Customizable Twitter Bots for Improved QoS in Product Marketing

A PROJECT REPORT

Submitted by

ANUSH BASKARAN (13BCE0800)
BALABADRAPATHRUNI RAMYA (13BCE0613)
TARUNI ANAND M (13BCE0036)

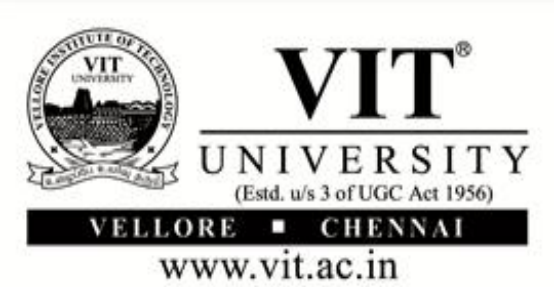
In partial fulfilment for the award of the

B.Tech. Degree

in

Computer Science and Engineering

School of Computing Science and Engineering



School of Computer Science and Engineering

DECLARATION

We hereby declare that the project entitled **"Customizable Twitter Bots for Improved QoS in Product Marketing"** submitted by us to the School of Computer Science and Engineering, VIT University, Vellore-14 in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by us under the supervision of **Dr. Iyengar N.Ch.S.N., Senior Professor.** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Taruni Anand M (13BCE0036)

Anush Baskaran (13BCE0800)

Baladhrapathruni Ramya(13BCE0613)

School of Computer Science and Engineering

CERTIFICATE

The project report entitled "**Customizable Twitter Bots for Improved QoS in Product Marketing**" is prepared and submitted by **Candidates Taruni Anand M (Register No: 13BCE0036), Anush Baskaran(Register No: 13BCE0800) and Balabadrapathruni Ramya (Register No: 13BCE0613).** It has been found satisfactory in terms of scope, quality and presentation as partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** in VIT University, India.

Guide

Dr. Iyengar N.Ch.S.N.

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We would like to take this opportunity to thank our internal guide, **Dr. Iyengar N.Ch.S.N., Senior Professor**, School of Computer Science and Engineering, VIT University, Vellore-14 for his constant guidance and encouragement, which helped us complete the project.

We express our sincere gratitude to **Prof. Senthilkumar R., Head of Department**, B.Tech Computer Science and Engineering, VIT University, Vellore-14 and **Dr. Arunkumar T., Dean**, School of Computer Science and Engineering, VIT University, Vellore-14, for giving us the opportunity to undertake our project work at the university.

We would also like to express our heartfelt gratitude to **Dr. G.Viswanathan**, **Chancellor**, VIT University for providing a wide range of facilities that helped complete our project. We are extremely grateful to our Vice Presidents. **Shri. Shankar Viswanathan, Shri. Sekar Viswanathan, Shri G.V.Selvam**, and **Dr. Anand A. Samuel, Vice chancellor** and **Dr.S.Narayanan, Pro-Vice Chancellor** for providing necessary resources.

Last but not the least, we would like to thank the management of VIT University for permitting us to use the library resources. This acknowledgement would be incomplete without expressing our whole hearted thanks to our family and friends for constantly motivating us during the course of the work.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Expansion
HTTP	Hyper Text Transfer Protocol
WWW	World Wide Web
JSON	JavaScript Object Notation
QOS	Quality of Service
API	Application Programming Interface
REST API	RESTful API

ABSTRACT

Microblogging is a combination of blogging and instant messaging that allows users to create short messages to be posted and shared with an audience online. Microblogging tends to offer a portable communication mode that feels organic and spontaneous to many and hence captures the public imagination. It attracts users due to its instant publication with few restrictions on content.

Social platforms like Twitter have become extremely popular means of this new type of blogging. As of March, 2016 Twitter has 320 million users worldwide. Twitter has become a favorite in data mining community due to its excellent API and support for academic research which makes scraping twitter data easy. Twitter has a limit of 140 characters, hence letting the users express their emotions in a concise manner through text, emoticons, hashtags, images etc; As twitter data has a variety of emotions expressed and is available for collection, it is the ideal dataset to perform sentiment analysis on.

The idea of sentiment analysis or opinion mining is to derive the opinion or attitude expressed by the writer. It is the process of computationally determining the emotion of the tweet. Using sentimental analysis we have identified and categorized the nature of the target tweets. This has been achieved through python. Python is an object-oriented, high level programming language with easily implementable syntax which focuses on readability. It is easier to create deep neural networks, that is, deep learning through python. Python also has primary built-in data analysis tools, packages and libraries which help keep the focus on the method of implementation rather than the implementation itself. In our paper we have automated the process of marketing by creating a bot which tweets a reply to the target tweet based on the sentimental analysis performed on it.

Traditionally bots have been used to reply to every tweet which causes spamming. These spam bots tend to like ancient posts, redirecting the user to spam websites which indirectly buries their PC in malware oblivion. An automated bot can be an extremely useful tool for marketing if used in the right way. We use keyword search to first find the target audience which are again refined based on their liking and positivity towards a particular product that we are marketing for. Our bot eliminates

the idea of spamming as we customize and auto tweet only to a specific target audience based on their interest in that particular item.

This paper aims to create an enhanced system for marketing which improves the QoS(Quality Of Service) and is extremely client friendly as all the client needs to do is enter the keywords to find the target tweets and rest will be taken care of by the system.

1. **Introduction**

In recent times, users have started spending a significant amount of their time on social networking platforms (Twitter, Facebook, Orkut), sharing a plethora of personal information and opinions on different aspects of life. Internet users around the world make use of this powerful tool of communication a.k.a microblogging, using Twitter, a web application that provides dual benefits of microblogging and social networking. In this paper, we propose an application that uses the open structure of twitter to perform 2 main tasks: (1) analyze data for sentiment analysis and opinion mining of a particular product and classify this data into positive, neutral and negative reviews (2) use this data to create automated programs, known as bots, which are useful for generating benign tweets. We use this double edged sword to provide companies with the opportunity to make use of these automated bots to generate the right tags to advertise products. This enhanced marketing opportunity provides Quality of Service in terms of cost, time and energy.

1.1. Aim and Objective

While social media is taking up the front scene in communication, Twitter has emerged as one of the most popular websites for people to share messages in a 140 character length message. This data conveys interests and emotions, which can be used to predict several outcomes. Several companies, start ups in particular, can use this data to determine the right tags to be included to use Twitter for easy advertising. There has been no solution for this purpose, and hence becomes an important need of the hour

1.2. Theoretical Background:

Twitter provides its data openly, as all of it is publicly available. The APIs are coded in Python, and can be analysed through several algorithms. The reports of such analysis can be used to predict possible tags to be included in tweets. An application will be made in order to enable the type of data to be collected, and for framing the tweet. This application will be called the customizable Twitter bot. The algorithm will include data regarding followers, topics of interest, as well as trends that are currently popular on the internet. Business

insights can also be developed on this basis, providing valuable information for successful advertising.

1.3. Motivation :

The motivation of Twitter bots will remain unchanged, as they are already present, and easily executable. However, they lack in customizability. A customizable application for tweeting will enhance quality of tweets, improve product reach, and most importantly save lots of time. This will in turn allow cost cutting and easy profile analysis, which can in turn provide extensibility.

1.4. Report Organization:

The report has been organized into four parts. It consists of the Introduction to the concept behind the project, followed by the software requirement specification, the software design specifications, the code and finally the results and conclusion. The SRS consists of the functional and non functional requirements, and the SDS consists of the screenshots and the design aspects of the project. Further, the code has been attached for reference. The results have been provided as a set of cases and our application of the same. The conclusion consists of the final decisions, drawbacks and consistency with the objective, along with any future works plausible in the field.

2. Literature Survey

Recently, there have been several methodologies to analyze sentiments of reviews and large textual matter, in order to aid in marketing and understand customer views and feedback. Using Twitter as a microblogging website, has also found its disadvantages in terms of large number of illegitimate botnets for marketing purposes. When sentiment analysis is used to find profiles of people who are currently interested in a specific product (based on search keys), the audience for marketing becomes targeted. To make this more time efficient, we can use bots to selectively tweet to interested users about the given product. This overcomes the disadvantage of bots spamming unnecessarily to irrelevant users.

Agarwal [1] et al proposed that posts on microblogging sites like Twitter can be classified based on polarity using three models. The models were created

for binary classification of sentiment into positive and negative; as well as a three way classification of tweets into positive, negative and neutral polarity. This was performed using tree kernel based, feature based and unigram models, as well combinations of the same. It was concluded that the combination of tree kernel based model and feature based model outperformed the other models.

Pak [2] et al proposed to build a classifier for microblogging websites by collecting a corpus of positive, negative and objective sentiments, followed by linguistic analysis on the corpora. The analysis was based on key features, based on a "salience" value in addition to entropy. It was concluded that salience provided more accurate discrimination of n-grams than entropy, increasing its efficiency as an analyzer.

Wilson [3] et al proposed a method to determine contextual polarity of a statement based on prior polarity key words. This was done by first classifying the text as neutral or polar based on clues that determine a hint of polarity. This is followed by disambiguating the text as positive, negative, neutral or both based on an inter annotator study. It was concluded that this method could determine polarity of large volumes of data, along with significantly better results than baseline methods.

Go [4] et al proposed a classifier based on distant supervision for microblogging on Twitter, owing to their popularity. They used Twitter data as their training dataset, and trained a binary classifier, to determine either positive or negative polarity of a tweet. This was done by acquainting the classifier with emoticons, and limiting the trainer data to tweets that contained emoticons. It was concluded that machine learning algorithms provide high accuracy in classifying tweets based on polarity.

Chu [5] et al proposed a classification system that was based on an entropy based component, a machine learning based component, an account properties based component and a decision maker based component that uses a combination of features from a user to determine its possibility of being a human, bot or cyborg. The efficiency was determined by the test dataset used. It was concluded that the human to bot to cyborg ratio was nearly 5:4:1, making exploitation of twitter accounts relatively simple.

Kouloumpis [6] et al proposed a sentiment analyzer for Twitter specific to microblogging, by means of comparing it to an existing lexicon. The lexicon consisted of a Hashtag dataset, an Emoticon dataset, iSieve dataset, along with POS features. It was concluded that POS feature analysis was less effective in Twitter sentiment analysis, unlike those of reviews. Hashtags and emoticons analysis proved to be more beneficial in evaluating the sentiment of the given Tweet.

Haustein [7] et al examined the existence and effects of automated bots prevailing in the Twitter environment. It was stated that most of these bots follow tweeters with an intention of the users following the bot back. In addition, there was also rampant usage of cyborgs on Twitter, primarily for spam and marketing purposes. It was specifically noted to identify bots that disseminate links to scientific papers, and concluded that identification of social bots are far more difficult than arXiv.

Wald [8] et al proposed a method to identify bots and cyborgs on Twitter. This was done by engaging ten different feature algorithms to calculate a Klout score, and further using six classification learning algorithms. It was concluded that bots with higher number of followers and lengthier posts were responded to. Also, RF100, the ensemble learner outperformed other learning algorithms in accurately identifying bots.

Boshmaf [9] et al proposed a method to test the activity of bots on Online Social Networks (OSN) by setting up a web based botnet called Social bot Network on Facebook. They had managed to infiltrate several user accounts, with an 80% success rate. They further found that this could breach a user's privacy beyond the public domain. It was concluded that the in practice, several OSN security practices were ineffective in detecting or stopping social bots.

3. Overview of the Proposed System

3.1. Introduction and Related Concepts

In recent times, users have started spending a significant amount of their time on social networking platforms (Twitter, Facebook, Orkut), sharing a plethora of personal information and opinions on different aspects of life. Internet users

around the world make use of this powerful tool of communication a.k.a microblogging, using Twitter, a web application that provides dual benefits of microblogging and social networking. In this project, we use the open structure of twitter to perform 2 main tasks: (1) analyze data for sentiment analysis and opinion mining of a particular product and classify this data into positive, neutral and negative reviews (2) use this data to create automated programs, known as bots, which are useful for generating benign tweets. We use this double edged sword to provide companies with the opportunity to make use of these automated bots to generate the right tags to advertise products. This enhanced marketing opportunity provides Quality of Service in terms of cost, time and energy.

3.1.1. Architecture for the Proposed System

System Design:

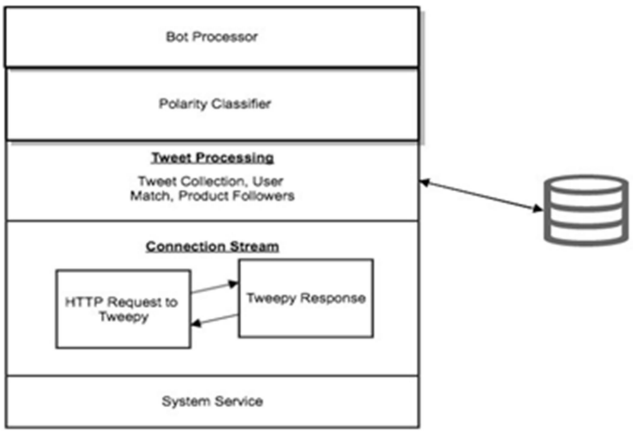


Figure 1. System Design

3.2.1. System Service

The system service component invokes all other components and provides a command line console for twitter to process its search for tweets, based on the search command.

3.2.2. Connection Stream

Once the HTTP server pings the Tweepy API with the search option, Tweepy induces the generation of the Tweet processing stream. This HTTP connection is open till a single request is served and is open every time a request is made.

3.2.3. Tweet Processing

The Tweet Processing component is generate during the middle of the Connection Stream process. The collection of new published tweets containing the search term is gathered. It also gathers tweet status id, tweet author id, and tweet author name. Information collected is stored in the database. This component also gathers the product tweeted about's followers.

3.2.4. Polarity Classifier

The polarity classifier is used to analyse tweets to determine whether a tag is positive, negative or neutral. The results are then written to the database to be used in the next stage. While neutral and negative tweets are ignored, the positive tweets are taken to be used for marketing purposes.

3.2.5. Bot Processor

The bot processor is responsible for analysis positive reviews to gather the user_id of the person who tweeted, and tag him in an automated tweet that a company wishes to publish. Also, in order to make sure that the messages do not spam user accounts by continuously tagging them, a counter is set to tag them in optimal number of posts, in order to serve a positive purpose.

3.3. Proposed System Model: UML Diagram

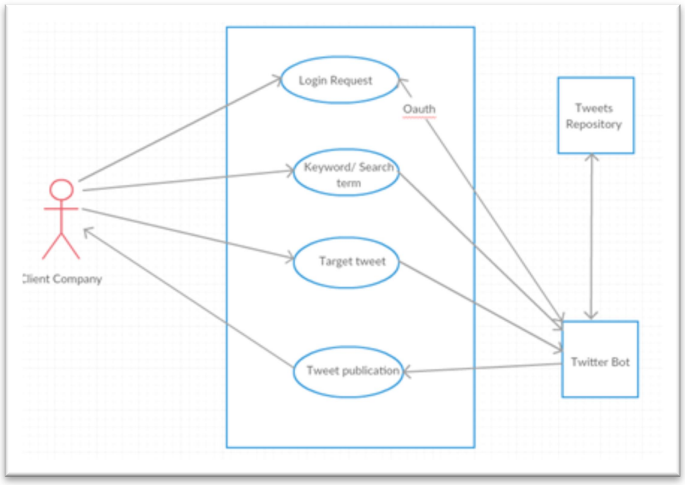


Figure 2. UML Diagram

4. Proposed System Analysis and Design

4.1. Introduction

4.2. Requirement Analysis

4.2.1. Functional Requirements

4.2.1.1. Product Perspective:

The product is a web application that allows users to customize their bots to tweet as per their target audience.

4.2.1.2. Product features:

Every client company may advertise their product on the basis of their requirement. The Tweet bot enables choice in audience based on their tweet polarity, and provide efficient publicity.

4.2.1.3. User characteristics:

There are two types of users: the public relations officer of a product based company and the administrator. The PRO will be enabled with options to search for tweets based on his product, and further specify his message to be tweeted by the bot. All of this will be provided in the form of a web based application. The administrator will be enabled with refining searches, a repository of related tweets along with their polarity analysis.

The bot can be customized further according to the company's directions.

4.2.1.4. Assumption & Dependencies:

The assumptions for this product is that there will be continuous internet connectivity and the Twitter server is working continuously. The bot may come across tweets that are redundant or unrelated, so, we assume that the tweets generated are not sarcastic in nature.

4.2.1.5. User Requirements:

The user must have an account in Twitter. The user must also have prior knowledge of the product to provide effective search terms for refined search.

4.2.2. Non Functional Requirements

4.2.2.1. Product Requirements

4.2.2.1.1. Efficiency (in terms of Time and Space):

Time is dependent on the speed of internet and the relevancy of tweets available for the given key term. Space requirements will be taken care of through MongoDB, again on the basis of search results.

4.2.2.1.2. Reliability:

The bot will be providing enough scope for advertising, however, there is an exception of sarcasm in the tweets, which will be handled with future development.

4.2.2.1.3. Compatibility:

It is compatible on web browsers, as it is a web application. Hence, portable with all web enabled systems.

4.2.2.1.4. Usability:

The system shall provide a uniform look and feel between all the web pages. The system shall provide a digital image for each product in the product catalog. The system shall provide use of icons and toolbars. The system shall provide handicap access. The system shall provide multi language support.

4.2.3. Implementation Requirements:

A working web browser is required, along with consistent internet connectivity.

4.2.4. System Requirements

4.2.4.1. User Interface:

The proposed system is a web application. The user interface will be limited to the types of controls that can be generated using HTML, Javascript, and Cascading Style Sheets. The user interface code will be generated by individual developers, as well as by Anaconda for Python.

4.2.4.2. Hardware Interfaces

The Hardware Interfaces of the system are handled by the Mac OS and the Windows 10 Operating System. No hardware dependent code will be written by the team in Phase 1 of the Customisable Twitter Bots system.

4.2.4.3. Software Interfaces

Operating System

The software is being designed to run on Windows 10 and Mac OS. This is specifically in order to have compatible internet browsers.

Web Server

The software is being designed to run on Internet Web browsers.

Database

The software will access MongoDB for storing data collected from the analysis phase.

Libraries / API

The software project utilises the Tweepy module from the list of Twitter API's. The application will be created using the Django Framework.

4.2.5. Engineering Standard Requirements

- Economic

  The application that is being built is primarily made to satiate economic thresholds in the world of marketing and business. The Twitter bots become a necessity in handling time, cost and labour, in order to obtain low cost, low time and low labour consumption.

- Environmental

  The bots do not have any impact on the environment, however, its impact can indirectly affect traditional paper consumption and wastage. Thus, becoming a benefit environmentally.

- Social

  The customizable bots work on the information provided and gained by social media. The bots are shaped to build according to the current social trends and interests of the social media users. Depending on the product, the bot may appeal socially to its target audience.

- Political

  The bot system may face a political majority using the same to advertise their campaigns. This way, it may impact the audience and their political view, according to the way the bots are used.

- Ethical

  The Twitter bots use only information that is available publicly. This allows ethical usage of user information.

- Health and Safety

  As Twitter has been used in several forms, awareness campaigns and health related matter can be publicized, making it an important asset to the field of health and safety.

- Sustainability

  The world is currently revolving around global marketing and social media. The idea of Twitter bots will be sustained as long as the internet and social media prevail.

- Legality

The Twitter bots use only information that is available publicly. This allows legal usage of user information.

5. Implementation

The scenario considers an example use case where the automated twitter bot software being created is being used by a new company to market its product. The product is being searched for at the system service stage. This is analogous to a user requesting for a resource from Twitter. In order for us to extract data from the web application, we are using a Twitter API called Tweepy which sets up a HTTP request to the twitter servers and the request is served in the forms of tweets relating to the type of the product. Along with the tweets, the user id and specific parts of the tweet are stored in the database for processing at a later stage. The tweet is then analysed at the next stage also known as sentiment analysis and opinion mining using a classifier algorithm to predict whether the tweets are neutral, positive or negative. The users who gave out positive reviews will then be tagged in optimal number of posts that the new company marketing a similar product makes. The negative reviewers will be tagged in new product posts that the company has to make. This is the stage when the company has to take a chance and find out whether they can acquire a new customer. In order to save the companies time, cost and man power, twitter bots will be used to automate the process of tagging and putting up these posts on behalf of the company. The flow chart given below explains the process described above.

Figure 3. Flow Chart

5.1. Data Collection

For a company trying to promote its product via twitter, it can optimize its marketing campaigning only if it tries to reach out to twitter users who may be potential buyers. For this purpose, we try to gather twitter data whose search terms are similar to the product being sold. For instance, if a company is selling shoes, we try to capture tweets about users review about a particular shoe. However, when a person is tweeting about a shoe, he may not always use the word shoe. Hence, we tried to use words that are synonymous to shoes like [boots, footwear] etc as shown in Table 1.

Word	Image
Shoe	· 7h So I got these **shoes** from @ and they're great, with awesome customer service 🙌🙌🙌... Contact her for more 👌
Footwear	™ @ · 18h Fresh hot from oven. #Seunclan **footwear** with elegant footwears that suits all dress sense. @

Table 1. Using synonymous terms to optimize collection of data.

While twitter provides several API's for its developers to provide access to its twitter servers, we used the tweepy API written in Python, due to the support

and the features it provides. We also preferred to use the streaming API as compared to the REST API to access live stream data by establishing a continuous HTTP connection. Twitter's API ensures the security of user accounts by providing an OAuth authentication model which supplies a unique set of tokens and secrets to access a users account through an application. This ensures that only the user that possesses these tokens can post or access the twitter account.

For this project, we ran the streaming API and collected around 2000 tweets from users. In order to store these tweets, we used MongoDB, which in turn neatly stores this data in the form of a JSON object. This allows for easy parsing of the components of twitter data such as text, hashtags, user_id, screen_name etc. The pseudo code for collection of data is shown below in Figure 4.

Algorithm 1 Data Collection

```
procedure COLLECTDATA
    MONGO_HOST ← path to which database you want to collect data
    search_words ← keywords for tweet search
    open streaming connection to twitter.
try:
    consumer_key ← consumer key to twitter
    consumer_secret ← consumer secret to twitter
    access_token ← access token to twitter
    access_secret ← access secret to twitter
    auth ← tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)
    client ← create cursor to mongoDB
    db ← client.database
    listener ← set up the listener for streaming and set rate limit
    streamer ← tweepy.stream(auth, listener
    streamer.filter(search_words)
    json_data ← collect data as a JSON object and store the same
except:
    if on_error(status_code) then
        print error connecting to the database
```

Figure 4. Pseudo Code for collecting Twitter data

5.2. Data Processing

Upon obtaining a stream of tweets in the form of a JSON file, we follow a few steps to process the text. This involves extraction of the text from the file, preprocessing, tokenization and finally analysis.

5.2.1. Extraction

Text from the JSON object is extracted into Python by using simple list methods, like list.append and list.insert. This ensures that only the textual matter from the JSON object is removed and available for processing. This is also done by reading the JSON file into a Python directory, followed by pretty printing to make the text readable.

5.2.2. Preprocessing and Tokenization

The language used in Twitter does not always conform to the rules and regulations of the English language. In order to pre process the collected tweets, we use regular expressions to separate the hashtags, emoticons, tags, hyperlinks and normal text. The text is then tokenised-separated into individual words, after removing stop words like prepositions and articles. This is called Natural Language Processing or NLP. This narrows down the data that needs to be analyzed and makes analysis more efficient.

5.2.3. Analysis

The tokenised data is now ready to be analysed. Analysis of text has been done using the textblob API. Textblob has a very user friendly and simple mechanism for analysing sentiments, using the bag of words method. The bag of words method consists of three steps-tokenisation, lexicon comparing and polarity calculation. As our data has been tokenised, the stop words and punctuation have been removed for easy and efficient analysis. Lexicon analysis, the next step, basically utilizes a pre existing lexicon (or bag) of words to compare every word in a given tweet. Assuming the tweet contains the word “awesome”, the lexicon helps to classify this as a positive word, and vice versa for a negative term. The step that follows this is most important. This involves checking every word in the tweet, and calculating its positivity or negativity by means of scoring. Assume that a given tweet consists of the words “awesome”, “amazing” and “uncomfortable”. The two positive words (awesome and amazing) are given a positive score, and the word uncomfortable is given a negative

score. The scores of the entire tweet is then calculated by averaging the individual scores. This is repeated for the entire repository of tweets.

Sample Tweet: Running with XYZ shoes is awesome and amazing but gets uncomfortable after some time #weird :P
Tweet after preprocessing: Running with XYZ shoes is awesome and amazing but gets uncomfortable after some time
Hashtags: weird
Emoticon: :P
Tokenised tweet: Running, shoes, amazing, awesome, uncomfortable
Scoring the tweet: running: 0, shoes: 0, amazing: 1, awesome: 1, uncomfortable: -1

$$Tweet\ score = \frac{\sum Token\ scores}{Number\ of\ tokens}$$

Tweet score for given tweet= 0.25

Tokenised tweet: Beach, amazing, awesome, boring
Scoring the tweet: Beach: 0, amazing: 1, awesome: 1, boring: -1

The bag of words method otherwise follows the Naïve Bayes Classification algorithm. This means that it uses the common probabilistic method to classify data as positive and negative. The simple formula that was mentioned above can be deduced directly from the one below, that uses Laplacian smoothing. P is the polarity, P(C) is the probability of d occurring in class C, and count (d,C) is the number of times d has occurred in class C. The decision tree of how this has been carried out has been given in Figure 5.

$$P = \frac{P(C)}{Number\ of\ words\ in\ class\ C} \cdot \prod_{i}^{n} count(d_i, C)$$

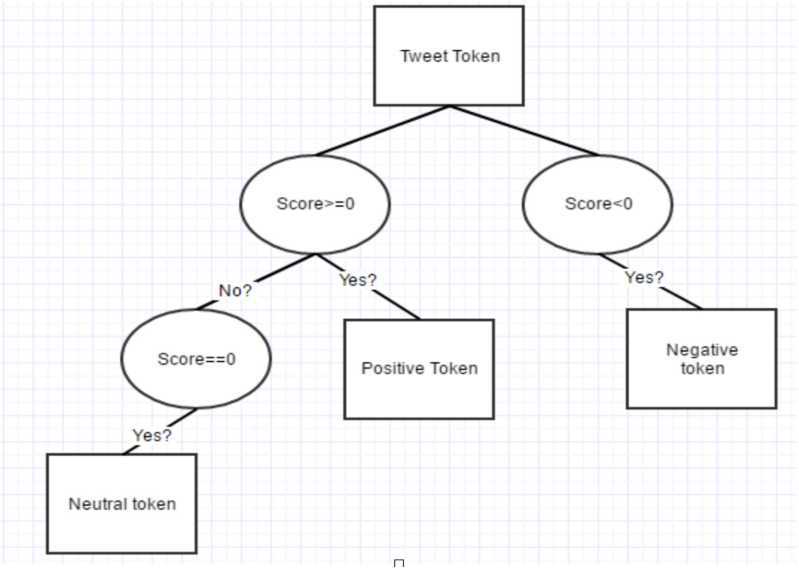


Figure 5. Categorization of Tweets

Application of this formula to the entire dataset can provide with a polarity for every tweet. This ensures that our data is successfully classified. The pseudo code for how we've carried out data processing is given in Figure 6.

Algorithm 1 Data Processing

```
procedure PROCESSDATA
    screen_names ← list to capture positive tweeters
    consumer_key ← consumer key to twitter
    consumer_secret ← consumer secret to twitter
    access_token ← access token to twitter
    access_secret ← access secret to twitter
    auth ← tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)
    client ← create cursor to mongoDB
    db ← client.database
    tweets_iterator ← object to iterate through the records in db collection
loop:
    if tweet_iterator(i)! = null then
        text ← tweet_iterator[i]['text'].
        text_data ← TextBlob(text).
        if text_data.sentiment.polarity is greater than 0 then
            add the twitter user name to screen_names
        i ← i + 1.
        goto loop.
        close;
```

Figure 6. Pseudocode for Processing Twitter data

5.3. Bot Processing

Once the screen names for the positive tweeters have been obtained, we use twitter bots (which were traditionally used for spamming) to our advantage. These twitter bots tag screen names to promote a product of the company by sending each user a tweet with an image about the product. In order to keep to the tweet limit, these bots send tweets after a random time interval to each of the twitter users from the screen name data set as shown in Table 2 in the bot processing output phase. In doing this, we make sure that the users remember the product name and eventually as their sent posts, they might tend to buy the product. In doing so, we make sure that the users are not continuously spammed with messages about the product. This concept of marketing is called as *positioning*. In such a scenario, the user tends to form a perception about a product. The pseudo code for the creation of a bot is shown in Figure 7.

Algorithm 1 Bot Processing

```
procedure CREATEBOT
    consumer_key ← consumer key to twitter
    consumer_secret ← consumer secret to twitter
    access_token ← access token to twitter
    access_secret ← access secret to twitter
    auth ← tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)
    screen_names ← list of positive tweeters
loop:
    if screen_names[i] != null then
        status ← tag screen_names[i] in status
        fn ← path in local host to image
        api.update_with_media(fn, status)
        sleep(random amount of time)
    i ← i + 1.
    goto loop.
    close;
```

Figure 7. Pseudocode for Bot Processing

6. Results and Discussion

6.1. Sample Test Cases

Test ID	Test Objectives	Preconditions	Steps	Test Data	Expected Results	PostConditions	Status
1	Collection of Product Data	Hosting of the Web App	Click on the required button	Search Terms	Values recorded in the desired file line by line	Values Stored in a text file	Pass
2	Collection of data	Keeping MongoDBrunning in the background	Run Python Script with valid twitter credentials	JSON Object with tweet data	Data stored in the desired database in the form of a collection	The recording is done	Pass
3	Valid Processing of Data	Keeping MongoDBrunning in the background	Run Python Script with TextBlob bag of words API downloaded on to the system	Sentiment with Polarity Values	Valid Usernames stored in a text file	The recording is done and analysis is complete	Pass
4	Bot posting tweet successfully	Valid URL to the product marketing poster	Read usernames from file and run python script	Twitter Post to username	Tweet	The tweet is posted	Pass

5.2. Expected Results

We present the code along with the outputs of each phases. As discussed, the first phase involves an application created for the company where they enter relevant product terms. The output for the data collection phase is stored in the form of a

JSON object which makes extracting certain parts of the tweet easier in later stages. In the data processing phase, we classify the tweets as positive, negative, and neutral and store the screen names of those people whose tweets seem positive. In the final phase, a tweet is sent out by our bot giving details about the companies products. The entire program was formulated in phases consisting of Data Processing, Data Collection and Twitter Bot Processing. For each of these phases, coding was done using Python and the Tweepy API provided by twitter. We make use of MongoDB for the storing of data. The web application was coded in HTML, CSS and PHP and later linked to the Python Backend. The tables shown below showcase the code along with the output at each stage.

```
from __future__ import print_function
import tweepy
import json
from pymongo import MongoClient

MONGO_HOST= 'mongodb://localhost/twitterdb'

f = open('mydata.txt')
WORDS = f.readlines()
for i, a in enumerate(WORDS):
    WORDS[i] = WORDS[i].strip(' \n')
f.close()

class StreamListener(tweepy.StreamListener):
    #This is a class provided by tweepy to access the Twitter Streaming API.

    def on_connect(self):
        # Called initially to connect to the Streaming API
        print("You are now connected to the streaming API.")

    def on_error(self, status_code):
        # On error - if an error occurs, display the error / status code
        print('An Error has occured: ' + repr(status_code))
        return False

    def on_data(self, data):
        #This is the meat of the script...it connects to your mongoDB and stores the tweet
        try:
            client = MongoClient(MONGO_HOST)

            # Use twitterdb database. If it doesn't exist, it will be created.
            db = client.twitterdb

            # Decode the JSON from Twitter
            datajson = json.loads(data)

            #grab the 'created_at' data from the Tweet to use for display
            created_at = datajson['created_at']

            #print out a message to the screen that we have collected a tweet
            print("Tweet collected at " + str(created_at))

            #insert the data into the mongoDB into a collection called twitter_search
            #if twitter_search doesn't exist, it will be created.
            collection = 'twitter_search_'+WORDS[0]
            db.collection.insert_one(datajson)
        except Exception as e:
            print(e)


consumer_key = ''
consumer_secret = ''
access_token = ''
access_secret = ''

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

#Set up the listener. The 'wait_on_rate_limit=True' is needed to help with Twitter API rate limiting.
listener = StreamListener(api=tweepy.API(wait_on_rate_limit=True))
streamer = tweepy.Stream(auth=auth, listener=listener)
print("Tracking: " + str(WORDS))
streamer.filter(track=WORDS)
```

Figure 8. Data Collection

```
import tweepy
import json
from pymongo import MongoClient
from textblob import TextBlob

#variable used for accessing positive product tweeters
screen_names = []

consumer_key = ''
consumer_secret = ''
access_token = ''
access_secret = ''

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

MONGO_HOST= 'mongodb://localhost/twitterdb'

client = MongoClient(MONGO_HOST)

db = client.twitterdb

tweets_iterator = db.collection.find().limit(100)
for tweet in tweets_iterator:
  text=tweet['text']
  text_data=TextBlob(text)
  print text + "\n"
  print text_data.sentiment
  print "\n"

  #getting positive tweeters
  if text_data.sentiment.polarity > 0:
    screen_names.append("@" + tweet['user']['screen_name'])

print(screen_names)

myfile = open("screenNames.txt","w")
for name in screen_names:
    myfile.writelines(name+"\n")

myfile.close()
```

Figure 9. Data Processing

```
#!/usr/bin/env python
import tweepy, sys, time
import os
from random import randint
from keys import keys

#f = open('screenNames.txt')
#h = f.readlines()
#for i, a in enumerate(h):
    #h[i] = h[i].strip(' \n')
#f.close()
#print h

CONSUMER_KEY = keys['consumer_key']
CONSUMER_SECRET = keys['consumer_secret']
ACCESS_TOKEN = keys['access_token']
ACCESS_TOKEN_SECRET = keys['access_token_secret']

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
api = tweepy.API(auth)

h = [u'@ATaruni', u'@anushbaskaran17']

for i in h:
    i = i.rstrip()
    status = i + " " + "! Our New Products Launched at a reasonable price!"
    fn = os.path.abspath('/Users/anushbaskaran/Desktop/Project_Boots.png')
    api.update_with_media(fn, status=status)
    nap = randint(1, 60)
    time.sleep(nap)
```

Figure 10. Bot Processing

5.2.1 Initial Web Application and Shoes as sample product

In this case, we have found the initial number of tweets collected were 2200, from the keywords shoes, boots and footwear.

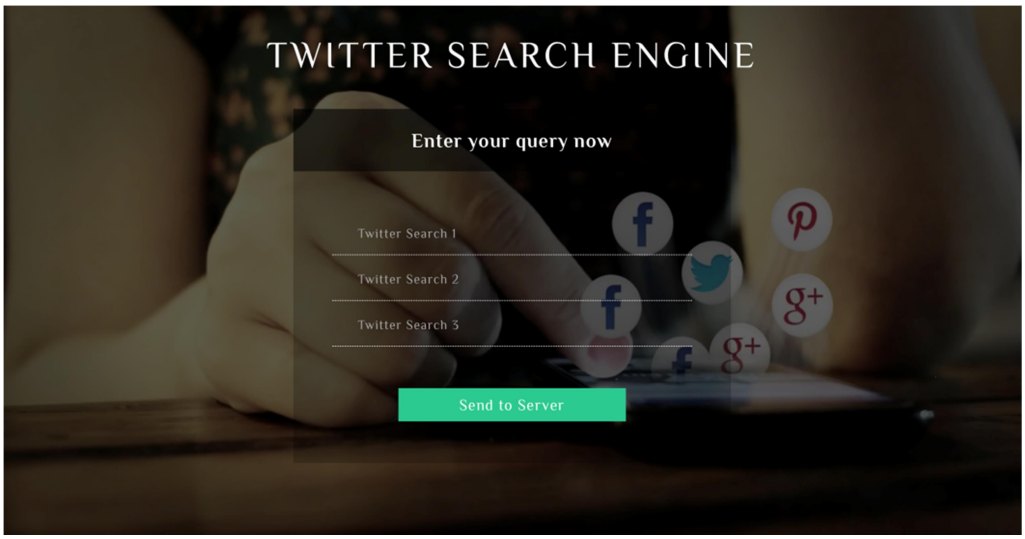


Figure 11. Initial Application

{"_id":ObjectId("58d4f8b4ff3c9d3e6f75c3ce"),"contributors":null,"truncated":false,"text":"#Good 2001 #Nike Air Jordan #Retro XI Concord 11 White Black Size 9.5 #Shoes #Share https://t.co/6RQytqcBcz https://t.co/EtoOu0c48t","is_quote_status":false,"in_reply_to_status_id":null,"id":NumberLong("845224956858318849"),"favorite_count":0,"source":"dlvr.it","retweeted":false,"coordinates":null,"timestamp_ms":"1490352307892","entities":{"user_mentions":[],"symbols":[],"hashtags":[{"indices":[0,5],"text":"Good"},{"indices":[11,16],"text":"Nike"},{"indices":[28,34],"text":"Retro"},{"indices":[70,76],"text":"Shoes"},{"indices":[77,83],"text":"Share"}],"urls":[{"url":"https://t.co/EtoOu0c48t","indices":[108,131],"expanded_url":"http://dlvr.it/NjSXdL","display_url":"dlvr.it/NjSXdL"}],"media":[{"source_user_id":414391672,"source_status_id_str":"845166093987569668","expanded_url":"https://twitter.com/Jimmie_Ramseyer/status/845166093987569668/photo/1","display_url":"pic.twitter.com/6RQytqcBcz","url":"https://t.co/6RQytqcBcz","media_url_https":"https://pbs.twimg.com/media/C7qiFNnVwAA56Y0.jpg","source_user_id_str":"414391672","source_status_id":NumberLong("845166093987569668"),"id_str":"845166090523099136","sizes":{"small":{"h":453,"resize":"fit","w":680},"large":{"h":1066,"resize":"fit","w":1600},"medium":{"h":800,"resize":"fit","w":1200},"thumb":{"h":150,"resize":"crop","w":150}},"indices":[84,107],"type":"photo","id":NumberLong("845166090523099136"),"media_url":"http://pbs.twimg.com/media/C7qiFNnVwAA56Y0.jpg"}]},"in_reply_to_screen_name":null,"id_str":"845224956858318849","retweet_count":0,"in_reply_to_user_id":null,"favorited":false,"user":{"follow_request_sent":null,"profile_use_background_image":true,"default_profile_image":false,"id":414391672,"verified":false,"profile_image_url_https":"https://pbs.twimg.com/profile_images/1839602324/66209717989_normal.jpg","profile_sidebar_fill_color":"DDFFCC","profile_text_color":"333333","followers_count":1489,"profile_sidebar_border_color":"BDDCAD","id_str":"414391672","profile_background_color":"9AE4E8","listed_count":688,"profile_background_image_url_https":"https://abs.twimg.com/images/themes/theme16/bg.gif","utc_offset":-28800,"statuses_count":565275,"description":"love anything that sparkles. #Celebs #Headline #CelebrityNews #Celeb #Fashion #Style #Entertainment #News #Topic #Story #Hot #Trend #Google #Yahoo","friends_count":11,"location":"San Francisco, CA","profile_link_color":"0084B4","profile_image_url":"http://pbs.twimg.com/profile_images/1839602324/66209717989_normal.jpg","following":null,"geo_enabled":false,"profile_background_image_url":"http://abs.twimg.com/images/themes/theme16/bg.gif","name":"Jimmie_Ramseyer","lang":"en","profile_background_tile":false,"favourites_count":0,"screen_name":"Jimmie_Ramseyer","notifications":null,"url":null,"created_at":"Thu Nov 17 00:31:02 +0000 2011","contributors_enabled":false,"time_zone":"Alaska","protected":false,"default_profile":false,"is_translator":false},"geo":null,"in_reply_to_user_id_str":null,"possibly_sensitive":false,"lang":"en","created_at":"Fri Mar 24 10:45:07 +0000 2017","filter_level":"low","in_reply_to_status_id_str":null,"place":null,"extended_entities":{"media":[{"source_user_id":414391672,"source_status_id_str":"845166093987569668","expanded_url":"https://twitter.com/Jimmie_Ramseyer/status/845166093987569668/photo/1","display_url":"pic.twitter.com/6RQytqcBcz","url":"https://t.co/6RQytqcBcz","media_url_https":"https://pbs.twimg.com/media/C7qiFNnVwAA56Y0.jpg","source_user_id_str":"414391672","source_status_id":NumberLong("845166093987569668"),"id_str":"845166090523099136","sizes":{"small":{"h":453,"resize":"fit","w":680},"large":{"h":1066,"resize":"fit","w":1600},"medium":{"h":800,"resize":"fit","w":1200},"thumb":{"h":150,"resize":"crop","w":150}},"indices":[84,107],"type":"photo","id":NumberLong("845166090523099136"),"media_url":"http://pbs.twimg.com/media/C7qiFNnVwAA56Y0.jpg"}]}}

Figure 12. Data Collection

```
#Good 2001 #Nike Air Jordan #Retro XI Concord 11 White Black Size 9.5 #Shoes #Sh
are https://t.co/6RQytqcBcz https://t.co/EtoOu0c48t

Sentiment(polarity=0.17777777777777778, subjectivity=0.3444444444444445)
```

Figure 13. Data Processing

Figure 14. Bot Processing

5.2.2. Pizzas as Sample Product

For the search term Pizza, we obtained 1500 search results, including opinions and posts by pizza vendors.

Figure 15. Data Collection

```
Pineapple is actually good on pizza why didn't I try it before

Sentiment(polarity=0.7, subjectivity=0.6000000000000001)
```

Figure 16. Data Processing

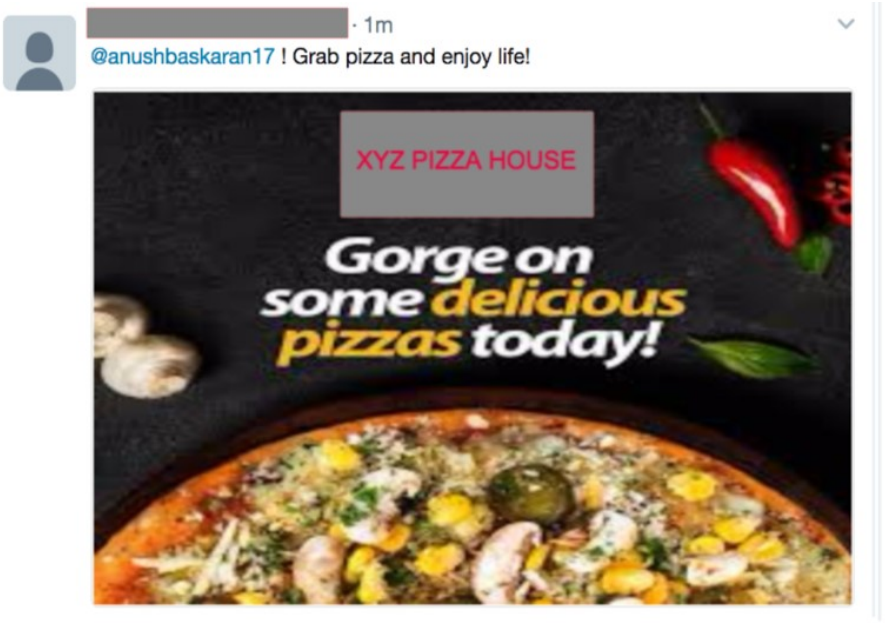


Figure 17. Bot Processing

5.2.3 Mobile Phones as a Sample Product

The search term “mobile phones” resulted in 500 tweets within the rate limits, and the processing has been provided below.

{ "_id" : ObjectId("58fd6932ff3c9d8f4f853832"), "contributors" : null, "truncated" : false, "text" : "Not saying cause I wanna delete my account I aint goin nowhere but its dumb how
you cant deactivate on mobile", "is_quote_status" : false, "in_reply_to_status_id" : null, "id" : NumberLong("856340860329500674"), "favorite_count" : 0, "source" : "<a href=\"http
://twitter.com/download/android\" rel=\"nofollow\">Twitter for Android", "retweeted" : false, "coordinates" : null, "timestamp_ms" : "1493002545803", "entities" : { "user_mentio
ns" : [ ], "symbols" : [ ], "hashtags" : [ ], "urls" : [ ] }, "in_reply_to_screen_name" : null, "id_str" : "856340860329500674", "retweet_count" : 0, "in_reply_to_user_id" : null, "
favorited" : false, "user" : { "follow_request_sent" : null, "profile_use_background_image" : true, "default_profile_image" : false, "id" : NumberLong("2598596234"), "verified" : fa
lse, "profile_image_url_https" : "https://pbs.twimg.com/profile_images/853251584339398657/YHkCZRyj_normal.jpg", "profile_sidebar_fill_color" : "EFEFEF", "profile_text_color" : "3333
33", "followers_count" : 2599, "profile_sidebar_border_color" : "EEEEEE", "id_str" : "2598596234", "profile_background_color" : "131516", "listed_count" : 30, "profile_background_im
age_url_https" : "https://pbs.twimg.com/profile_background_images/582962281878679552/3ymKSpJV.png", "utc_offset" : -18000, "statuses_count" : 65234, "description" : "WWF SmackDown!
Juuuust Bring it!", "friends_count" : 677, "location" : "A. ♥ ", "profile_link_color" : "009999", "profile_image_url" : "http://pbs.twimg.com/profile_images/853251584339398657/YHkCZ
Ryj_normal.jpg", "following" : null, "geo_enabled" : true, "profile_banner_url" : "https://pbs.twimg.com/profile_banners/2598596234/1492701120", "profile_background_image_url" : "ht
tp://pbs.twimg.com/profile_background_images/582962281878679552/3ymKSpJV.png", "name" : "Drew.", "lang" : "en", "profile_background_tile" : true, "favourites_count" : 12483, "screen
_name" : "REIGN5YARD", "notifications" : null, "url" : "http://xvideos.com", "created_at" : "Tue Jul 01 21:37:57 +0000 2014", "contributors_enabled" : false, "time_zone" : "Central
Time (US & Canada)", "protected" : false, "default_profile" : false, "is_translator" : false }, "geo" : null, "in_reply_to_user_id_str" : null, "lang" : "en", "created_at" : "Mon Ap
r 24 02:55:45 +0000 2017", "filter_level" : "low", "in_reply_to_status_id_str" : null, "place" : null }

Figure 18. Data Collection

Too much mobile addiction sometimes.

https://t.co/wixW5Wq7bQ

Figure 19. Data Processing

Figure 20. Bot Processing

5.3. Performance Measures and Accuracy

In this section, we present the different methods by which we can calculate the accuracy and measures of performance of out proposed system. The repository of words along with their sentiments are numerous. And on a similar note, we also can classify emoticons as positive and negative. A scatter plot of a sample tweet space of 20 have been made, the x axis representing the polarity and y axis representing the subjectivity of the tweet.

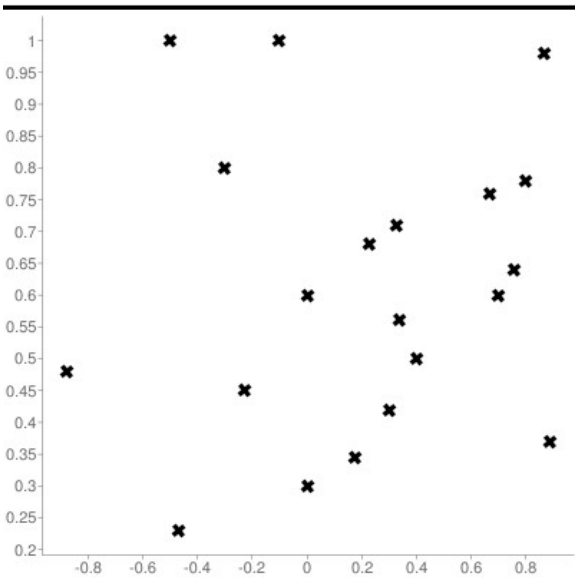


Figure 8. Scatter Plot of Subjectivity to Polarity

The tables 4 and 5 below describe the polarity calculation of the words used in the sample space above, along with the emoticons used in analysis.

Word	**Polarity**
Amazing	1
Awesome	1
Beautiful	1
Lovely	1
Hopeful	1
Wonderful	1
Splendid	1
Great	1
Good	1
Bad	-1
Poor	-1

Disgusting	-1
Awful	-1

Table 4. Words based Polarity

Emoticon	Polarity
:), :-)	1
: P, :-P	1
XD	1
XP	1
:*, :-*	1
:O	-1
:(, :-(	-1
:/,:\	-1

Table 5. Emoticon based Polarity

7. Future Works

Marketing is an ever growing field of economics. The QoS in terms of microblogging has great scope of improvement owing to its novelty in the field. The bots can be improved to analyse and process public images, such that privacy is not disrupted. Further, this can extended to other social media platforms, such as Facebook and Instagram. Also, the analysed sentimental data can be organized to provide meaningful feedback and reviews to the company. The bots can be utilised to improve efficacy in such terms, bringing advantages to the users and consumers alike. Measures can be taken to filter other bot tweets, and detect sarcasm in tweets. This will enable the system to become a powerful tool in the world of business.

8. Conclusion

It can be concluded that bots can be utilised for a positive outlook in terms of marketing. The bot can be tailored to work towards attracting a customer base who is genuinely interested in utilising the product after analysing their sentiment towards it. This has been found to reduce spam, owing to the selective nature of delivering its advertisements. Also, having sentiments of the users known, the user has ample scope to review, revise and upgrade their products. The efficiency in terms of time and

energy has also been optimized by automating the process using bots. This in turn helps in reducing delays, and cuts monetary costs too.

APPENDIX

Before the execution of the program can be done on your system, the following packages needs to be installed. The installation instructions are given for any UNIX operating system. Windows installation instructions can be obtained from the links provided. The commands following the '$' symbol are to be executed from the terminal.

1. Python 2.7 installation:
   (https://www.python.org/downloads/release/python-2713/)

2. MongoDB installation:
   $ brew update
   $ brew install mongodb
   $ mkdir -p /data/db
   (http://treehouse.github.io/installation-guides/)

3. PyMongo module:
   $ python -m pip install pymongo

4. TextBlob module installation:
   $ pip install -U textblob
   $ python -m textblob.download_corpora

5. Tweepy API installation:
   $ pip install tweepy

Once the following packages are installed, the following is to be run to access the application:

1. Host the web application using an open source host like MAMP and then navigate to the www folder and click on 'index.html'. Enter your search term.

2. Open a terminal and type in the following:

2.1. $ mongod //to be running till the end

2.2. $ mongo //to be running till the end

2.3. Phase 1 - Data Collection - $ python data_collection.py

2.4. Phase 2 - Data Processing - $ python data_processing.py

2.5. Phase 3 - Bot Processing:

Modify line 27 and 28 according to your marketing strategy.

$ python yolobot.py

NOTE: before you run Phase1, Phase2 and Phase3 enter your twitter credentials.

Data Collection File:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""


@author: anushbaskaran
"""

from __future__ import print_function
import tweepy
import json
from pymongo import MongoClient

MONGO_HOST= 'mongodb://localhost/twitterdb'
```

```
f = open('mydata.txt')
WORDS = f.readlines()
for i, a in enumerate(WORDS):
   WORDS[i] = WORDS[i].strip(' \n')
f.close()

class StreamListener(tweepy.StreamListener):
   #This is a class provided by tweepy to access the Twitter Streaming API.


   def on_connect(self):
      # Called initially to connect to the Streaming API
      print("You are now connected to the streaming API.")


   def on_error(self, status_code):
      # On error - if an error occurs, display the error / status code
      print('An Error has occured: ' + repr(status_code))
      return False


   def on_data(self, data):
      #This is the meat of the script...it connects to your mongoDB and stores the
tweet
      try:
         client = MongoClient(MONGO_HOST)


         # Use twitterdb database. If it doesn't exist, it will be created.
         db = client.twitterdb


         # Decode the JSON from Twitter
```

```
            datajson = json.loads(data)


            #grab the 'created_at' data from the Tweet to use for display
            created_at = datajson['created_at']


            #print out a message to the screen that we have collected a tweet
            print("Tweet collected at " + str(created_at))


            #insert the data into the mongoDB into a collection called twitter_search
            #if twitter_search doesn't exist, it will be created.
            collection = 'twitter_search_'+WORDS[0]
            db.collection.insert_one(datajson)
        except Exception as e:
           print(e)


consumer_key = ''
consumer_secret = ''
access_token = ''
access_secret = ''


auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)


#Set up the listener. The 'wait_on_rate_limit=True' is needed to help with Twitter API
rate limiting.
listener = StreamListener(api=tweepy.API(wait_on_rate_limit=True))
streamer = tweepy.Stream(auth=auth, listener=listener)
print("Tracking: " + str(WORDS))
```

```
streamer.filter(track=WORDS)
```

Data Processing Code:

```
import tweepy
import json
from pymongo import MongoClient
from textblob import TextBlob

#variable used for accessing positive product tweeters
screen_names = []

consumer_key = ''
consumer_secret = ''
access_token = ''
access_secret = ''

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

MONGO_HOST= 'mongodb://localhost/twitterdb'

client = MongoClient(MONGO_HOST)


db = client.twitterdb
```

```
tweets_iterator = db.collection.find().limit(100)
for tweet in tweets_iterator:
  text=tweet['text']
  text_data=TextBlob(text)
  print text + "\n"
  print text_data.sentiment
  print "\n"

  #getting positive tweeters
  if text_data.sentiment.polarity > 0:
        screen_names.append("@" + tweet['user']['screen_name'])


print(screen_names)


myfile = open("screenNames.txt","w")
for name in screen_names:
        myfile.writelines(name+"\n")


myfile.close()
```

Bot Processing Code:

```
#!/usr/bin/env python
import tweepy, sys, time
import os
from random import randint
from keys import keys


#f = open('screenNames.txt')
```

```
#h = f.readlines()
#for i, a in enumerate(h):
   #h[i] = h[i].strip(' \n')
#f.close()
#print h



CONSUMER_KEY = keys['consumer_key']
CONSUMER_SECRET = keys['consumer_secret']
ACCESS_TOKEN = keys['access_token']
ACCESS_TOKEN_SECRET = keys['access_token_secret']

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
api = tweepy.API(auth)

h = [u'@username1', u'@username2']


for i in h:
   i = i.rstrip()
   status = i + " " + "! Our New Products Launched at a reasonable price!"
   fn = os.path.abspath('Enter Image Location')
   api.update_with_media(fn, status=status)
   nap = randint(1, 60)
   time.sleep(nap)
```

PHP Connection:

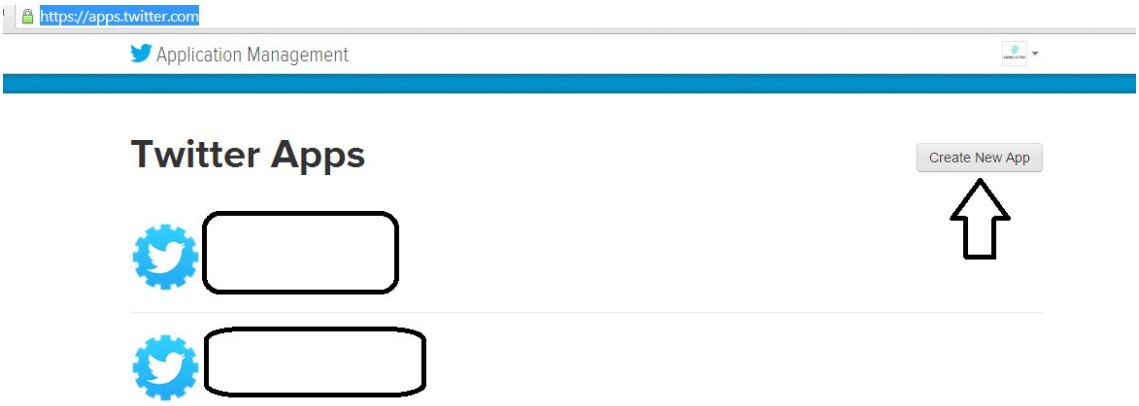
```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Final year Project</title>
<!-- Meta tag Keywords -->
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="application/x-javascript">
addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false);
function hideURLbar(){ window.scrollTo(0,1); } </script>
<!-- Meta tag Keywords -->
<!-- css files -->
<link rel="stylesheet" href="css/style.css" type="text/css" media="all" /> <!-- Style-CSS -->
<link rel="stylesheet" href="css/font-awesome.css"> <!-- Font-Awesome-Icons-CSS -->
<!-- //css files -->
<!-- web-fonts -->
<link
href="//fonts.googleapis.com/css?family=Philosopher:400,400i,700,700i&amp;subset=cyrillic,cyrillic-ext,vietnamese" rel="stylesheet">
<!-- //web-fonts -->
</head>
<body>
<div data-vide-bg="video/social2">
	<div class="center-container">
		<!--header-->
		<div class="header-w3l">
			<h1>Twitter Search Engine</h1>
		</div>
		<!--//header-->
		<!--main-->
		<div class="main-content-agile">
```

```
                    <div class="wthree-pro">
                        <h2>Enter your query now</h2>
                    </div>
                    <div class="sub-main-w3">

                    <!--
                    CHANGE action parameter to file name that accepts the parameter

                    Refer to this link for explanation
                    http://stackoverflow.com/questions/15965646/posting-html-form-values-to-python-script

                    -->
                        <form action="ChangeNameHere.extension" method="post">
                            <input placeholder="Twitter Search" name="mail" type="text" required="">

                            <input type="submit" value="Login">
                        </form>
                    </div>
                </div>
                <!--//main-->
                <!--footer-->
                <div class="footer">
                    <p></p>
                </div>
                <!--//footer-->
        </div>
</div>
<!-- js -->
<script type="text/javascript" src="js/jquery-2.1.4.min.js"></script>
<script src="js/jquery.vide.min.js"></script>
```

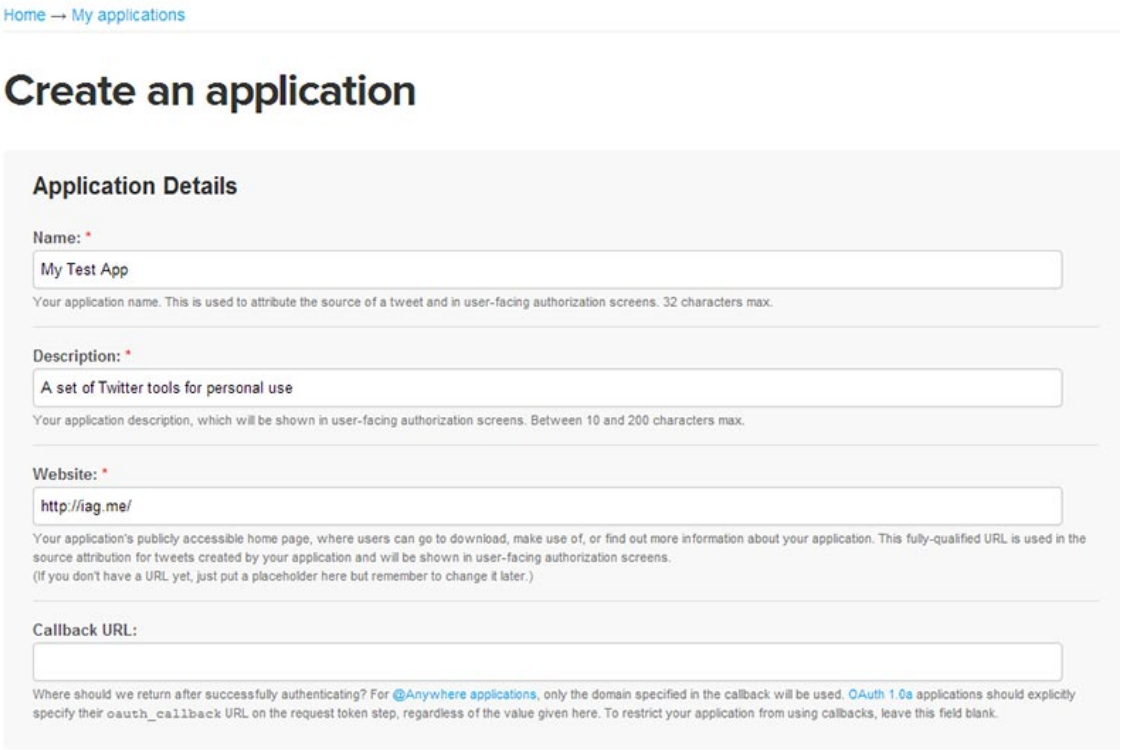
```
<!-- //js -->
</body>
</html>
```

Annexure – 1

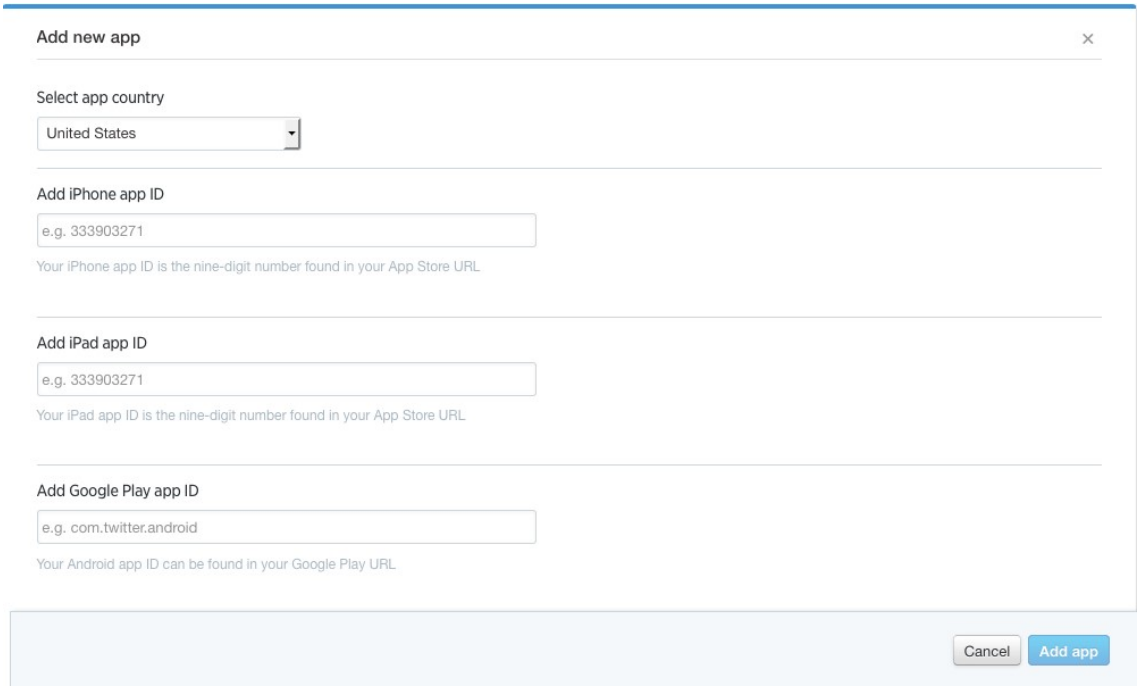
Steps to create your Twitter app to interact with the API:



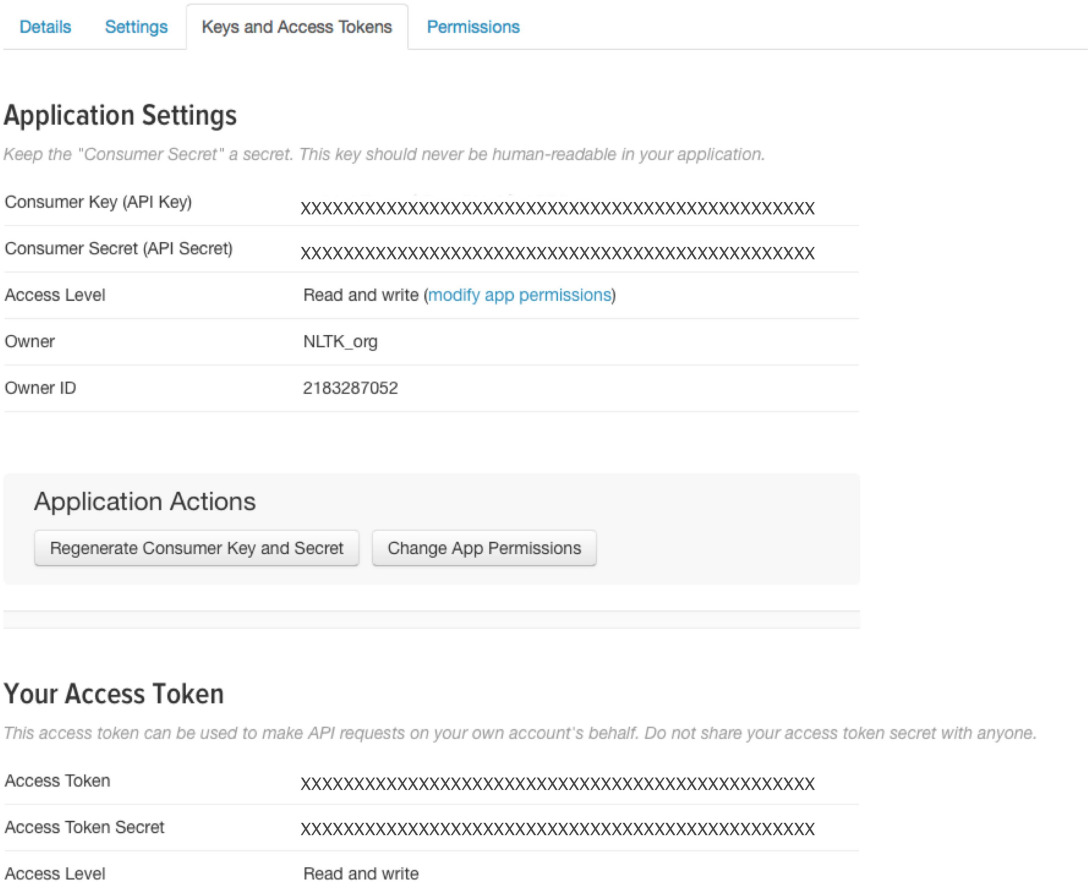
1. Select Create New App



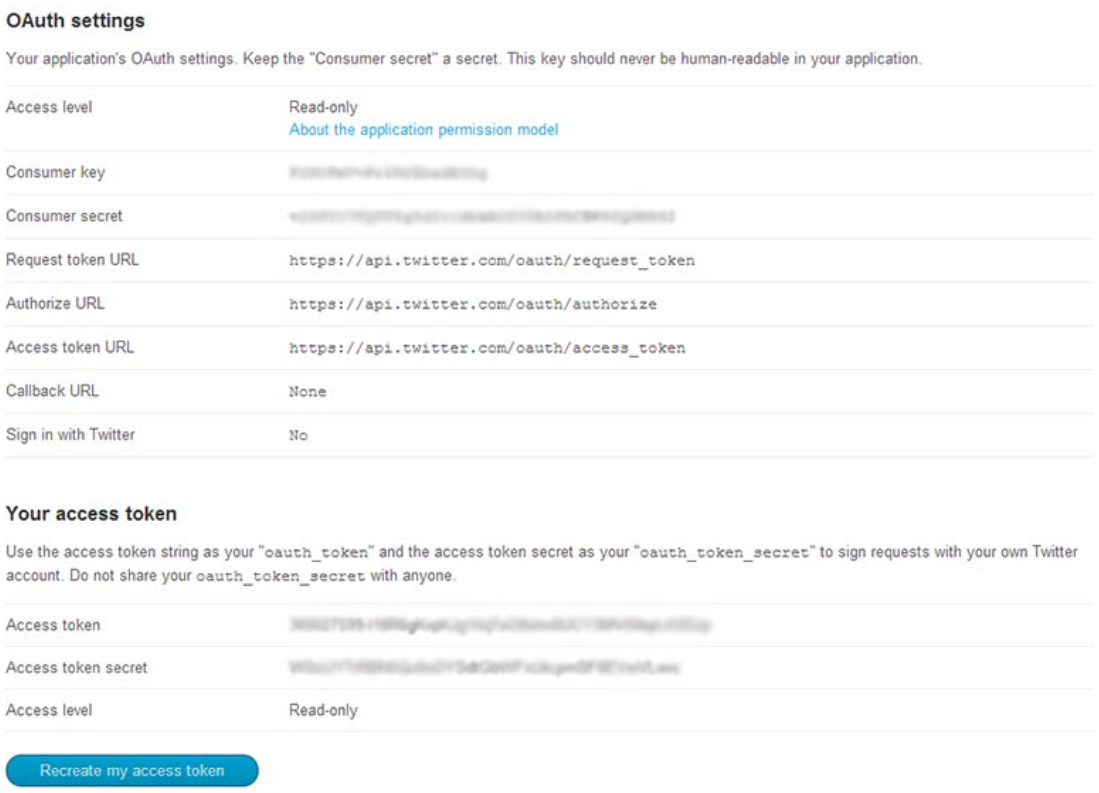
2. Fill all relevant details



3. Add any relevant app ID. Fill in your mobile number.



4. Generate tokens and secrets, allowing access to read and write via app.



5. Generate access tokens via the OAuth portal, and test.

About OAuth:

OAuth is an authentication protocol that allows users to approve application to act on their behalf without sharing their password. Access tokens are not explicitly expired. An access token will be invalid if a user explicitly rejects an application from their settings, or if Twitter suspends an application. If an application is suspended, there will be a note on the apps.twitter.com page stating that it has been suspended. You should plan that a user's access token may become invalid at any time and you will need to re-authorize for that user in the case that it does. Ensuring you handle this situation gracefully is imperative for a quality user experience. Many users trust an application to read their information but not necessarily change their name or post new statuses. Updating information via the Twitter API - be it name, location or adding a new status - requires and HTTP POST. We stuck with the same restriction when implementing this. Any API method that requires an HTTP POST is considered a write method and requires read & write access.

9. References

[1] Agarwal, Apoorv, et al. "Sentiment analysis of twitter data." Proceedings of the workshop on languages in social media. Association for Computational Linguistics, 2011.

[2] Pak, Alexander, and Patrick Paroubek. "Twitter as a Corpus for Sentiment Analysis and Opinion Mining." LREc. Vol. 10. No. 2010. 2010.

[3] Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. "Recognizing contextual polarity in phrase-level sentiment analysis." Proceedings of the conference on human language technology and empirical methods in natural language processing. Association for Computational Linguistics, 2005.

[4] Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." CS224N Project Report, Stanford 1.12 (2009).

[5] Chu, Zi, et al. "Who is tweeting on Twitter: human, bot, or cyborg?." Proceedings of the 26th annual computer security applications conference. ACM, 2010.

[6] Kouloumpis, Efthymios, Theresa Wilson, and Johanna D. Moore. "Twitter sentiment analysis: The good the bad and the omg!." Icwsm 11.538-541 (2011): 164.

[7] Haustein, Stefanie, et al. "Tweets as impact indicators: Examining the implications of automated “bot” accounts on Twitter." Journal of the Association for Information Science and Technology 67.1 (2016): 232-238.

[8] Wald, Randall, et al. "Predicting susceptibility to social bots on twitter." Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on. IEEE, 2013.

[9] Boshmaf, Yazan, et al. "The socialbot network: when bots socialize for fame and money." Proceedings of the 27th annual computer security applications conference. ACM, 2011.

[10] Azeem, Mohd Abdul. "Consumers' Att itudes toward Commercial E-mail Spam and Web pop-ups: Interference, Perceived Loss of Control, and Irritation." (2012).

[11] Bonzanini, Marco. Mastering Social Media Mining with Python. Packt Publishing Ltd, 2016.

[12] Lu, Rong, and Qing Yang. "Trend analysis of news topics on twitter." International Journal of Machine Learning and Computing 2.3 (2012): 327.

[13] Kumar, Akshi, and Teeja Mary Sebastian. "Sentiment analysis on twitter." IJCSI International Journal of Computer Science Issues 9.3 (2012): 372-378.