**M S Ramaiah Institute of Technology**

(An Autonomous Institute, Affiliated to VTU)

MSR nagar, MSRIT post, Bangalore-54

A Dissertation Report on

PREDICTION OF MOVIE SUCCESS USING SENTIMENT ANALYSIS OF TWEETS

Submitted by

**Raksha Singh** 1MS12CS089

**Sweety Kumari** 1MS12CS119

**Priya Sharma** 1MS12CS144

**Aishwarya S S** 1MS12CS142

*in partial fulfillment for the award of the degree of*

# *Bachelor of Engineering in Computer Science & Engineering*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**M.S.RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

[www.msrit.edu](http://www.msrit.edu), **May 2015**

# Abstract

Now-a-days social networking sites are at the boom, so large amount of data is generated. Millions of people are sharing their views daily on micro blogging sites, since it contains short and simple expressions. In this project, we will discuss about a paradigm to extract the sentiment from a famous micro blogging service, Twitter, where users post their opinions for everything. In this project, we will discuss the existing analysis of twitter dataset with data mining approach such as use of Sentiment analysis algorithm using machine learning algorithms. An approach is introduced that automatically classifies the sentiments of Tweets taken from Twitter dataset. These messages or tweets are classified as positive, negative or neutral with respect to a query term. This is very useful for the companies who want to know the feedback about their product brands or the customers who want to search the opinion from others about product before purchase. We will use machine learning algorithms for classifying the sentiment of Twitter messages using distant supervision. The training data consists of Twitter messages with emoticons, acronyms which are used as noisy labels.

1 **INTRODUCTION**

* 1. General Introduction……………….
  2. Statement of the Problem…………..
  3. Objectives of the project……………
  4. Project deliverables……………
  5. Current Scope………………………
  6. Future Scope……………………….

1. **PROJECT ORGANIZATION**
   1. Software Process Models
   2. Roles and Responsibilities
2. **LITERATURE SURVEY**

3.1….Introduction

3.2…Main Body

3.3 Conclusion of Survey

1. **SOFTWARE REQUIREMENT SPECIFICATIONS**

5.1 Product Overview

5.2 External Interface Requirements

5.2.1 User Interfaces

5.2.2 Hardware Interfaces

5.2.3 Software Interfaces

5.2.4 Communication Interfaces

5.3 Functional Requirements

5.3.1 Functional Requirement 1.1

:

5.3.n Functional Requirement 1.n

5.4 Software System Attributes

5.4.1 Reliability

5.4.2 Availability

5.4.3 Security

5.4.4 Portability

5.4.5 Maintainability

5.4.6 Performance

5.5 Performance Requirements

5.6 Database Requirement

5.7 Design Constraints

5.8 Other Requirements

1. **DESIGN**
   1. Introduction
   2. Architecture Design
   3. Graphical User Interface
   4. Class Diagram and Classes (represent Inheritance, Aggregation and Association)
   5. Sequence Diagram
   6. Data flow diagram
   7. Metric calculation
2. **IMPLEMENTATION**
   1. Tools Introduction
   2. Technology Introduction
   3. Overall view of the project in terms of implementation
   4. Explanation of Algorithm and how it is been implemented
   5. Information about the implementation of Modules
3. **TESTING**

**7.1** Results and Snapshots

1. **CONCLUSION & SCOPE FOR FUTURE WORK**
2. **REFERENCES**

**1. GENERAL INTRODUCTION**

**1.1 Introduction**

The emergence of social media has given web users a venue for expressing and sharing their thoughts and opinions on all kinds of topics and events. Twitter, with nearly 600 million users1 and over 250 million messages per day, has quickly become a gold mine for organizations to monitor their reputation and brands by extracting and analyzing the sentiment of the Tweets posted by the public about them, their markets, and competitors. Sentiment analysis over Twitter data and other similar microblogs faces several new challenges due to the typical short length and irregular structure of such content. Two main research directions can be identified in the literature of sentiment analysis on microblogs. First direction is concerned with finding new methods to run such analysis, such as performing sentiment label propagation on Twitter follower graphs , and employing social relations for user-level sentiment analysis. The second direction is focused on identifying new sets of features to add to the trained model for sentiment identification, such as microblogging features including hashtags, emoticons, the presence of intensifiers such as all-caps and character repetitions etc., and sentiment-topic features.

By investigating movie features derived from the semantic conceptual representation of the entities that appear in tweets. The semantic features consist of the semantic concepts (e.g. “person”, “company”, “city”) that represent the entities (e.g. “Steve Jobs”, “Vodafone”, “London”) extracted from tweets. The rational behind introducing these features is that certain entities and concepts tend to have a more consistent correlation with positive or negative sentiment. Knowing these correlations can help determining the sentiment of semantically relevant or similar entities, and thus increasing accuracy of sentiment analysis.

**1.1 Statement**

To provide a Sentiment analysis system for movies’ review classification, that may be helpful to analyze the information in the form of the number of tweets where opinions are highly unstructured and are either positive or negative.

**1.2 Objectives**

To implement an algorithm for automatic classification of text into positive, negative or neutral.

Sentiment analysis to determine the attitude of mass is positive, negative or neutral towards the subject of interest.

Graphical representation of sentiment in the form of graph.

**1.3 Deliverables**

Project Proposal

Final presentation

Final report

**1.4 Current Scope**

It can involve social conversations and also direct and indirect feedback (such surveys, contact-center notes, and warranty and insurance claims), online news, presentations, even scientific papers. Any information source that captures subjective information.

Sentiment analysis lets marketers (and market researchers, customer service and support staff, product managers, etc.) get at root causes, at explanations of behaviors that are captured in transaction and tracking records. Sentiment analysis means better targeted marketing, faster detection of opportunties and threats, brand-reputation protection, and the ultimate aim, profit.

Interesting choice of platforms. Facebook and Twitter are major sources of sentiment (and also of complementary social connectedness data). Facebook and Twitter accounts have profile data attached to them, but nothing that matches the detailed, usably-structured information you can find on LinkedIn. Google is the ultimate information-access engine, capable of bringing together information from a huge variety of disparate sources, including sentiment information such as product, restaurant, and hotel ratings, although when corporations wish to find, mine, and exploit sentiment they need to turn to deeper BI and analytics tools.

There’s no one-size-fits-all sentiment solution, not Google or one of the several as-a-service solutions out there or any of the capable analysis workbenches or social-media analytics tools. Instead, there’s a whole spectrum of sentiment sources and analysis possibilities.

**1.5 Future Scope**

Sentiment analysis methods till now have been used to detect the polarity in the thoughts and opinions of all the users that access social media. Researchers and Businesses are very interested to understand the thoughts of people and how they respond to everything happening around them. Companies use this to evaluate their advertisement campaigns and to improve their products.

There is a lot of scope in analyzing the video and images on the web. Now a days, with the advent of Facebook, Instagram and Video vines people are expressing their thoughts with pictures and videos along with text. Sentiment analysis will have to pace up with this change. Tools which are helping companies to change strategies based on Facebook and Twitter will also have to accommodate the number of likes and re-tweets that the thought is generating on the Social media. People follow and unfollow people and comments on Social Media but never comment so there is scope in analyzing these aspects of the Web as well.

As new text types appear on the Social Web, the techniques to pre-process, as well as to tackle their informal style must be adapted, so as to obtain acceptable levels of performance of the sentiment analysis systems. The field will have to combine with effective computing, psychology and neuroscience to converge on a unified approach to understanding the sentiments better.

**2. PROJECT ORGANIZATION**

**2.1 Software Process Models.**

We have adopted RUP because of the following reasons

Easily resolves risks

Supports iterative development.

RUP Stands for "Rational Unified Process." RUP is a software development process from Rational, a division of IBM. It divides the development process into four distinct phases that each involve business modeling, analysis and design, implementation, testing, and deployment. The four phases are:

1. **Inception** - The idea for the project is stated. The development team determines if the project is worth pursuing and what resources will be needed.
2. **Elaboration** - The project's architecture and required resources are further evaluated. Developers consider possible applications of the software and costs associated with the development.
3. **Construction** - The project is developed and completed. The software is designed, written, and tested.
4. **Transition** - The software is released to the public. Final adjustments or updates are made based on feedback from end users.

The RUP development methodology provides a structured way for companies to envision create software programs. Since it provides a specific plan for each step of the development process, it helps prevent resources from being wasted and reduces unexpected development costs.

2.2 roles and responsibilities

Each and every member in our group referred to many research papers and decided to perform sentiment anlaysis using RStudio. All the activities were performed in a group and has the effort of each individual.

For preparing the reports : Aishwarya and Sweety took the responsibility of finishing the report for the project. Priya and Raksha took the responsibility of preparing the paper. Power point presentation were made by all the members in the team.

**3. LITERATURE SURVEY**

**3.1 Introduction**

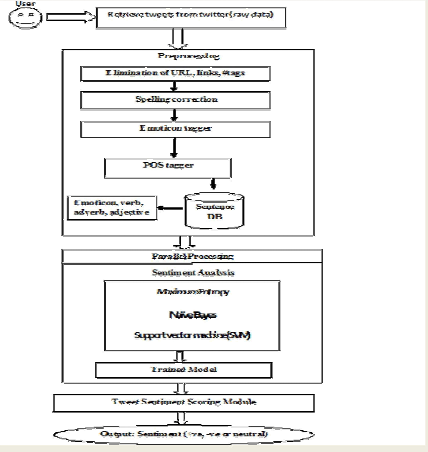
We know that there are almost 111 micro blogging sites. Micro blogging websites are nothing but social media site to which user makes short and frequent posts. Twitter is one of the famous micro blogging services where user can read and post messages which are 148 characters in length. Twitter messages are also called as Tweets. We will use these tweets as raw data. We will use a method that automatically extracts tweets into positive, negative or neutral sentiments. By using the sentiment analysis the customer can know the feedback about the product or services before making a purchase. The company can use sentiment analysis to know the opinion of customers about their products, so that they can analyze customer satisfaction and according to that they can improve their product. Sentiment analysis has become one of popular research area in computational linguistics, because of the explosion of sentiment information from social web sites (i.e., Twitter and Facebook), online forums, and blogs as in paper. In paper, there is use of two resources : 1) a hand annotated dictionary for emoticons 2) an acronym dictionary gathered from web. The approach is the use of different machine learning classifiers and feature extractors. Naive Bayes, Maximum Entropy (MaxEnt), and Support Vector Machines (SVM) are the machine learning classifiers. Unigrams, bigrams, unigrams and bigrams, and unigrams with part of speech tags are the feature extractors.

In paper , there is use of two resources : 1) a hand annotated dictionary for emoticons 2) an acronym dictionary gathered

from web. The approach is the use of different machine learning classifiers and feature extractors. Naive Bayes, Maximum Entropy (MaxEnt), and Support Vector Machines (SVM) are the machine learning classifiers. Unigrams, bigrams, unigramsand bigrams, and unigrams with part of speech tags are the feature extractors.There are three different machine learning algorithms who achieved great success for text categorization but we’ll be using Natural Language Processor(NLP).

**3.2 Main Body**

Sentiments are the words or sentences that represent view or opinion that is held or expressed that can be positive, negative or neutral. We are going to propose a novel hybrid approach involving both corpus-based and dictionary-based techniques, which will find the semantic orientation of the sentiments words in tweets. We will also consider features like emoticons,neutralization, negation handling and capitalization as they have recently become a huge part of the internet language.The proposed Sentiment Analysis on twitter data is based on two important parts viz Data Extraction, pre-processing of extracted data and classification.



Fig

To uncover the sentiments, we will first extract the opinion words from tweets and then we find out their orientation, i.e., to decide whether each sentiment word reflects exaggerated and self-indulgent feelings of tenderness, sadness, or nostalgia.

The following steps will expound the process of the proposed system are :

1. Retrieval of tweets

2. Pre-processing of extracted data

3. Parallel processing

4. Sentiment scoring module

5. Output sentiment

These steps are explained below:

**1. Retrieval of tweets :**

As twitter is the most exaggerated part of social networking site, it consists of various blogs which are related to various topics worldwide. Instead of taking whole blogs, we will rather search on particular topic and download all its web pages then extracted them in the form of text files by using mining tool i.e. Weka which provides sentiment classifier..

**2. Pre-processing of extracted data:**

After retrieval of tweets Sentiment analysis tool is applied on raw tweets but in most of cases results to very poor performance.Therefore, preprocessing techniques are necessary for obtaining better results as given extract tweets i.e. short messages from twitter which are used as raw data. This raw data needs to be preprocessed. So, preprocessing involves

following steps which constructs n-grams:

**i) Filtering:**

Filtering is nothing but cleaning of raw data. In this step, URL links (E.g. http://twitter.com), special words in twitter (e.g.“RT” which means ReTweet), user names in twitter (e.g. @Ron - @ symbol indicating a user name), emoticons are removed.

ii) Tokenization:

Tokenization is nothing but Segmentation of sentences. In this step, we will tokenize or segment text with the help of splitting text by spaces and punctuation marks to form container of words.

iii) Removal of Stopwords:

Articles such as “a”, “an”, “the” and other stopwords such as “to”, “of”, “is”, “are”, “this”, “for” removed in this step.

**iv) Construction of n-grams:**

Set of n-grams can make out of consecutive words. Negation words such as “no”, “not” is attached to a word which follows or precedes it. For Instance: “I do not like remix music” has two bigrams: “I do+not”, “do+not like”, “not+like remix music”. So the accuracy of the classification improves by such procedure, because negation plays an important role in sentiment analysis.

Paper represents that negation needs to be taken into account, because it is a very common linguistic construction that affects polarity.

**3. Sentiment scoring module:**

Prior polarity of words is the basic of our number of features. The dictionary is used in which English language words assigns a score to every word, between 1 (Negative) to 3 (Positive). So, this scoring module is going to determine score of sentiments in the sentiment analysis of data.

**4. Output sentiment**:

Based on the dictionary assignment of score, the proposed system interprets whether the tweet is positive, negative or neutral.

**3.3 Conclusion of Survey**

Twitter is a demandable micro blogging service which has been built to discover what is happening at any moment of time and anywhere in the world. In the survey, we found that social media related features can be used to predict sentiment in Twitter.

We will use two machine learning algorithms which will contribute to out perform three models namely unigram and feature based model . So, our proposed system concludes the sentiments of tweets which are extracted from twitter. The difficulty increases with the nuance and complexity of opinions expressed. Product reviews, etc are relatively easy. Books, movies, art, music are more difficult. We can also implement features like emoticons, neutralization, negation handling and capitalization/internationalization as they have recently become a huge part of the internet.

**4. SOFTWARE REQUIREMENT SPECIFICATION (SRS)**

**DATA COLLECTION**

Register Your App

In order to have access to Twitter data programmatically, we need to create an app that interacts with the Twitter API.

The first step is the registration of your app. In particular, you need to point your browser to [http://apps.twitter.com](http://apps.twitter.com/), log-in to Twitter (if you’re not already logged in) and register a new application. You can now choose a name and a description for your app (for example “Mining Demo” or similar). You will receive a *consumer key* and a *consumer secret*: these are application settings that should always be kept private. From the configuration page of your app, you can also require an access token and an access token secret. Similarly to the consumer keys, these strings must also be kept private: they provide the application access to Twitter on behalf of your account. The default permissions are read-only, which is all we need in our case, but if you decide to change your permission to provide writing features in your app, you must negotiate a new access token.

Important Note: there are rate limits in the use of the Twitter API, as well as limitations in case you want to provide a downloadable data-set, see:

<https://dev.twitter.com/overview/terms/agreement-and-policy>

<https://dev.twitter.com/rest/public/rate-limiting>

**Accessing the Data**

Twitter provides REST APIs you can use to interact with their service. There is also a bunch of Python-based clients out there that we can use without re-inventing the wheel. In particular, Tweepy in one of the most interesting and straightforward to use.

importtweepy

fromtweepy import OAuthHandler

consumer\_key = 'YOUR-CONSUMER-KEY'

consumer\_secret = 'YOUR-CONSUMER-SECRET'

access\_token = 'YOUR-ACCESS-TOKEN'

access\_secret = 'YOUR-ACCESS-SECRET'

auth = OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_secret)

api = tweepy.API(auth)

The api variable is now our entry point for most of the operations we can perform with Twitter.

For example, we can read our own timeline (i.e. our Twitter homepage) with:

|  |  |
| --- | --- |
|  | for status in tweepy.Cursor(api.home\_timeline).items(10):      # Process a single status      print(status.text) |

Tweepy provides the convenient Cursor interface to iterate through different types of objects. In the example above we’re using *10* to limit the number of tweets we’re reading, but we can of course access more. The status variable is an instance of the Status() class, a nice wrapper to access the data. What Tweepy doesn’t provide is the raw JSON response from the API, which could be useful if we want to store it and process it later.

**Streaming**

In case we want to “keep the connection open”, and gather all the upcoming tweets about a particular event, the streaming API is what we need. We need to extend the StreamListener() to customise the way we process the incoming data. A working example that gathers all the new tweets with the #python hashtag:

fromtweepy import Stream

fromtweepy.streaming import StreamListener

classMyListener(StreamListener):

    defon\_data(self, data):

        try:

            with open('python.json', 'a') as f:

                f.write(data)

                return True

        exceptBaseException as e:

            print("Error on\_data: %s" % str(e))

        return True

    defon\_error(self, status):

        print(status)

        return True

twitter\_stream = Stream(auth, MyListener())

twitter\_stream.filter(track=['#python'])

**Specific Requirements**

**4.1 External Interface Requirements**

**4.1.1 Hardware Interfaces:**

The application is intended to be a stand-alone, single-user system. The application will run on Windows operating system. No further hardware devices or interfaces will be required.

**4.1.2 Software Interfaces**

**Inputs**

The software will receive input from two sources. First, the user interface and second, the Twitter API. The user interface will supply the keywords, while the Twitter API will supply the Tweet text.

**Outputs**

The output will portray the current mood of the Twitter community on a given topic in the form of a simple graph.

Operating System

The software will run on Windows.

**4.1.3 User Interfaces**

The interface will meet the following requirements to conform to the users’ needs. It will be simple and easy to understand. Controls which allow the user to interact with the application will be clear and imply their functionality within the application. The interface will include user inputs as well as graphics. The graphics displayed to the user will provide a visual representation of the output produced.

**User Inputs (Mandatory)**

The user will be able to control the sentiment analysis of topics: by adding, editing, or removing keywords for each topic.

**Graphic: Topic Mood Gauge (Mandatory)**

This graphic will consist of a simple gauge which shows the current mood of the Twitter community on a given topic. This will be done by displaying the percentage of the positive, negative and neutral tweets.

**4.2 Functional Requirements**

**4.2.1 Retrieving Input**

The software will receive two inputs: keywords and Tweets.

● Keywords will be entered by the user for each topic.

● Tweets will be retrieved with the Twitter Streaming API.

**4.2.2 Real-Time Processing**

The software will take input, process data, and display output in real-time. This will enforce that the snapshot provided by the simple gauge is a current view of the Twitter community’s mood on the chosen topic.

**4.2.3 Sentiment Analysis**

Sentiment analysis will be performed on the user-specified keywords within the Tweet to determine the overall mood of the Tweet relative to the topic. The sentiment analysis will provide a negative, neutral, or positive numeric sentiment value.

**4.2.4 Output**

The software must output real time data in the form of a simple gauge. This output should be clear and easy to understand.

**4.3 Software System Attributes**

**4.3.1 Reliability**

The software will meet all of the functional requirements without any unexpected behavior. At no time should the graph output display incorrect or outdated information.

**4.3.2 Availability**

The software will be available at all times on the OS, as long as the device is in proper working order. The functionality of the software will depend on any external services such as internet access that are required. If those services are unavailable, the user should be alerted.

**4.3.3 Security**

The software should never disclose any personal information of Twitter users, and should collect no personal information from its own users.

**4.3.4 Maintainability**

The software should be written clearly and concisely. The code will be well documented. Particular care will be taken to design the software modularly to ensure that maintenance is easy.

**4.3.5 Portability**

This software will be designed to run On windows OS and forward compatible.

**4.4 Performance Requirements**

**4.4.1 Real-Time**

The software will provide up-to-date information, limited only by the rate of Twitter input. The gauge output should display the latest results at all times.

**4.4.2 Use Cases**

This software will serve as a tool of interest, providing users with the current mood of the Twitter community on any specified topic

**4.5 Database Requirements**

The tweets taken from Twitter will be stored on an excel spreadsheet. Excel is an excellent programme for storing large amounts of data as well as being easy to upload the data to R Studio .The data will have two columns, column one will have the score of the tweets ( pos, neg, very pos and very neg), column two will store the actual tweet content. Each row will represent an individual tweet.

**4.6 Design Constraints**

Twitter API has some limitations such as Twitter API can only return a fixed maximum amount of tweets (1500). The return of a maximum number of tweets may not be met sometimes as there are not enough tweets for the particular keyword.

**5. DESIGN**

**5.1 Introduction**

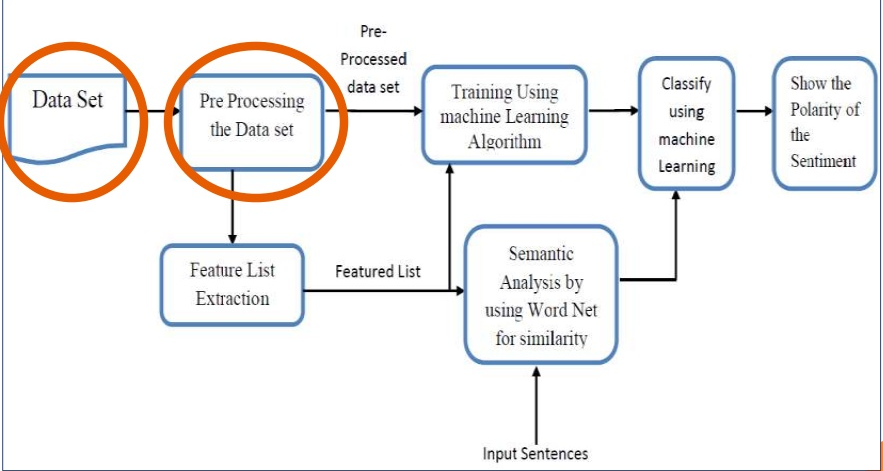
To do sentiment analysis ,we first need to extract tweets from twitter .For extracting tweets we are making a twitter API and then we are extracting live tweets. Tweets contain many redundant fields which we don’t need for processing ,so we ll do preprocessong to remove punctuation marks,numbers,whitespaces etc. After preprocessing we will do processing part .In this we are find the polarity of testing tweets .For this we are using naive bayes algorithm.At the end we are drawing bar graph to represent polarity of tweets.

Number of modules: 2

First module is used for preprocessing .

Second module is used for classification of tweets.

**5.2 Architecture Design**

****

**Explanation of each module:**

**DATASET AND PRE-PROCESSING :**

The tweets contain a lot of opinions about the data which are expressed in different ways by individuals . The twitters dataset used in this work is already labeled. Labeled dataset has a negative and positive polarity and thus the analysis of the data becomes easy. The raw data having polarity is highly susceptible to inconsistency and redundancy. It deals with the preparation that removes the repeated words and punctuations and improves efficiency of the data.

**FEATURE EXTRACTION :**

The improved dataset after pre- processing has a lot of distinctive properties. The feature extraction method, extracts the aspect (adjective) from the dataset. Later this adjective is used to show the positive and negative polarity in a sentence. Unigram model extracts the adjective and segregates it. It discards the preceding and successive word occurring with the adjective in the sentences. Example “painting Beautiful” through unigram model, only Beautiful is extracted from the sentence.

.

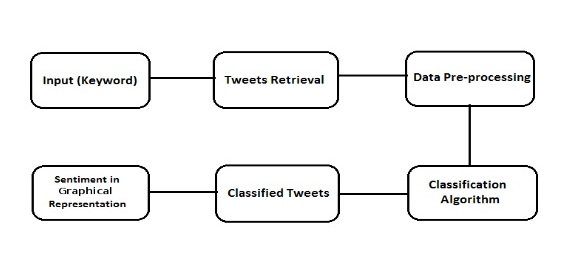
**SEMANTIC ANALYSIS** :

Semantic analysis is derived from the WordNet database where each term is associated with each other. This database is of English words which are linked together. If two words are close to each other, they are semantically similar. More specifically, we are able to determine synonym like similarity. The key task is to use the stored documents that contain terms and then check the similarity with the words that the user uses in their sentences. For example in the sentence ”I am happy” the word ‘’happy’’ being an adjective gets selected and is compared with the stored feature vector for synonyms. Let us assume 2 words; ‘glad’ and ‘satisfied’ tend to be very similar to the word ‘happy’. Now after the semantic analysis, ‘glad’ replaces ‘happy’ which gives a positive polarity.

* 1. **Graphical User Interface**

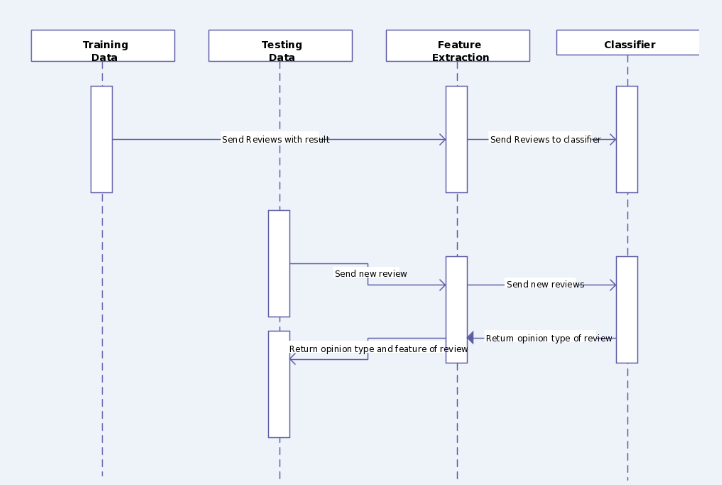
For GUI we are using R console. From R we are extracting tweets from twitter by making an twitter application .Then using various packages and function present in R we are doing preprocessing and processing of tweets.At the end we are representing polarity of tweets using Bargraph. We are also using saplumira for plotting piechart i.e graphical representation of output.

**5.4 Class Diagram and Classes**

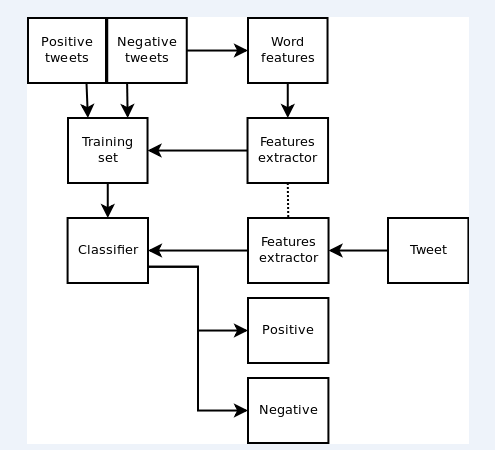
****

We are using 6 classes :

1. Input
2. Tweets Retrieval
3. Data pre-processing
4. Classification algorithm
5. Classified tweets
6. Sentiment in graphical representation
   1. **Sequence Diagram**

****

**5.6 Data flow diagram**

****

**5.7 Metric calculation**

Using NLP algorithm

**6. IMPLEMENTATION**

**6.1Tools Introduction**

R studio

RStudio is a [free](https://en.wikipedia.org/wiki/Free_software) and [open source](https://en.wikipedia.org/wiki/Open-source_software) [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for [R](https://en.wikipedia.org/wiki/R_(programming_language)), a [programming language](https://en.wikipedia.org/wiki/Programming_language) for [statistical computing](https://en.wikipedia.org/wiki/Statistical_computing) and graphics. RStudio is available in two editions: RStudio Desktop, where the program is run locally as a regular [desktop application](https://en.wikipedia.org/wiki/Desktop_application); and RStudio Server, which allows accessing RStudio using a web browser while it is running on a remote [Linux](https://en.wikipedia.org/wiki/Linux) server. Prepackaged distributions of RStudio Desktop are available for [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [Mac OS X](https://en.wikipedia.org/wiki/Mac_OS_X), and Linux.

Twitter

Twitter is an online [social networking](https://en.wikipedia.org/wiki/Social_networking_service) service that enables users to send and read short 140-[character](https://en.wikipedia.org/wiki/Character_(computing)) messages called "tweets". Registered users can read and post tweets, but unregistered users can only read them. Users access Twitter through the website interface, [SMS](https://en.wikipedia.org/wiki/Short_Message_Service), or mobile device [app](https://en.wikipedia.org/wiki/Application_software). Twitter Inc. is based in [San Francisco](https://en.wikipedia.org/wiki/San_Francisco) and has more than 25 offices around the world.

**6.2 Technology Introduction**

**R**  [programming language](https://en.wikipedia.org/wiki/Programming_language)

**R** is a [programming language](https://en.wikipedia.org/wiki/Programming_language) and software environment for [statistical computing](https://en.wikipedia.org/wiki/Statistical_computing) and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among [statisticians](https://en.wikipedia.org/wiki/Statistician) and [data miners](https://en.wikipedia.org/wiki/Data_mining) for developing [statistical software](https://en.wikipedia.org/wiki/Statistical_software) and data analysis. Polls, [surveys of data miners](https://en.wikipedia.org/wiki/Rexer%27s_Annual_Data_Miner_Survey), and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.

R is an implementation of the [S programming language](https://en.wikipedia.org/wiki/S_(programming_language)) combined with [lexical scoping](https://en.wikipedia.org/wiki/Lexical_scoping) semantics inspired by [Scheme](https://en.wikipedia.org/wiki/Scheme_(programming_language)). [S](https://en.wikipedia.org/wiki/S_(programming_language)) was created by [John Chambers](https://en.wikipedia.org/wiki/John_Chambers_(programmer)) while at [Bell Labs](https://en.wikipedia.org/wiki/Bell_Laboratories). There are some important differences, but much of the code written for S runs unaltered.

R is [a GNU project](https://en.wikipedia.org/wiki/List_of_GNU_packages). The [source code](https://en.wikipedia.org/wiki/Source_code) for the R software environment is written primarily in [C](https://en.wikipedia.org/wiki/C_(programming_language)), [Fortran](https://en.wikipedia.org/wiki/Fortran), and R. R is freely available under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), and pre-compiled binary versions are provided for various [operating systems](https://en.wikipedia.org/wiki/Operating_system). While R has a[command line interface](https://en.wikipedia.org/wiki/Command_line_interface), there are several [graphical front-ends](https://en.wikipedia.org/wiki/Graphical_user_interface) available.

**6.3 Over all View of implementation**

We started off by installing R..It is a free “software environment for statistical computing and graphics” and is available for Unix platforms, Windows and MacOS.Next we installed twitteR package and CRAN mirrors.

A mirror is a distribution site for software. Generally the mirrors will all have copies of the software (libraries, source code, documentation) identical to those available from the 'main' download site for the software. This allows you to choose a download site close to you for better bandwidth/latency, balances load across many servers, and provides redundancy so you can always obtain the software even if one or more of the mirror sites is offline. Sometimes there is no primary site and the software is simply distributed via a network of mirror sites.

**The Twitter Authentication**

First we need to create an app at Twitter where we generate consumer key, consumer secret key, access token, access secret token in order to access the tweets application.

We used these commands in order to authenticate our twitter application

|  |
| --- |
| reqURL <- "https://api.twitter.com/oauth/request\_token"    accessURL <- "http://api.twitter.com/oauth/access\_token"    authURL <- "http://api.twitter.com/oauth/authorize"    consumerKey <- "yourconsumerkey"    consumerSecret <- "yourconsumersecret"    twitCred <- OAuthFactory$new(consumerKey=consumerKey,consumerSecret=consumerSecret,requestURL=reqURL,accessURL=accessURL,authURL=authURL)    download.file(url="http://curl.haxx.se/ca/cacert.pem", destfile="cacert.pem")    twitCred$handshake(cainfo="cacert.pem")    registerTwitterOAuth(twitCred) |

Sentiment Analysis on Twitter:

After passing the authentication we need to extract the tweets.

For example:

tweets = searchTwitter("#apple", n=200, cainfo="cacert.pem")

This makes twitteR get 200 Tweets with the keyword #apple in it

The Analysis:

To be able to analyze our tweets, we have to extract their text and save it into the variable

What we also need are our lists with the positive and the negative words.

We have to insert a small algorithm

|  |  |
| --- | --- |
|  | score.sentiment = function(sentences, pos.words, neg.words, .progress='none')    {    require(plyr)    require(stringr)    # we got a vector of sentences. plyr will handle a list    # or a vector as an "l" for us    # we want a simple array ("a") of scores back, so we use    # "l" + "a" + "ply" = "laply":    scores = laply(sentences, function(sentence, pos.words, neg.words) {    # clean up sentences with R's regex-driven global substitute, gsub():    sentence = gsub('[[:punct:]]', '', sentence)    sentence = gsub('[[:cntrl:]]', '', sentence)    sentence = gsub('\d+', '', sentence)    # and 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)    } |

## The final steps:

analysis = score.sentiment(Tweets.text, pos, neg)

sentiment Analysis was now saved.

get a histogram with:

hist(analysis$score)

**6.4Explanation of Algorithm and how it is been implemented**

Natural language processing (NLP) is a field of computer science, artificial intelligence, and [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) concerned with the interactions between [computers](https://en.wikipedia.org/wiki/Computer) and [human (natural) languages](https://en.wikipedia.org/wiki/Natural_language). As such, NLP is related to the area of [human–computer interaction](https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction). Many challenges in NLP involve [natural language understanding](https://en.wikipedia.org/wiki/Natural_language_understanding), that is, enabling computers to derive meaning from human or natural language input, and others involve [natural language generation](https://en.wikipedia.org/wiki/Natural_language_generation).

NLP algorithms are based on [machine learning](https://en.wikipedia.org/wiki/Machine_learning)

**Major tasks in NLP**

[**Automatic summarization**](https://en.wikipedia.org/wiki/Automatic_summarization)

Produce a readable summary of a chunk of text. Often used to provide summaries of text of a known type, such as articles in the financial section of a newspaper.

[**Coreference resolution**](https://en.wikipedia.org/wiki/Coreference)

Given a sentence or larger chunk of text, determine which words ("mentions") refer to the same objects ("entities"). [Anaphora resolution](https://en.wikipedia.org/wiki/Anaphora_resolution) is a specific example of this task, and is specifically concerned with matching up [pronouns](https://en.wikipedia.org/wiki/Pronoun) with the nouns or names that they refer to. The more general task of coreference resolution also includes identifying so-called "bridging relationships" involving [referring expressions](https://en.wikipedia.org/wiki/Referring_expression). For example, in a sentence such as "He entered John's house through the front door", "the front door" is a referring expression and the bridging relationship to be identified is the fact that the door being referred to is the front door of John's house (rather than of some other structure that might also be referred to).

[**Discourse analysis**](https://en.wikipedia.org/wiki/Discourse_analysis)

This rubric includes a number of related tasks. One task is identifying the [discourse](https://en.wikipedia.org/wiki/Discourse) structure of connected text, i.e. the nature of the discourse relationships between sentences (e.g. elaboration, explanation, contrast). Another possible task is recognizing and classifying the [speech acts](https://en.wikipedia.org/wiki/Speech_act) in a chunk of text (e.g. yes-no question, content question, statement, assertion, etc.).

[**Machine translation**](https://en.wikipedia.org/wiki/Machine_translation)

Automatically translate text from one human language to another. This is one of the most difficult problems, and is a member of a class of problems colloquially termed "[AI-complete](https://en.wikipedia.org/wiki/AI-complete)", i.e. requiring all of the different types of knowledge that humans possess (grammar, semantics, facts about the real world, etc.) in order to solve properly.

[**Morphological segmentation**](https://en.wikipedia.org/wiki/Morphology_(linguistics))

Separate words into individual [morphemes](https://en.wikipedia.org/wiki/Morpheme) and identify the class of the morphemes. The difficulty of this task depends greatly on the complexity of the [morphology](https://en.wikipedia.org/wiki/Morphology_(linguistics)) (i.e. the structure of words) of the language being considered. [English](https://en.wikipedia.org/wiki/English_language) has fairly simple morphology, especially [inflectional morphology](https://en.wikipedia.org/wiki/Inflectional_morphology), and thus it is often possible to ignore this task entirely and simply model all possible forms of a word (e.g. "open, opens, opened, opening") as separate words. In languages such as [Turkish](https://en.wikipedia.org/wiki/Turkish_language) or [Manipuri](https://en.wikipedia.org/wiki/Manipuri_language), a highly agglutinated Indian language, however, such an approach is not possible, as each dictionary entry has thousands of possible word forms.

[**Named entity recognition**](https://en.wikipedia.org/wiki/Named_entity_recognition)**(NER)**

Given a stream of text, determine which items in the text map to proper names, such as people or places, and what the type of each such name is (e.g. person, location, organization). Note that, although [capitalization](https://en.wikipedia.org/wiki/Capitalization) can aid in recognizing named entities in languages such as English, this information cannot aid in determining the type of named entity, and in any case is often inaccurate or insufficient. For example, the first word of a sentence is also capitalized, and named entities often span several words, only some of which are capitalized. Furthermore, many other languages in non-Western scripts (e.g. [Chinese](https://en.wikipedia.org/wiki/Chinese_language) or [Arabic](https://en.wikipedia.org/wiki/Arabic_language)) do not have any capitalization at all, and even languages with capitalization may not consistently use it to distinguish names. For example, [German](https://en.wikipedia.org/wiki/German_language) capitalizes all [nouns](https://en.wikipedia.org/wiki/Noun), regardless of whether they refer to names, and [French](https://en.wikipedia.org/wiki/French_language) and [Spanish](https://en.wikipedia.org/wiki/Spanish_language) do not capitalize names that serve as [adjectives](https://en.wikipedia.org/wiki/Adjective).

[**Natural language generation**](https://en.wikipedia.org/wiki/Natural_language_generation)

Convert information from computer databases into readable human language.

[**Natural language understanding**](https://en.wikipedia.org/wiki/Natural_language_understanding)

Convert chunks of text into more formal representations such as [first-order logic](https://en.wikipedia.org/wiki/First-order_logic) structures that are easier for [computer](https://en.wikipedia.org/wiki/Computer) programs to manipulate. Natural language understanding involves the identification of the intended semantic from the multiple possible semantics which can be derived from a natural language expression which usually takes the form of organized notations of natural languages concepts. Introduction and creation of language metamodel and ontology are efficient however empirical solutions. An explicit formalization of natural languages semantics without confusions with implicit assumptions such as [closed-world assumption](https://en.wikipedia.org/wiki/Closed-world_assumption) (CWA) vs. [open-world assumption](https://en.wikipedia.org/wiki/Open-world_assumption), or subjective Yes/No vs. objective True/False is expected for the construction of a basis of semantics formalization.

[**Optical character recognition**](https://en.wikipedia.org/wiki/Optical_character_recognition)**(OCR)**

Given an image representing printed text, determine the corresponding text.

[**Part-of-speech tagging**](https://en.wikipedia.org/wiki/Part-of-speech_tagging)

Given a sentence, determine the [part of speech](https://en.wikipedia.org/wiki/Part_of_speech) for each word. Many words, especially common ones, can serve as multiple [parts of speech](https://en.wikipedia.org/wiki/Parts_of_speech). For example, "book" can be a [noun](https://en.wikipedia.org/wiki/Noun) ("the book on the table") or [verb](https://en.wikipedia.org/wiki/Verb) ("to book a flight"); "set" can be a [noun](https://en.wikipedia.org/wiki/Noun), [verb](https://en.wikipedia.org/wiki/Verb) or [adjective](https://en.wikipedia.org/wiki/Adjective); and "out" can be any of at least five different parts of speech. Some languages have more such ambiguity than others. Languages with little [inflectional morphology](https://en.wikipedia.org/wiki/Inflectional_morphology), such as [English](https://en.wikipedia.org/wiki/English_language) are particularly prone to such ambiguity. [Chinese](https://en.wikipedia.org/wiki/Chinese_language) is prone to such ambiguity because it is a [tonal language](https://en.wikipedia.org/wiki/Tonal_language) during verbalization. Such inflection is not readily conveyed via the entities employed within the orthography to convey intended meaning.

[**Parsing**](https://en.wikipedia.org/wiki/Parsing)

Determine the [parse tree](https://en.wikipedia.org/wiki/Parse_tree) (grammatical analysis) of a given sentence. The [grammar](https://en.wikipedia.org/wiki/Grammar) for [natural languages](https://en.wikipedia.org/wiki/Natural_language) is [ambiguous](https://en.wikipedia.org/wiki/Ambiguous) and typical sentences have multiple possible analyses. In fact, perhaps surprisingly, for a typical sentence there may be thousands of potential parses (most of which will seem completely nonsensical to a human).

[**Question answering**](https://en.wikipedia.org/wiki/Question_answering)

Given a human-language question, determine its answer. Typical questions have a specific right answer (such as "What is the capital of Canada?"), but sometimes open-ended questions are also considered (such as "What is the meaning of life?"). Recent works have looked at even more complex questions.

[**Relationship extraction**](https://en.wikipedia.org/wiki/Relationship_extraction)

Given a chunk of text, identify the relationships among named entities (e.g. who is married to whom).

[**Sentence breaking**](https://en.wikipedia.org/wiki/Sentence_breaking)**(also known as**[**sentence boundary disambiguation**](https://en.wikipedia.org/wiki/Sentence_boundary_disambiguation)**)**

Given a chunk of text, find the sentence boundaries. Sentence boundaries are often marked by [periods](https://en.wikipedia.org/wiki/Full_stop) or other [punctuation marks](https://en.wikipedia.org/wiki/Punctuation_mark), but these same characters can serve other purposes (e.g. marking [abbreviations](https://en.wikipedia.org/wiki/Abbreviation)).

[**Sentiment analysis**](https://en.wikipedia.org/wiki/Sentiment_analysis)

Extract subjective information usually from a set of documents, often using online reviews to determine "polarity" about specific objects. It is especially useful for identifying trends of public opinion in the social media, for the purpose of marketing.

[**Speech recognition**](https://en.wikipedia.org/wiki/Speech_recognition)

Given a sound clip of a person or people speaking, determine the textual representation of the speech. This is the opposite of [text to speech](https://en.wikipedia.org/wiki/Text_to_speech) and is one of the extremely difficult problems colloquially termed "[AI-complete](https://en.wikipedia.org/wiki/AI-complete)" (see above). In [natural speech](https://en.wikipedia.org/wiki/Natural_speech) there are hardly any pauses between successive words, and thus [speech segmentation](https://en.wikipedia.org/wiki/Speech_segmentation) is a necessary subtask of speech recognition (see below). Note also that in most spoken languages, the sounds representing successive letters blend into each other in a process termed [coarticulation](https://en.wikipedia.org/wiki/Coarticulation), so the conversion of the analog signal to discrete characters can be a very difficult process.

[**Speech segmentation**](https://en.wikipedia.org/wiki/Speech_segmentation)

Given a sound clip of a person or people speaking, separate it into words. A subtask of [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition) and typically grouped with it.

[**Topic segmentation**](https://en.wikipedia.org/wiki/Topic_segmentation)**and recognition**

Given a chunk of text, separate it into segments each of which is devoted to a topic, and identify the topic of the segment.

[**Word segmentation**](https://en.wikipedia.org/wiki/Word_segmentation)

Separate a chunk of continuous text into separate words. For a language like [English](https://en.wikipedia.org/wiki/English_language), this is fairly trivial, since words are usually separated by spaces. However, some written languages like [Chinese](https://en.wikipedia.org/wiki/Chinese_language), [Japanese](https://en.wikipedia.org/wiki/Japanese_language) and [Thai](https://en.wikipedia.org/wiki/Thai_language) do not mark word boundaries in such a fashion, and in those languages text segmentation is a significant task requiring knowledge of the [vocabulary](https://en.wikipedia.org/wiki/Vocabulary) and [morphology](https://en.wikipedia.org/wiki/Morphology_(linguistics)) of words in the language.

[**Word sense disambiguation**](https://en.wikipedia.org/wiki/Word_sense_disambiguation)

Many words have more than one [meaning](https://en.wikipedia.org/wiki/Meaning_(linguistics)); we have to select the meaning which makes the most sense in context. For this problem, we are typically given a list of words and associated word senses, e.g. from a dictionary or from an online resource such as [WordNet](https://en.wikipedia.org/wiki/WordNet).

In some cases, sets of related tasks are grouped into subfields of NLP that are often considered separately from NLP as a whole. Examples include:

[**Information retrieval**](https://en.wikipedia.org/wiki/Information_retrieval)**(IR)**

This is concerned with storing, searching and retrieving information. It is a separate field within computer science (closer to databases), but IR relies on some NLP methods (for example, stemming). Some current research and applications seek to bridge the gap between IR and NLP.

[**Information extraction**](https://en.wikipedia.org/wiki/Information_extraction)**(IE)**

This is concerned in general with the extraction of semantic information from text. This covers tasks such as [named entity recognition](https://en.wikipedia.org/wiki/Named_entity_recognition), [Coreference resolution](https://en.wikipedia.org/wiki/Coreference), [relationship extraction](https://en.wikipedia.org/wiki/Relationship_extraction), etc.

**6.5 Information about the implementation of Modules**

Module is used for preprocessing and classification of tweets.

score.sentiment = function(sentences, pos.words, neg.words, .progress='none')

{

require(plyr)

require(stringr)

# we got a vector of sentences. plyr will handle a list

# or a vector as an "l" for us

# we want a simple array ("a") of scores back, so we use

# "l" + "a" + "ply" = "laply":

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

# clean up sentences with R's regex-driven global substitute, gsub():

sentence = gsub('[[:punct:]]', '', sentence)

sentence = gsub('[[:cntrl:]]', '', sentence)

sentence = gsub('\d+', '', sentence)

# and 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)

}

This module is used to remove all unnecessary fields from tweets like numbers,whitespaces etc.After that we are classifying tweets as neutral ,positive and negative.

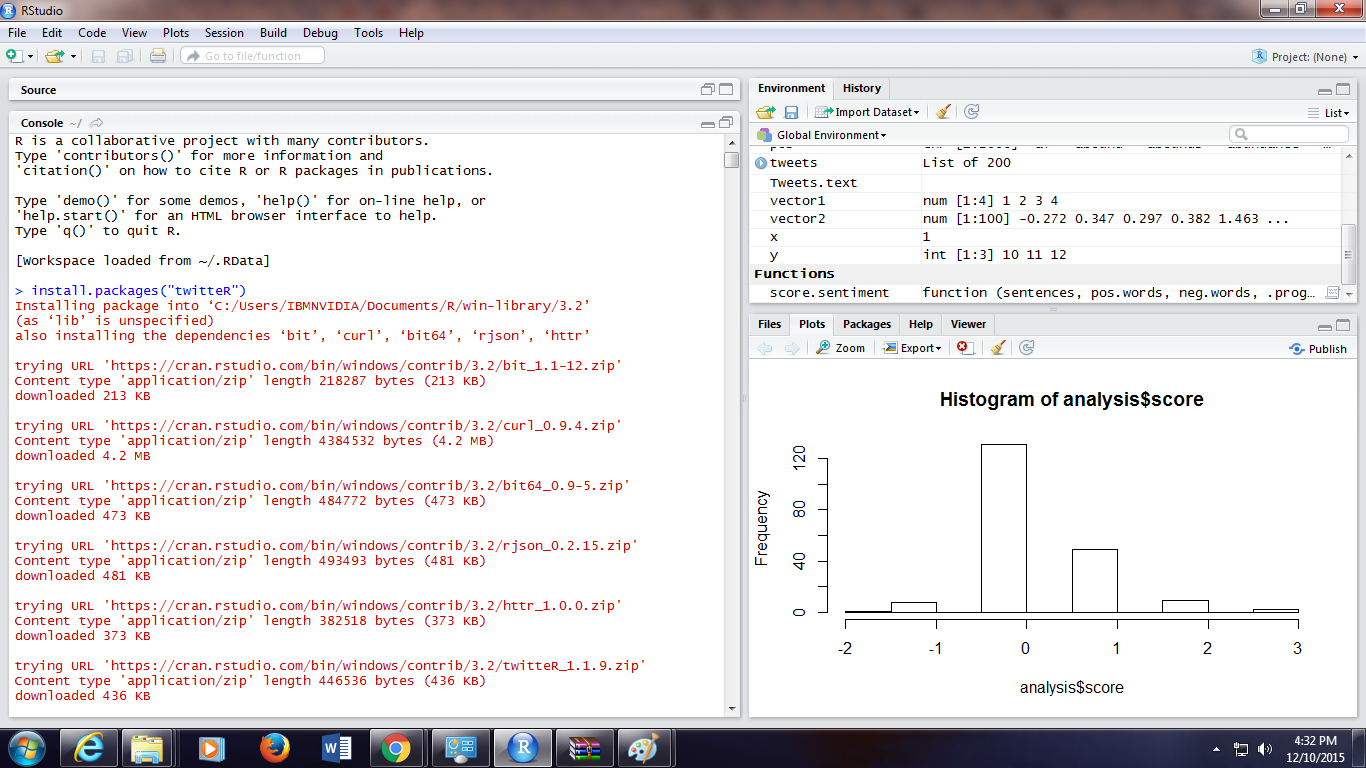
**FEATURE EXTRACTION :**

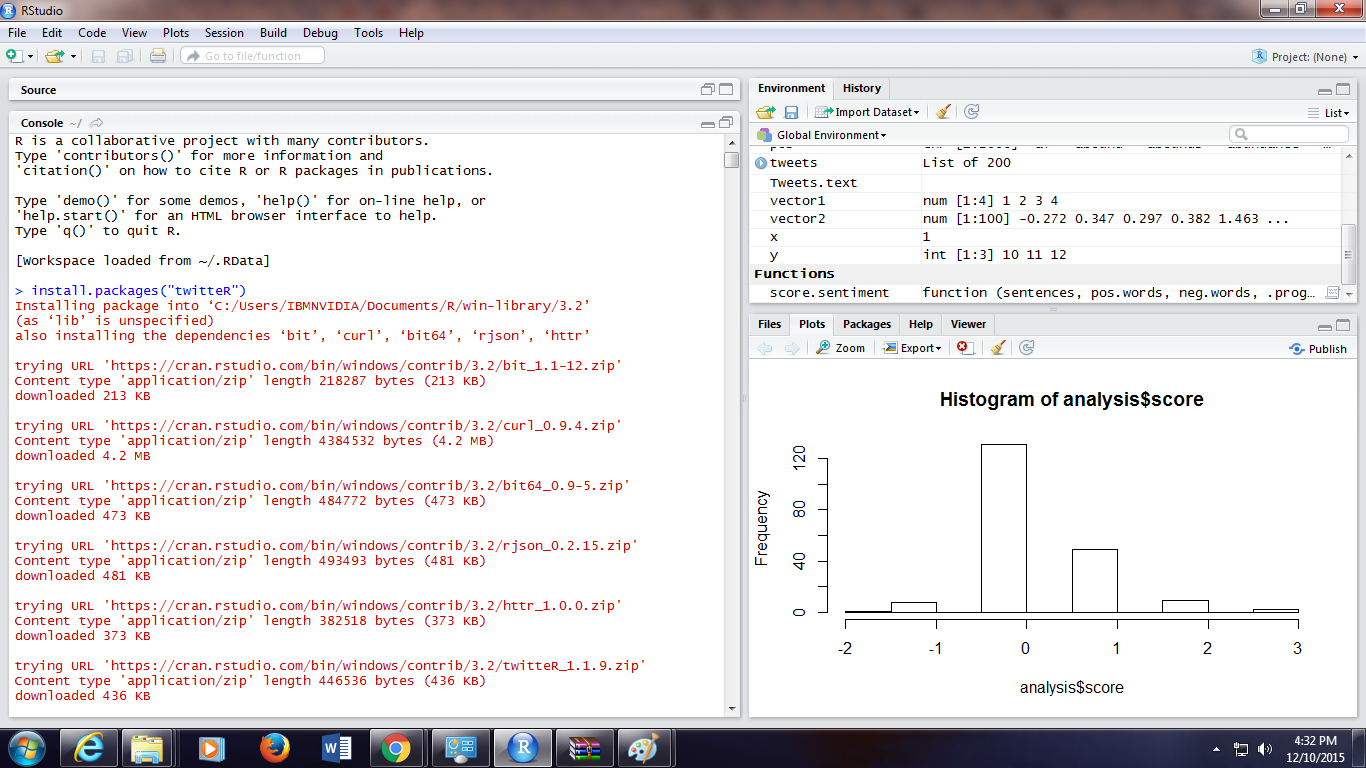
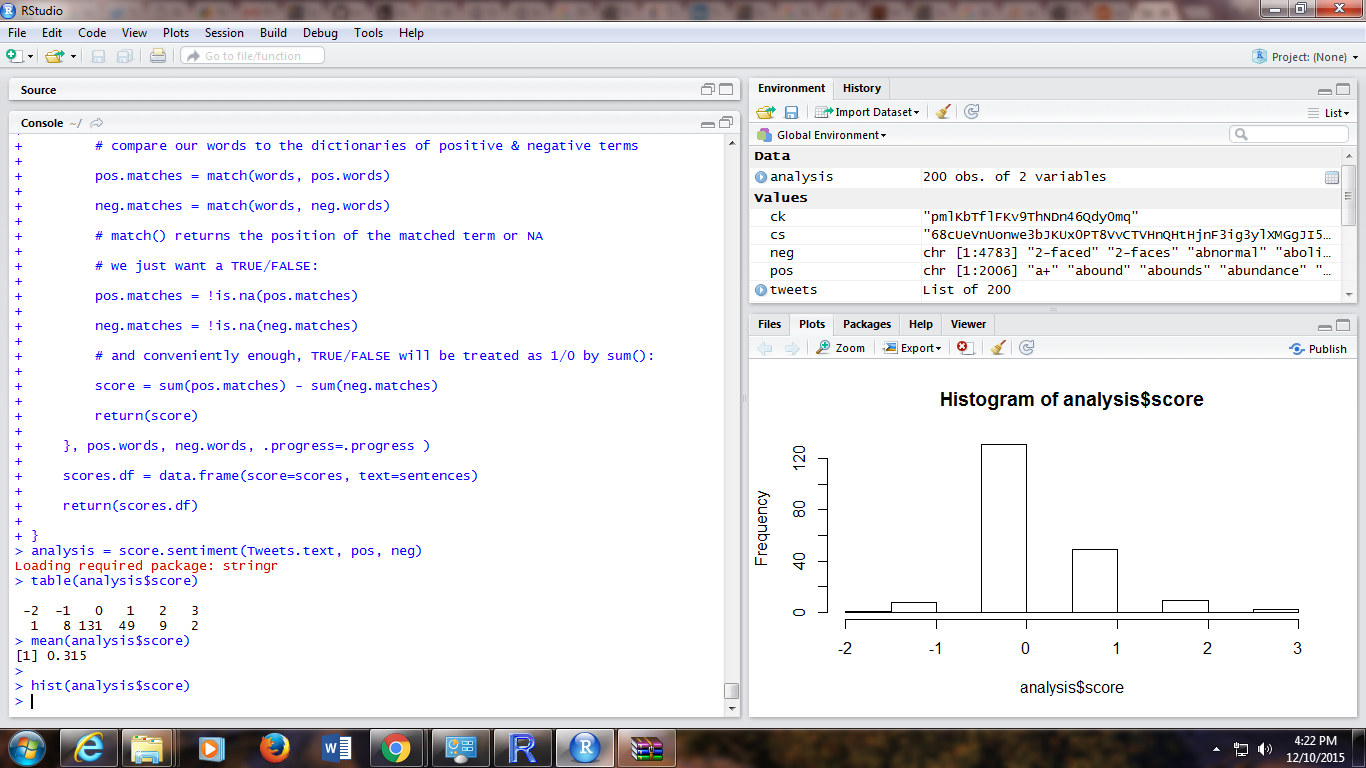
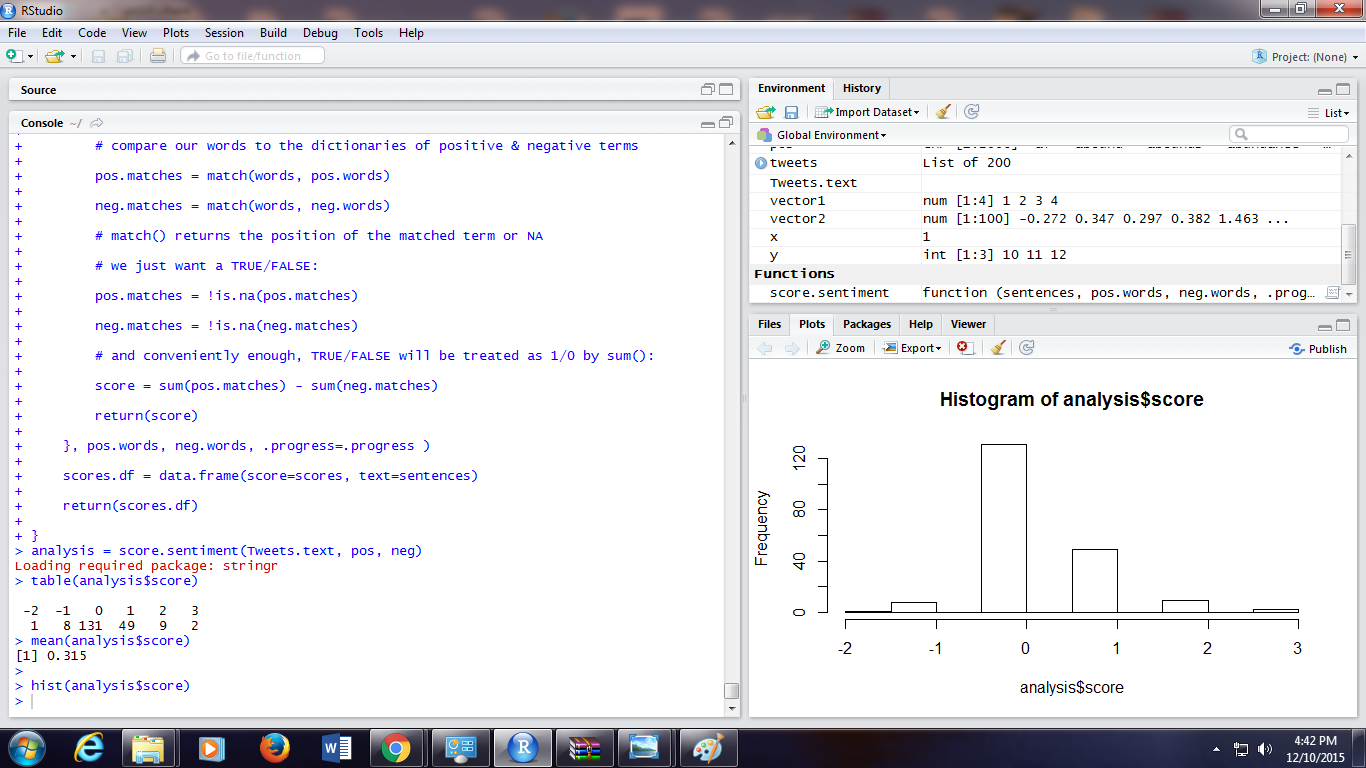
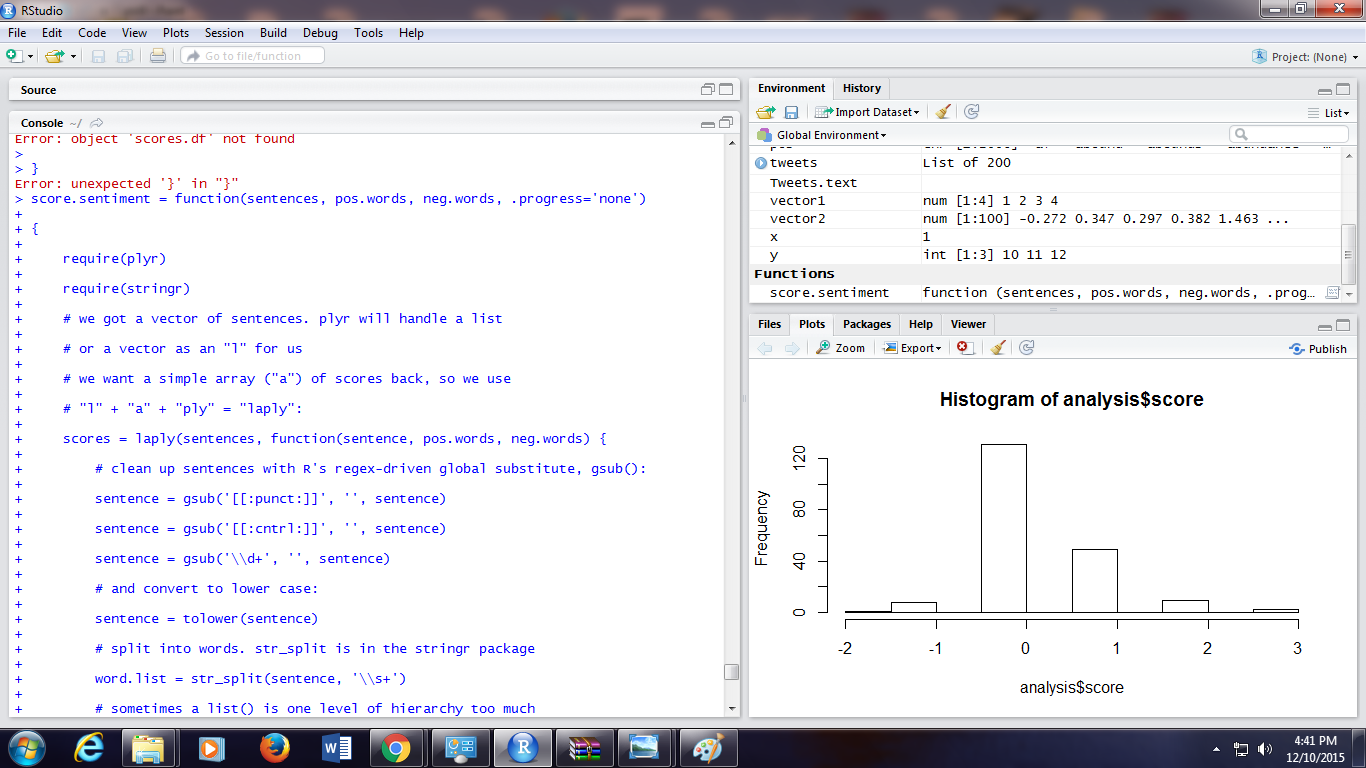
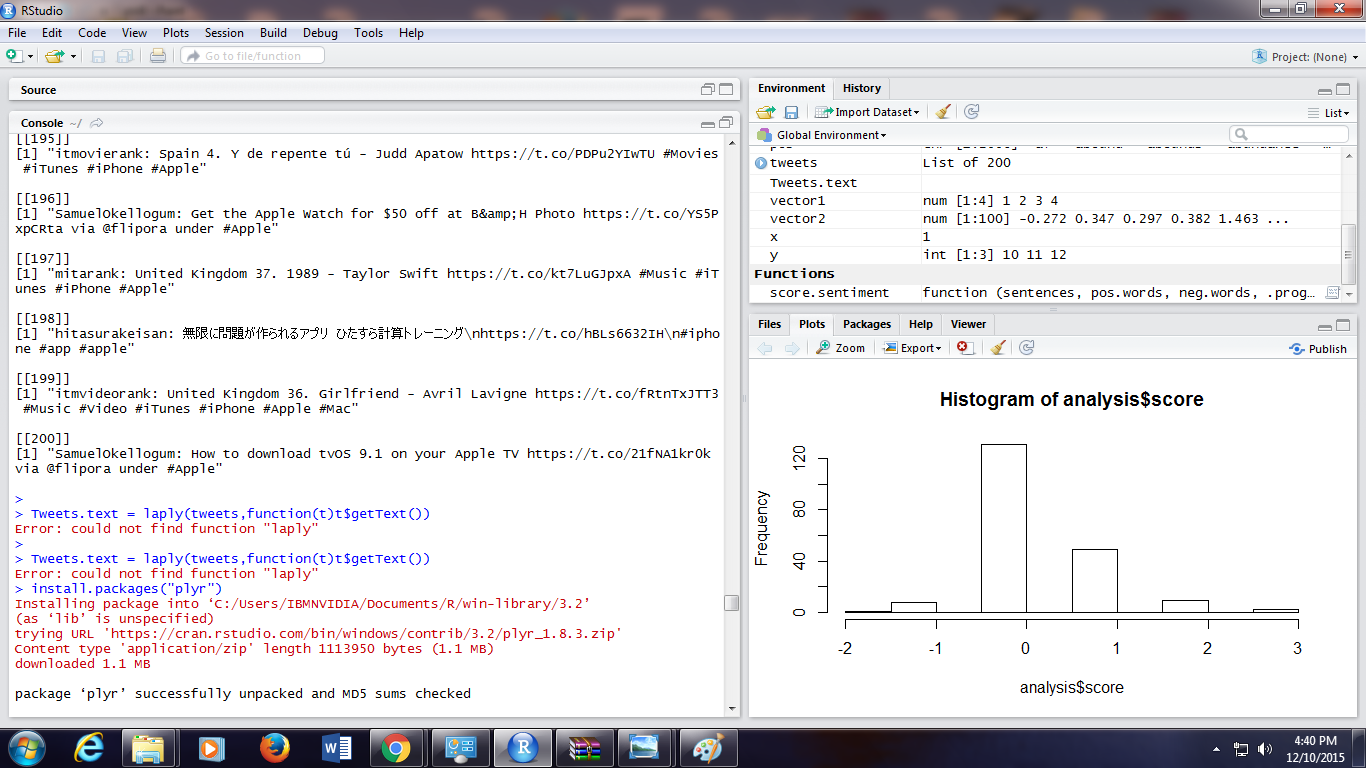
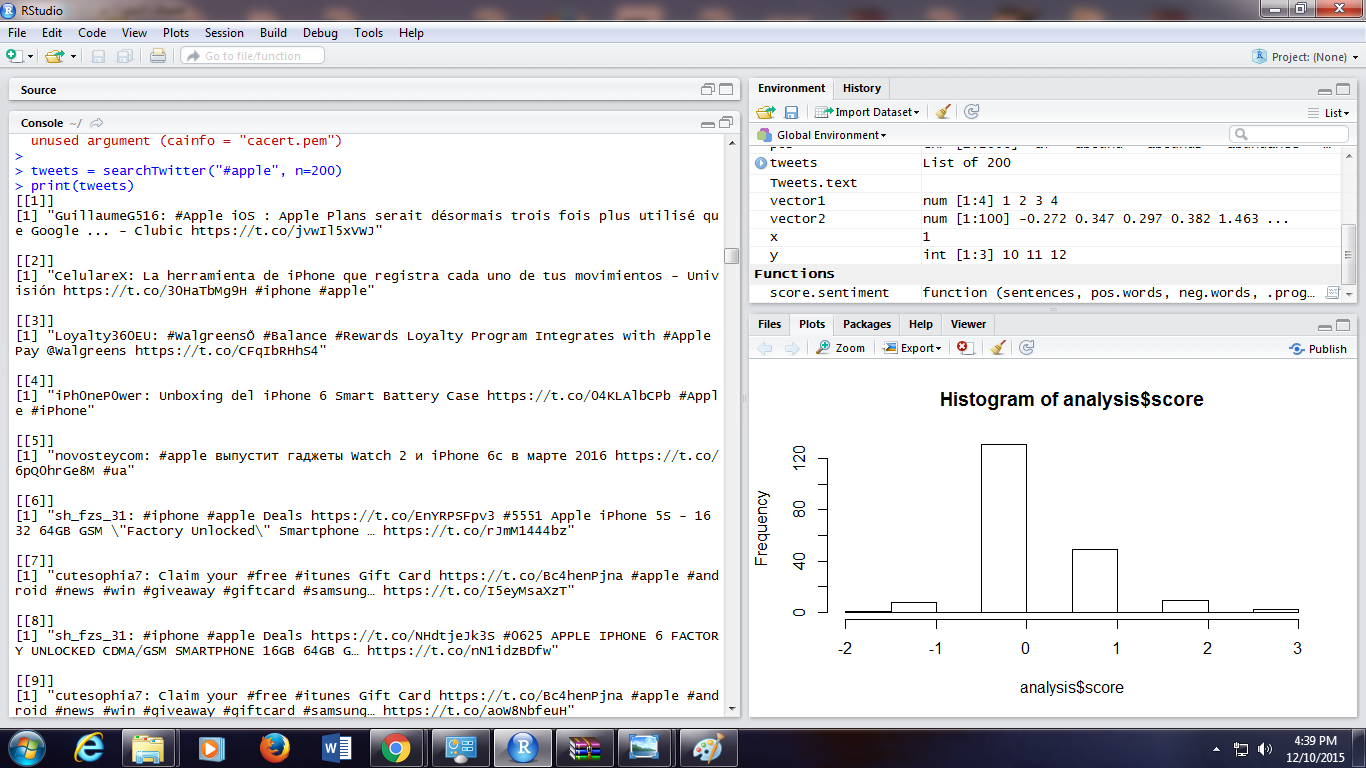
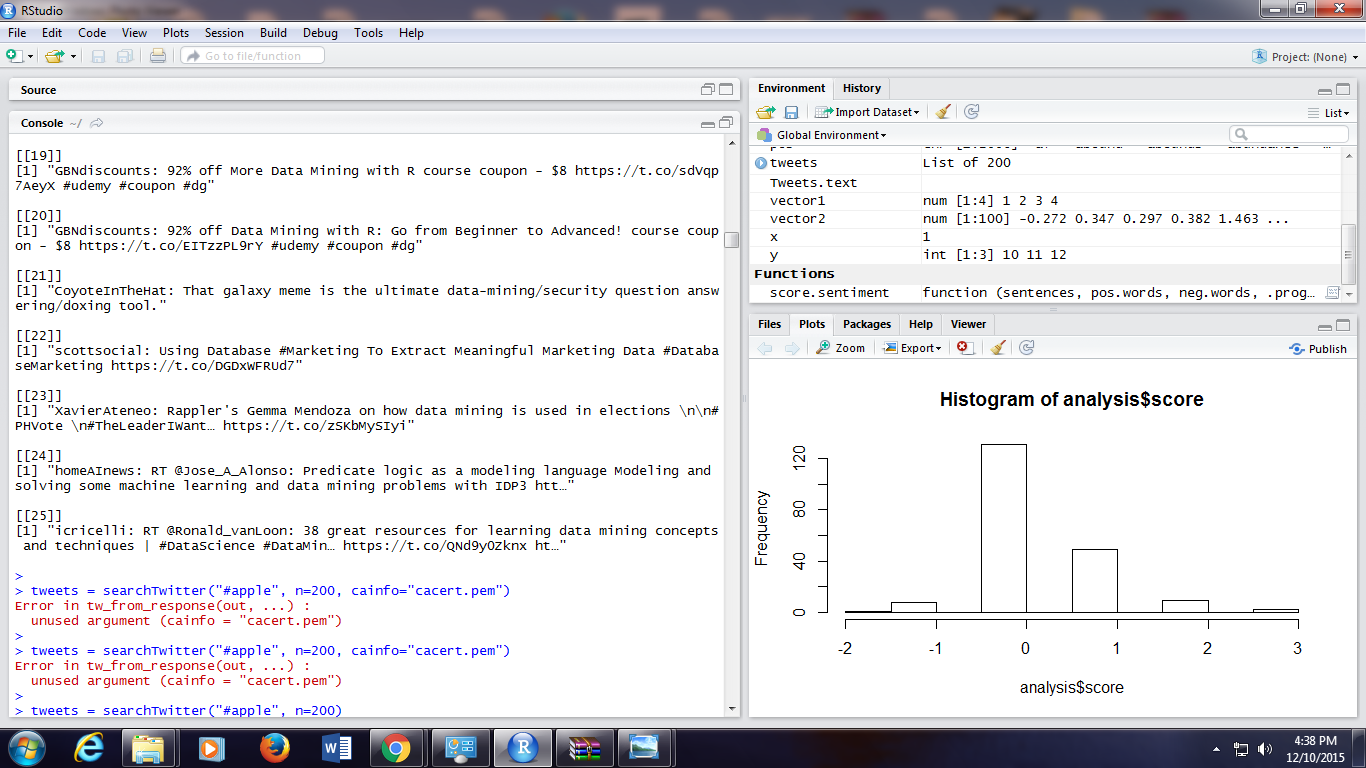
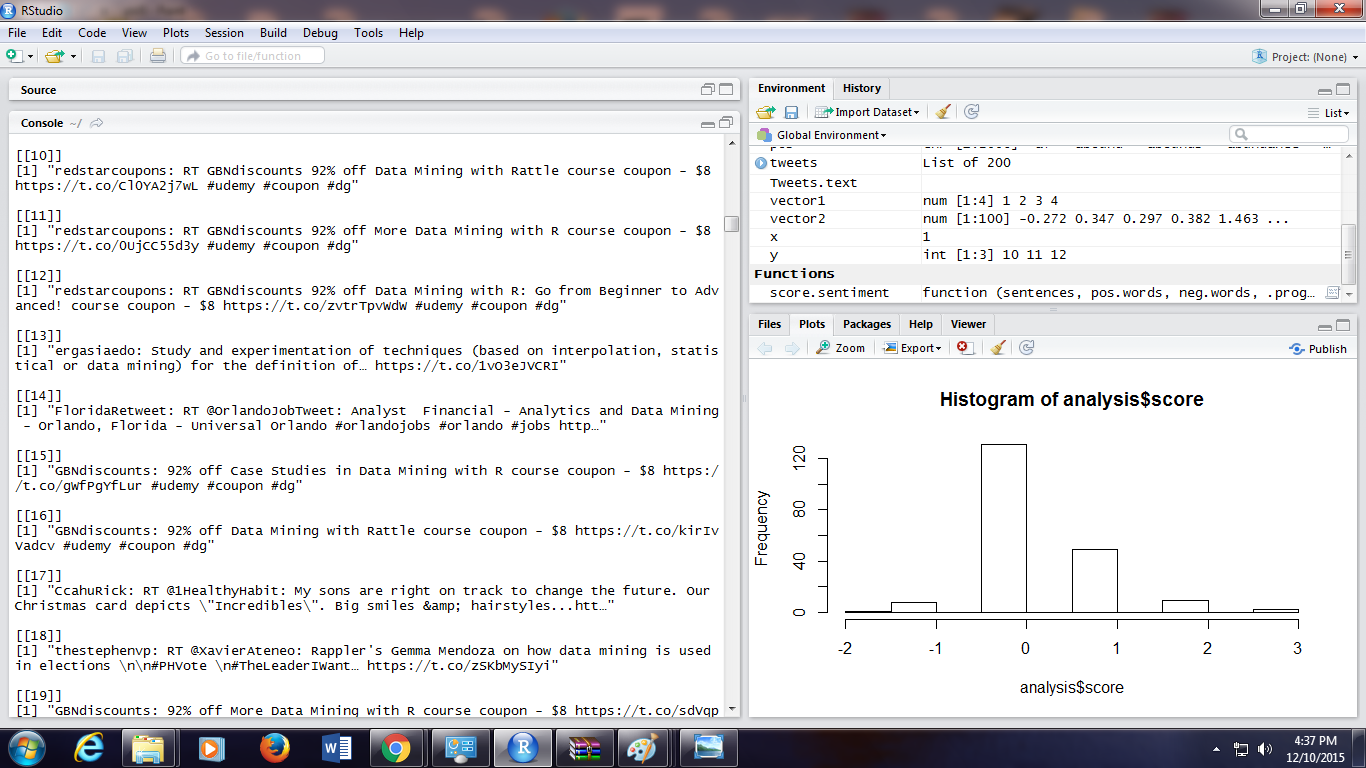
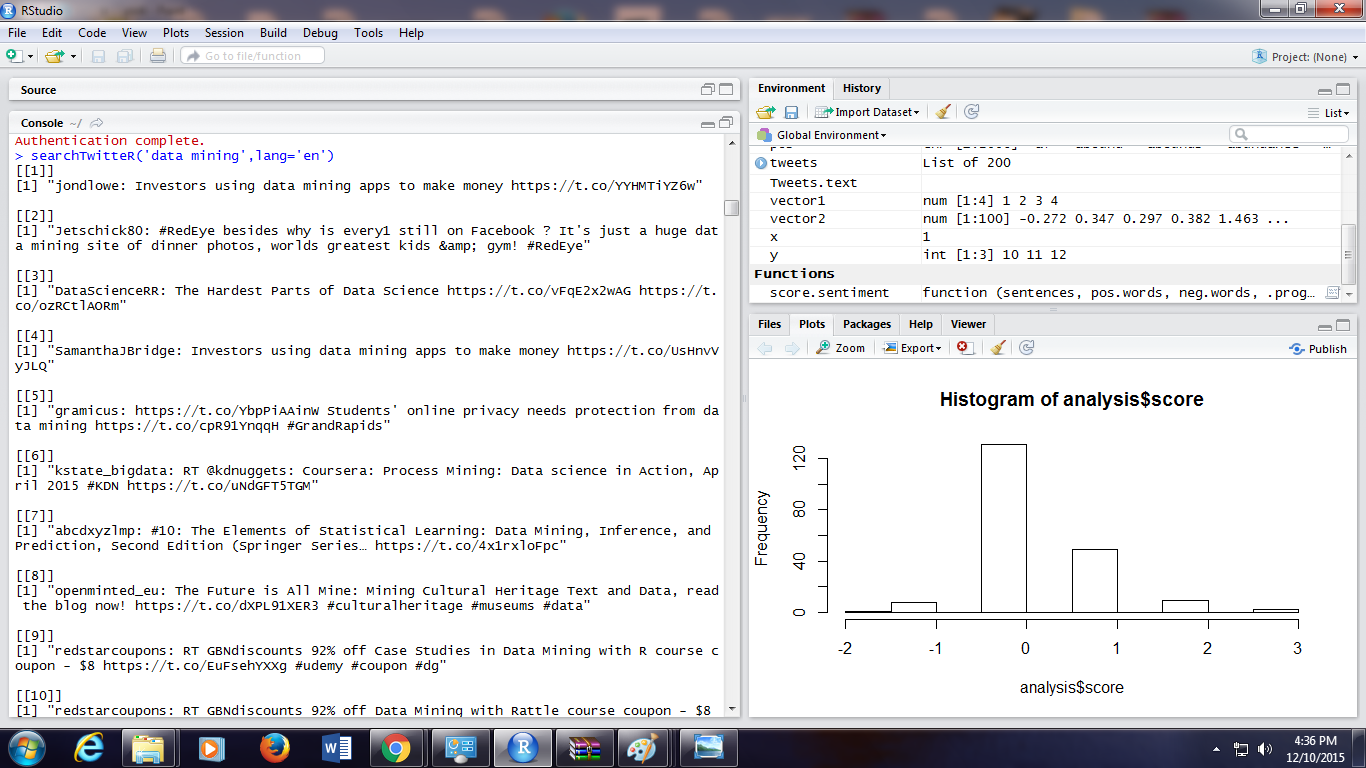
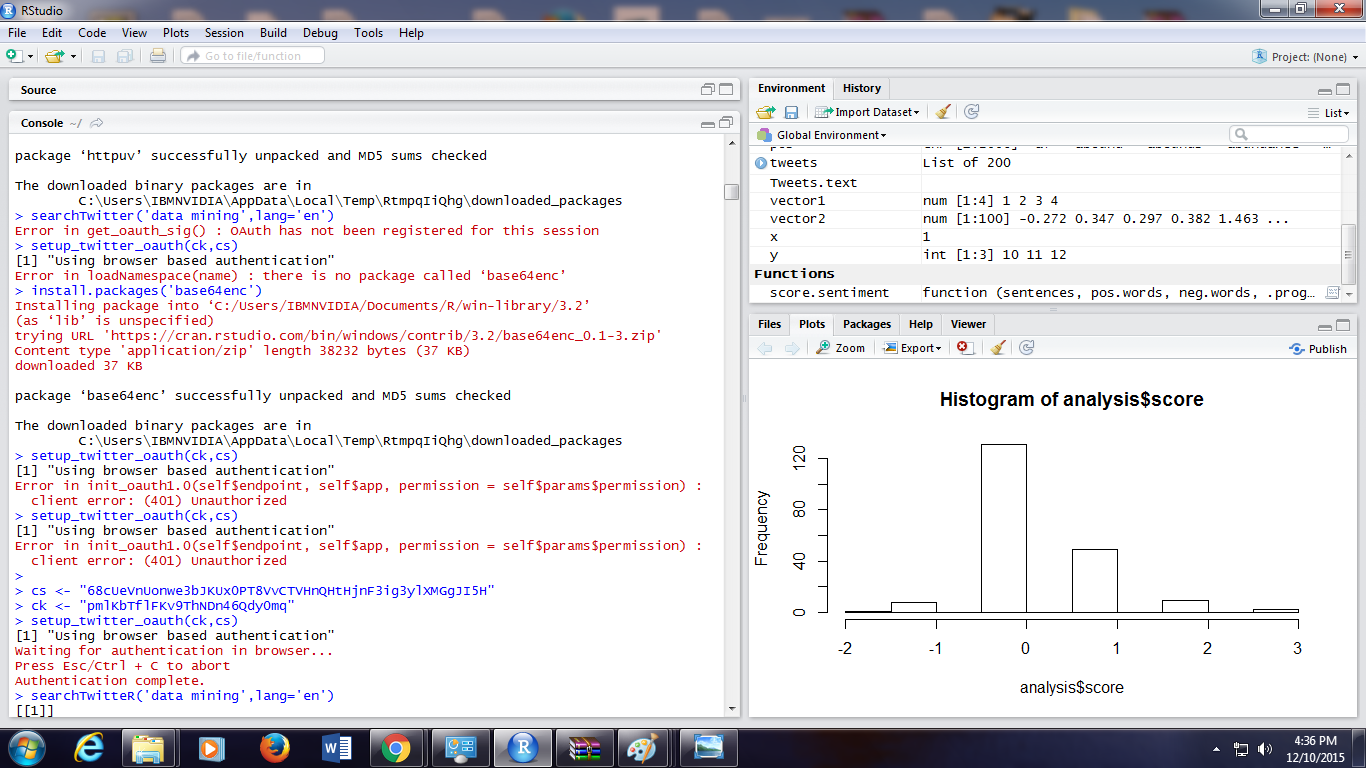
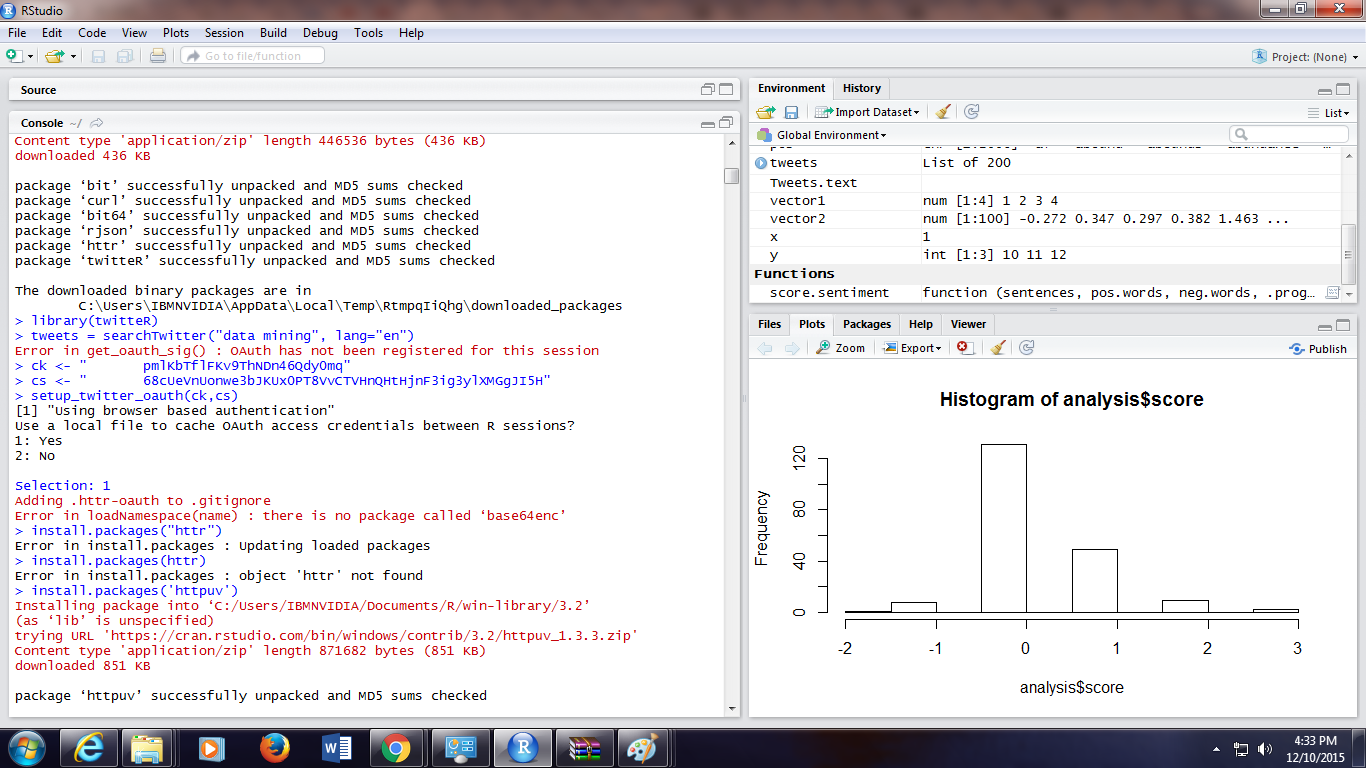
The improved dataset after pre- processing has a lot of distinctive properties. The feature extraction method, extracts the aspect (adjective) from the dataset. Later this adjective is used to show the positive and negative polarity in a sentence. Unigram model extracts the adjective and segregates it. It discards the preceding and successive word occurring with the adjective in the sentences. Example “painting Beautiful” through unigram model, only Beautiful is extracted from the sentence.

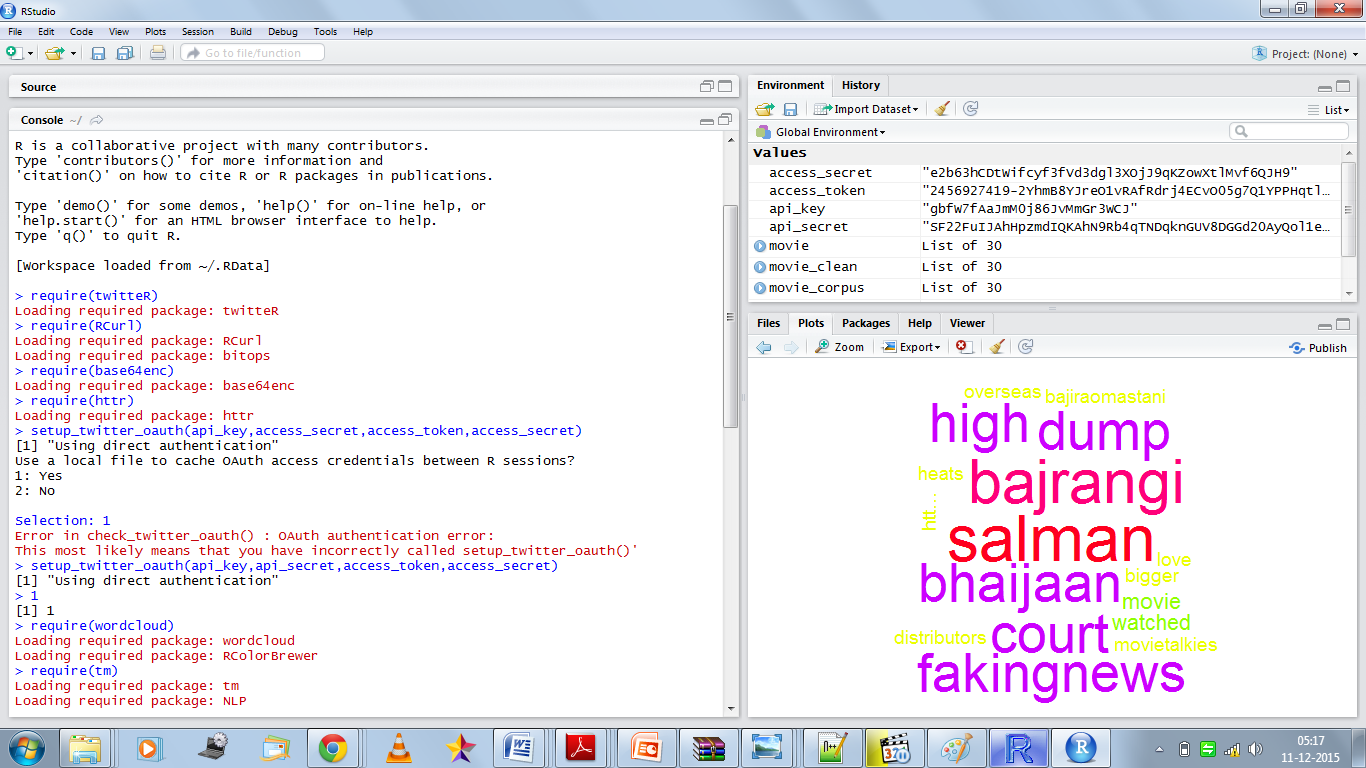
**SEMANTIC ANALYSIS :**

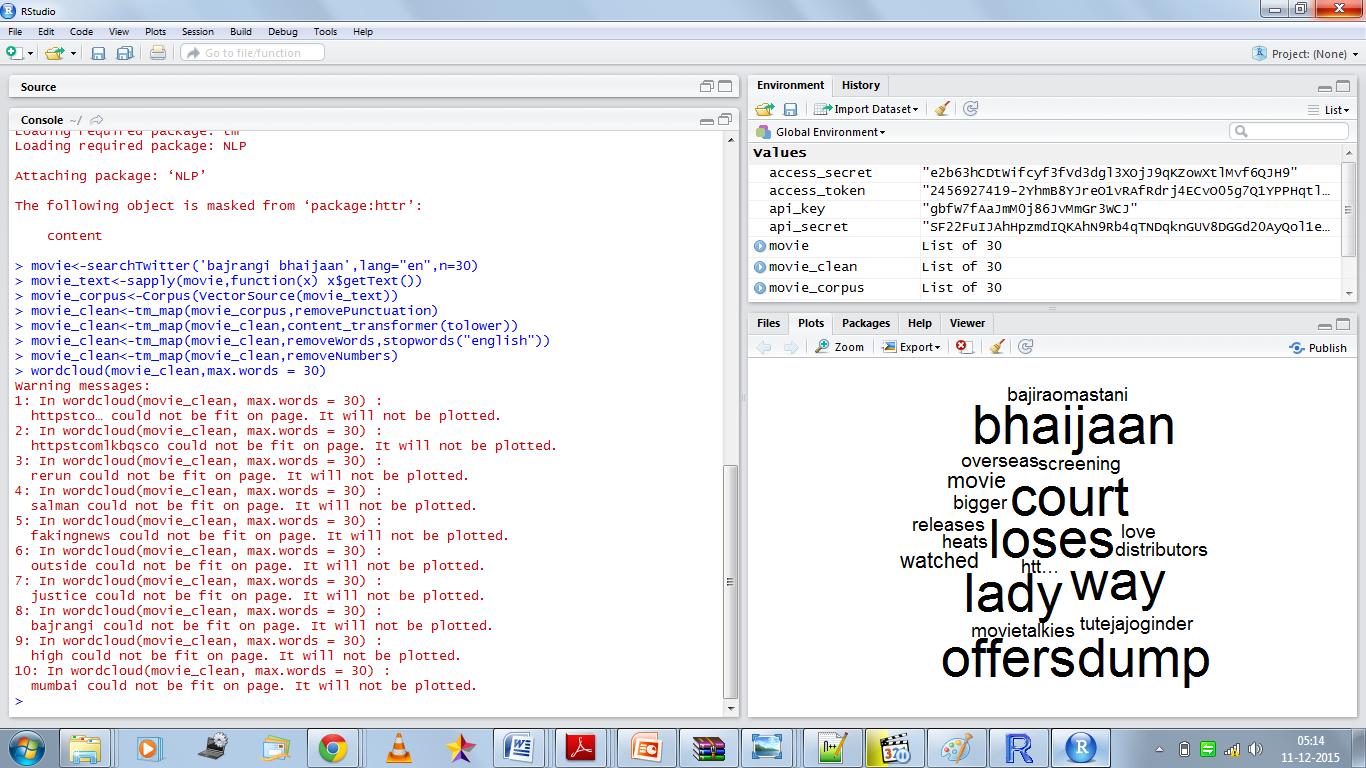
Semantic analysis is derived from the WordNet database where each term is associated with each other. This database is of English words which are linked together. If two words are close to each other, they are semantically similar. More specifically, we are able to determine synonym like similarity. The key task is to use the stored documents that contain terms and then check the similarity with the words that the user uses in their sentences. For example in the sentence ”I am happy” the word ‘’happy’’ being an adjective gets selected and is compared with the stored feature vector for synonyms. Let us assume 2 words; ‘glad’ and ‘satisfied’ tend to be very similar to the word ‘happy’. Now after the semantic analysis, ‘glad’ replaces ‘happy’ which gives a positive polarity.

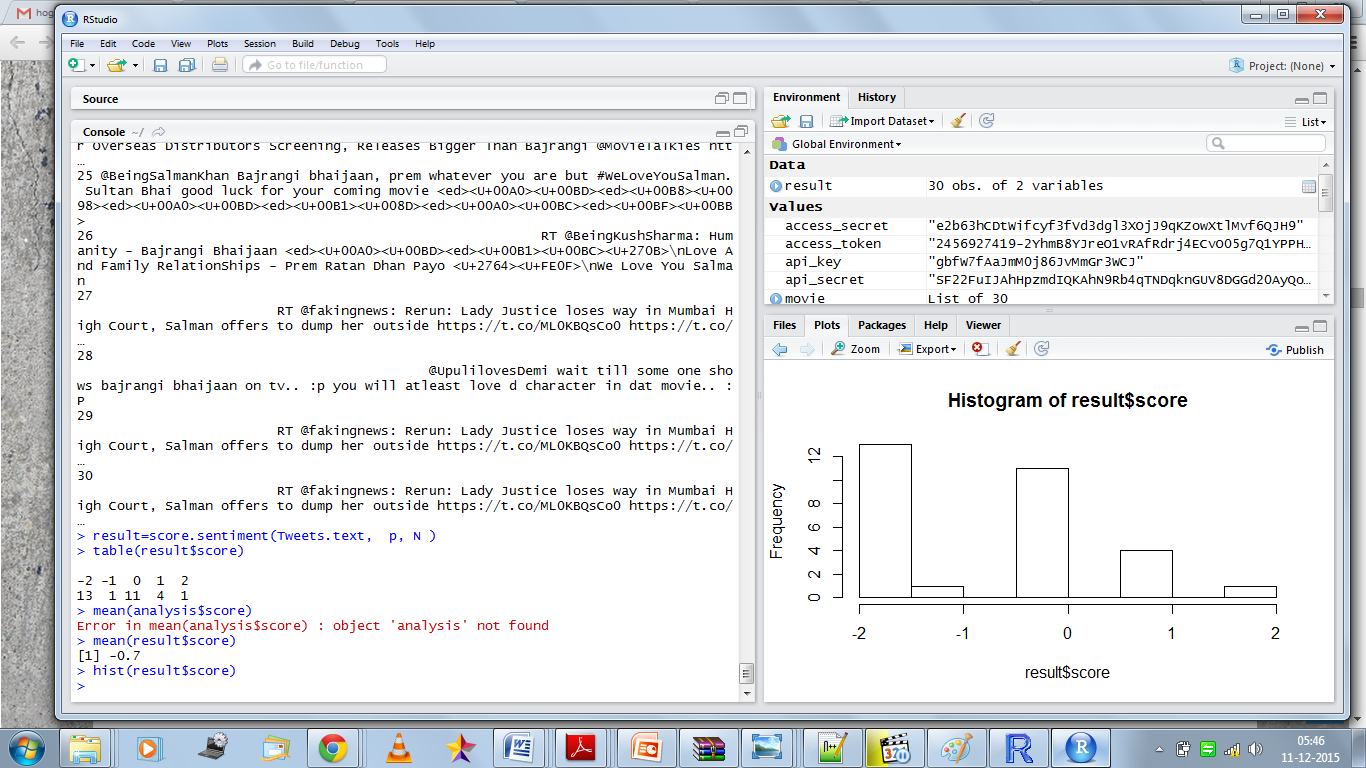
**Testing and results.**

****

****







**CONCLUSION**

In this project knowledge based methodologies and machine learning methodologies were used in order to give a thorough examination of the tweets of movies were extracted from twitter.Tweets were then classified into positive and negative classes using the machine learning classifier Natural Language Processor(NLP).The extraction of tweets from twitter proved to be more difficult than expected and several attempts were made to produce a dataset.The presence of white spaces, punctuations and numbers had to be confronted in the preprocessing stage.

It was found that Machine learning algorithms were simpler to implement and more efficient than other aspects of the paper as they produced a table which allowed for transparency in the accuracy of the NLP classification. However, the accuracy of the NLP still leaves room for improvement this may be achieved by better preprocessing.

**FUTURE WORK**

The applicability of sentiment analysis for future businesses and marketing in using a keywords and analysis of the sentiments around that keyword by the public is only going to increase as the popularity of Twitter grows over the next few years. However, in terms of long-term development or research, the ability of the twitter API to pull data that is older, should be developed as well as other social media API’s so that sentiment analysis could be performed over a period of time, especially in the realm of social sciences where researchers could enquire into social and political shifts of opinion on the social media sites. Equally the lack of change in opinion over time on some issues might be worth pursuing as a topic of research for twitter sentiment analysis. The usefulness of such a

sentiment analyzer would allow for an interesting analysis of social and

political issues.

**9. REFERENCES**

1. Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau, *“Sentiment Analysis of Twitter Data”* Department of Computer Science, Columbia University, New York, 2009.

2. Akshi Kumar and Teeja Mary Sebastian, *“Sentiment Analysis on Twitter”* department of Computer Engineering,Delhi Technological University, Delhi, India, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4,No 3, July 2012

3. G. Vinodhini, R. M. Chandrasekaran *“Sentiment Analysis and Opinion Mining: A Survey”* Assistant Professor, Department of Computer Science and Engineering, Annamalai University, Annamalai Nagar-608002, Volume 2, Issue 6, June 2012, IEEE paper.

4.Luciano Barbosa and Junlan Feng, “*Robust sentiment detection on twitter from biased and noisy data.*” Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pages 36–44, 2010.

5. Adam Bermingham and Alan Smeaton, “*Classifying sentiment in microblogs: is brevity an advantage?” A*CM, pages 1833–1836, 2010.

6. Pak and P. Paroubek. “*Twitter as a Corpus for Sentiment Analysis and Opinion Mining”*, In Proceedings of the Seventh Conference on International Language Resources and Evaluation, 2010.