Twitter Sentiment Analysis

Project Report
CMPE 239: Web and Data Mining

By
Team: Clique

Anuradha Vakil [008016216]
Dhanvantari Tilak [010014147]
Gaurangi Bhangale [009303944]

Submitted To
Prof. Chandrasekar Vuppalapati

Date: 28 April, 2015

Table of Contents

List of Figures:

1. Project Description

Recently, there has been an enormous growth in the use of social networking websites such as Facebook, Twitter etc. Encouraged by that growth, media organizations and companies are progressively looking for ways to mine these social networking sites for information about what people think and feel about their services and products. Organizations such as Twitter (twitrratr.com), tweetfeel (www.tweetfeel.com), and Social Mention (www.socialmention.com) are some of those companies who make use of Twitter sentiment analysis as one of their products.

Micro-blogging sites have become a source of variety of information. We can say this because microblogs have become very popular now a days and a variety of people post real time comments about their opinions on a various topics, complains, current issues, and express what they feel for products they use in their life. Many times, companies developing such products poll these micro-blogging sites to get an idea of general sentiment for their product. They study user reactions and reply to users on such sites. One challenge is to build an application to sense and summarize an overall sentiment called sentiment analysis.

Sentiment analysis is a task that is becoming increasingly important for many companies because of emergence of social media sites. Twitter is one such a very popular micro-blogging website where users can create their status messages - Tweets. These tweets convey opinions about several topics. Our project helps in extracting positive or negative sentiments from the tweets. Our customers can use sentiment analysis to research products or current events and compare their results. Marketers can use our tool in researching public opinions of their products and company decisions, they will also be able to analyze if their customers are satisfied or not. This tool can also be useful for organizations to collect feedback about issues in new releases.

The main goal of our project is to classify the polarity of the given text and to check whether the said opinion is positive or negative. In order to do this, we first obtain twitter data from their API. Then we remove unnecessary information by cleansing the data. Our next step is to parse through the data and tokenize it in the required format. We then try to find the sentiments of each token by checking it against the dictionaries. The sentiment of the entire sentence is then computed called the ‘sentiment score’. We then visualize these sentiment scores for every tweet in graphical format so that it is easy to understand from the end user perspective.

Our current visualization shows the comparisons of results from currently popular topics such as checking if iWatch will be a success vs Microsoft Band. Or user can see if people are talking more about Walmart or Target? What's their favourite shopping spot? Or they can also check who has more chances to win in the presidential election next year. About whom people love to tweet more, Rand Paul or Hillary Clinton? For each comparison pair, tweets for both of the topics are collected separately and are cleansed for better results. Their sentiment score for each tweet is then calculated and presented in graphical format.

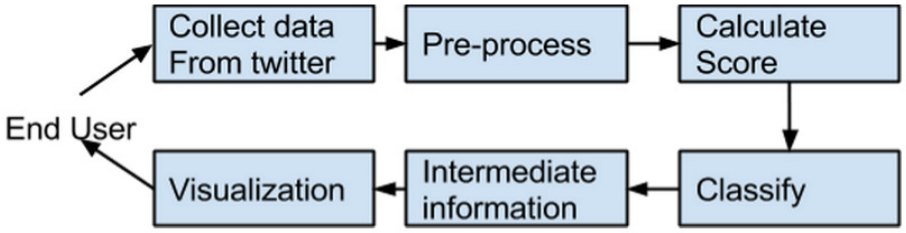


Figure 1: Process in Twitter Sentiment Analysis

2. Requirements

The proposed application is developed for the purpose of identifying sentiments from the tweets obtained from tweeter.

Requirements of the proposed system are described as follow:

1. User should be able to select the topic he wants to analyse
2. System should gather the tweets from Twitter API and apply preprocessing on them in order to remove unwanted data
3. System should calculate analysis result and store it internally
4. The result should be displayed to users in graphical format so that he can perform comparison of peoples' sentiments.

The requirements to get these tasks done are as follows,

1. Create a twitter Development account: It involves following steps,
   - Visit https://dev.twitter.com/user/login and log in with your credentials.**Sign up** if you dont have a twitter account.
   - Authorize the site to use your account by clicking **Authorize app**.

- Go to https://dev.twitter.com/apps (Twitter applications page) and click **Create a new application**.
- Follow instructions written on screen. We can write any text as desired in the fields, **Name**, its **Description**, and **Website**. Give the link of the website if you have one.
- In the details page, you can find the Key and Access Tokens tab. This is the place where you can get Twitter developer credentials. We can note down the **Consumer key** and **Consumer secret**.
- Also make a note of **Access token** and **Access token secret** which you can find at the bottom of the page.

2. Install Python: Our backend code works in Python. We have used Python 2.7 eclipse IDE.

3. Install NLTK and yaml: NLTK library is an essential platform for developing Python programs so that they can process natural languages. It has easy-to-use interfaces with a number of lexical resources and corpora for example WordNet, along with a list of libraries for tokenization, classification, and stemming, parsing, tagging, and semantic analysis.

3. Spring XD: Spring XD (eXtreme Data) joins Spring Boot and Grails as part of the execution portion of the Spring IO platform. It can be used to create a stream of twitter data

4. Install Google App Engine for Python: Google App Engine executes a Python code using a pre-loaded Python interpreter in a sandboxed environment. We hosted our website using App Engine deployment.

3. UI Design Principles

1. The structure principle. An ideal design should arrange the UI firmly, in meaningful ways based on consistent and clear models that are noticeable and recognizable to users. We tried to follow this principle by putting related things together and also by separating unrelated things.

2. The simplicity principle. The UI design should be simple. The tasks should be simple to do, conversing simply and clearly in users own language. We tried to implement this principle by providing good shortcuts that are related to longer procedures.

3. The visibility principle. Our UI should keep all required options for a given task visible but at the same time it should not distract end user with additional or redundant information. We have our made our basic and required functionalities clearly visible to the end users.

4. The feedback principle. Our UI design should let users be informed of interpretations or actions. We try to follow this principle by notifying users on change of state or condition, and if errors or exceptions are generated.

5. The tolerance principle. We tried to keep our UI flexible and tolerant, reducing the risk of misuse by allowing undoing. We try to avoid errors wherever possible by testing varied inputs and sequences.

6. The reuse principle. We also try to should reuse some internal and external components at the same time try to maintain consistency.

7. Learnability and readability: We try to keep our UI simple and elegant so that it is easy to learn

8. Focus: Interface and content is designed so that the main focus is not lost.

9. Content Sensitive: The aesthetics and quality of the content is designed by keeping user in mind.

3.1 **Wireframe**

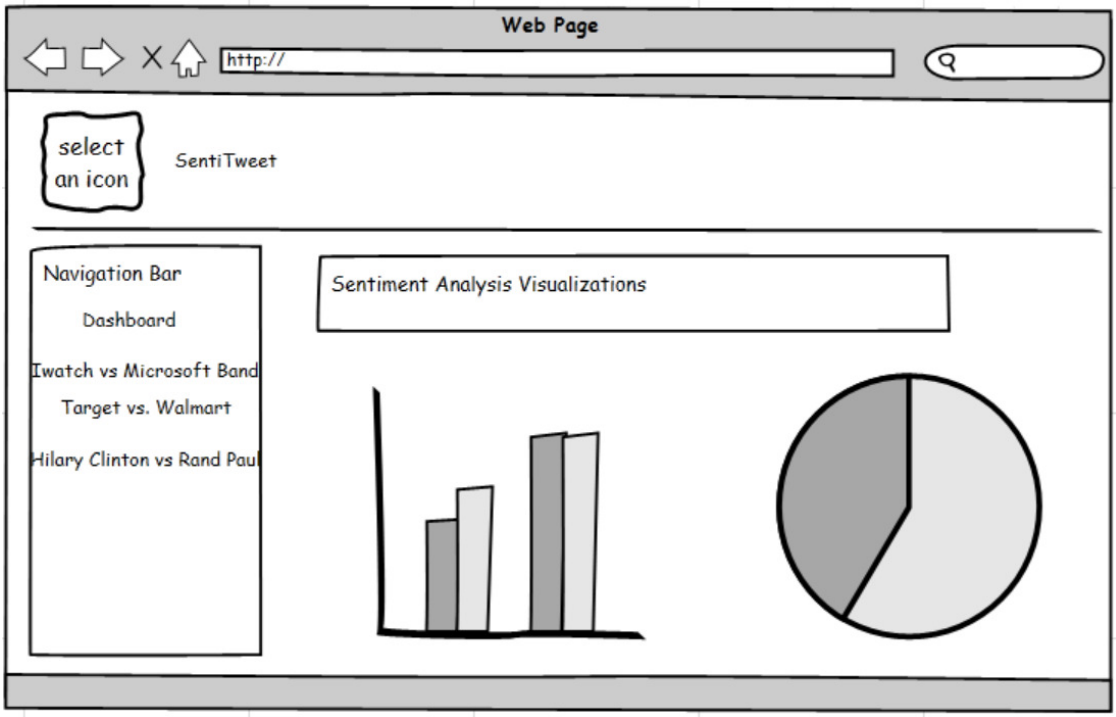


Figure 2: Wireframe-Dashboard

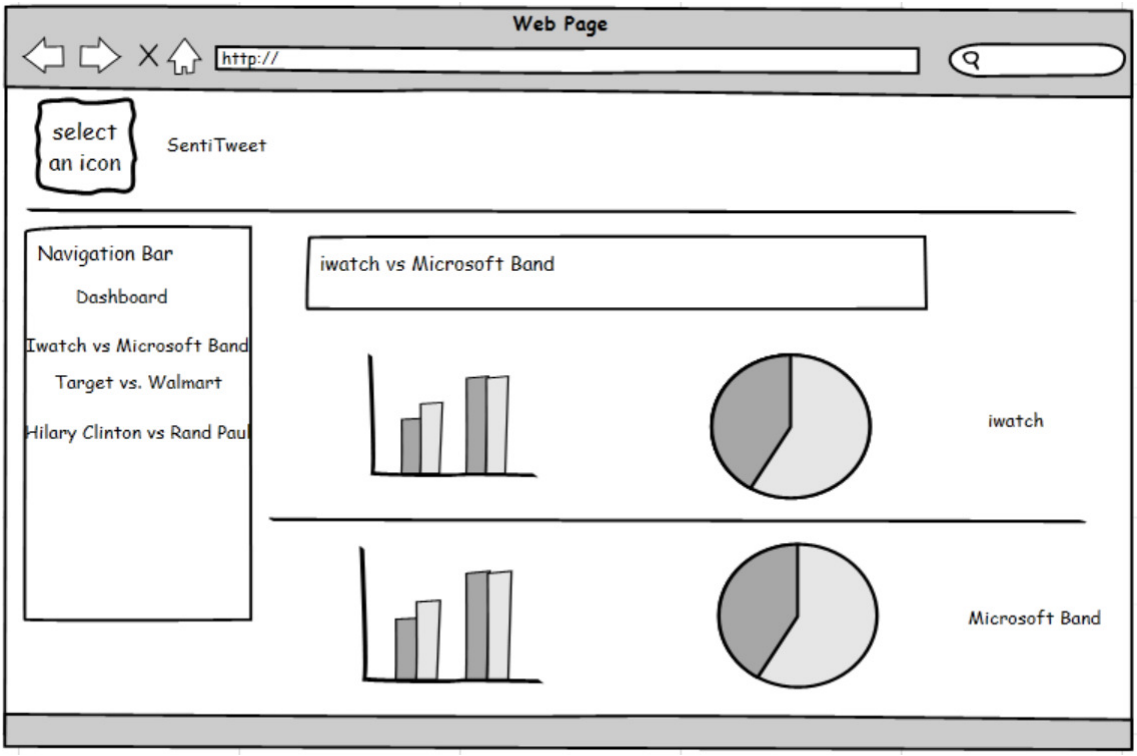


Figure 3: Wireframe- Comparison of Products

4. High Level Architecture Design

The high level architecture design describes various components used for sentiment analysis process. The Twitter data is fetched using Spring Xd Server, which provides fast access to the tweets data as it runs on a distributed environment. This fetched data is stored in HDFS which allows us to process the tweets at a faster speed. Once the data is cleaned in Hadoop, it gets stored in Apache Hive. This data can be queried and fed to the Sentiment Analysis engine build using python. Lexicon based approach is used to calculate the sentiments of the tweets and score is generated for each tweet. This generated score is then fed to the visualization framework which build using java script to create meaningful charts. This project is hosted on Google cloud Engine and can be accessed using http://sentitweets.appspot.com

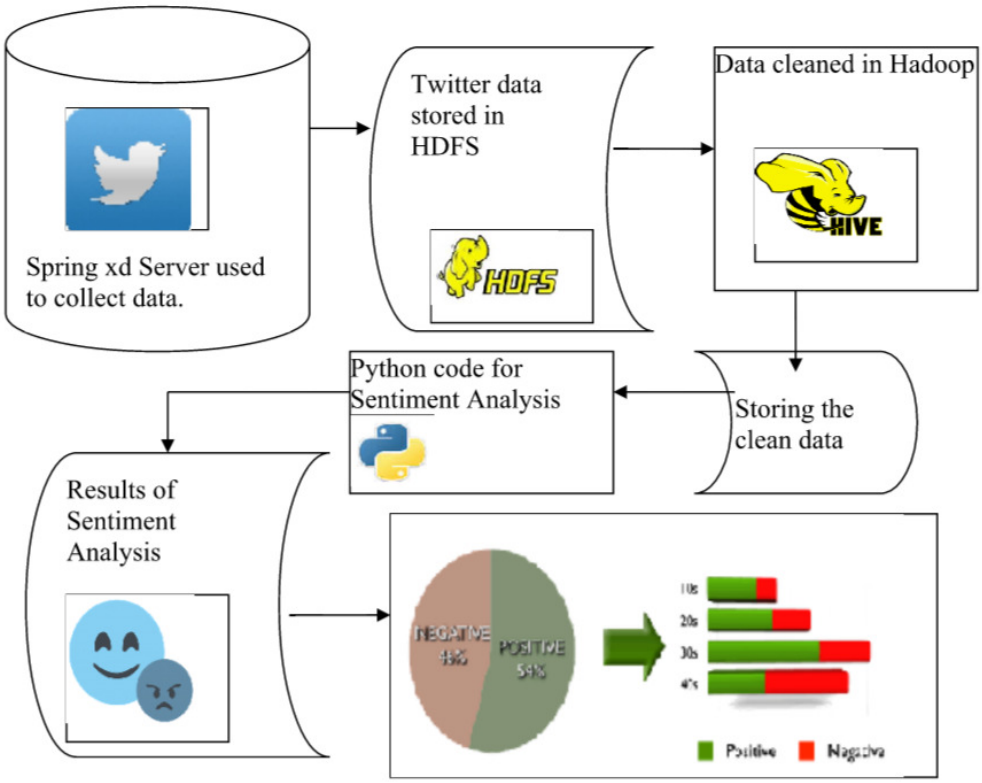


Figure 4: High-level Architecture

5. Datasets & Data patterns

All data is taken from twitter using Spring xd. The dataset is stored in HDFS and after the cleaning process is done it is stored in the hive tables.

We followed the following steps to get the twitter data:

- Downloaded and installed the Spring XD. Spring XD is a unified, distributed, and extensible system for data ingestion, real time analytics, batch processing, and data export.

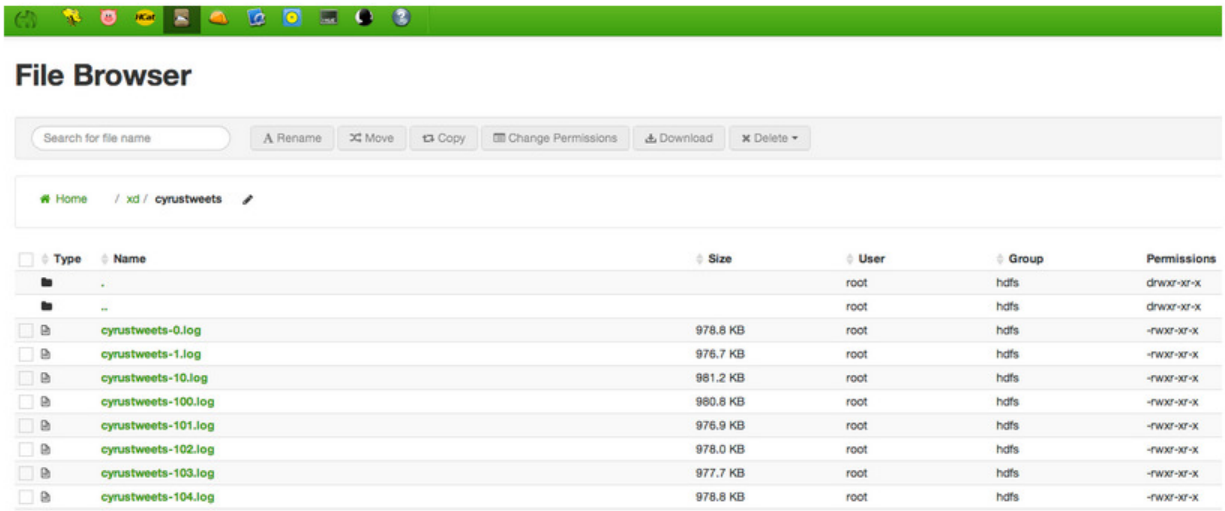
```
create stream --name ticktock --definition "Time | Log"
```

```
2013-10-12 17:18:09
2013-10-12 17:18:10
2013-10-12 17:18:11
2013-10-12 17:18:12
2013-10-12 17:18:13
2013-10-12 17:18:14
```

- Downloaded and install Horton works sandbox and configure the network setting for it so that the virtual sandbox can be connected to the Spring Xd.
- Configure the Spring XD to use Hadoop by editing the Hadoop.properties.
- Create the Tweet Stream in Spring XD and stream information from Twitter by using the twitter developer APIs.

```
stream create --name cyrustweets --definition "twitterstream --
track='mileycyrus, miley cyrus' | hdfs"
```

- Refine the data using Apache hive by importing the data from the HDFS to Hive. Once the data is stored in Hive , we can query the database for tweets and feed these tweets to the sentiment analysis engine.

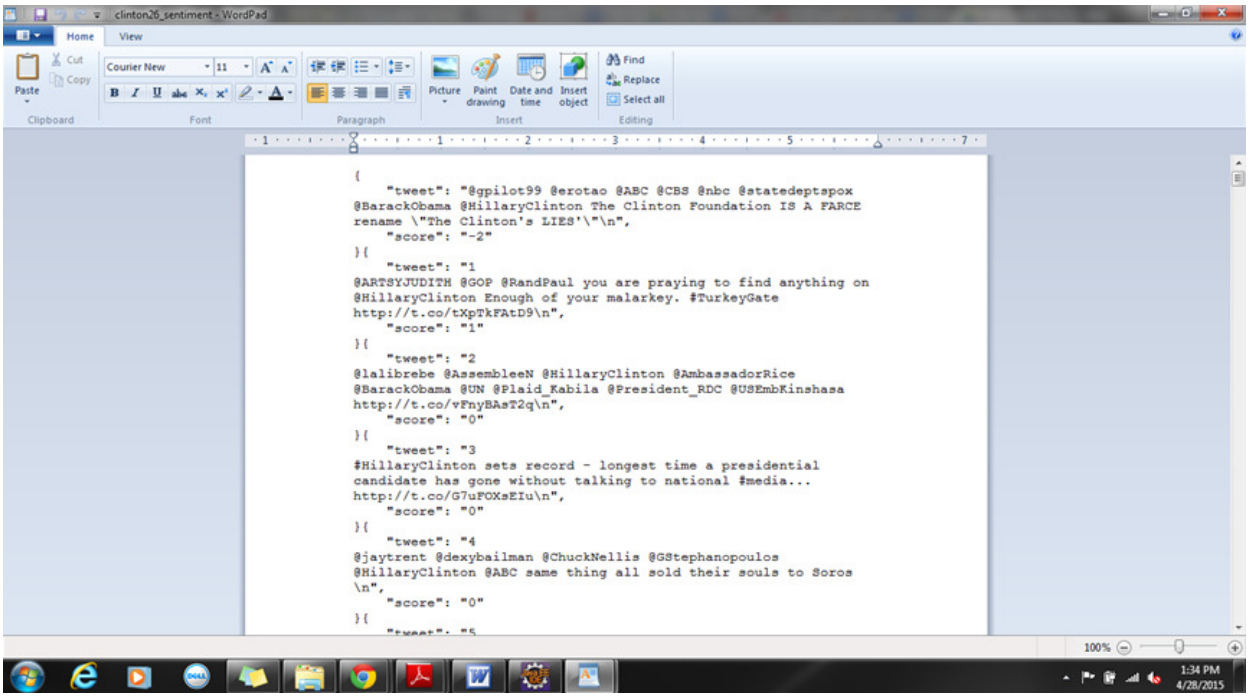


Since we have done sentiment analysis on aggregated data, every day at same time we used to take dump for the same.

Data Patterns:

Field	Type	Description
Text	String	The actual UTF-8 text of the status update
Contributors	Collection of contributors	Collection of users indicating the list of users who contributed to authorship of the tweet.
Created_at		UTC time when this time was created.
Number of Tweets	Integer	The total number of tweets captured in data dump
Sentiment score	Calculated -integer	The real number indicating the score. If above 0 it is positive, if below 0 it is negative and if 0 it is neutral.

Screenshot of data dump used for Sentiment analysis



6. Data Flow Diagrams & Architecture

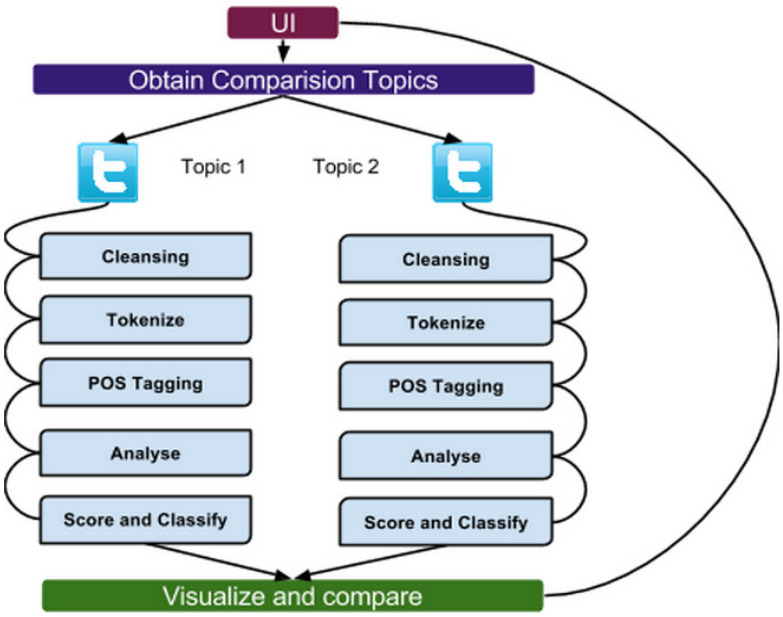


Figure 5: System Architecture

Stepwise Data flow information:

1. Collecting data From twitter API

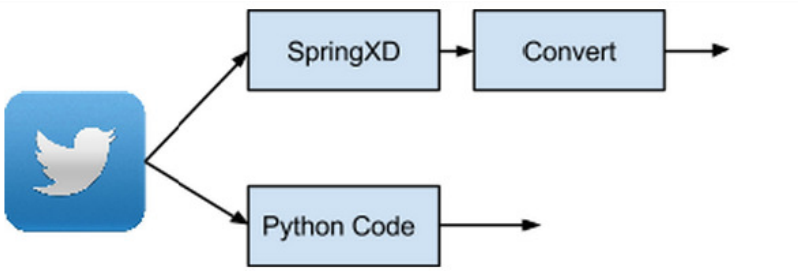


Figure 6: Data Flow for Collecting Tweets

In this phase, we either use SpringXD or our python script which uses Twitter API to get tweets for that day. The advantage of using SpringXD is that the result is very fast. But we had to convert our output in required format.
We also wrote a python script which can access twitter API and get recent (In 24 hour window) tweets.

2. Preprocessing data

Figure 7:Data flow for Preprocessing data

I this phase, we convert the raw data in the form appropriate for us. Conversion includes removing unnecessary data and creating a JSON file format so that the key value pair can be easily accessed.

3. Analysing and classifying data

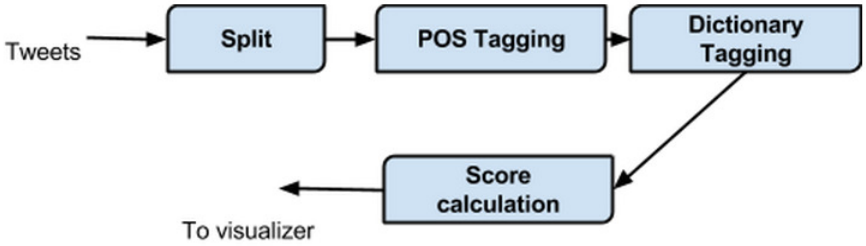


Figure 8:Data flow for analysis of tweets

This is the crucial step of our software. We use our preprocessed tweets and split and tokenize them. We then also apply POS tagging and dictionary tagging in order to classify the given text. A dictionary a list of words that possess a category. For example, one can have a dictionary for positive expressions, and another one for stop words.

4. Visualization

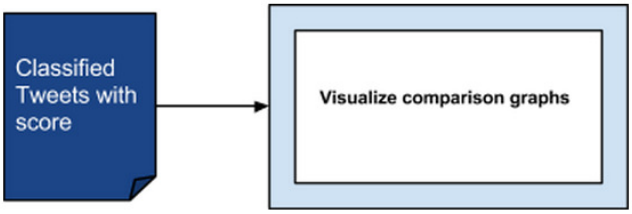


Figure 9: Data flow for visualizing results

In the end, we visualize the data using our website which is hosted on Google app Engine Cloud platform.

7. Data Mining Principles & Algorithms

- We have used Lexicon based dictionary approach for identifying positive and negative words.
- Since the code is written in Python we have used nltk library.
- There are 2006 positive words and 4782 negative words. The dictionary has been adopted from :
- Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA
- There are 3 other small dictionaries also included which help in calculating the weightage factor depending on the adverb which increases or decreases the effect of main verb. The third dictionary handles flips in sentences. Ex. “not bad” which means something non negative.
- The algorithm for doing the sentiment analysis does text, splitting, tagging and extracting information from the same.

- There are numerous ways of handling the same, some very simple structured for example defining the text simply as list of words or we could have complex structure which would carry processed text.
- We are going to follow following structure :
- 1. Each text is a list of sentences
- 2. Each sentence is a list of tokens
- 3. Each token is a tuple of three elements: a word form (the exact word that appeared in the text), a word lemma (a generalized version of the word), and a list of associated tags.
- For Example: The tweet collected on Wholefoods – "RT @scooter bunny: Inspiring demonstration in San Francisco by @DxEverywhere today. The greenwashing by #WholeFoods will be exposed.", would appear in following manner after POS-tagging:

```
[('The', 'The', ['DT']),
('greenwashing', 'greenwashing', ['NN']),
('by', 'by', ['IN']),
('#', '#', ['NN']),
('WholeFoods', 'WholeFoods', ['NNS']),
('will', 'will', ['MD']),
('be', 'be', ['VB']),
 ('exposed', 'exposed', ['VBN']),
 ('.', '.', ['.'])]
```

Step -1 : Tokenize, split into sentences and POS tag.

- We are making use of NLTK library for the same.

```
import nltk

class Splitter(object):
        def __init__(self):
        self.nltk_splitter = nltk.data.load('tokenizers/punkt/english.pickle')
        self.nltk_tokenizer = nltk.tokenize.TreebankWordTokenizer()

        def split(self, text):

     sentences = self.nltk_splitter.tokenize(text)
        tokenized_sentences = [self.nltk_tokenizer.tokenize(sent) for sent in sentences]
        return tokenized_sentences
```

· Here the above code is used to split the tweets which would be read from a file.

```
class POSTagger(object):
        def __init__(self):
        pass

    def pos_tag(self, sentences):
        pos = [nltk.pos_tag(sentence) for sentence in sentences]
        pos = [[(word, word, [postag]) for (word, postag) in sentence] for sentence in pos]
        return pos
```

- Here POS tag is attached to the tokenized sentences.
- Using the above two wrapper functions we perform basic text preprocessing, where the input is individual tweet which is read from the txt file and out put is collection of sentences , each of which is collection of tokens.
- Since NLTK does not lemmatize the words our forms and lemmas would be always identical. Till now the tag which is attached to the word is the one provided by NLTK library.

Step -2 : Tagging the text with dictionaries

Next, we need the dictionaries, as mentioned earlier we are using dictionary for positive and negative word each. Which is stored in .yml form.

```
class DictionaryTagger(object):
        def __init__(self, dictionary_paths):
        files = [open(path, 'r') for path in dictionary_paths]
        dictionaries = [yaml.load(dict_file) for dict_file in files]
        map(lambda x: x.close(), files)
        self.dictionary = {}
        self.max_key_size = 0
        for curr_dict in dictionaries:
        for key in curr_dict:
        if key in self.dictionary:
                self.dictionary[key].extend(curr_dict[key])
        else:
                self.dictionary[key] = curr_dict[key]
        self.max_key_size = max(self.max_key_size, len(key))
```

· In the above class all the dictionaries are read and each word is stored in map with inbuilt lamda function of python.

· Then , the key of the dictionaries are set so as to make up next function of lookup easy.

```
def tag(self, postagged_sentences):
return [self.tag_sentence(sentence) for sentence in postagged_sentences]

def tag_sentence(self, sentence, tag_with_lemmas=False):
"""
the result is only one tagging of all the possible ones.
The resulting tagging is determined by these two priority rules:
- longest matches have higher priority
- search is made from left to right
"""
tag_sentence = []
N = len(sentence)
if self.max_key_size == 0:
self.max_key_size = N
i = 0
while (i < N):
j = min(i + self.max_key_size, N) #avoid overflow
tagged = False
while (j > i):
expression_form = ' '.join([word[0] for word in sentence[i:j]]).lower()
  expression_lemma = ' '.join([word[1] for word in sentence[i:j]]).lower()
if tag_with_lemmas:
        literal = expression_lemma
else:
        literal = expression_form
if literal in self.dictionary:
        #self.logger.debug("found: %s" % literal)
        is_single_token = j - i == 1
        original_position = i
        i = j
        taggings = [tag for tag in self.dictionary[literal]]
        tagged_expression = (expression_form, expression_lemma, taggings)
        if is_single_token: #if the tagged literal is a single token, conserve its previous taggings:
```

```
                original_token_tagging = sentence[original_position][2]
                tagged_expression[2].extend(original_token_tagging)
                tag_sentence.append(tagged_expression)
                tagged = True
        else:
            j = j - 1
        if not tagged:
        tag_sentence.append(sentence[i])
        i += 1
        return tag_sentence
```

- The above class shows how the word is searched in map and tagged

The output for the above function would be:

```
[ ('RT', 'RT', ['NN']),
 ('@', '@', ['NN']),
 ('scooter_bunny', 'scooter_bunny', ['NN']),
 (':', ':', ['NN']),
 ('inspiring', 'inspiring', ['positive', 'NN']),
 ('demonstration', 'demonstration', ['NN']),
 ('in', 'in', ['IN']),
 ('San', 'San', ['NN']),
 ('Francisco', 'Francisco', ['NN']),
 ('by', 'by', ['IN']),
 ('@', '@', ['NN']),
 ('DxEverywhere', 'DxEverywhere', ['NN']),
 ('today', 'today', ['NN']),
 ('.', '.', ['.'])],
```

- Here as each tagged token is compared to dictionary words, inspiring comes up in positive list and hence tagged positive.

Step-3: Calculation of Sentiment

- By this time our data would have been tagged as positive or negative hence summing up would provide the complete score.

```
def value_of(sentiment):
```

```
        if sentiment == 'positive': return 1
        if sentiment == 'negative': return -1
        return 0

def sentiment_score(review):
   return sum ([value_of(tag) for sentence in dict_tagged_sentences for token in sentence for tag in token[2]])
```

- We save this values to json file, which is further used for visualization.

Step-4: Inverts, incrementers and decrementers

- These features are taken care by respective files and following code, which attaches some weight to the corresponding word and adjusts score accordingly.

```
def sentence_score(sentence_tokens, previous_token, acum_score):
   if not sentence_tokens:
        return acum_score
        else:
        current_token = sentence_tokens[0]
        tags = current_token[2]
        token_score = sum([value_of(tag) for tag in tags])
      if previous_token is not None:
        previous_tags = previous_token[2]
        if 'inc' in previous_tags:
        token_score *= 2.0
        elif 'dec' in previous_tags:
        token_score /= 2.0
        elif 'inv' in previous_tags:
        token_score *= -1.0
        return sentence_score(sentence_tokens[1:], current_token, acum_score + token_score)

def sentiment_score(review):
        return sum([sentence_score(sentence, None, 0) for sentence in review])
```

Overview:

· Just to sum up we first, divided sentences into tokens, secondly, we added POS (Part Of Speech) tags to tokenized text using nltk, thirdly used the dictionaries to tag for “positive", "negative", "inverter", "incrementer" and "decrementer" and calculated the score.

8. KDD Process

Knowledge Discovery in Databases, or KDD refers to the broad process of finding knowledge in data, and emphasizes the "high-level" application of particular data mining methods.

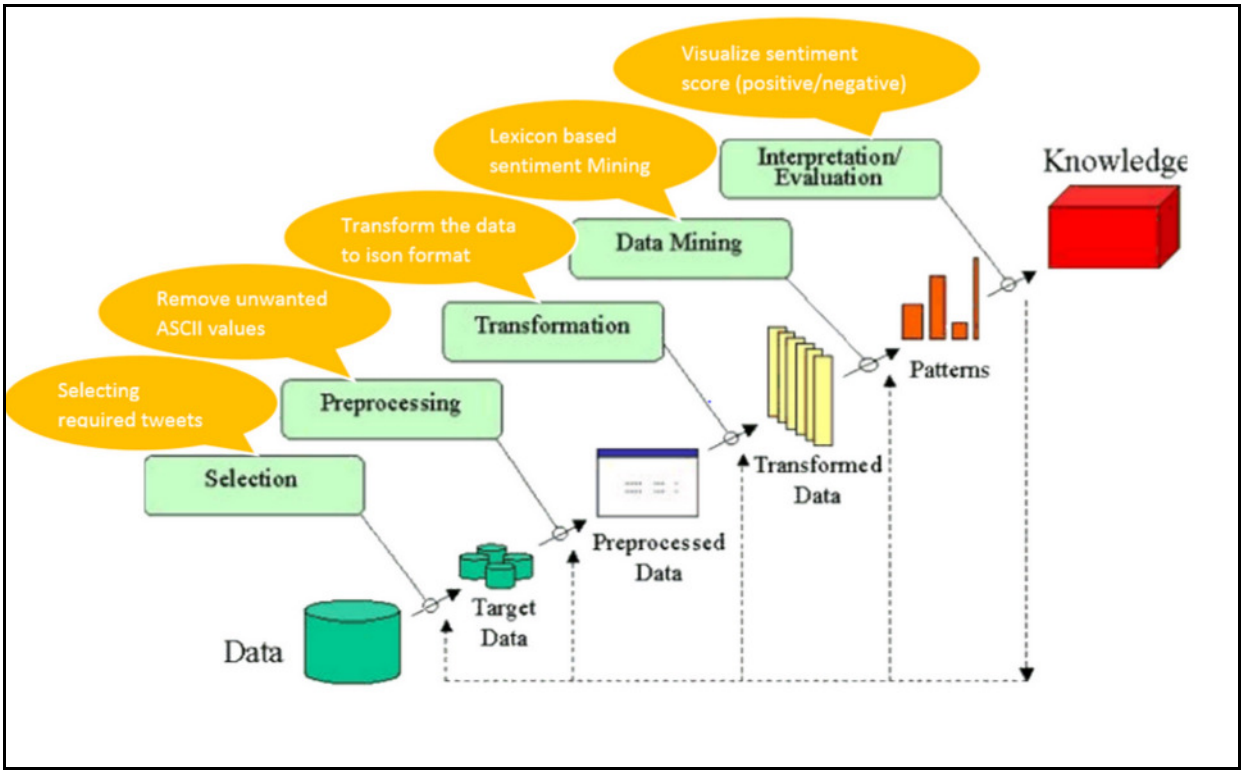


Figure 10: KDD process in sentiment Analysis

The steps involved in the KDD process are as follows:

1. Identifying the goal of the KDD process for Twitter sentiment analysis.
2. Understanding the different application domains involved and the discovery of knowledge that is required. Application domain involved is the twitter tweets to detect the sentiment of people towards a given domain.
3. Selecting the target data set on which the knowledge discovery is to be performed. For analyzing the sentiment we took Twitter data using the twitter api. The tweets with the required hashtags was selected in this process.

4. Cleansing and preprocessing the data to handle the missing fields or removing the unwanted fields as per the requirements. The tweets were preprocessed to remove the unwanted ascii values and data irrelevant for the analysis.
5. The file was cleansed and then converted to json format which served as a input file to the sentiment analysis engine.
6. Matching the Knowledge Discovery goals with various data mining methods to derive the sentiments.
7. Choosing the data mining algorithm to analyse the sentiments of people over some topic. The model suitable for entire KDD process is selected.

8. Lexicon based sentiment analysis approach is used for analyzing the sentiment from the tweets.
9. Interpretation of essential knowledge from the mined data. We interpreted how many people tweeted positive or negative towards a given topic. The interpreted knowledge was visualized using various JavaScript frameworks
10. Documenting the KDD results in reports for user reference.

Following is the process used for performing sentiment analysis:

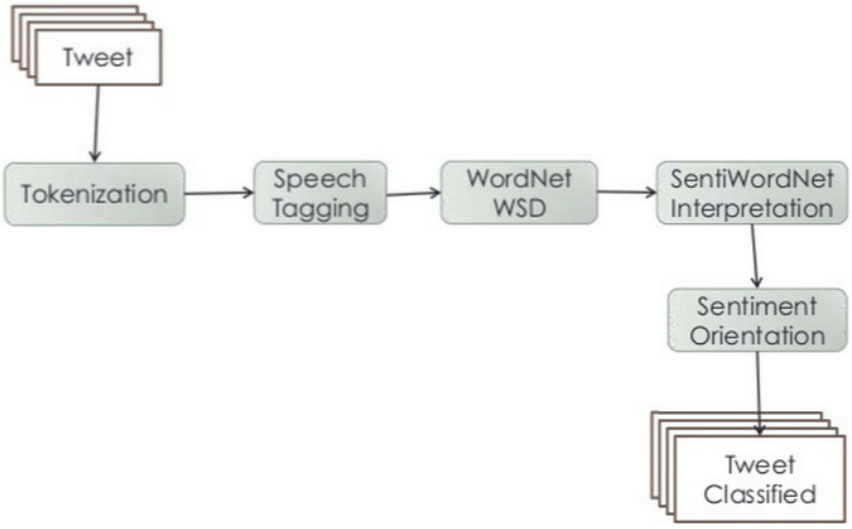


Figure 11: Process Flow for Sentiment Analysis

9. Data Tools

Tools	Description

Spring XD	Tool to gather twitter data for sentiment analysis
Hortonworks Hadoop 1.3	Tool to clean twitter data
Eclipse	The code for calculating sentiment score is done in PyDev in eclipse. Python packages used: Nltk- for POS tagging Yaml Pandas- it is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
BootStrap Framework	Framework for developing responsive, mobile projects on the web.
Apache Server	Tool to develop User Interface
Google App Engine	Tool to host the site
Google drive	Collaboration tool for reports, drawing and presentation

10. Client Side Design

Our client side application is purely JavaScript, html5, JQuery and CSS application. We have implemented the visualization functionality on the client side. Once the home page is loaded all the computations are calculated and the output.json is given to the visualization framework to generate charts.

Following is the basic client side design where the user can select what he wants to compare to generate the sentiment charts. The charts analyses the positive, negative and neutral sentiment towards the product/domain.

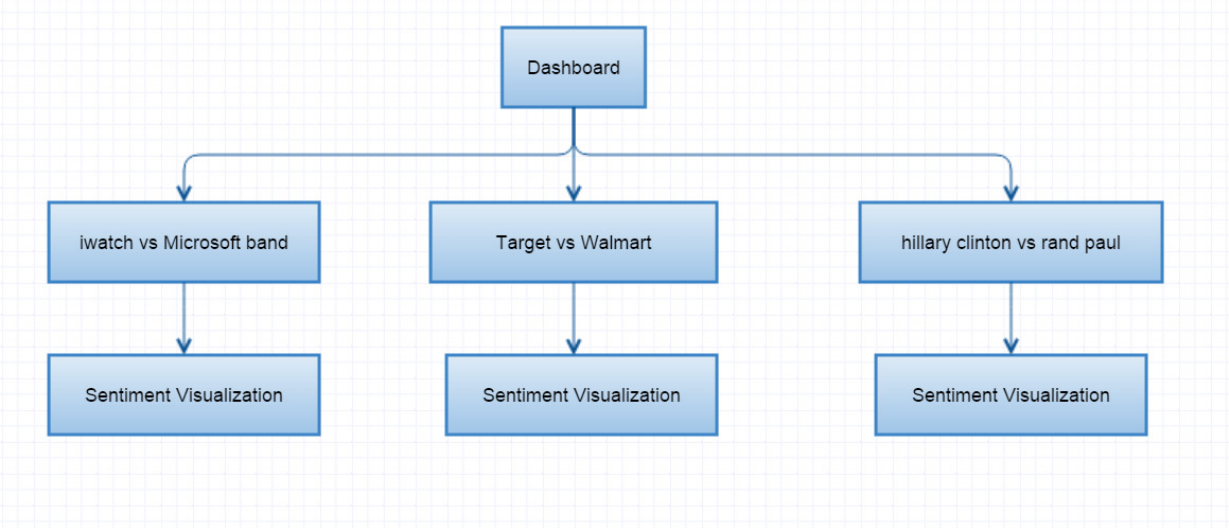
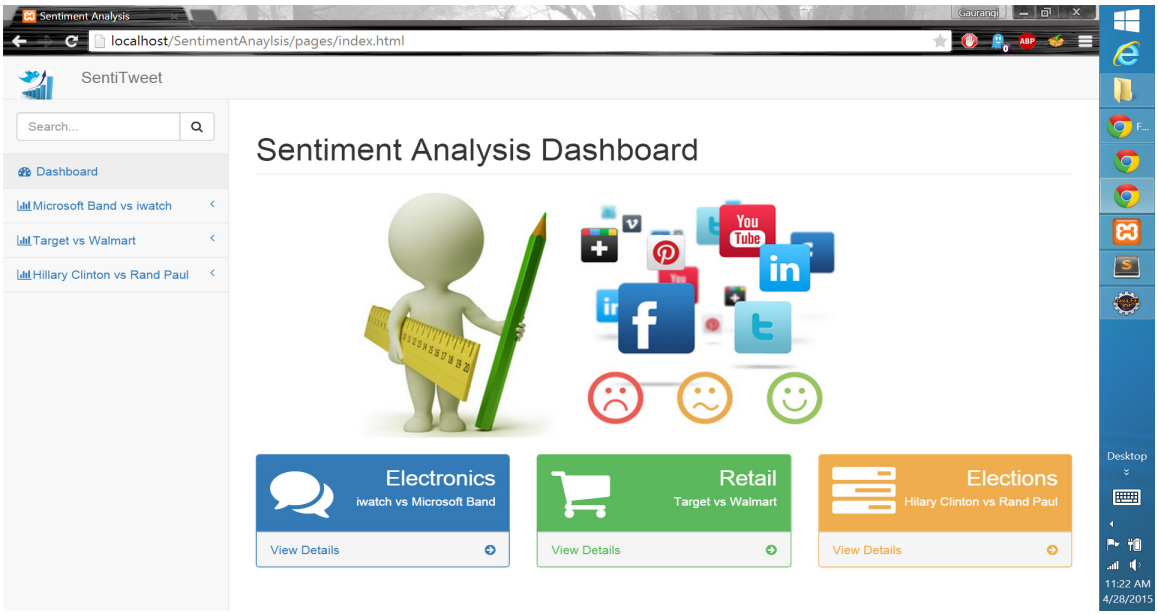


Figure 12: Client Side Design



11. Design Patterns

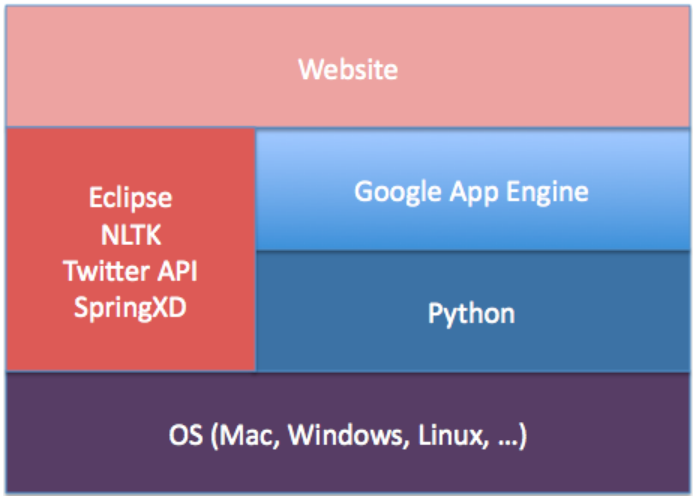


Figure 13: Technology Stack

Front End: For our user interface we created a website using HTML, CSS and Bootstrap.
We made use of JavaScript as a client side scripting language. This UI shows comparison graphs between the results of the analysis performed. User can choose a pair from the dropdown list for which he wants to analyse sentiments and see the comparison result.

Middle Tier: The middle tier is a python backend code. These functions are triggered based on Menu option selected on UI. The actual computation of all KDD process takes place here. It includes cleansing, and retrieving required data, analyzing and computing the sentiment score, generating the output files.

Data Store: Data Store is a simple directory which store the generated output in JSON format. This intermediate data is then used by our UI code. Such data is stored in the same directory.

Cloud: We have used Google App Engine for Python in order to host our application on cloud.

12. Testing (UI or Stress test)

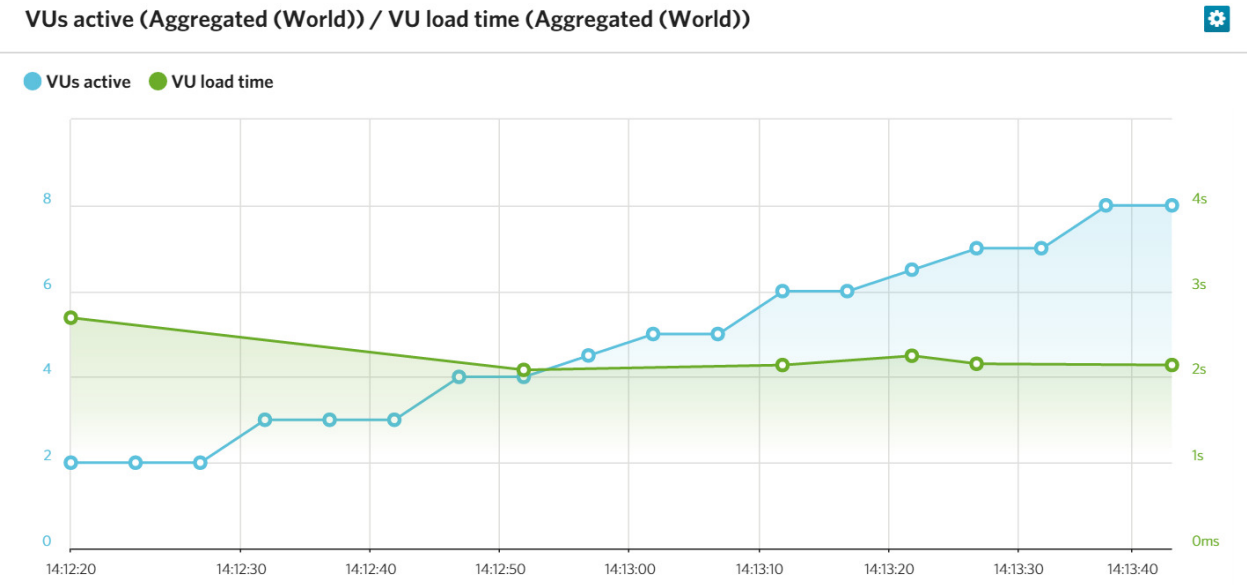
1. Specification based testing:

We tested the software based on the requirements specified. The given inputs and outputs i.e behavior of the software are tested to verify if they satisfy the expected values. Specification based testing is used for functional testing of our project.

2. Unit Testing:

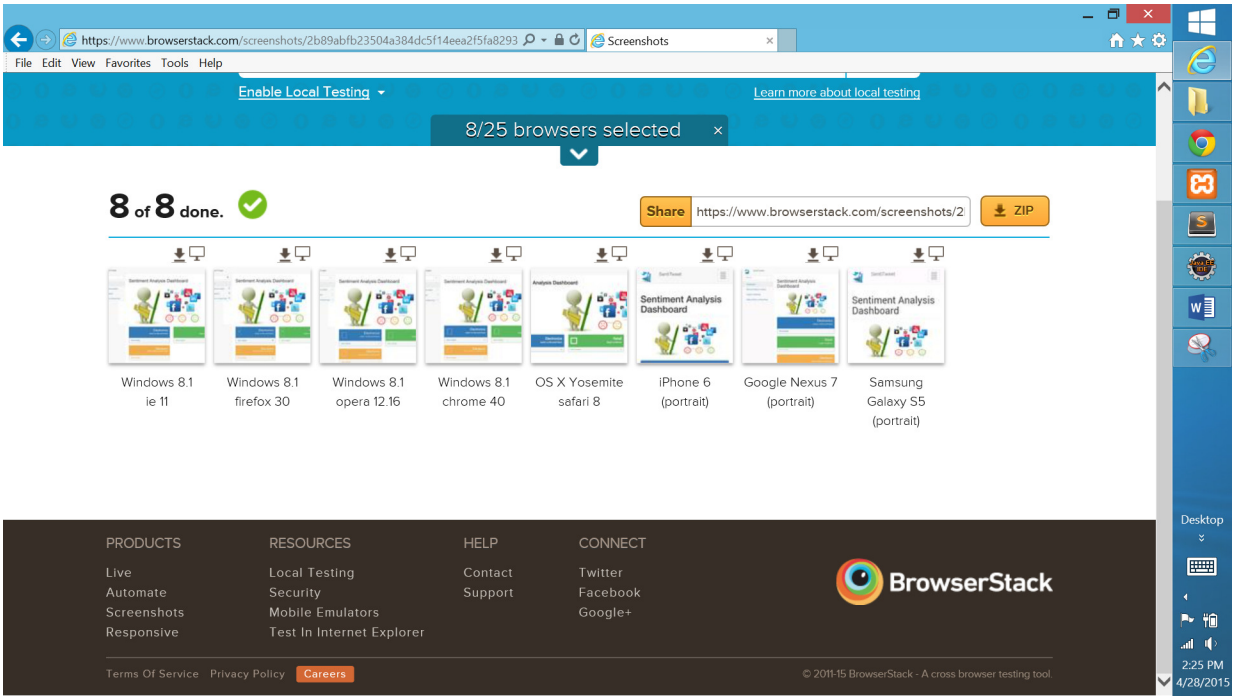
We performed unit testing to verify the functionality of specific sections of the code. The code for various steps like image segmentation, image extraction, and classification was tested independently to ensure that that part of the code performs as desired.

Stress Test for the website:



URL	Load zone	User scenario	Successful	Failed	Last avg
sentitweets.appspot.com/../sb-admin-2....	Sydney, AU (Amazon)	Auto generated scenario	39	0	319.94ms
sentitweets.appspot.com/../sb-admin-2....	Aggregated (World)	Auto generated scenario	39	0	319.94ms
sentitweets.appspot.com/../logo1.jpg	Aggregated (World)	Auto generated scenario	39	0	318.08ms
sentitweets.appspot.com/../logo1.jpg	Sydney, AU (Amazon)	Auto generated scenario	39	0	318.08ms
sentitweets.appspot.com/../index.html	Sydney, AU (Amazon)	Auto generated scenario	39	0	459.07ms

Browser Compatibility test:



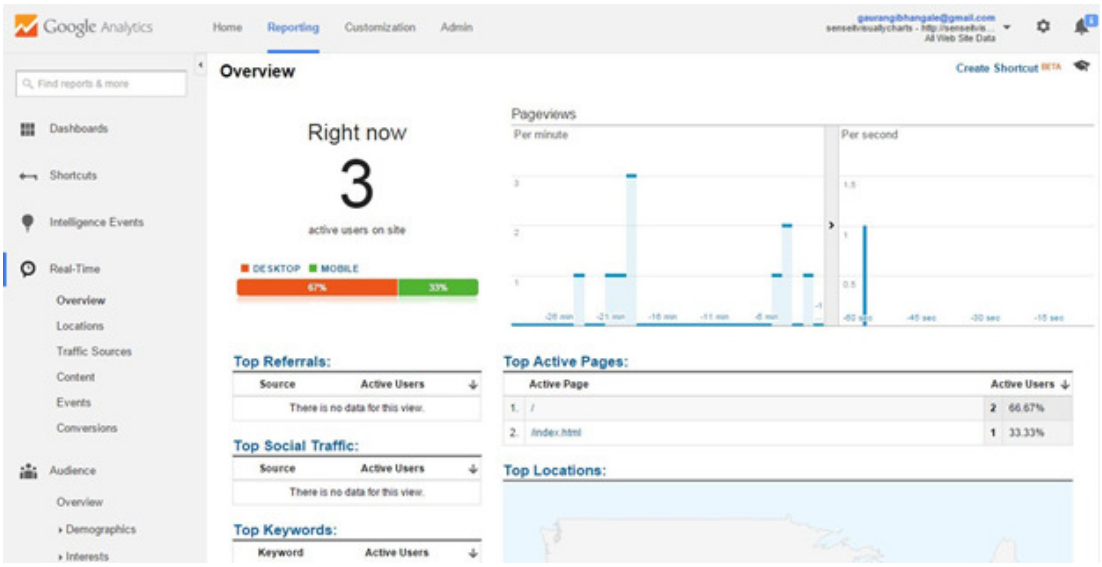
13. Google Analytics

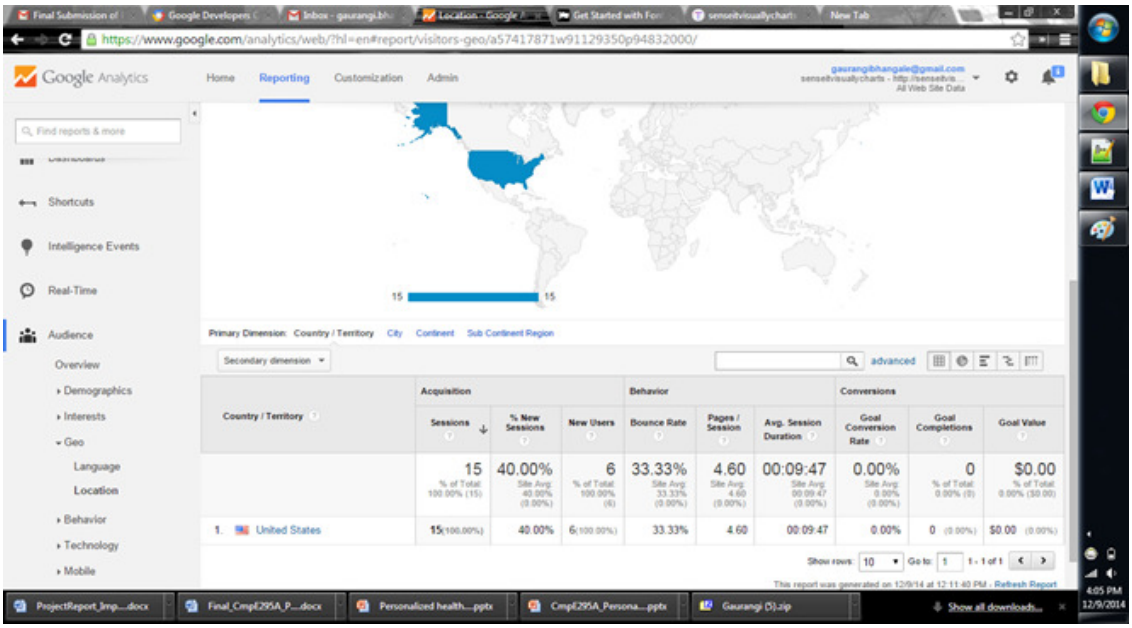
We have used google analytics to track the usage and performance of the system in real time. Following is the code for google analytics.

```
<script>
 (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
 (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
 m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
 })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

 ga('create', '**********', 'auto');
 ga('send', 'pageview');

</script>
```


Google Analytics
Home
Reporting
Customization
Admin
Overview
Create Shortcut BETA
Dashboards
Shortcuts
Intelligence Events
Real-Time
Overview
Locations
Traffic Sources
Content
Events
Conversions
Audience
Overview
Demographics
Interests
Right now
3
active users on site
DESKTOP
MOBILE
67%
33%
Pageviews
Per minute
Per second
Top Referrals:
Source
Active Users
There is no data for this view.
Top Social Traffic:
Source
Active Users
There is no data for this view.
Top Keywords:
Keyword
Active Users
Top Active Pages:
Active Page
Active Users
1. /
2
66.67%
2. /index.html
1
33.33%
Top Locations:

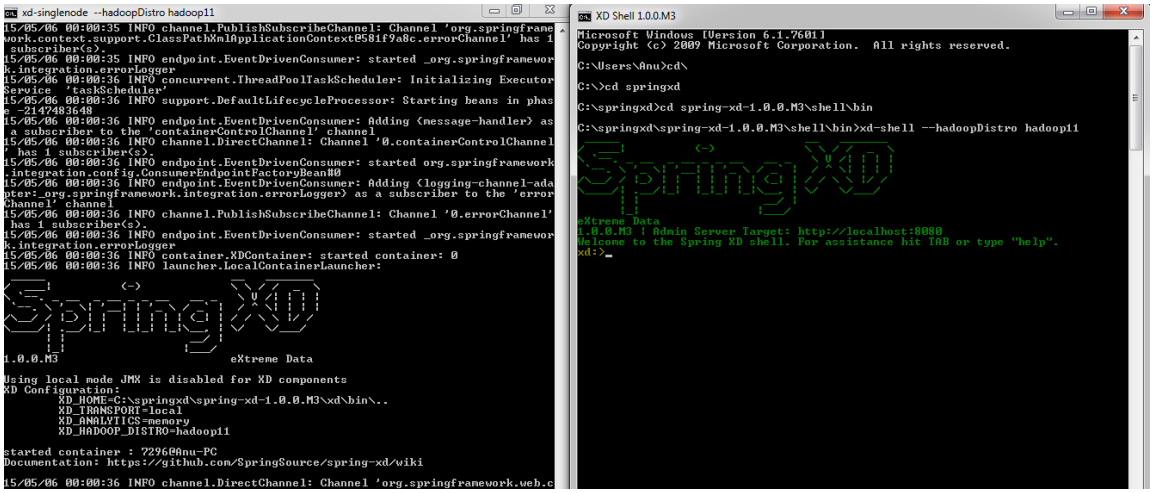

https://www.google.com/analytics/web/?hl=en#report/visitors-geo/a57417871w91129350p94832000/
Google Analytics
Home
Reporting
Customization
Admin
Shortcuts
Intelligence Events
Real-Time
Audience
Overview
Demographics
Interests
Geo
Language
Location
Behavior
Technology
Mobile
Primary Dimension: Country / Territory
City
Continent
Sub Continent Region
Secondary dimension
advanced
Country / Territory
Acquisition
Behavior
Conversions
Sessions
% New Sessions
New Users
Bounce Rate
Pages / Session
Avg. Session Duration
Goal Conversion Rate
Goal Completions
Goal Value
15
40.00%
6
33.33%
4.60
00:09:47
0.00%
0
$0.00
1. United States
15
40.00%
6
33.33%
4.60
00:09:47
0.00%
0
$0.00

14. Screenshots

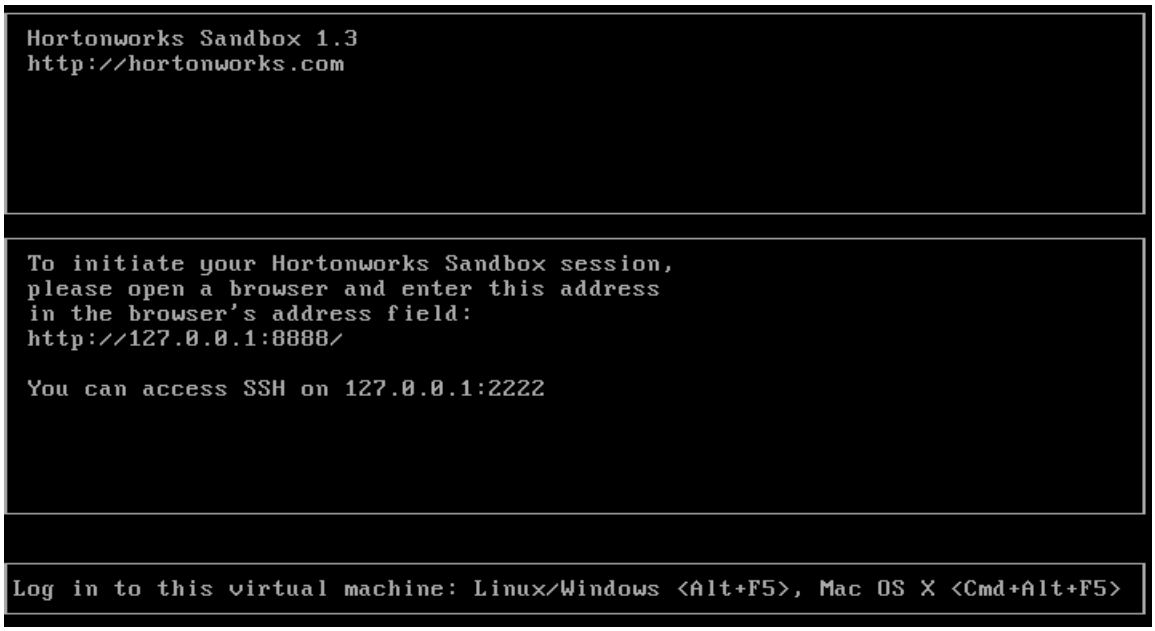
Start Spring xd server and shell with following commands:

C:\springxd>cd spring-xd-1.0.0.M3\xd\bin-----server

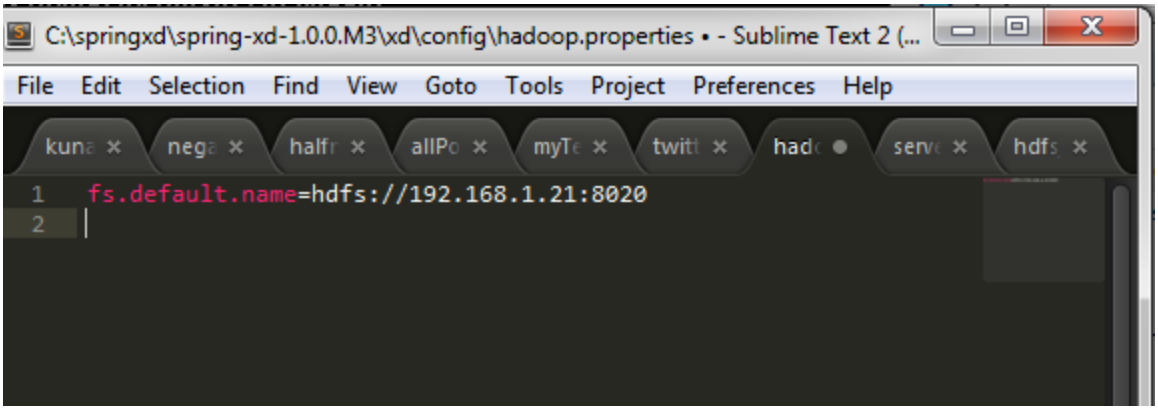
C:\springxd>cd spring-xd-1.0.0.M3\shell\bin—shell



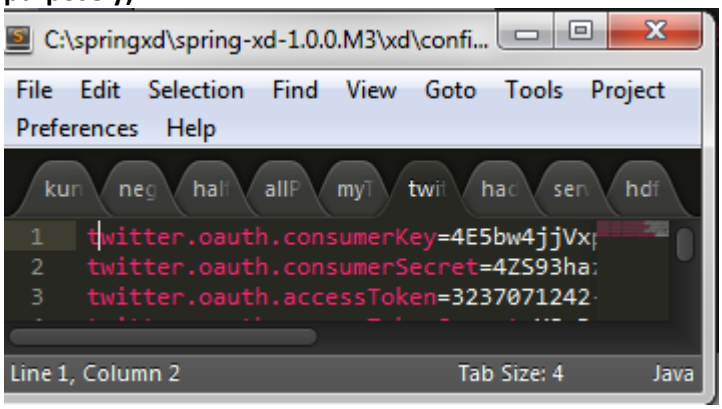
start Hortonworks Sandbox 1.3



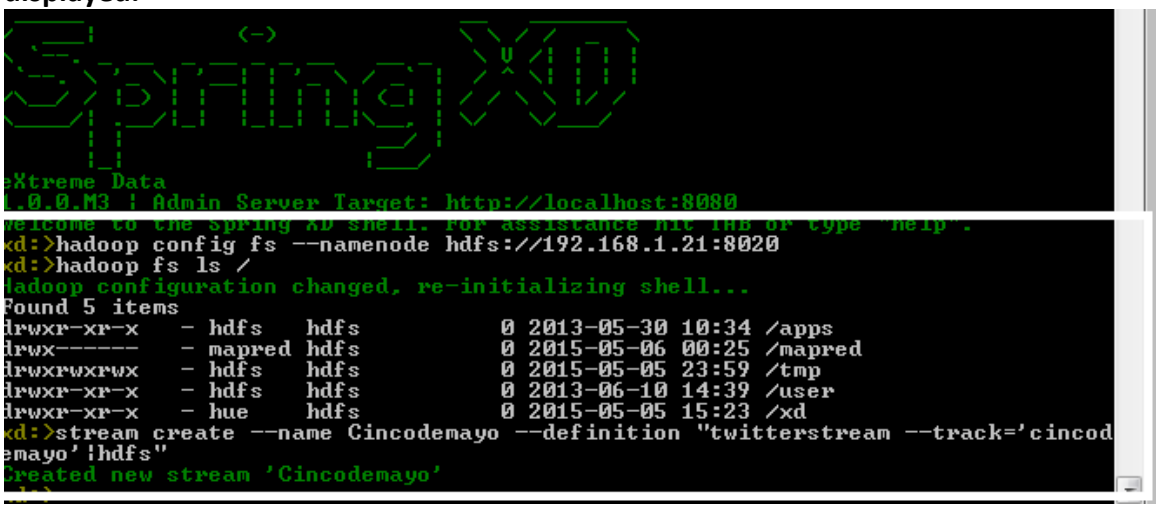
Add the hdfs ip given by hortonworks to hadoop.properties in config folder of xd in Spring xd.



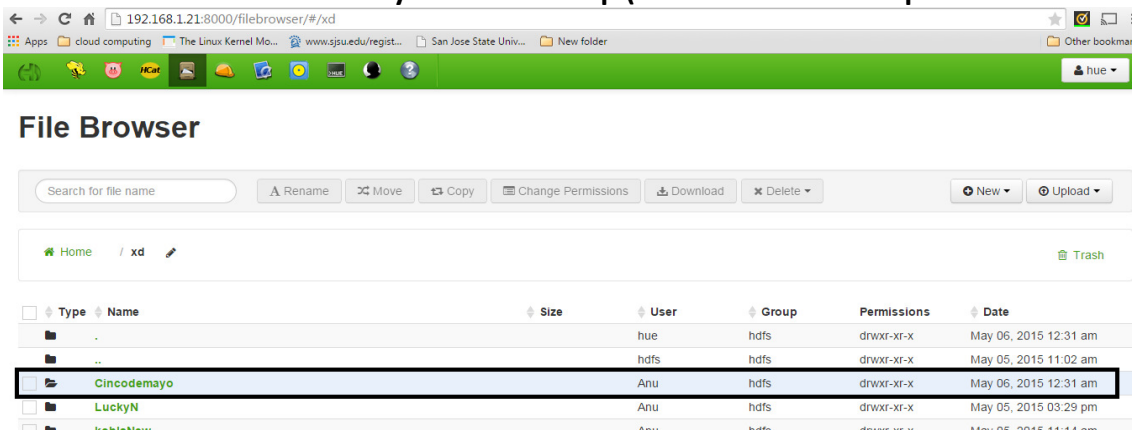
Add twitter Access and consumer keys to twitter.properties (complete picture not shown purposely)



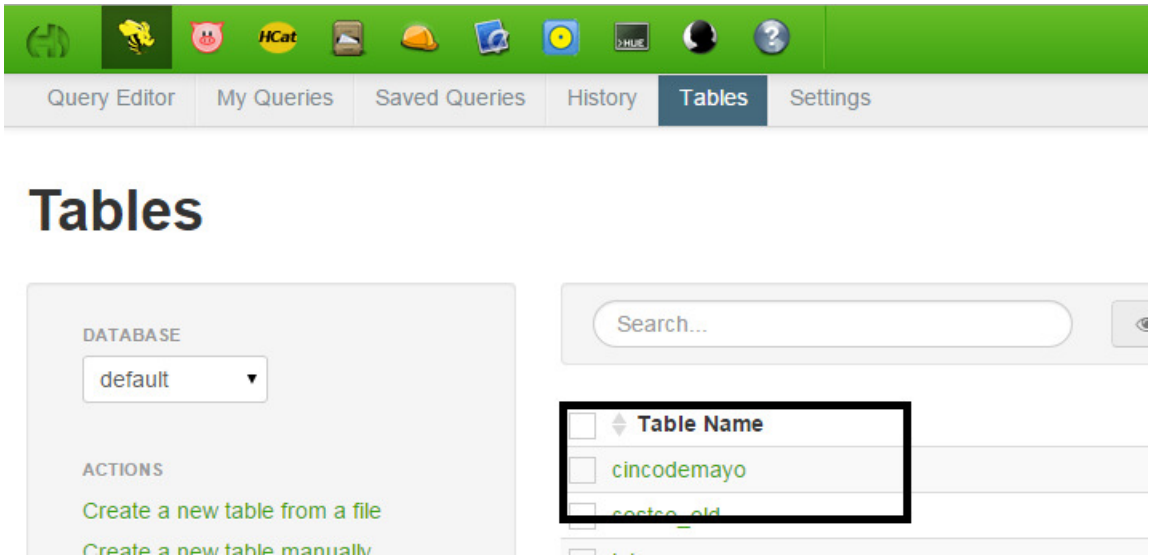
In shell script point to hdfs ip with port number and then start the query. Give the location for saving the file in hdfs. Once the query is created and message would be displayed.



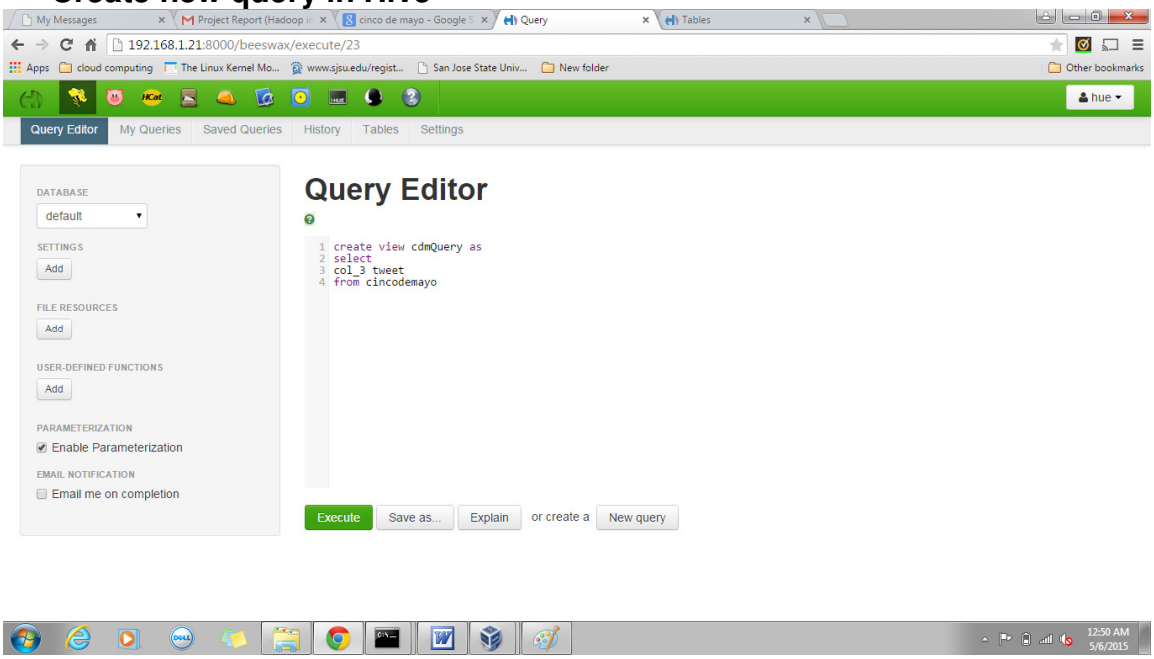
Now we move to Filesystem in Hadoop (Hortonworks Hadoop 1.3- stand alone cluster)



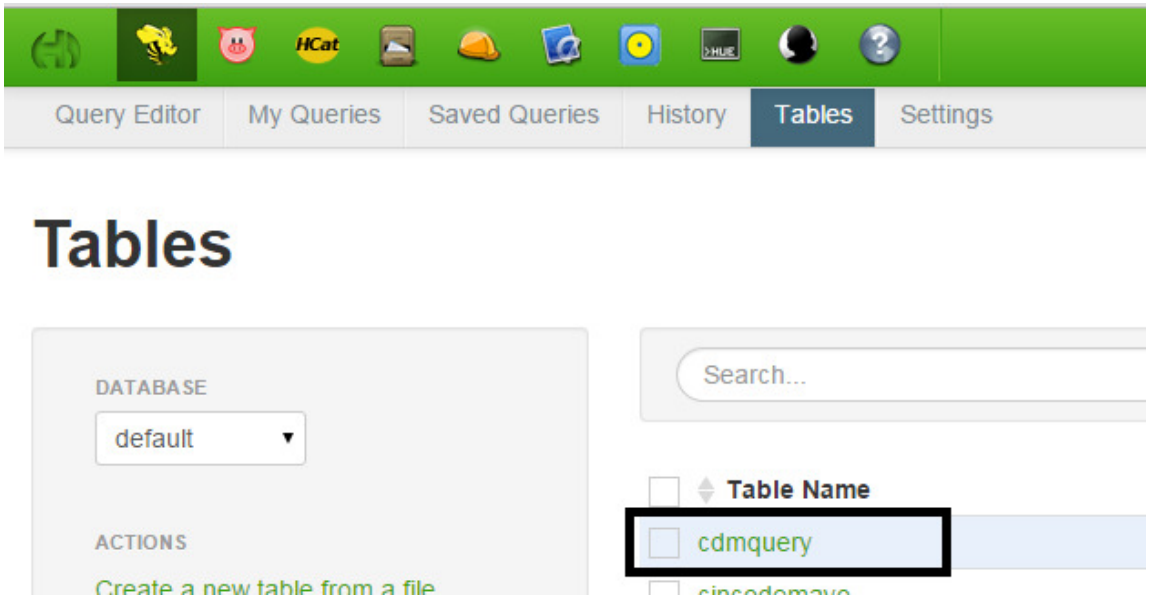
Create a new table in Hive



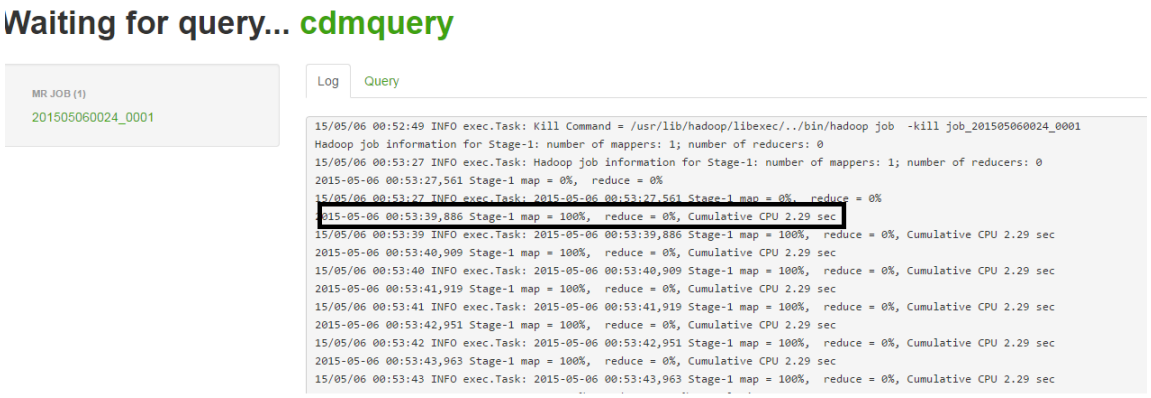
Create new query in Hive



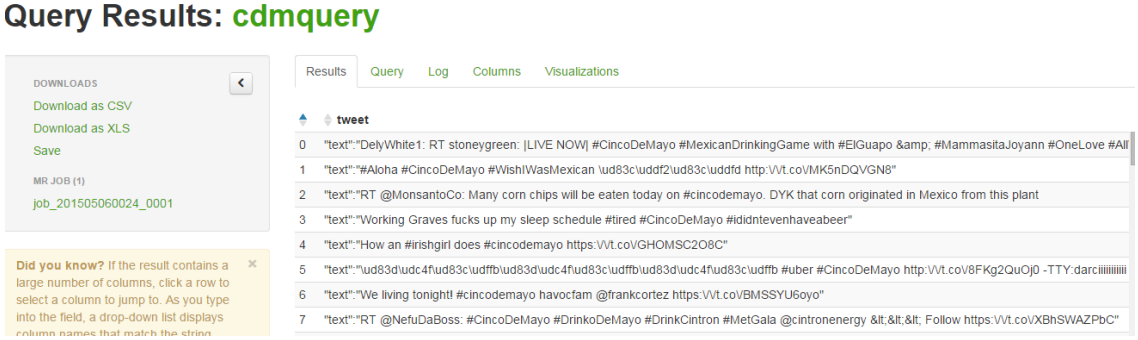
Once query is executed you can see table created under Tables



When you click on browse data in the query table, you can see Map Reduce job running



And then the data itself..



Sentiment Analysis in progress

basic_sentiment_analysis

```
59          files = [open(path, 'r') for path in dictionary_paths]
60          dictionaries = [yaml.load(dict_file) for dict_file in files]
61          map(lambda x: x.close(), files)
62          self.dictionary = {}
63          self.max_key_size = 0
64
65
66          for curr_dict in dictionaries:
```

Console | PyUnit

```
<terminated> C:\Users\Anu\Desktop\SJSU\cmpe239\project\basic_sentiment_analysis-master\basic_sentiment_analysis.py
('#', '#', ['#']),
('Starbucks', 'Starbucks', ['NNS']),
(',', ',', [',']),
('#', '#', ['#']),
('Target', 'Target', ['NN']),
('and', 'and', ['CC']),
('#', '#', ['NN']),
('Costco', 'Costco', ['NN']),
('stores', 'stores', ['NNS']),
('were', 'were', ['VBD']),
('burned', 'burned', ['negative', 'VBN']),
('to', 'to', ['TO']),
('the', 'the', ['DT']),
('ground', 'ground', ['NN']),
('in', 'in', ['IN']),
('#', '#', ['NN']),
('Baltimore \x85', 'Baltimore \x85', ['NN'])]]
analyzing sentiment...
-2
```

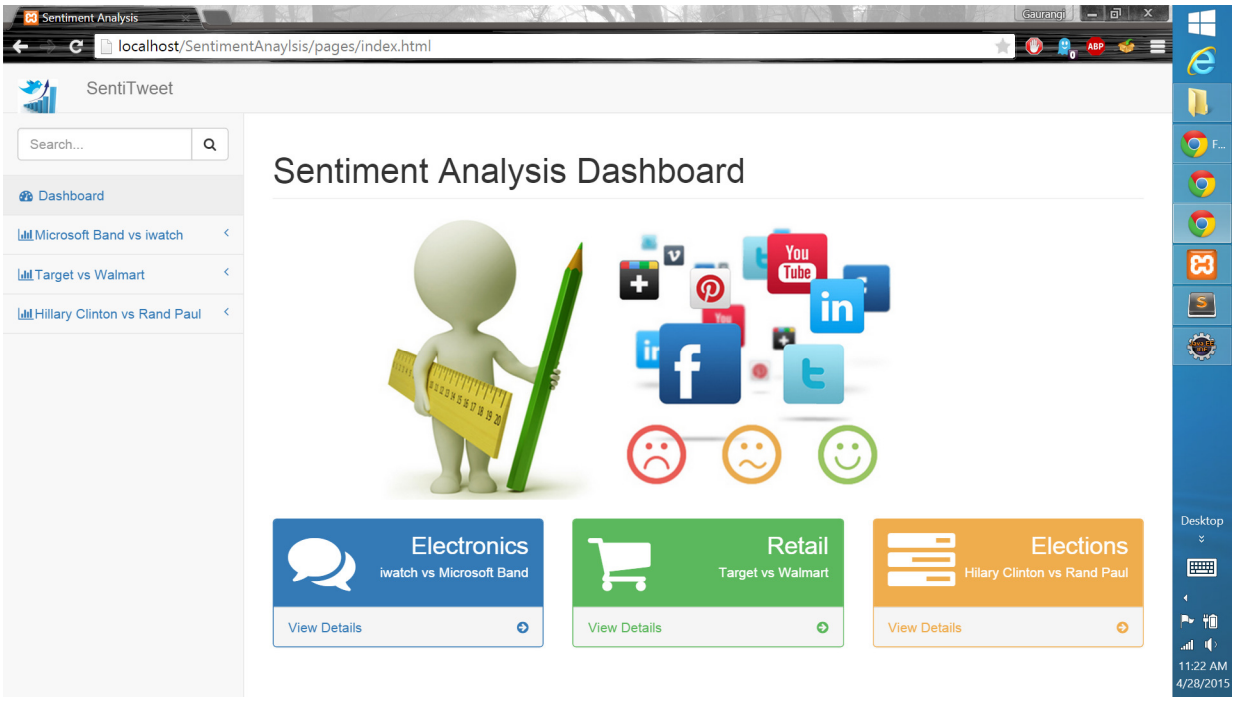
Screen shot for sentiment.json file which stores the score for each tweet. It acts as input for visualization

basic_sentiment_analysis | wholefoods_no_tweets.json | clinton26_sentiment.json | microsoftband26.json

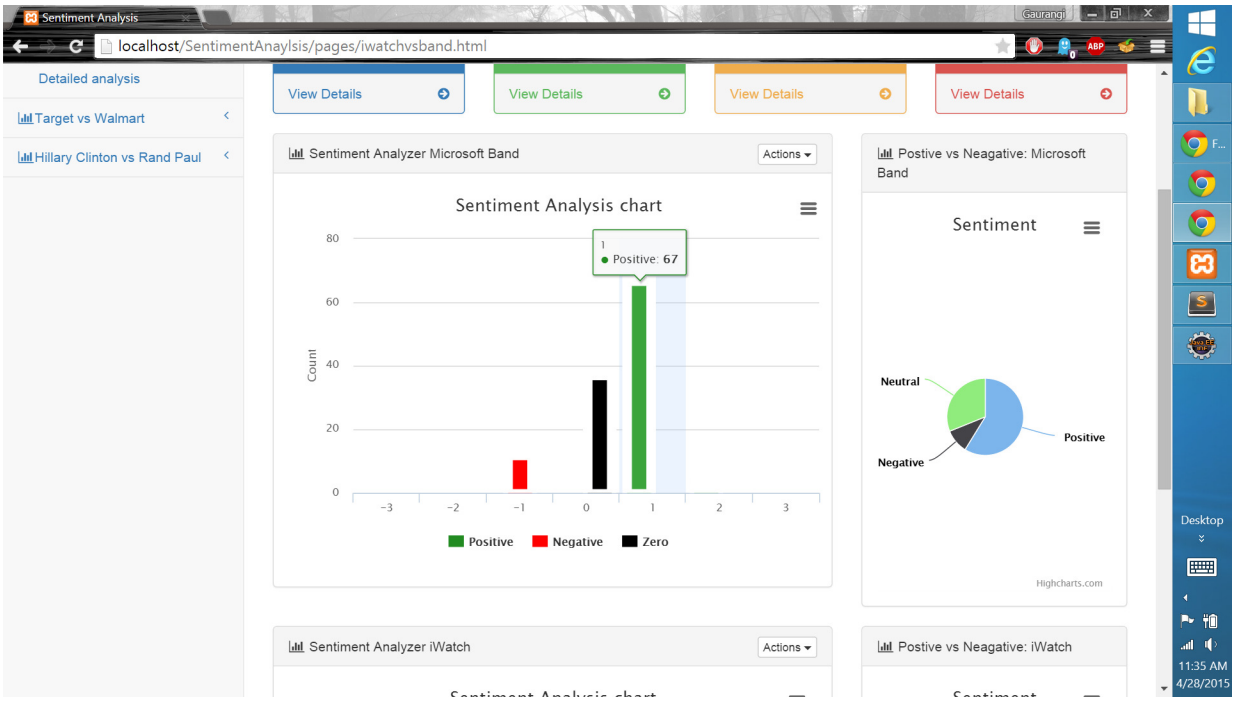
```
1 {
2     "line": "
3     "score": "0"
4 }{
5     "line": "0            Microsoft support for #veterans, #data collaboration, and the #MicrosoftBand: http://t.co/CnyG2GXSlW  #
6     "score": "1"
7 }{
8     "line": "1                    @microsoftband Is the heartbeat monitor availble when you are wearing the screen both outward
9     "score": "0"
10 }{
11     "line": "2                                  Chalk up the win for team @NaviNet! @microsoftband #grandhack #HealthIT #Wearab
12     "score": "1"
13 }{
14     "line": "3              @microsoftband are there issues with syncing with MapMyFitness or RunKeeper since the update today.
15     "score": "-1"
16 }{
17     "line": "4                    RT @NokiaDay: Un peu trop fan? J'avoue... Merci l'application FanBand, must have! #Microsoft
18     "score": "0"
```

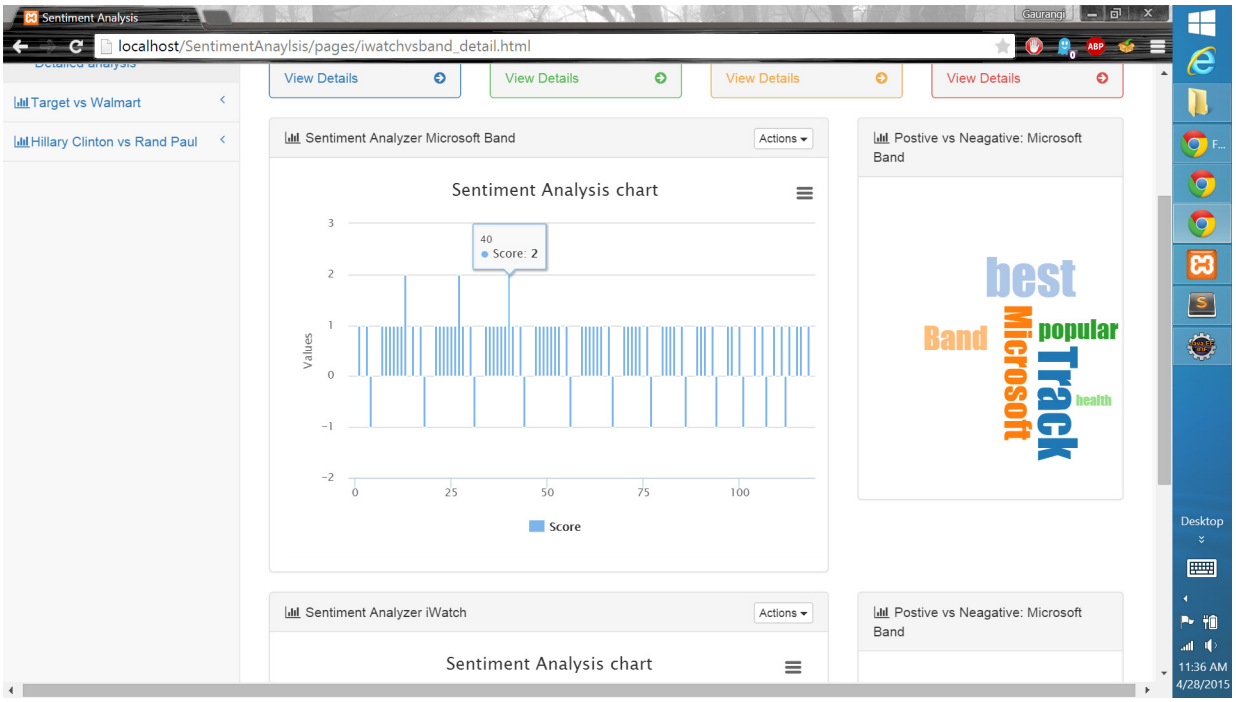
Front End Visualizations:

HomePage



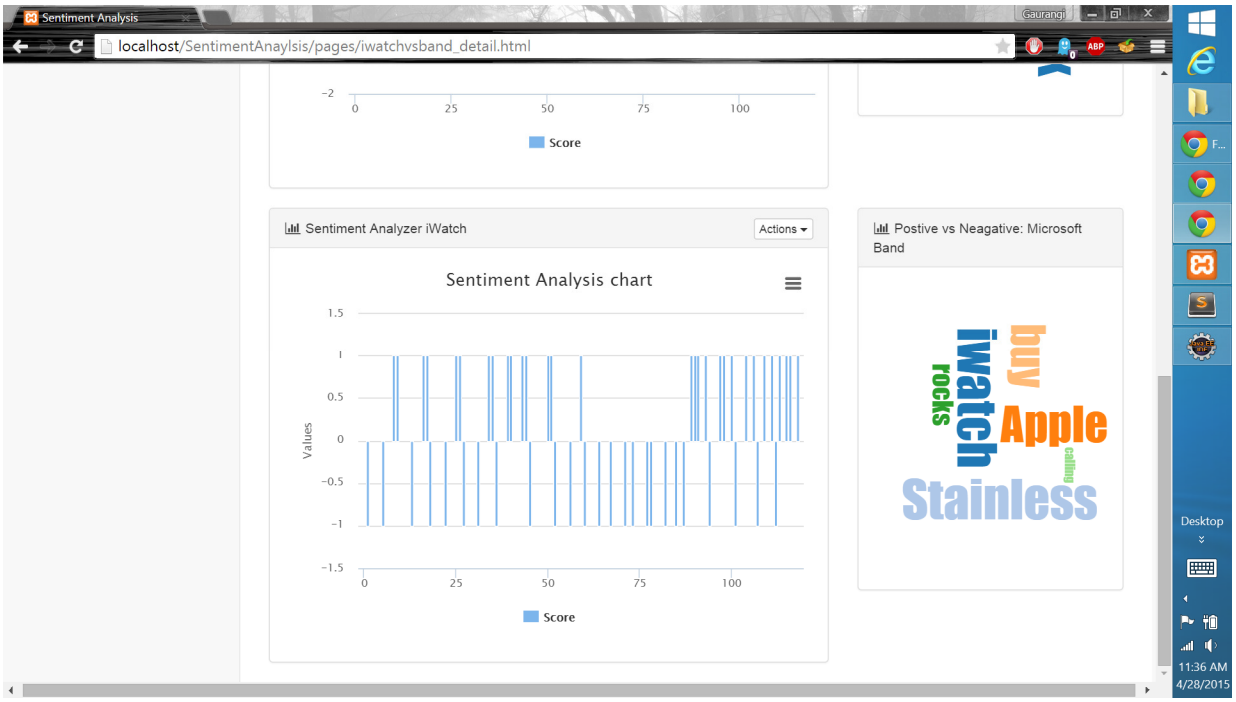
Sentiment Analysis: MicrosoftBand



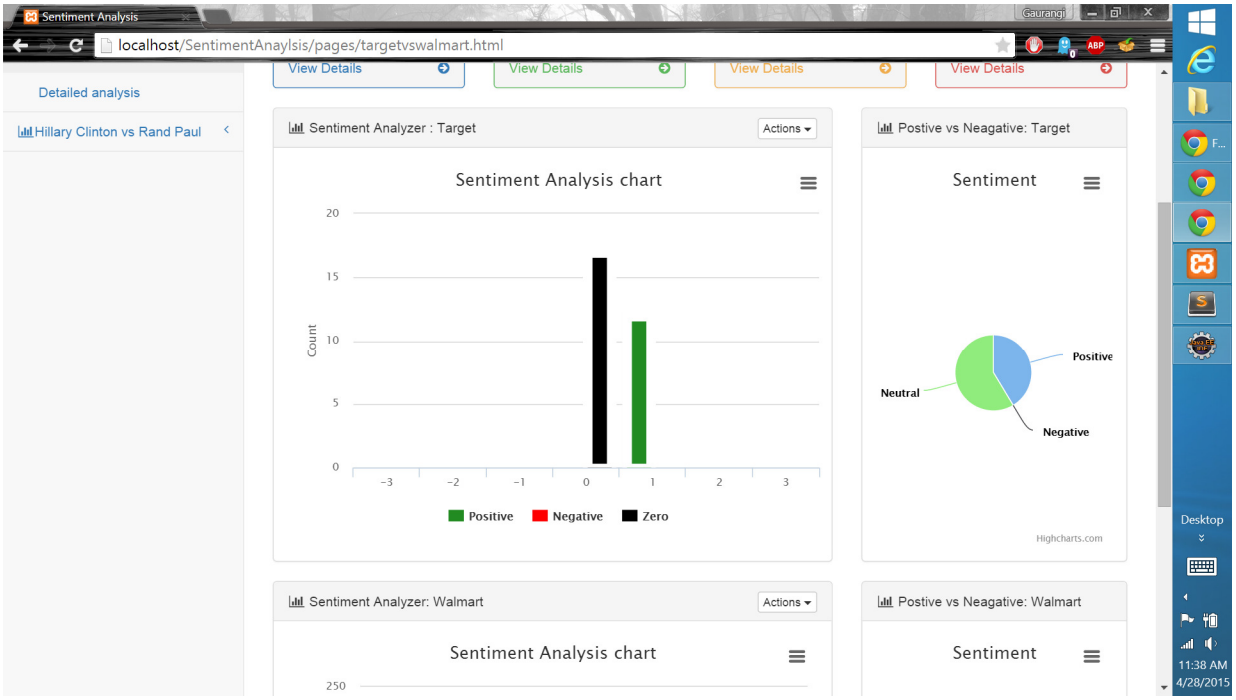


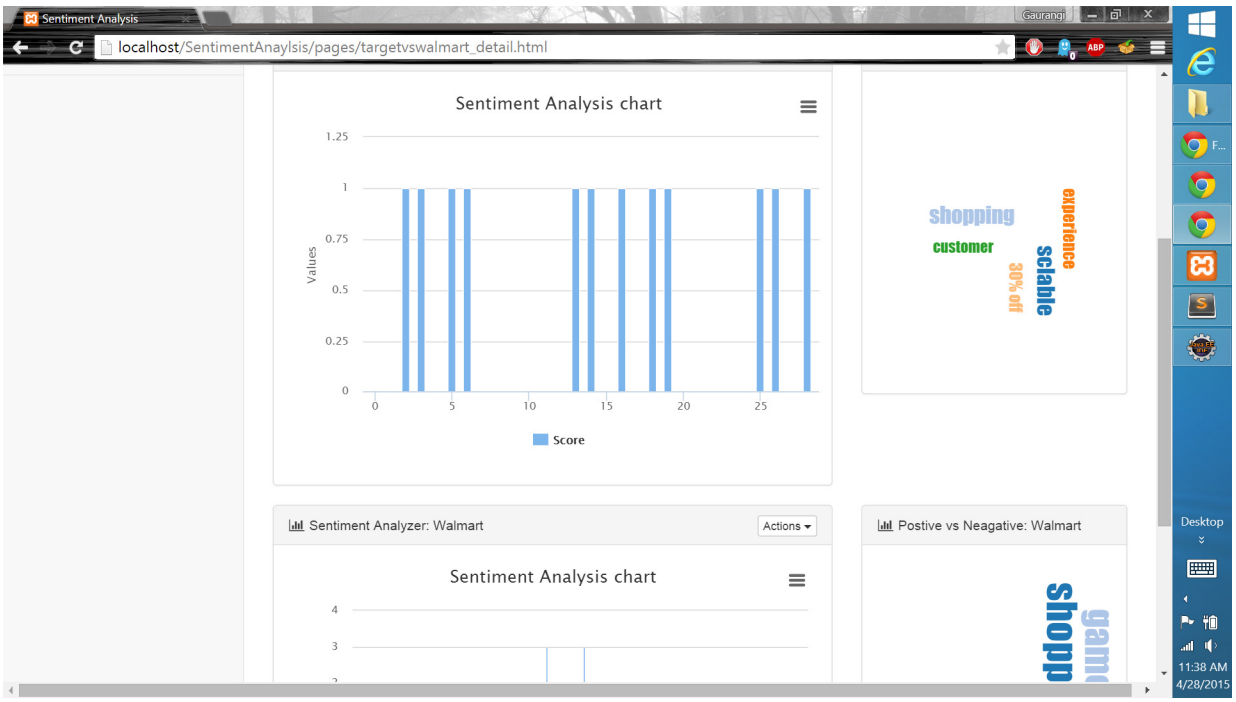
Sentiment Analysis: iwatch


Sentiment Analysis
localhost/SentimentAnaylsis/pages/iwatchvsband.html
Positive
Negative
Zero
Highcharts.com
Sentiment Analyzer iWatch
Actions
Sentiment Analysis chart
Count
Positive
Negative
Zero
Postive vs Negative: iWatch
Sentiment
Negative
TweetShare: 23.9%
Neutral
Highcharts.com
Desktop
11:35 AM
4/28/2015

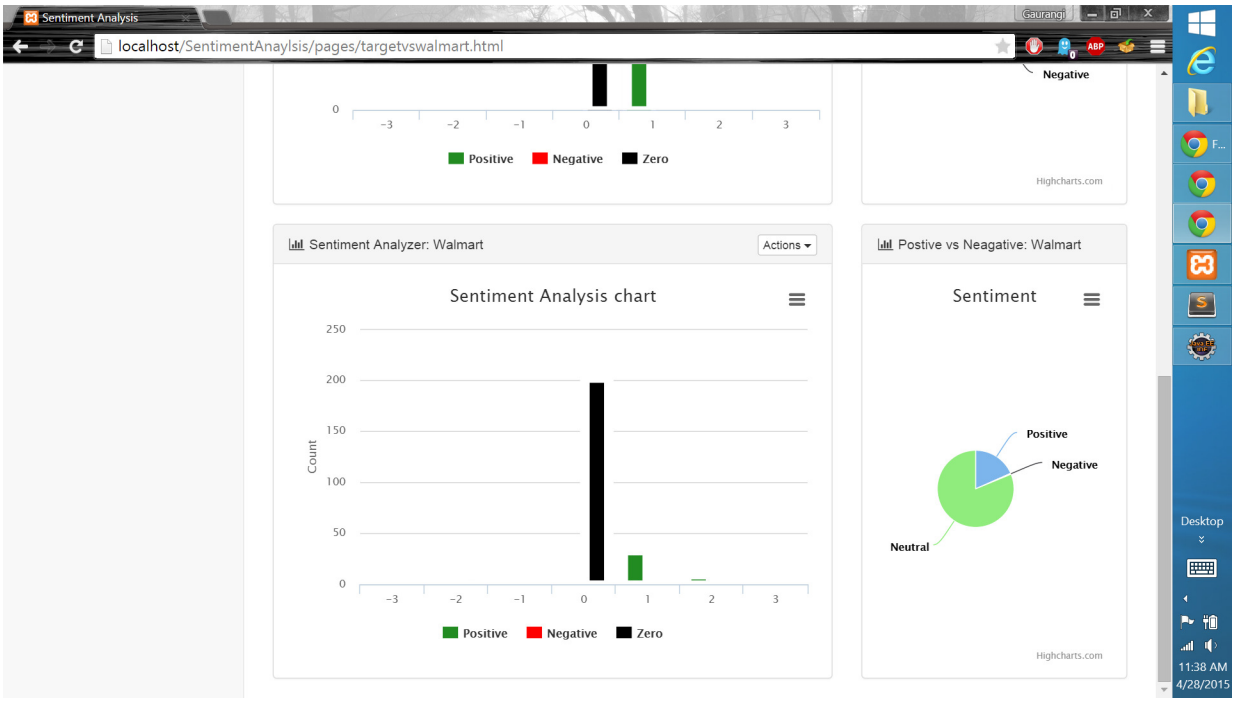

Sentiment Analysis
localhost/SentimentAnaylsis/pages/iwatchvsband_detail.html
Score
Sentiment Analyzer iWatch
Actions
Sentiment Analysis chart
Values
Score
Postive vs Negative: Microsoft Band
buy
iWatch
rocks
Apple
calling
Stainless
Desktop
11:36 AM
4/28/2015

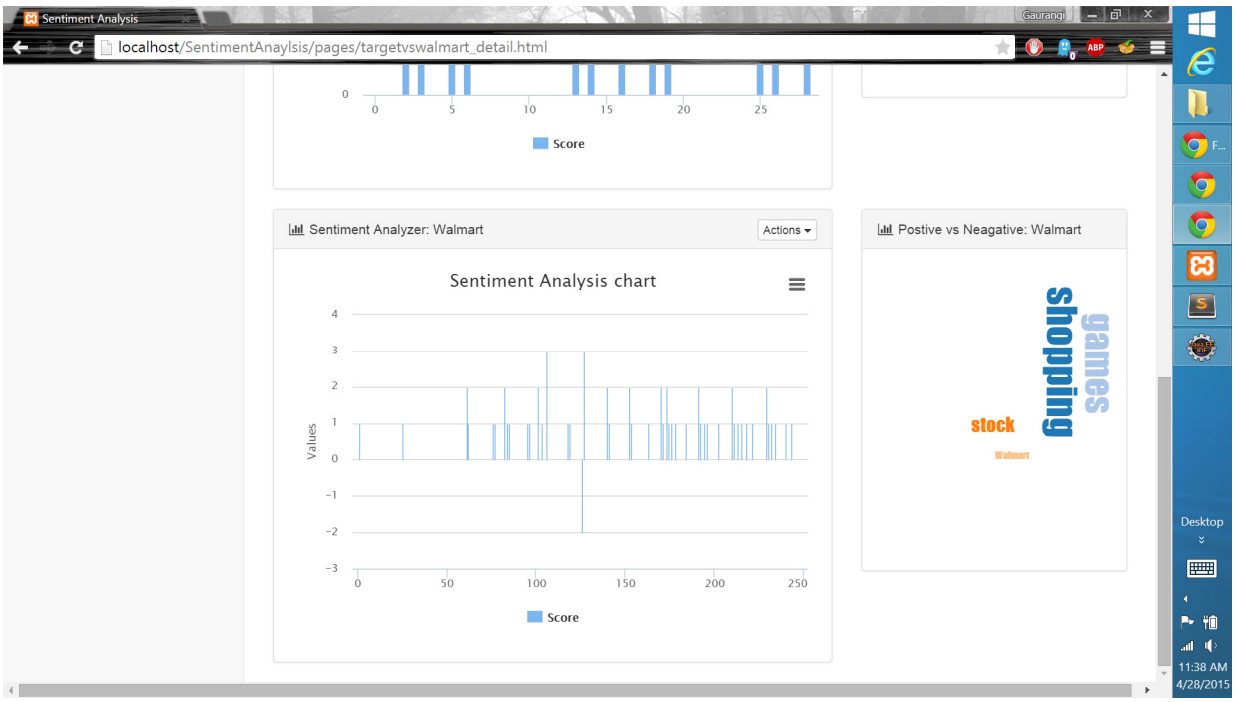
Sentiment Analysis: Target



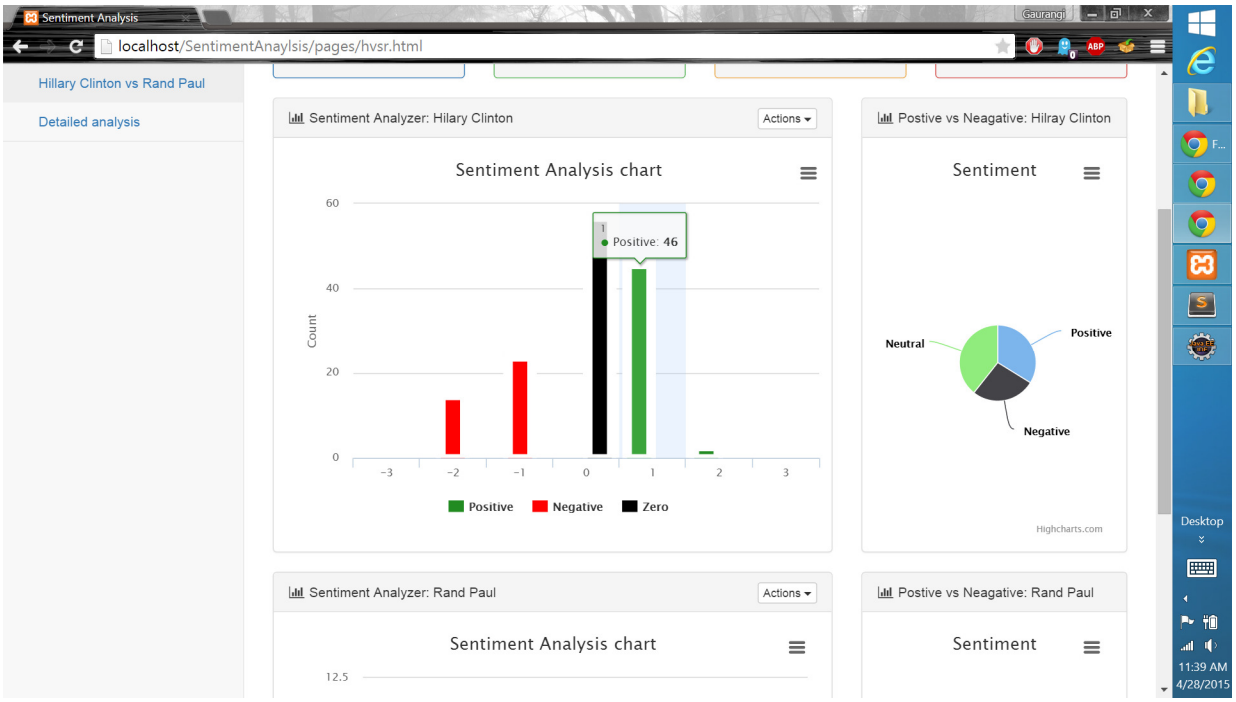


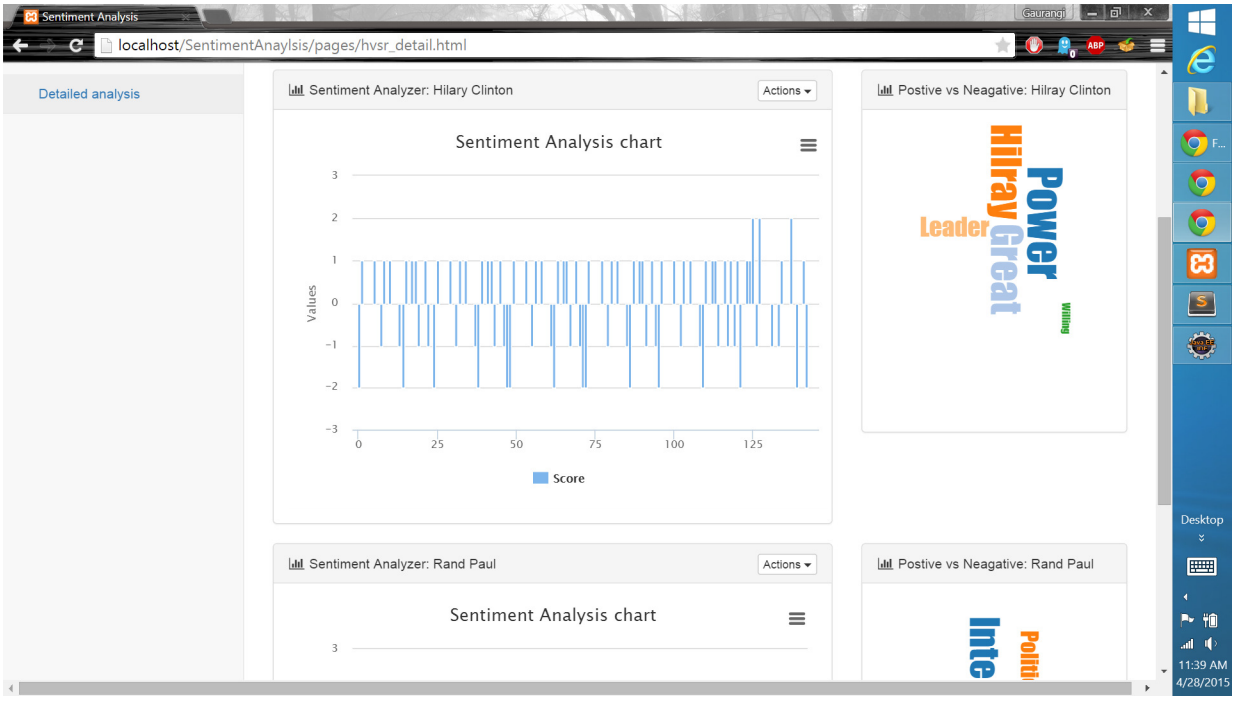
Sentiment Analysis: Walmart



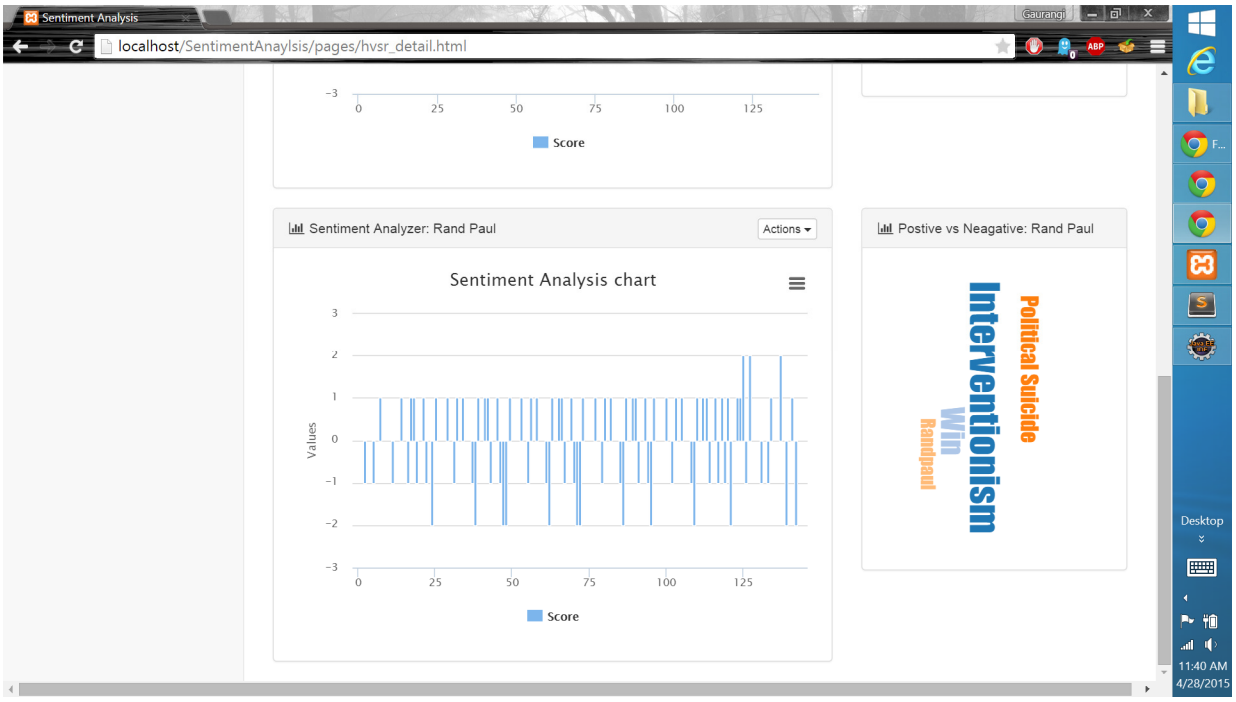


Sentiment Analysis: Hillary Clinton





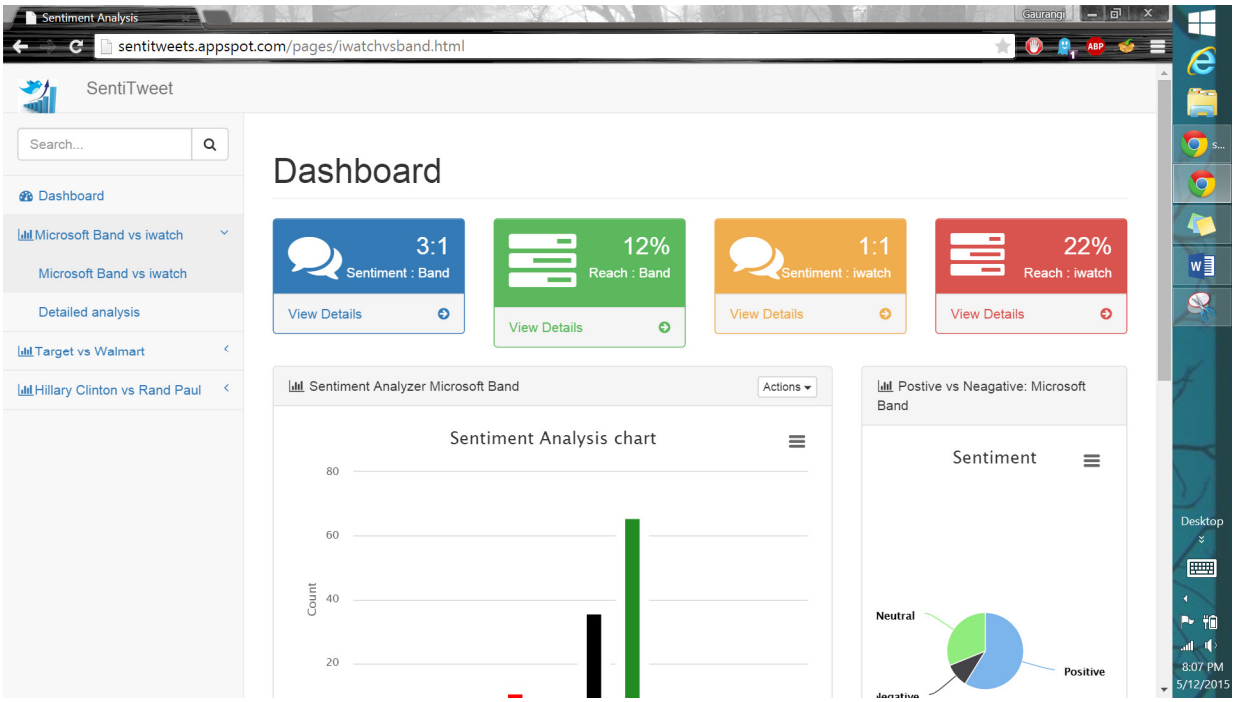
Sentiment Analysis: Rand Paul

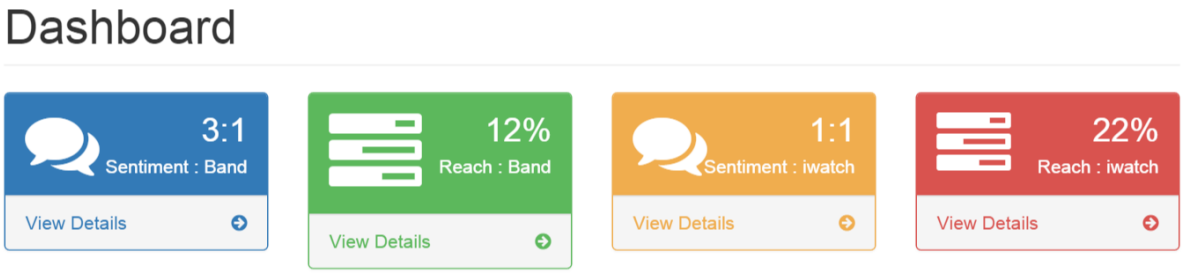


15. Challenges faced:

- All of us had experience programming in java, but we planned to learn Python as a new language. So it took some time to understand its basics and then code it.
- We also faced some problems in installing 'nltk' library in our machine. It was due to version issues we had in python installation with nltk and eclipse.
- Other problem was in the type of data we got from user, some of the tweets were written in native languages, which were difficult to process since we used English dictionary in order to identify the meaning of the data. So we removed such data in cleaning process.
- Other technical challenge was with the limit of tweets we can access from twitter.com (new Twitter API). In order to check the functionality of our system we had to obtain tweets from twitter every other day. But because of the limit they put on the tweet access, we had to sign up several accounts for different token.
- Hadoop installation with Hortonworks was working well with our home WiFi, but there was some problem running it with college network. We fixed this issue by activating hotspot on our mobile device.

16. Sentiment Score Checking Criteria:





For each of the topics, the Sentiment Ratio and the Reach is calculated so that the user can easily compare the popularity of the subject or topic

Sentiment Ratio: Sentiment Ratio is the ratio based on the positive and negative score

$$\text{Sentiment Ratio} = \frac{Total\ Number\ of\ Positive\ Score}{Total\ Number\ of\ Negative\ score}$$

And similarly, the Reach is calculated based on the number of unique users and total number of tweets. Reach is a measure of the range of influence.

$$\text{Reach} = \frac{Number\ of\ Unique\ Authors\ referencing\ your\ brand}{Total\ Number\ of\ mentions}$$

17. **References**

[1] "User Interface Design Tips, Techniques, and Principles." *User Interface Design Tips, Techniques, and Principles*. N.p., n.d. Web. 27 Apr. 2015. <http://www.ambysoft.com/essays/userInterfaceDesign.html>.

[2] "Overview of the KDD Process." *KDD Process/Overview*. N.p., n.d. Web. 28 Apr. 2015. <http://www2.cs.uregina.ca/~dbd/cs831/notes/kdd/1_kdd.html>.

[3] "Basic Sentiment Analysis with Python" Web. 28 Apr. 2015. <http://fjavieralba.com/basic-sentiment-analysis-with-python.htm>