Abstract

As e-commerce is becoming more and more popular, the number of customer reviews that a product receives grows rapidly. For a popular product, the number of reviews can be in hundreds or even thousands. This makes it difficult for a potential customer to read them to make an informed decision. This project is helps in getting the reviews about a particular project using data from Twitter. For this the tweets about a particular product are collected and processed in order to determine whether the tweet is in favour of the product or in oppose of the product by using language processing.

The aim of this project is to make an application in the field of natural language processing in order to find and implement an algorithm to solve the problem of measuring real-time comments made by user on products. For this we have collected the reviews form twitter that is tweets by different users about a particular product and then the polarity of the tweet is determined that either it is negative, positive and neutral. For determining the polarity of the tweet the sentiment of each word of the tweet is calculated and scores are given to public opinions of the products which could being used to compare between the products where a statistical output was produced to show the results. The methods used in this project can be used for any specific product with public opinions.

CONTENTS

INTRODUCTION

OBJECTIVE OF THE PROJECT

Web data mining generally refers to crawling through the web locating and fetching from pages containing desired valuable information mostly with the use of web crawlers. Web crawlers can be built to fetch information of desired target or in other words they can be made application specific. They find high applications in search engines to give up-to-date information.

Nowadays social networks have covered hundreds of millions active and passive web users around the planet. The fast and exponential growth of social networking sites has proven undeniable, facilitating interconnection between users and high rate of information exchange. According to the Nielsen report in March 2009, Social Networking has been the global consumer phenomenon of 2008. Two-thirds of the world's internet population visits a social network or blogging site and the sector now accounts for almost 10% of all internet time. With this amount of large user information exchange, social media have become a good platform for research and data mining.

The valuable information retrieved from social networking sites can be utilized in many ways one of which can be to study, understand and predict the market for specific products which is very essential to improve qualities of the respective product. Due to scrutiny certain social networking sites are continuously updating their user-dependent privacy policies for their users, which in turn are becoming a bit of a challenge for mining them. After collecting the desired information the most important part would be to understand the contents of this information. NLP is a field of computer science and linguistics concerned with the interactions between computers and human (natural) languages. One specific application in NLP that can be used for this purpose is sentiment analysis. It can be used to identify and extract subjective information from the information source collected. With all these processes and methods, it is possible to build a system which can extract application dependent information, process it and produce data which can be used for studying and deductions based on the information retrieved.

The explosion of Web opinion data has made essential the need for automatic tools to analyze and understand people's sentiments toward different topics. Important application area of sentiment/opinion identification is business intelligence as a product manufacturer always wants to know consumers' opinions on its products. So our project is to make an application in the field of natural language processing in order to find and implement an algorithm to solve the problem of measuring real-time comments made by user on products.

DESCRIPTION OF THE PROJECT

We have collected the reviews form twitter that is tweets by different users about a particular product and then the polarity of the tweet is determined that either it is negative, positive and neutral and that determine the total review of the people about a particular product this is done in following steps:

- Collecting the relevant data from twitter i.e. tweets related to the product.
- Identifying opinion tweets and fact tweets and deciding whether each opinion tweet is positive or negative, i.e., its polarity.
- After getting the polarity of the tweets we get a total score of the product positive and negative reviews by the people.

For getting the sentiment of the tweet first we pos tagged the tweet and then for each word we determined the sentiment of the word using Sentiword-Net. Generally the sentiment of the sentence is reflected by the adjective of the sentence. Negation is handled by inversing the sentiment of the word following not.

So a final score of the tweet is obtained so that the polarity of the tweet can be determined using the score if the score is positive than the tweet is favour of the product else it is in oppose of the product.

By this we can also analyse views of people on current affairs or some event or on a political issue. Since the task of manually scanning through large amounts of tweets about any keyword one by one is computational burden and is not practically implemented with respect to businesses and customer perspectives so we have automated this task.

Problem Definition

In our project, the aim is to check the polarity of a tweet about a product (**keyword**), as an average twitter user publishes about their reviews of a product in there tweets and so this helps ion getting the user opinion about a product and also know the problem faced by the users related to the product.

Online customer reviews posted on twitter is considered as a significant informative resource which is useful for both potential customers and product manufacturers. They supplement information provided by electronic storefronts such as product descriptions, reviews from experts, and personalized advices as they are the end users to use it. In Twitter, due to legal constraints, the text of every tweet was not in the original corpus, but only the tweet ID. In order to get the text of tweet, one should use the Twitter API accordingly. This API returns limited tweets. So we have a limited amount of data available for our algorithm. The main challenges of processing tweets emerge because of the size restriction of 140 characters. This restriction makes people use shortcuts, omit prepositions and use emoticons. Moreover, In Twitter, people are often use hash-tags attached to a single word expression, for referring to a trending topic (e.g. #FavouritePlayerFromEachPremierLeague) and the @ character when referring to another Twitter account (e.g. @nokia). Another problem is the language the tweets are not language specific that is it contains different languages so it is very difficult to analyse the opinion.

The task of manually scanning through large amounts of tweets about any keyword one by one is computational burden and is not practically implemented with respect to businesses and customer perspectives. Therefore it is more efficient to automatically process the various tweets and provide the necessary information in a suitable form as “positive” or “negative”. But they can also be any other elements such as 'relevant' and 'irrelevant', 'in favour of' or 'against'.

In this report, we dedicate our work to the main subtask of opinion summarization by the polarity analysis of tweets. We proposed an approach for mining the tweet about any identifying product feature or any relevant data and opinion based on the consideration of syntactic information and semantic information. For this we are extracting the tweets from the Twitter and the tagging them to extract the opinion and to do sentiment analysis on the given reviews using SentiWordNet & Stanford POS Tagger. Hence, we are calculating the polarity values for the respective tweets.

PROPOSED WORK

Sentiment analysis of Twitter Data performs its operation in **three** major steps: **Tweeter Object creation**, **determining Fact tweets** & **Opinion tweets** and **Polarity Determination**. The input to the system is a keyword for all the related tweets relevant to the keyword. The output is the sentiment polarity of the tweet as the same quoted in the introduction. The inputs are given from the Search Box. The Tweeter Object creation takes the entered keyword and searches using the current tweets and returns the twitter object. The opinion direction identification function takes the given tweet and summarizes the whole tweet into **3** categories: **Positive**, **Negative** and N**eutral**. Then finally summarising the result to calculate the number of reviews in favour of the product and the opinion that are in oppose of the product. So user can directly enter the product name and can get the reviews of the people about the product.

3.1 Architecture

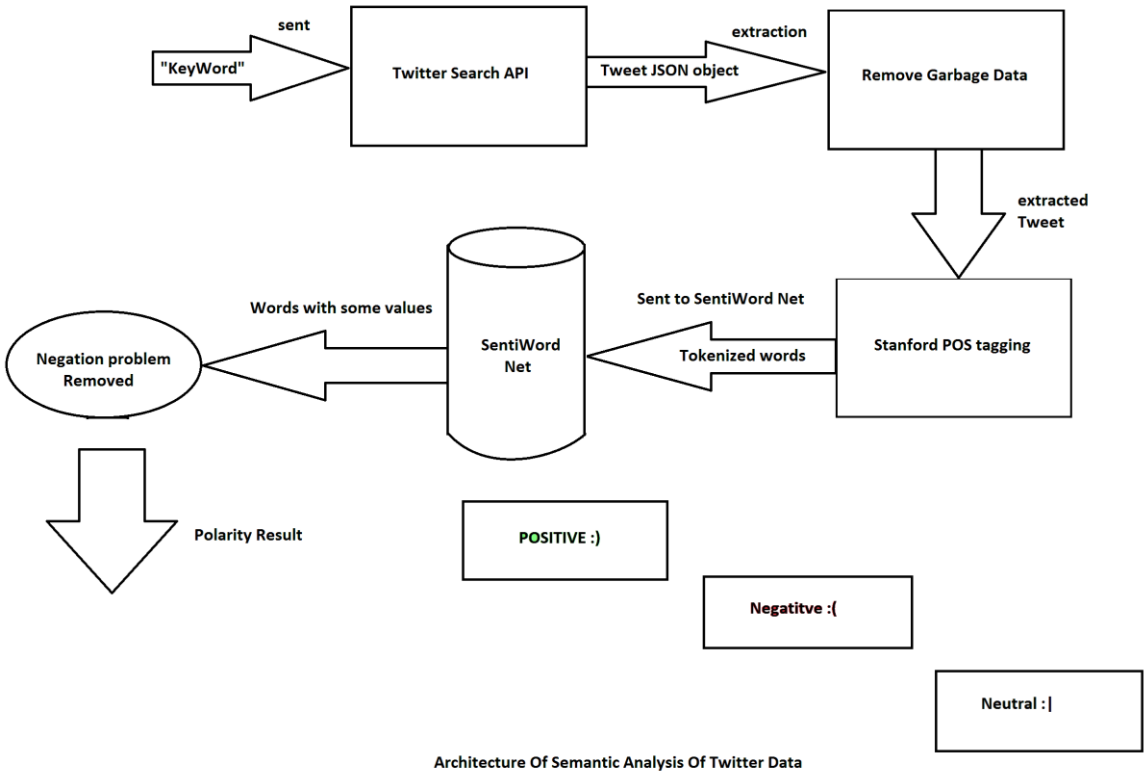


Figure 3.1 Architecture of Semantic Analyser of Twitter data

The architecture of semantic analyser of twitter data consists of three parts:

1) Twitter data collection

2) POS tagging the tweet text

3) Semantic orientation of the tweet

We have collected the reviews form twitter that is tweets by different users about a particular product and then the polarity of the tweet is determined that either it is negative, positive and neutral. For determining the polarity of the tweet the sentiment of each word of the tweet is calculated and scores are given to public opinions of the products which could being used to compare between the products where a statistical output was produced to show the results.

3.1.1 TWITTER DATA COLLECTION:

Tweeter Object creation

Twitter is a real-time information network that connects you to the latest stories, ideas, opinions and news about what you find interesting. Simply find the accounts you find most compelling and follow the conversations. At the heart of Twitter are small bursts of information called Tweets. Each Tweet is 140 characters long, but don't let the small size fool you—you can discover a lot in a little space.

Hence, for sentiment analysis of any tweet first we have to send a "**keyword**" .This keyword is heart of our mining process. This is one of the major advantages of the streaming API, one of type Twitter Search API. Tweets delivered with this method are basically real-time, with a lag of a second or two at most between the times the tweet is posted and it is received from the API. But this lag can be greater at times of Twitter overload. Generally a tweet aggregation system based on the streaming API can be assumed to be real-time. Since aggregation with the search API is run as a repeated series of requests, the lag will be be equal to the delay between requests. Research says that never run a search based aggregation at a rate faster than one request each 60 seconds. By the use of this streaming **Twitter Search API**, we collect the tweets which are related to the entered "**keyword**". The JSON creates JavaScript Object of each tweet and returns the same to the calling function.

For Example: **Searched Keyword: "ipl6"**

Retuned sample Tweets:

"@**theiplgamer**: ipl 6? hell no ! its champions league all the way **#ipl6** "

"@**rosstaylor**: @punewarriorsipl Congo to all team and best of luck for next match ☺ **#ipl6**"

3.1.2 Tweet Data Refinement

Twitter Data Refinement

The main challenges of processing tweets emerge because of the size restriction of 140 characters. This restriction makes people use shortcuts, omit prepositions and use emoticons. Moreover, In Twitter, people are often use hash-tags attached to a single word expression, for referring to a trending topic (e.g. #ipl6) and the '@' character when referring to another Twitter account (e.g. @rosstaylor). Another problem is the language the tweets are not language specific that is it contains different languages so it is very difficult to analyse the opinion.

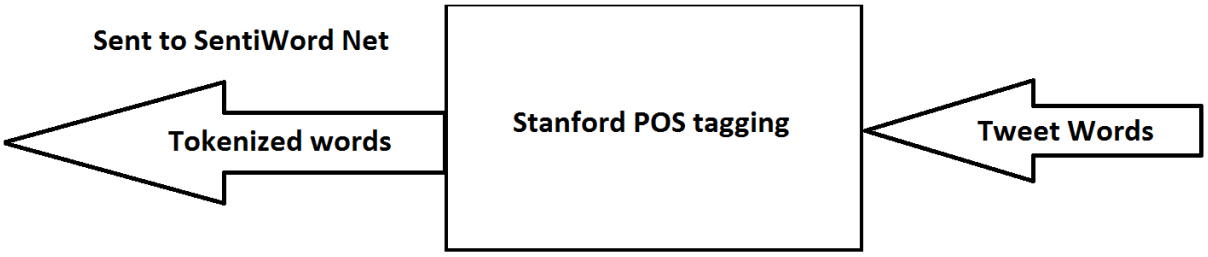
So, we will use extraction technique of removal of garbage data from our tweet. Garbage words associated with a general tweet are username, time, date and shortcuts with no significant meaning. As we are no longer interested say in time and date to make a distinction about a particular tweet. Hence we manually remove these words from our tweet. This may harm the accuracy of the result, but we can achieve the max accuracy if there were no garbage data. As we are using **SentiWordNet** for checking the polarity value of each word from a tweet. This process will make our analysis result computation process easy to implement and process with large data set.

Example:

Tweet: "@**theiplgamer**: ipl 6? hell no ! its champions league all the way **#ipl6** "

Refined tweet: ipl 6? hell no ! its champions league all the way

3.2 POS TAGGING THE TWEET TEXT



(Parts Of Speech Tagging)

3.2.1 Part-of-Speech Tagging (POS)

The refined tweet is needed to be analysed in a better way according to their type. For this purpose, we are using **Stanford POS tagger** for the determination of a word type in the particular tweet.

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. This tagger uses **Penn Tree Bank** tagset for the determination of word type.

JJ	Adjective	JJR	Adjective, comparative
JJS	Adjective, superlative	NN	Noun, singular or mass
NNS	Noun, plural	NP	Proper noun, singular
NPS	Proper noun, plural	VB	Verb, base form
VBD	Verb, past tense	VBG	Verb, gerund
VBN	Verb, past participle	VBZ	Verb 3rd person singular

Few Penn Tree Bank Tag set Examples

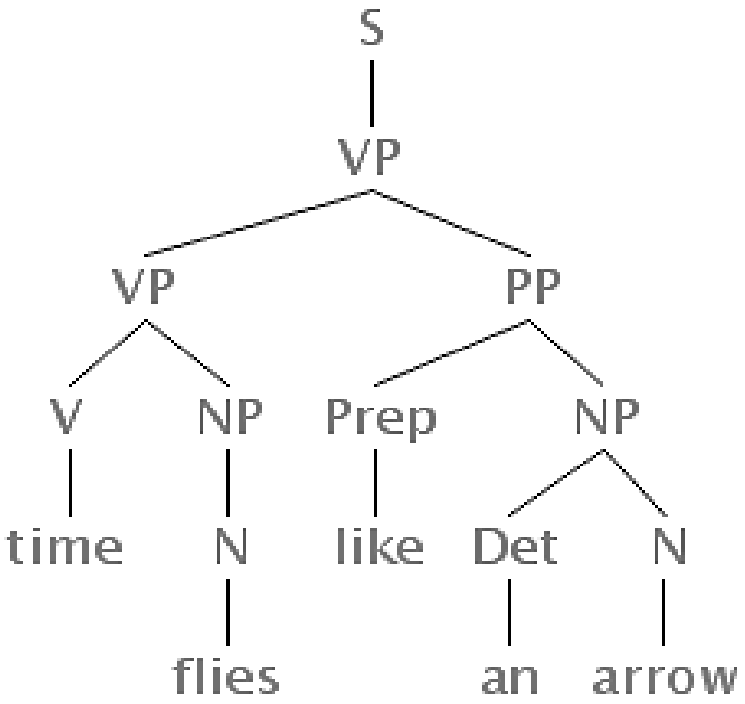
Example:

Tweet: **"@**HTC is a worst mobile."

POS tagged sentence: @HTC_**NN** is_**VB** a_**DT** worst_**JJ** mobile_**NN**.

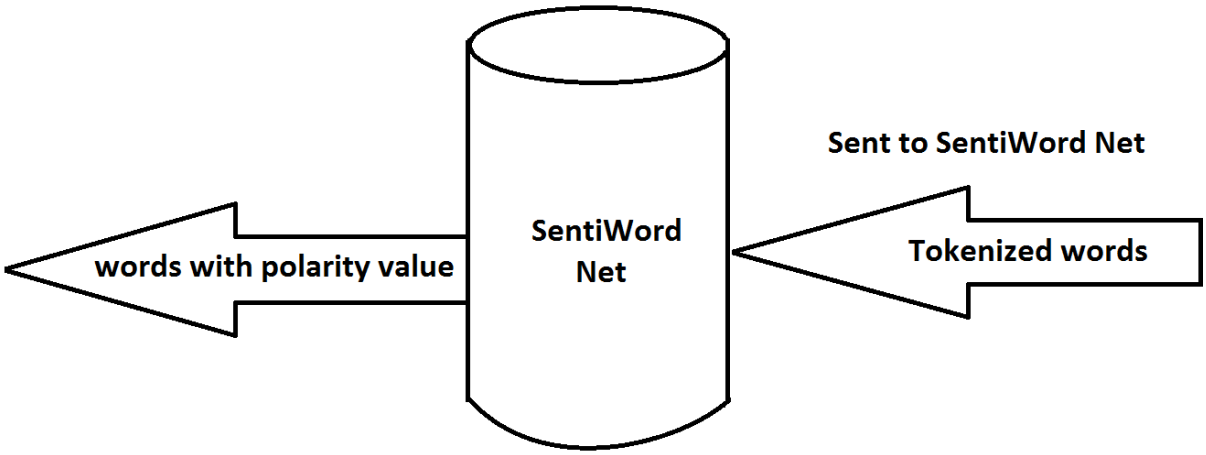
Another Tweet Example: **"**time flies like an arrow**"**

The words in the tweet text are tagged using a POS-tagger so that it assigns a label to each word, allowing the machine to do something with it. It looks like this:



POS tagged sentence: time_**V** flies_**N** like**_PP** an**_det_NP** arrow**_NP**

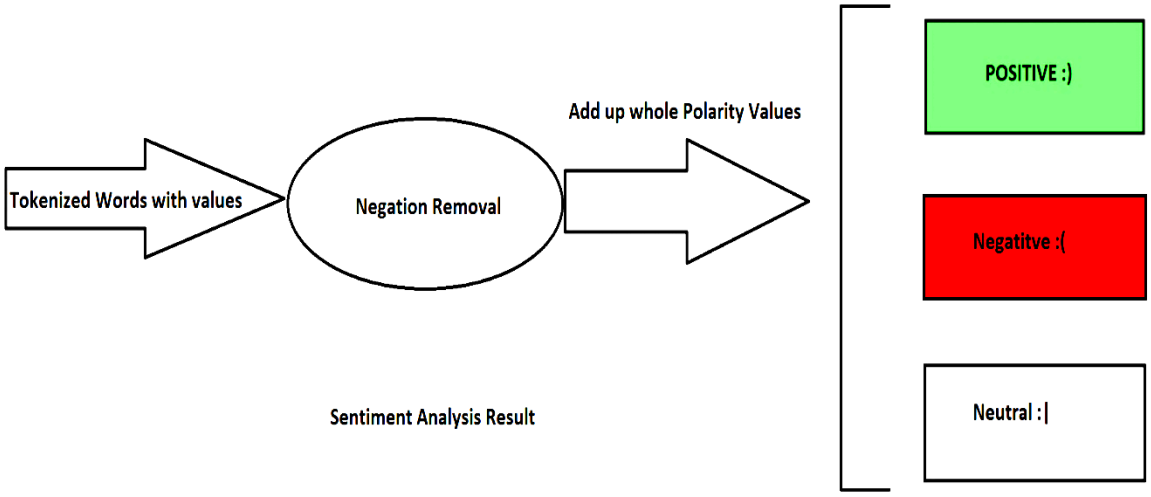
3.3.1 SEMANTIC ORIENTATION OF THE TWEET



Polarity Determination with SentiWordNet

SentiWordNet is used for automatically extracting opinions, emotions and sentiments in text. It allows us to track attitudes and feelings on the web. People write tweets about all sorts of different topics. We can track products, brands and people for example and determine whether they are viewed positively or negatively on the web. By this sentiment analysis, we can categorize the tweets as facts and opinions. This allows business to track the **bad rants**,**new product perception**,**brand perception**, **reputation management** as well as allows individuals to get an **opinion** on something (reviews) on a global scale.

3.2.3 Negation Problem Removal & Polarity Result



The tweet which has negation words simply reverses the overall polarity. Since this is no auto implementation is done; so we have to detect the problem of negation if any,and give the exact polarity result .
Tweet Example: "samsung mobile is not bad"

Outcome: **"Negative tweet"**

"*If we calculate the polarity sum of this tweet example, it comes* ***negative****; whereas it contains problem of negation, then output must be* ***positive*** *(weak)***"**

This evaluation is to measure the reliability of the attached polarity scores of sentiment lexicons. A typical approach to sentiment analysis is to start with a lexicon of positive and negative words and phrases. In these lexicons, entries are tagged with their prior out of context polarity. How far the present SentiWordNet Set, a prior polarity lexicon can help to identify polarity in text.

MODULE WISE FLOW DIAGRAM

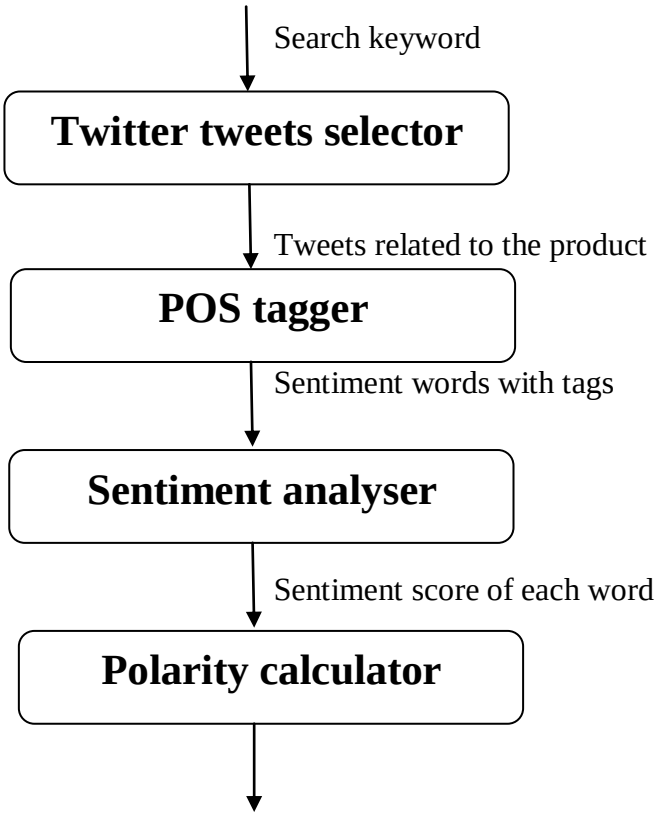


Figure: Data flow between the modules

Description:

Tweet selector is responsible for collecting tweets about a particular topic the input to tweet selector is the search query and the output is the relevant tweets. Then these tweets are send input to the POS tagger here the tweet is tagged that is each word of the tweets is determined that whether it is a noun or adjective or verb etc. after this for each word sentiment is determined that is the sentiment value of each word is obtained and send to the polarity calculator where the polarity of the tweet is calculated using the sentiment of each of the word and a final result is produced the whether the tweet is positive or negative.

PROPOESED ALGORITHM

Twitter tweet selector:

Twitter is a popular social media place to look for information about literally any product. Also the nature of tweets in twitter that its length cannot be greater tha 140 characters interests us because we deal with sentence level sentiment detection in the following sections. We implemented the Twitter API to search the web in an automated manner and collect data of interest. Given a product, our desired data are the tweets that reflect some opinion about the product. To extract text from the tweet we need JSON object we used the JSON-java-master for getting the tweet text from the JSON object that is returned by the search API. The API allows to input a search query along with various other preferences like

1) Location of the tweet

2) Username of the person who tweeted, but we do not take in this information.

3) Tweets between certain period, or tweets since a date or tweets until a date. Twitter does not provide any data older than a week hence it is necessary to run the data retrieval program continuously in order to obtain updated tweets.

Since many of the tweets are advertisement and promotional tweets and many tweets refer to some other links or web pages. These tweets are also removed from the data since these tweets don't contain opinion and is noise in our data. Now that data is on hand we move to sentiment analysis module to associate the data with numerical score.

POS tagger:

After removing the garbage data the tweets are sent to the POS tagger.
In most sentences a user talks about a noun and we capture all forms of noun. Detecting the part-of-speech of a word is by itself a complex task and it is left to the Stanford POS Tagger library. There are few real time tweets/comments which fail using this library because they are not grammatically correct or constructed with mostly colloquial phrases.

Sentiment Analyser:

Sentiment analyser calculates the sentiment of each word using the SentiWordNet data. So we get a positive or negative score of a word accordingly. The words are sent to sentiment analyser with their respective tags. Generally the sentiment in a sentence is reflected by the adjective but some sentences also uses the verb as the sentiment (ex-loving, caring).so these words are also sent with adjective tags to analyse the value.

Pseudo Code

For each word in a sentence, identify its Part-of-speech using Stanford POS Tagger

if the word is adjective, noun, adverb

total score=total score +score of the word

if the word is verb

total score=total score +max(score of the word as verb, score of the word as adjective)

if the word is not

that inverse the sentiment of the words following it

Polarity calculator

Detecting emoticons is done at the final stage of scoring. Emoticons are a very straightforward way of expressing opinions. When we encounter a positive emoticon we add value to the positive feature and similarly on finding a negative emoticon we deduct value from a negative feature thus making it more negative. If not is used before an adjective then the sentiment words following it is reversed (like “not bad”, “not good”).

Finally the output for the tweet is either positive or negative dependent upon the sentiment. So from this the count of all the positive reviews is calculated and the count of negative reviews is calculated and a total score is generated for the product.

Pseudo Code

For each of the tweet

if the total score is positive

positive

else if the total score is negative

negative

else

neutral

EXPLANATION

General Sentiment analysis

Sentiment analysis has been done on a range of topics. For example, there are sentiment analysis studies for movie reviews, product reviews, sports, news and politics. Research shows that sentiment analysis is more difficult than traditional topic-based text classification, despite the fact that the number of classes in sentiment analysis is less than the number of classes in topic-based classification. In sentiment analysis, the classes to which a piece of text is assigned are usually negative or positive. They can also be other binary classes or multi valued classes like classification into 'positive', 'negative' and 'neutral'. The main reason that sentiment analysis is more difficult than topic-based text classification is that topic-based classification can be done with the use of keywords while this does not work well in sentiment analysis. Other reasons for difficulty are: sentiment can be expressed in subtle ways without any ostensible use of negative words; it is difficult to determine whether a given text is objective or subjective.

3.4.1 Classification and approaches

As elaborated in the introduction, sentiment analysis is formulated as a text-classification problem. However, the classification can be approached from different perspectives suited to the work at hand. Depending on the task at hand and perspective of the person doing the sentiment analysis, the approach can be discourse-driven, relationship-driven, language-model-driven, or keyword-driven. Some of the perspectives that can be used in sentiment classification are discussed briefly in the subsequent subsections.

3.4.1.1 Knowledge-based approach

In this approach, sentiment is seen as the function of some keywords. The main task is the construction of sentiment discriminatory-word lexicons that indicate a particular class such as positive class or negative class. The polarity of the words in the lexicon is determined prior to the sentiment analysis work. There are variations to how the lexicon is created. For example, lexicons can be created by starting with some seed words and then using some linguistic heuristics to add more words to them, or starting with some seed words and adding to these seed words other words based on frequency in a text. For some domains of tasks, there are publicly available discriminatory word lexicons for use in sentiment analysis.

3.4.1.2 Relationship-based approach

This classification task can be approached from the different relationships that may exist in or between features and components. Such relationships include relationships between discourse participants, relationships between product features. For example, if one wants to know the sentiment of customers about a product brand, one may compute it as a function of the sentiments on different features or components of it.

3.4.1.3 Language models

In this approach the classification is done by building n-gram language models. Presence or frequency of n-grams might be used. In traditional information retrieval and topic-oriented classification, frequency of n-grams is shown to give better results. In sentiment classification of movie reviews found that term-presence gives better results than term frequency. They indicate that unigram presence is more suited for sentiment analysis. But among researchers found that bi-grams and tri-grams worked better than unigrams in sentiment classification of product reviews.

3.4.1.4 Discourse structures and semantics

In this approach, discourse relation between text components is used to guide the classification. For example in reviews, the overall sentiment is usually expressed at the end of the text. As a result the approach to sentiment analysis, in this case, might be discourse-driven in which the sentiment of the whole review is obtained as a function of the sentiment of the different discourse components in the review and the discourse relations that exist between them. In such an approach, the sentiment of a paragraph that is at the end of the review might be given more weight in the determination of the sentiment of the whole review. Semantics can be used in role identification of agents where there is a need to do so, For example **[@: KKR beats MI**] is different from **[@: MI beats KKR**].

3.4.2 Sentiment analysis techniques

Using one or a combination of the different approaches in this subsection, one can employ one or a combination of machine learning techniques. Specifically, one can use unsupervised techniques, supervised techniques or a combination of them.

3.4.2.1 Unsupervised Techniques

In unsupervised technique, classification is done by a function which compares the features of a given text against discriminatory-word lexicons whose polarity are determined prior to their use. For example, starting with positive and negative word lexicons, one can look for them in the text whose sentiment is being sought and register their count. Then if the document has more positive lexicons, it is positive, otherwise it is negative. A slightly different approach in which a simple unsupervised technique to classify reviews as recommended (thumbs up) or not recommended (thumbs down) based on se-mantic information of phrases containing an adjective or adverb. He computes the semantic orientation of a phrase by mutual information of the phrase with the word 'excellent' minus the mutual information of the same phrase with the word 'poor'. Out of the individual semantic orientation of phrases, an average semantic orientation of a review is computed. A review is recommended if the average semantic orientation is positive, not recommended otherwise.

3.4.2.2 Supervised Techniques

The main task here is to build a classifier. The classifier needs training examples which can be labelled manually or obtained from a user-generated user-labelled online source. It has been shown that Supervised Techniques outperform unsupervised techniques in performance. Supervised techniques can use one or a combination of approaches we saw above. For supervised techniques, the text to be analysed must be represented as a feature vector.

3.4.2.3 Combined Techniques

There are some approaches which use a combination of other approaches. They start with two word lexicons and unlabelled data. With the two discriminatory-word lexicons (negative and positive), they create pseudo-documents containing all the words of the chosen lexicon. After that, they compute the cosine similarity between these pseudo-documents and the unlabelled documents.

3.4.2.4 Feature Engineering

Since most of sentiment analysis approaches use or depend on machine learning techniques, the salient features of text or documents are represented as feature vector. The following are the features used in sentiment analysis. Term presence or term frequency. In standard Information retrieval and text classification, term frequency is preferred over term presence. However, Research shows that words that appear only once in a given corpus are good indicators of high-precision subjectivity. Term can be unigrams, bi-grams or other higher-order n-grams. Whether unigrams or higher order n-grams give better results is not clear. POS (Part of speech) Tags: POS is used to disambiguate sense which in turn is used to guide feature selection. For example, with POS tags, we can identify adjectives and adverbs which are usually used as sentiment indicators. But, It was found that adjectives performed worse than the same number of unigrams selected of the basis of frequency. Syntax and negation: Collocations and other syntactic features can be employed to enhance performance. In some short text (sentence-level) classification tasks, algorithms using syntactic features and algorithms using n-gram features were found to give same performance. Negation is also an important feature to take into account since it has the potential of reversing a sentiment.

Technology used.

1) Twitter search API v1:

Returns relevant tweets that match a specified query The Twitter search API is a dedicated API for running searches against the real-time index of recent Tweets. The Search API is not complete index of all Tweets, but instead an index of recent Tweets. At the moment that index includes between 6-9 days of Tweet. HTTP method is GET and the return format is atom or JSON.

2) Stanford POS Tagger, v3.1.4:

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. This software is a Java implementation of the log-linear part-of-speech taggers. The English taggers use the Penn Treebank tag set. The tagger was originally written by Kristina Toutanova. Since that time, Dan Klein, Christopher Manning, William Morgan, Anna Rafferty, Michel Galley, and John Bauer have improved its speed, performance, usability, and support for other languages.

3) SENTIWORDNET 3.0:

SentiWordNet is lexical resource explicitly devised for supporting sentiment classification and opinion mining applications. SENTIWORDNET 3.0 is an improved version of SENTIWORDNET 1.0, a lexical resource publicly available for research purposes, now currently licensed to more than 300 research groups and used in a variety of research projects worldwide. SENTIWORDNET 1.0 and 3.0 are the result of automatically annotating all WORDNET synsets according to their degrees of positivity, negativity, and neutrality. SENTIWORDNET 1.0 and 3.0 differ (a) in the versions of WORDNET which they annotate (WORDNET 2.0 and 3.0, respectively), (b) in the algorithm used for automatically annotating WORDNET, which now includes (additionally to the previous semi-supervised learning step) a random-walk step for refining the scores. SENTIWORDNET 3.0 manually annotated for positivity, negativity, and neutrality; these results indicate accuracy improvements of about 20% with respect to SENTIWORDNET 1.0.

Limitations

During data collection one issue to be raised is making a compromise between large data extraction and low quality and lesser amount of data extracted with high quality. Writing a specific program for general data collection is quite challenging. One of the limitations of this project was collecting data of certain specific products there by collecting a larger amount of data. Due to the above mentioned problem certain unwanted noises are introduced. Another limitation in this project was during the sentiment analysis phase where statements made by people are not always in correct grammar and with spelling errors. For tagging different parts of a sentence, the used sentence parser often finds wrong identifications of sentence parts limiting the efficiency of the method used. We have collected the data using the Twitter API which also return a specific amount of the tweet data. So we have applied the algorithm to the available data.

EXPERIMENTAL RESULTS

For the search query “Samsung glaxay s3”

Positive opinion tweet:

“awesome phone I love it the samsung galaxy s3 is amazing”

Output of the POS tagger:

“awesome_JJ phone_NN I_PRP love_VBP it_PRP the_DT samsung_JJ galaxy_NN s3_NN is_VBZ amazing_JJ”

Sentiment of each word:

Score of awesome is 0.75

Score of phone is 0.0

Score of love is 0.030000000000000006

Score of samsung is 0.0

Score of galaxy is 0.0

Score of s3 is 0.0

Score of is is 0.0

Score of amazing is 0.25

Total score is 1.03

So a total positive score is obtained so the opinion reflected in the tweet is positive and is in favour of the product.

Negative opinion tweet:

“my samsung galaxy s3 has frozen twice this morning. loads of problem. terrible phone. never have a fault free day with it!!!”

Output of the POS tagger:

“my_PRP$ samsung_JJ galaxy_NN s3_NN has_VBZ frozen_VBN twice_RB this_DT morning_NN ._. loads_NNS of_IN problem_NN ._. terrible_JJ phone_NN ._. never_RB have_VBP a_DT fault_NN free_JJ day_NN with_IN it_PRP !!!_IN”

Sentiment of each word:

Score of samsung is 0.0

Score of galaxy is 0.0

Score of s3 is 0.0

Score of has is 0.0

Score of frozen is 0.0

Score of twice is 0.0

Score of morning is 0.0

Score of loads is -0.25

Score of problem is -0.045454545454545456

Score of terrible is -0.625

Score of phone is 0.0

Score of never is 0.0

Score of have is 0.0

Score of fault is -0.625

Score of free is 0.0

Score of day is 0.0

Total score is -1.5454545454545454

So a total negative score is obtained so the opinion reflected in the tweet is negative or not in favour of the product.

Neutral opinion tweet(fact tweets):

1) “my mum got the samsung galaxy s3”

Output of the POS tagger:

“my_PRP$ mum_NN got_VBD the_DT samsung_JJ galaxy_NN s3_NN”

Sentiment of each word:

Score of mum is 0.0

Score of got is 0.0

Score of samsung is 0.0

Score of galaxy is 0.0

Score of s3 is 0.0

Total score is 0.0

2) “ready to become a samsung galaxy s3 owner”

Output of the POS tagger:

“ready_JJ to_TO become_VB a_DT samsung_JJ galaxy_NN s3_NN owner_NN”

Sentiment of each word:

Score of ready is 0.0

Score of become is 0.0

Score of samsung is 0.0

Score of galaxy is 0.0

Score of s3 is 0.0

Score of owner is 0.0

total score is 0.0

So a total neutral score is obtained so no opinion reflected in the tweet is neither negative nor positive tweet.

CODE RESULT SNAPSHOTS

Positive opinion tweet output:

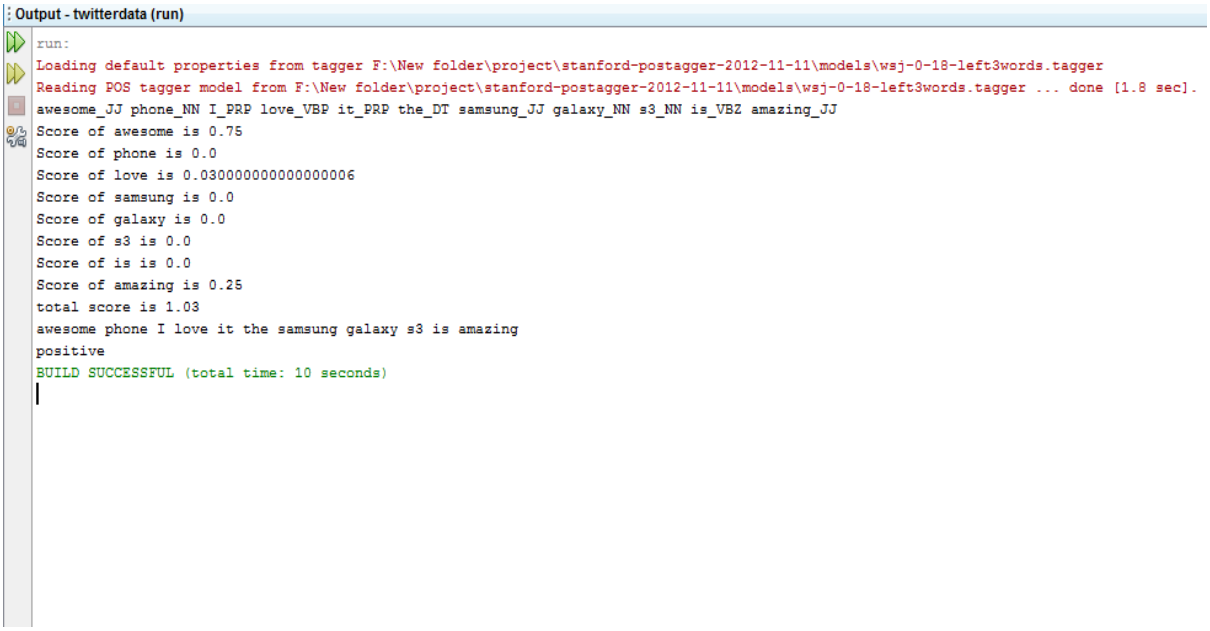


Figure 1. Output for positive tweet: “awesome phone I love it the samsung galaxy s3 is amazing”

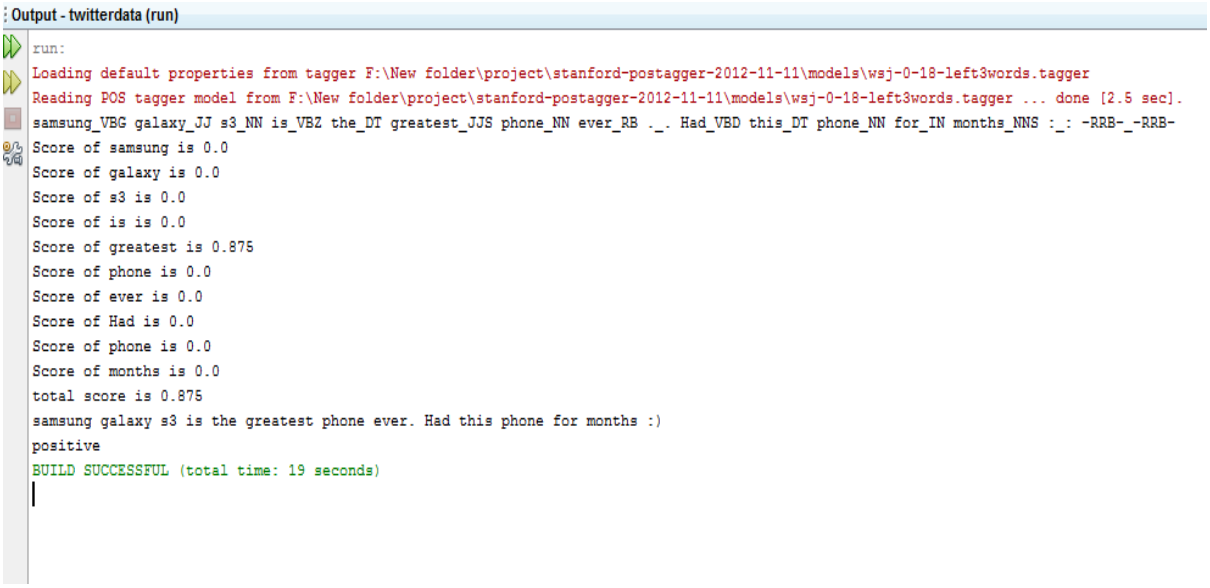


Figure 2. Output for positive tweet: “samsung galaxy s3 is the greatest phone ever. Had this phone for months”

Negative opinion tweet output:

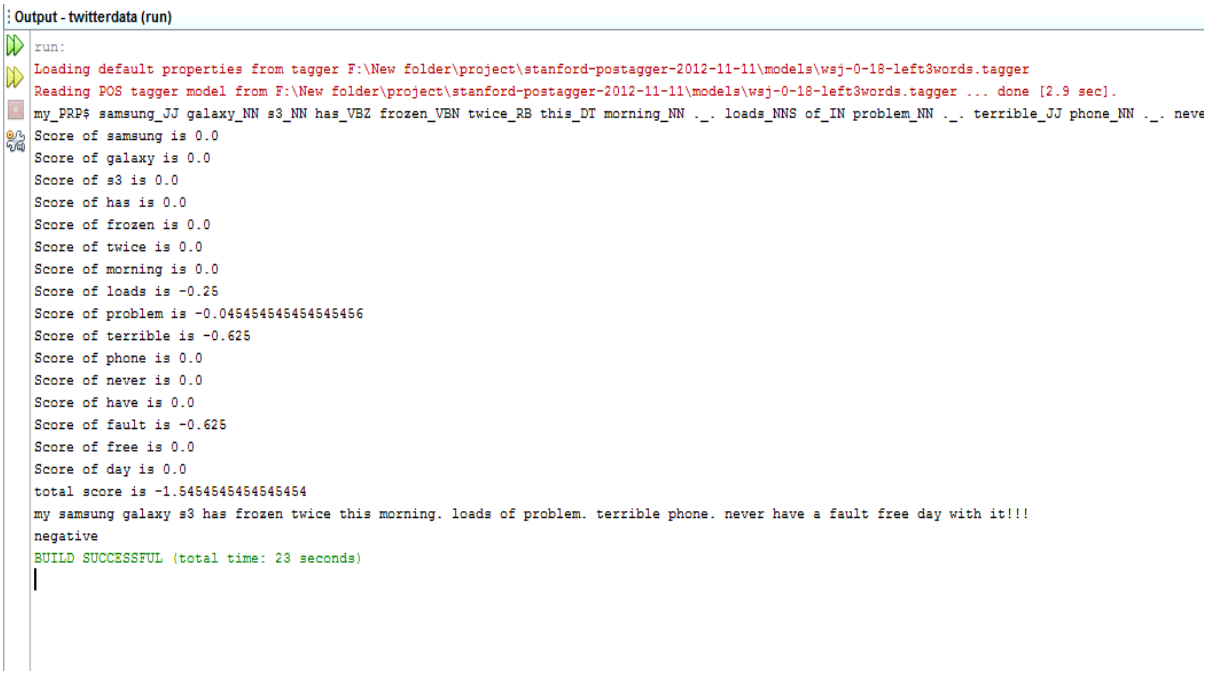


Figure 3. Output for negative tweet: “my samsung galaxy s3 has frozen twice this morning. loads of problem. terrible phone. never have a fault free day with it!!!”

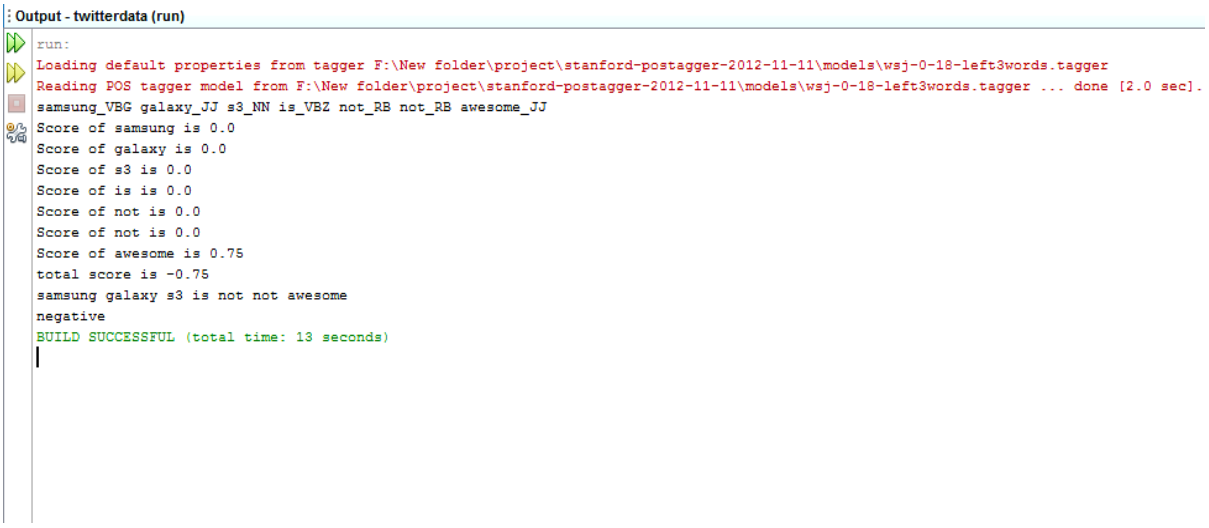


Figure 4 . Output for negative tweet: “samsung galaxy s3 is not not awesome”

Neutral opinion tweet output:

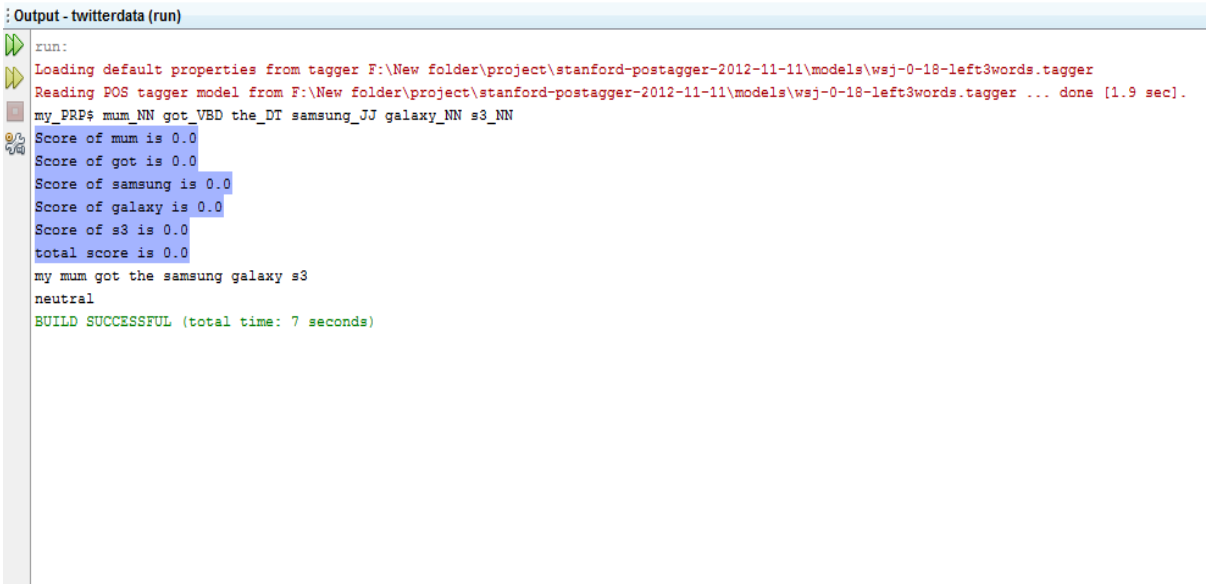


Figure 5. neutral tweet output: “my mum got the samsung galaxy s3”

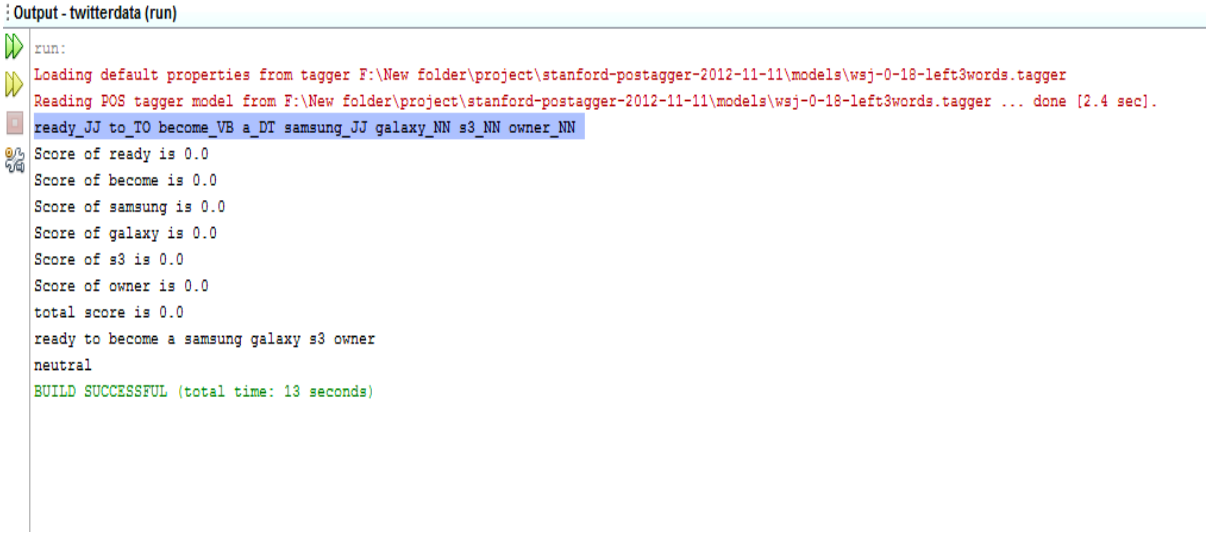


Figure 6. Neutral tweet output: “my mum got the samsung galaxy s3”

CONCLUSION AND FUTURE WORK

We searched the data for mainly 4 mobile brands. The following output is produced for the search :

Product	Tweets without noise
Samsung galaxy	72
Nokia lumia	101
blackberry	95
Iphone	112

Table 1: Output of tweets about different products

Product	Positive tweets	Negative tweets	Neutral tweets
Samsung galaxy	**35**	**26**	**21**
Nokia lumia	**34**	**8**	**59**
Blackberry	**30**	**25**	**40**
Iphone	**51**	**24**	**37**

Table 2: Output of reviews of different products

This eases a user to visualize the review on the product and also makes it easy to the user to compare the scores with other products. The features from the negative opinion can also be extracted and comparison can also be done on the basis of features. So the user can also analyse why the reviews are negative or positive

CODE

Module 1:Main.java

```
package twitterdata;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.net.URL;

import org.json.JSONArray;

import org.json.JSONObject;

import edu.stanford.nlp.tagger.maxent.MaxentTagger;

public class Main {

   public static boolean selectTweet(String s)

   {

      return !s.contains("http://");

   }

   public static void main(String[] args) throws Exception

   {

      Polarity pol = new Polarity();

      MaxentTagger tagger = new MaxentTagger("F:\\New folder\\project\\stanford-
postagger-2012-11-11\\models\\wsj-0-18-left3words.tagger");

      BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

      String urlstr = "http://search.twitter.com/search.json?q=";

      System.out.print("Search for : ");

      int counter_neutral=0,counter_positive=0,counter_negative=0;

      String inp1=in.readLine();

      String inp=inp1.toLowerCase();

      String temp="";
```

```
for(int i=0;i<inp.length();i++)
	{
	if(inp.charAt(i)==' ')
	{
	urlstr+=temp;
	urlstr+="%20";
	temp="";
	}
	else
	{
	temp+=inp.charAt(i);
	}
	}
	urlstr +=temp;
	urlstr +="&page=";
	int stop=0;
	int count=0;
	int pageno=0;
	while(stop==0)
	{
	StringBuffer buff = new StringBuffer();
	pageno++;
	URL url = new URL( urlstr+Integer.toString(pageno));
	BufferedReader br = new BufferedReader(
	new InputStreamReader(
	url.openConnection().getInputStream()));
```

```
 int c;
while((c=br.read())!=-1)
{
buff.append((char)c);
}
br.close();
JSONObject js = new JSONObject(buff.toString());
JSONArray tweets = js.getJSONArray("results");
JSONObject tweet;
for(int i=0;i<tweets.length();i++) {
tweet = tweets.getJSONObject(i);
String s=tweets.getJSONObject(i).getString("text");
if(selectTweet(s))
{
count++;
String tweettext1=tweets.getJSONObject(i).getString("text");
String tweettext=tweettext1.toLowerCase();
int resultscore = Polarity.getPolarity(tweettext, inp, tagger);
    if (resultscore == 0)
    {
    System.out.println(tweettext);
    System.out.println("neutral");
    counter_neutral++;
    }
    else if (resultscore > 0)
    {
```

```
            System.out.println(tweettext);
            System.out.println("positive");
            counter_positive++;
            }
            else
            {
            System.out.println(tweettext);
            System.out.println("negative");
            counter_negative++;
            }
    }
        if((counter_positive+counter_negative+counter_neutral)==100)
       {
       break;
       }
 }
}
     System.out.println("Total tweets related to the product
"+(counter_positive+counter_negative+counter_neutral));
     System.out.println("positive tweets "+counter_positive);
     System.out.println("negative tweets "+counter_negative);
     System.out.println("neutral tweets "+counter_neutral)
}
}
```

Module 2:TagText.java

```
package twitterdata;

import java.io.IOException;

import edu.stanford.nlp.tagger.maxent.MaxentTagger;

public class TagText

{

 public  String stringtagger(String sample,MaxentTagger tagger) throws IOException,

 ClassNotFoundException

    {

     String tagged = tagger.tagString(sample);

     return tagged;

     }

}
```

Module 3: SWN3.java

```
package twitterdata;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Set;

import java.util.Vector;

public class SWN3

{

   public  String pathToSWN = "F:\\New
folder\\project\\SentiWordNet_3.0.0_20130122.txt";

   private HashMap<String, String> _dict;

   public Double score(String testword,String type){

   double _score = 0.0;

   int flag=0;

   HashMap<String, Vector<Double>> _temp = new HashMap<String,
Vector<Double>>();

   try

   {

   BufferedReader csv = new BufferedReader(new FileReader(pathToSWN));

   String line = "";

   while((line = csv.readLine()) != null)

   {

   String[] data = line.split("\t");
```

```
Double score = Double.parseDouble(data[2])-Double.parseDouble(data[3]);
String[] words = data[4].split(" ");
for(String w:words)
{
String[] w_n = w.split("#");
if(w_n[0].equals(testword) && data[0].equals(type))
{
w_n[0] += "#"+data[0];
int index = Integer.parseInt(w_n[1])-1;
if(_temp.containsKey(w_n[0]))
{
   Vector<Double> v = _temp.get(w_n[0]);
   if(index>v.size())
   for(int i = v.size();i<index; i++)
   v.add(0.0);
   v.add(index, score);
   _temp.put(w_n[0], v);
}
else
{
   Vector<Double> v = new Vector<Double>();
   for(int i = 0;i<index; i++)
   v.add(0.0);
   v.add(index, score);
   _temp.put(w_n[0], v);
 }
```

```
 flag=1;
    break;
}
}
if(flag==1)
break;
}
Set<String> temp = _temp.keySet();
for (Iterator<String> iterator = temp.iterator(); iterator.hasNext();)
{
String word = (String) iterator.next();
Vector<Double> v = _temp.get(word);
_score = 0.0;
double sum = 0.0;
for(int i = 0; i < v.size(); i++)
_score += ((double)1/(double)(i+1))*v.get(i);
for(int i = 1; i<=v.size(); i++)
sum += (double)1/(double)i;
_score /= sum;
}
}
catch(Exception e)
{
}
return _score;
}
}
```

Module 4:Polarity.java:

```
package twitterdata;

import java.io.IOException;

import edu.stanford.nlp.tagger.maxent.MaxentTagger;

public class Polarity {

    public static String sentiwordType(String str)

    {

    if(str.equals("JJ")|| str.equals("JJR")||str.equals("JJS"))

    return "a";

    if(str.equals("NN")||str.equals("NNS")||str.equals("NP")||str.equals("NPS"))

    return "n";

    if(str.equals("RB")||str.equals("RBR")||str.equals("RBS"))

    return "r";


if(str.equals("VB")||str.equals("VBD")||str.equals("VBG")||str.equals("VBN")||str.equals(
"VBP")||str.equals("VBZ"))

    return "v";

    return "s";

    }

    public static int getPolarity(String tweet,String inp,MaxentTagger tagger)throws
IOException,

 ClassNotFoundException

    {

    int result=0;

    TagText p=new TagText();

    double resultscore=0;

    String tagged=p.stringtagger(tweet,tagger);

    String[] data = tagged.split(" ");

    String ngram="";
```

```
for(String W:data)
{
   String[] wt=W.split("_");
   ngram =ngram+wt[1]+" ";
}
System.out.println(tagged);
    for(String w:data)
    {
       String[] wt = w.split("_");
       SWN3 s = new SWN3();
       String type=sentiwordType(wt[1]);
       if(!type.equals("s"))
       {
          if(type.equals("v")||type.equals("r"))
          {
             double tempscore=s.score(wt[0],type);
             if(tempscore<s.score(wt[0],"a"))
             {
                resultscore+=s.score(wt[0],"a");
                System.out.println("Score of "+wt[0]+" is "+s.score(wt[0],"a"));
             }
             else
             {
                resultscore += s.score(wt[0], type);
                System.out.println("Score of "+wt[0]+" is "+s.score(wt[0], type));
             }
          }
          else
```

```
                {
                    resultscore += s.score(wt[0], type);
                    System.out.println("Score of "+wt[0]+" is "+s.score(wt[0], type));
                 }

               }
              }
        if(tweet.contains("not"))
        {
           resultscore=resultscore*(-1);
        }
System.out.println("total score is "+resultscore);
if(resultscore>0)
{
   result=1;
}
else if(resultscore<0)
{
   result=-1;
}
else
{
   result=0;
}
return result;
}
}
```

REFERENCES

[1] https://dev.twitter.com/docs/api/1/get/search

[2] http://nlp.stanford.edu/software/tagger.shtml

[3] http://sentiwordnet.isti.cnr.it/

[4] B. Pang and L. Lee. “Opinion Mining and Sentiment Analysis, volume 2 of Foundations and Trends in Information Retrieval”.

[5] Akshay Minocha and Navjyoti Singh “Generating domain specific sentiment lexicons using the Web Directory”.