print("Sentiment for each sentence for:"+doc)
      #print(sentence_sentiment)
return sentence_sentiment

for row in x.itertuples():
x['SentiWordNetScore'] = x.apply(lambda row: sentiwordnet_python(row.Text),
axis=1)

x.to_csv('C:/Users/Sachi/Desktop/MscProject/Outputs/step01_sentiwordnet.txt',
sep='\t', encoding='latin-1')




#Step 02 - Evaluation on R

#Calculate Sentiwordnet
y <- fread ("C:/Users/Sachi/Desktop/MscProject/Outputs/step01_sentiwordnet.txt",
header = TRUE, select = c("Text", "Rating", "SentiWordNetScore"))

#Remove Null values
y$SentiWordNetScore <- gsub("[[]]", "0.0", y$SentiWordNetScore)

#Remove Special characters from score column
y$SentiWordNetScore <- gsub("[[]", "", y$SentiWordNetScore)
y$SentiWordNetScore <- gsub("[]]", "", y$SentiWordNetScore)


#Assign to x
x <- y

#Convert SentiWordNet Score to numeric value
x$SentiWordNetScore = as.numeric(x$SentiWordNetScore)

#Replace NUll values to 01m1q
x$SentiWordNetScore = ifelse(is.na(x$SentiWordNetScore) == 'TRUE', 0.00,
x$SentiWordNetScore)

#Calculate Total Score
x$TotalScore = (x$SentiWordNetScore)

#Assign Predicted Rating
```

```
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))


Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
```

```
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+P
erformance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step01_SentiWordNet = Performance_matrix
Performance_matrix_Step01_SentiWordNet
```

Appendix C – R code for using Lexicon Dictionary with Stop Words Amendments

```
#Import Twitter Feeds
x <- fread ("C:\\Users\\Sachi\\Desktop\\MscProject\\TwitterFeeds\\TestData01.txt",
header = TRUE, select = c("Text","Rating"))

#Remove Duplicate tweets
x <- x[!duplicated(x), ]

#Duplicate "Text" field column
x$OriginalText = x$Text

#Lower all the letters
x$Text <- tolower(x$Text)

#Text Preprocessing
##Removal of Retweets
x$Text <- gsub("RT @[a-z,A-Z]*:", "", x$Text)

##Removal of HTML Links - Need qdapRegex Package to use rm_url
x$Text <- rm_url(x$Text, pattern=pastex("@rm_twitter_url", "@rm_url"))

##Removal of @People
x$Text <- gsub("@\\w+", "", x$Text)

##Removal of Special Characters ?& .
x$Text <- gsub("?", " ", x$Text, fixed = TRUE)
x$Text <- gsub(".", " ", x$Text, fixed = TRUE)
x$Text <- gsub("!", " ", x$Text, fixed = TRUE)
x$Text <- gsub("\"", " ", x$Text, fixed = TRUE)


#Text Refinement
#Remove Stop words
rm_words <- function(string, words) {
stopifnot(is.character(string), is.character(words))
spltted<- strsplit(string, " ", fixed = TRUE) # fixed = TRUE for speedup
vapply(spltted, function(x) paste(x[!tolower(x) %in% words], collapse = " "),
character(1))
}

#Customize the stop words
exceptions<- c("no")
my_stopwords <- setdiff(tm::stopwords("en"), exceptions)

#x$Text <-  rm_words(x$Text, tm::stopwords("en"))
x$Text <-  rm_words(x$Text, my_stopwords)

#Score based on Positive & Negative words
result1 <- score.sentiment( x$Text, pos_words, neg_words)
```

```
#Calculate Lexicon Score
x$Lexicon = result1$score

#Calculate Total Score
x$TotalScore = (x$Lexicon)

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))

Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
```

```
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+P
erformance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step03_Lexicon_StopWords = Performance_matrix
Performance_matrix_Step03_Lexicon_StopWords
```

Appendix D – R code for comparing Lexicon Dictionary with SentiWordNet

```
#Step 01-Input to Python

library(NLP)
library(tm)
library(regexr)

# Loading Negative words into two files
# Here *words have to be removed such as d*mn
neg_words_part1 <-
scan("C:/Users/Sachi/Desktop/MscProject/Dictionaries/Negative_Part1.txt",
what='character', comment.char=';')
neg_words_part2 <-
scan("C:/Users/Sachi/Desktop/MscProject/Dictionaries/Negative_Part2.txt",
what='character', comment.char=';')

# Remove positive words
a = removeWords(x$Text,pos_words)

# Remove negative words
b = removeWords(a,neg_words_part1)
x$SentiWordNet_Text = removeWords(b,neg_words_part2)


# Export data to a text file
write.table(x, "C:/Users/Sachi/Desktop/MscProject/Outputs/rdata_Step04.txt",
sep="\t",row.names=F)


#Step 02 - Run Python Script
import pandas as pd

x = pd.read_csv('C:/Users/Sachi/Desktop/MscProject/Outputs/rdata_Step04.txt',
sep='\t' , encoding='latin-1', skiprows=1, names = ["Text", "Rating", "OriginalText",
"Lexicon", "TotalScore", "PredictedRating", "SentiWordNet_Text"])

x = x.fillna('')

def sentiwordnet_python(doc):
import nltk
from nltk.corpus import sentiwordnet as swn
   #doc=   "Nice and friendly place with excellent food and friendly and helpful staff.
You need a car though. The children wants to go back! Playground and animals
entertained them and they felt like at home. I also recommend the dinner! Great value
for the price!"
sentences = nltk.sent_tokenize(doc)
stokens = [nltk.word_tokenize(sent) for sent in sentences]
taggedlist=[]
for stoken in stokens:
```

```
taggedlist.append(nltk.pos_tag(stoken))
wnl = nltk.WordNetLemmatizer()

   score_list=[]
for idx,taggedsent in enumerate(taggedlist):
       score_list.append([])
for idx2,t in enumerate(taggedsent):
newtag=''
lemmatized=wnl.lemmatize(t[0])
if t[1].startswith('NN'):
newtag='n'
elif t[1].startswith('JJ'):
newtag='a'
elif t[1].startswith('V'):
newtag='v'
elif t[1].startswith('R'):
newtag='r'
else:
newtag=''
if(newtag!=''):
synsets = list(swn.senti_synsets(lemmatized, newtag))
            #Getting average of all possible sentiments, as you requested
score=0
if(len(synsets)>0):
for syn in synsets:
score+=syn.pos_score()-syn.neg_score()
                score_list[idx].append(score/len(synsets))

   #print(score_list)
   sentence_sentiment=[]

for score_sent in score_list:
if len(score_sent)>0:
          sentence_sentiment.append(sum([word_score for word_score in
score_sent])/len(score_sent))
       #print("Sentiment for each sentence for:"+doc)
       #print(sentence_sentiment)
return sentence_sentiment

for row in x.itertuples():
x['SentiWordNetScore'] = x.apply(lambda row:
sentiwordnet_python(row.SentiWordNet_Text), axis=1)

x.to_csv('C:/Users/Sachi/Desktop/MscProject/Outputs/step04_sentiwordnet.txt',
sep='\t', encoding='latin-1')
```

```
#Step 03 - Evaluation on R

#Calculate Sentiwordnet
y <- fread ("C:/Users/Sachi/Desktop/MscProject/Outputs/step04_sentiwordnet.txt",
header = TRUE, select = c("Text", "Rating", "OriginalText", "Lexicon", "TotalScore",
"PredictedRating", "SentiWordNet_Text", "SentiWordNetScore"))

#Remove Null values
y$SentiWordNetScore <- gsub("[[]]", "0.0", y$SentiWordNetScore)

#Remove Special characters from score column
y$SentiWordNetScore <- gsub("[[]", "", y$SentiWordNetScore)
y$SentiWordNetScore <- gsub("[]]", "", y$SentiWordNetScore)


#Assign to x
x <- y

#Convert SentiWordNet Score to numeric value
x$SentiWordNetScore = as.numeric(x$SentiWordNetScore)

#Replace NULL values to 0
x$SentiWordNetScore = ifelse(is.na(x$SentiWordNetScore) == 'TRUE', 0.00,
x$SentiWordNetScore)

#Calculate Total Score
x$TotalScore = (x$Lexicon)+(x$SentiWordNetScore)

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))


Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
```

```
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+P
erformance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step04_Lexicon_SentiWordNet = Performance_matrix
Performance_matrix_Step04_Lexicon_SentiWordNet
```

Appendix E – R code for using Lexicon Dictionary with Slang Replacements

```
#Import Slang Dictionary-special charators
slangs_sc <- fread
("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\slangs_specialchar.csv",
header = TRUE, select = c("Slang","Slang_Desc"))

slangs_other <- fread
("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\slangs_others.csv", header =
TRUE, select = c("Slang","Slang_Desc"))

slangs_sc$Slang <- tolower(slangs_sc$Slang)
slangs_other$Slang <- tolower(slangs_other$Slang)

z <- x

#Duplicate "Text" field column
#z$OriginalText = z$Text


#Replace twitter feeds with slangs
for (q in 1:nrow(z))
for(t in 1:nrow(slangs_sc))
   #For special characters
  { z[q,1] <- gsub((str_c(" \\", slangs_sc[t,1]," ")), str_c(" ", slangs_sc[t,2]," "),
z[q,1])}

for (q in 1:nrow(z))
z[q,1] <- gsub(":\\)", "Positive", z[q,1])

for (q in 1:nrow(z))
z[q,1] <- gsub(":\\(", "Negative", z[q,1])

for (q in 1:nrow(z))
z[q,1] <- gsub(":p", "Positive", z[q,1])

for (q in 1:nrow(z))
z[q,1] <- gsub(":d", "Positive", z[q,1])

#for (q in 1:nrow(z))
#z[q,1] <- gsub("xd", "Positive", z[q,1])



for (q in 1:nrow(z))
for(t in 1:nrow(slangs_other))
   #For others characters
  { z[q,1] <- gsub((str_c(" ",slangs_other[t,1]," ")),  (str_c(" ", slangs_other[t,2]," ")),
z[q,1])}
```

```
#Replace twitter feeds with slangs
#for (q in 1:nrow(z))
# for(t in 1:nrow(slangs))
#  #For special characters
# if (substring(slangs[t,1], 1, 1) %in% c("$","*","/","<",">","?","@","^",":"))
#{ z[q,1] <- gsub((str_c(" \\", slangs[t,1]," ")), str_c(" ", slangs[t,2]," "), z[q,1])} else {
z[q,1] <- gsub((str_c(" ",slangs[t,1]," ")),  (str_c(" ", slangs[t,2]," ")), z[q,1])}


x$Slangs_Text <- tolower(z$Text)

#Text Preprocessing
##Removal of Retweets
x$Slangs_Text <- gsub("RT @[a-z,A-Z]*:", "", x$Slangs_Text)

##Removal of HTML Links - Need qdapRegex Package to use rm_url
x$Slangs_Text <- rm_url(x$Slangs_Text, pattern=pastex("@rm_twitter_url",
"@rm_url"))

##Removal of @People
x$Slangs_Text <- gsub("@\\w+", "", x$Slangs_Text)

##Removal of Special Characters ?& .
x$Slangs_Text <- gsub("?", " ", x$Slangs_Text, fixed = TRUE)
x$Slangs_Text <- gsub(".", " ", x$Slangs_Text, fixed = TRUE)
x$Slangs_Text <- gsub("!", " ", x$Slangs_Text, fixed = TRUE)
x$Slangs_Text <- gsub("\"", " ", x$Slangs_Text, fixed = TRUE)

#Text Refinement
#Remove Stop words
rm_words <- function(string, words) {
stopifnot(is.character(string), is.character(words))
spltted<- strsplit(string, " ", fixed = TRUE) # fixed = TRUE for speedup
vapply(spltted, function(x) paste(x[!tolower(x) %in% words], collapse = " "),
character(1))
}
#Customize the stop words
exceptions<- c("no")
my_stopwords <- setdiff(tm::stopwords("en"), exceptions)

#x$Text <-  rm_words(x$Text, tm::stopwords("en"))
x$Slangs_Text <-  rm_words(x$Slangs_Text, my_stopwords)

#Score based on Positive & Negative words
result1 <- score.sentiment(x$Slangs_Text, pos_words, neg_words)

#Calculate Lexicon Score
x$Lexicon_Slangs_Score = result1$score
```

```
#Calculate Total Score
x$TotalScore = (x$Lexicon_Slangs_Score)

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))


Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))
```

```
#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+P
erformance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step05_Lexicon_Slang = Performance_matrix
Performance_matrix_Step05_Lexicon_Slang
```

Appendix F – R code for using Lexicon Dictionary with Slang Replacements and Emoticons

```
#Import Emoticon Dictionary
emoticons<- fread
("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\Emoticons.txt", header =
TRUE, select = c("Emoticon","Sentiment_score"))

emoticons$Tag <- ""

#Tag Positive & Negative Emoticons
for (q in 1:nrow(emoticons))
  emoticons[q,3] <-
ifelse(emoticons[q,2]>0,"Positive",ifelse(emoticons[q,2]==0,"Neutral","Negative"))

emoticons_pos <-  emoticons[emoticons$Tag %in% "Positive"]
emoticons_neg <-  emoticons[emoticons$Tag %in% "Negative"]


#Score based on Emoticons
result2 <- score.sentiment(x$Text, tolower(emoticons_pos$Emoticon),
tolower(emoticons_neg$Emoticon))

#Calculate Lexicon Score
x$Emoticons = result2$score

#Calculate Total Score
x$TotalScore = x$Lexicon_Slangs_Score  + x$Emoticons

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))

Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")

#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
```

```
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+P
erformance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step06_Lexicon_Slang_Emoticons = Performance_matrix
Performance_matrix_Step06_Lexicon_Slang_Emoticons
```