Style Guidelines for Final Year Project Reports

COMP30440

School of Computer Science and Informatics

University College Dublin

17 October 2014

Twitter Sentiment Analysis

Project Specification

Sai-Keerthi-Chandu Seshadri

13205471

Project Specification

The purpose of this projects to retrieve tweets from the twitter and then send them for pre processing and then for analysis and then show them properly with html and css code embedded in to .jsp files and run on server with pie charts with some additional features included.

The development and implementation of such a product requires several steps to be followed:

- Tweets retrieval – There is a search bar which takes the keyword form the user and then retrieves the tweets using an api called Twitter4j.

- Sentimental analysis – There is a small step before sending the tweets to sentimental analysis, which is called filtering which takes away all the junk form th tweets and then analyses all the tweets based on the java code written in the algorithm developed manually, so all the analysis will be done using .java class files.

- Database (MySQL) – The database is used to store the count of the sentiments of the tweets, but not the tweets, and this is used for one of the features(history of a keyword).

- Display of Results – The results will be displayed on the browser using CSS and HTML for the text and a pie chart shows the analysis, which is made using java script.

- Server on browser – All the html and css code is embedded in to .jsp files which should be run on the server which is Apache Tomcat 7.0

Table Of Contents

Table of Contents

1 Introduction

It's a pretty safe bet to assume that just about everyone over the age of seven has a Facebook account, or a Twitter account. Social media has become an integral part of the social landscape, it can be an extremely powerful networking tool on anything ranging from podcasts, businesses, schools, etc., can effectively utilize social media to promote their product or promote upcoming events they have going on.

There are many social networking sites, Twitter among them is very popular where the user can upload some text called as Tweet by Twitter, with a maximum length of 140 characters, that text can say his mood or his assumptions or his activities or his opinions or his reviews and could be anything there can be people who can follow him and they can re-tweet the same tweet he posted, that means that tweet is again posted by the other user. Twitter is very popular that amost 500 million tweets will be generated per day on ana average. So it's a very famous and powerful platform where one can communicate with the world.

Twitter is used widely, analyzing the tweets of the particular product can serve the company as product reviews by the users, so it can help a business measure overall consumer attitudes and customer satisfaction, and it also serves to provide a warning when there is a sudden change in sentiment. Sentiment Analysis is referred to (Maite Taboada, 2010) as the general method of extracting subjectivity and polarity from a text resource.

The advantage of Twitter analysis is that it allows its scaling and handling. Analysis can be based on areas, type of users and references included in the text messages, thus we can target all the features in this way, also Twitter analysis with a good accuracy rate could make a competitive advantage to many business organisations and is very important in decision making processes.

In this project I address the problem of classifying the sentiment in posts from Twitter. The tweets can be analyzed by retrieving posts, preprocessing on the data, and analyzing it using various techniques to tell it's polarity as it is either neutral, positive, or negative. I have investigated various techniques for this analysis and then found the Lexicon based approach is the best among all.

2 Design

After researching the methods available for performing a sentiment analysis on Twitter, I chose to use a lexicon-based approach and to develop my **own algorithm** for a lexicon classification of tweets. A lexicon classifier could provide better accuracy rates for different domain types, while machine-learning usually provide good results only on the specific domains they are trained for.

For the implementation of the Twitter Sentiment Analysis tool, I have completed the following requirements of such a project: I have implemented a Twitter scrapping tool, I have developed a lexicon text classifier, that creates methods for pre-processing the text and analysing it,, I have implemented a database for storing the results of a search, I have developed a web interface so that users could access the tool and perform searches, and I have introduced charts for a better representation and understanding of the results. Each of these steps will be described in the following sections.

The project has been developed using Java, the web interface has been created in JSP and the charts have been produced within JavaScript.

2.1 Twitter4J setup

The main purpose of this project is to analyze the tweets, for that purpose we need to get the tweets based on a keyword entered, so for this, I am using an in built api in java called **Twitter4j,** which is written in java already for us, so we should download it and then import it to the current library using. This library also provides various other methods where w can set language, get tweets according to a geo location etc.

As my files are all on English, the language I can analyze is only English, so while retrieving the tweets, we should be careful that only English language tweets are fetched. so can make the tweets to be only from English using the following method

```
query.setLang("en");
```

Also while displaying and analyzing the tweets there should be a specific number which all the keywords should be followed, I took the number as 100, such that 100 tweets are retrieved which makes the analysis some strong information to show, we set the number of tweets to be fetched as 100 by the following method

```
query.setCount(100);
```

The main problem with this api is that it gives lot of re-tweets, i.e the same tweets again, the tweet which is re tweeted by a person is considered as a different tweet and that will also be given, those re tweets problem can be solved to some extent by using " -RT" appended to the keyword.

2.2 Making the Lexicon files

To make the Lexicon files, I have got the list of all the English words from the following address http://www-01.sil.org/linguistics/wordlists/english/wordlist/wordsEn.txt, and then modified the code of the senti-strength api (which gives the score of a word) such that the scores and the words will be sent to a new file with a tab space between them, also the words with score 0 are neglected because they don't carry any sentiment, this way we can get rid of a number of words in English with out storing them, so if I don't find a word in my lists, they are given a score 0, this reduces redundancy. this file is named as posAndNeg.txt, because those words carry either positive or negative sentiment and all the other text files are fetched through internet.

The algorithm separates each word of a tweet using String-tokenizer and then checks for the words which are stored in **maps** data structure (key) and then sums up their respective scores (value) from the map, and the final total score of all the sentiments of the words is used to tell if

the tweet is positive or negative or neutral. Positive words are those whose score is greater than 0, negative are those whose score is less than 0 and neutral are those whose score is 0.

I have used six different files for this project, they are

- Positive and negative
- Multipliers
- Emoticons
- Negations
- Short cuts

- Special chars

2.2.1 Positive and Negative words

Positive and negative words list will contain the list of all the English words which has some sentiment to them, say positive word has a positive sentiment and negative word has a negative sentiment(excluding neutral words which doesn't have any sentiment), so when ever the tweet contains any of these words, the score of the word will be added to the total score.

Table 1. Few examples from Positive and negative words file

Word	Score
accuse	-2
attract	1
hate	-4
kind	2

Example : The tweet, "I hate you", is calculated 0 + (-4) + 0 = -4, which is negative in sentiment.

2.2.2 Multipliers

Multipliers are the words which multiplies or intensifies the next word by a certain degree, say 'incredibly' multiplies the next word by 2 times, where as 'very' multiplies the next word by only 1.25 times, so if good would get 3.0 as the score, where as very good will get 3.75, and if we get two intensifiers next to each other, the multipliers will be added to each other and then multiplied with the next word, so for very very good will multiply (1.25 + 1.25) * 3.0, so if we have two intensifiers next to each other, they make more impact.

Table 2. Few examples from multipliers words file

Word	Score
extremely	2
extra	1.25

absolutely	1.5
some	-0.75

Example : "This is extremely bad" multiplies the intensity of the word 0 + 0 + 2*(-2) = -4

2.2.3 Emoticons

Emoticons are used widely now a days for text communication in all the social networking sutes and also in normal messages, so I got a list of emoticons and their scores next to it and then stored them in a map, these scores are also added up to the total score and plays a major role when some one types a text with a lot of spelling mistakes, these could identify his mood easily and helps in evaluating the sentiment.

Table 3. Few examples from emoticons words file

Word	Score
:)	1
:D	2
:(	-1
:P	1
(:	1

The emoticons now a days are used graphically, all when a graphical emoticon is sent for analysing, it is even converted as it's appropriate symbol in the file automatically.

Example : "I am nat haapppy with his behavuior :(" will be analysed 0 + 0 + 0 +.. + -1 = -1.

Thus when some one makes a spelling mistake, or some grammatical mistakes, these emoticons play a major role in analysing the text.

2.2.3 Negations

Negations are a set of words which change the polarity of the text, so when ever I get a negation, the polarity of the next word is changed, so the word next to negation becomes positive if it's negative and negative if it's positive.

Table 4. Few examples from emoticons words file

Word

don't
isn't
never
can't

Example : "This book isn't good" gets 0 + 0 + -(3) = -3

2.2.5 Short cuts

Short cut words are now widely used all over the world in text communication, when ever people something funny, the first word which comes in to most of the people's mind is "lol", which means "laughing out loudly", so people found so much easier to use short cut words instead of typing lengthy meanings, but for analysing, those short cuts makes no sense as they mean nothing according to english words, so I have made a list of short cut words with their abbreviations nxt to them, and when ever the word is from that list, they will be replaced by the appropriate word and then the abbreviation is sent for analysis.

Table 5. Few examples from shortcut words file

Word	**abbreviation**
lol	laughing out loudly
rofl	rolling on the floor laughing
ttyl	talk to you later
asap	as soon as possible
idk	I don't know

Example : "lol.... I didn't go there" will actually be analysed as "laughing out loudly.... I didn't go there" and will be given a score of 2.0, which is positive.

2.2.6 Special Chars

Now a days people started using special characters in between words although they mean the same with those characters substituted, people are replacing with special characters which looks similar to the replaced characters, say to write "Hello", people are using "H€llo" instead. So, to get rid of these special characters, I am replacing them with their appropriate characters only if they are present in between words.

Table 6. Few examples from special chars words file

special character	**appropriate character**

@	a
€	e
&	and
$	s
!	i

Example : "I am H@ppy on your re$ult" will be analysed as "I am Happy on your result", and will be givena score of 3.0 which is positive.

2.2.7 Longitudes and latitudes of countries.

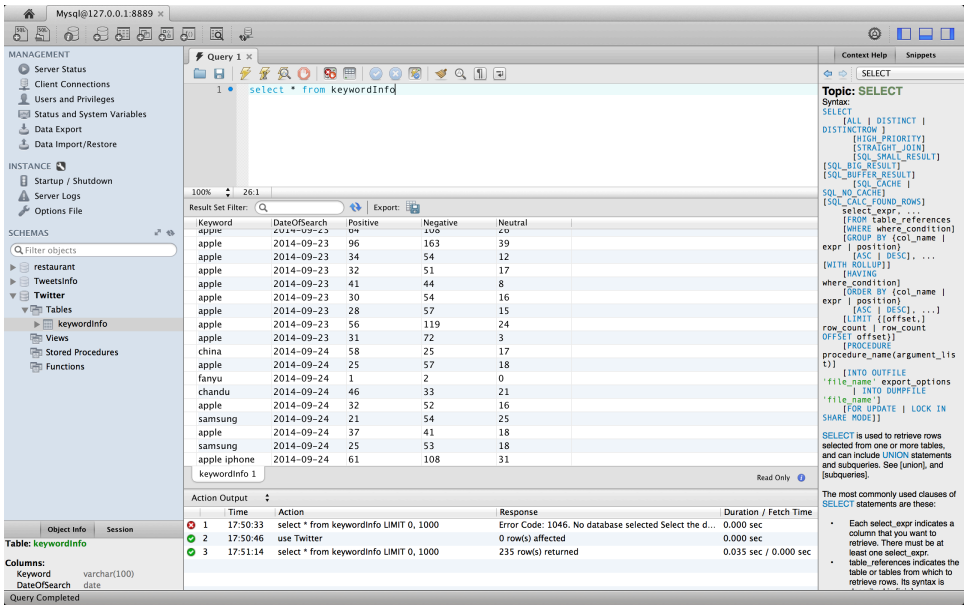
One of my features fetches the tweets based on the country, so I made a file with the list of all the countries in the world, and there is an in built method in twitter4j api, which fetches the tweets based on the location when the longitude and latitude of a place are passed to it, so I made this file, which is used to give the longitude and latitude of the user entered country, and these values are then passed to the in built methods of the api to get the tweets based on the country asked for.

Table 7. Few examples from countries file

Country	Latitude	Longitude
Ireland	55.32443216228089	-9.602050778750026
India	20.593684	78.96288
Germany	51.165691	10.451526
United States of America	37.09024	95.712891

2.3 Database for history of the sentiments

One of my features gets sentiment history of the given key word with in 7 days, for this purpose, I need to store the sentiments of the keyword in a database, so I am storing the keyword, it's number of positive, negative, neutral tweets and the day of search, if the tweet is searched more than once in a day, I will display the average of the scores, so for all these functions, I need a database, I made mysql database, and connected it to my java files, using JDBC connection. I have named the database as Twitter and I need only one table for all these functions, the table is named keywordInfo.



The above picture shows the database and the table, with it's values in it.

3 Implementation

To develop the sentimental analysis tool, I went through a lot of ways that I could implement my algorithm based on, finally I found lexicon approach as most accurate and also challenging, I used Java as my programming language over all, on the server side and also on the client side.

3.1 Initialising Maps :

Firstly, I made all the files with words and their scores next to them as explained above, after getting all the files, I stored them in a **HashMap,** which could if themap cntains a word oand and also can get a word with **O(1)** time complexity, which is very efficient to use. I stores all the words on my desktop and then sent them to different maps based on the files with the following code.

```
private static Map<String, Integer> emoticons;

private static Map<String, Double> multipliers;

private static Map<String, Integer> posAndNeg;

private static Map<String, String> shortCuts;

private static Map<String, String> specialChars;

private static Map<String, String[]> countries;

private static List<String> negations;

public static Map<String, Integer> getEmoticonMap()

        throws FileNotFoundException, IOException {
```

```
            if (emoticons == null) {
                    emoticons = new HashMap<String, Integer>();
                    InputStream in = new FileInputStream(new File(
                    "/Users/KeerthiChandu/Desktop/TwitterFile/emoticons.txt"));
                    BufferedReader reader = new BufferedReader(
                    new InputStreamReader(in));
                    String line;
                    while ((line = reader.readLine()) != null) {
                            String[] s = line.split("\t");
                            emoticons.put(s[0],
                            Integer.parseInt(s[1]));
                    }
            reader.close();
            }
            return emoticons;
    }
```

The above code initializes the emoticons map (initialisation is done only once, because of the 'if' condition, this improves efficiency), all the words and the scores in the file are seperated exactly by a **tab** space, so the String.split() method splits and then stores them in to a map. All the maps are made **private** because the data should be secured, all the maps should be accessed by get methods, in this way I am **restricting** others to change the content in the map.

3.2 Remove Hyperlinks

So, now I have my list of words with their sentiments next to them, now I have to to send the tweets for analysing, so the tweets should be sent word by word, but before sending the tweets for analystion, we need to do some **pre processing.** It is because, the tweets contain hyperlinks, and lot of junk characters, they should be filtered, the hyper links can be filtered by the follwing code,

```
public void removeHyperLink() {
    String[] hyperlinks = { "www.", "https://", "ssh://", "ftp://",
    "http://" };
    for (String index : hyperlinks) {
            while (Tweet.contains(index)) {
```

```
                StringBuilder sb = new StringBuilder(0);
                for (int i = Tweet.indexOf(index); i < Tweet.length(); i++)
                        sb.append(Tweet.charAt(i));
                        if (Tweet.charAt(i) == ' ')
                                break;
                }
            Tweet = Tweet.replace(sb, "");
        }
}
```

The above code, takes the tweet as a String and finds if the tweet contains any of those formats, where most of the hyperlinks starts with, and then once they find them, they will replace that hyperlink link with empty string, so the hyper link will be removed from the String.

3.3 Analyse Emoticons :

Analysing emoticons should be done before analysing the words, because in later methods, I am sending the tweet to remove junk, which takes away all the characters other than alphabets, so the emoticons can be analysed by the following code.

```
        public void analyseEmoticons() throws FileNotFoundException, IOException
{
            StringTokenizer str = new StringTokenizer(Tweet);
            while (str.hasMoreTokens()) {
                String s = str.nextToken();
                if (maps.getEmoticonMap().containsKey(s)) {
                        score = score + maps.getEmoticonMap().get(s);
                }
            }
        }
```

So the above method, takes the tweet as a string and then tokenizes it, which seperates the whole tweet in to group of words, each time of the loop, one word is given and then if the word is contained in the emoticons map, the scored will be added to the total score of the tweet.

3.4 Replace Special Chars

Now a days people started using special characters in between words although they mean the same with those characters substituted, people are replacing with special characters which looks similar to the replaced characters, say to write "Hello", people are using "H€llo" istead. So those special charaters should be replaced with ther appropriate ones only if they are a part of the word. The following code does that

```
public void replaceSpecialChar() throws FileNotFoundException, IOException {

	for (String s : maps.getSpecialCharsMap().keySet()) {

		if (Tweet.contains(s)) {

			while (Tweet.indexOf(s) != Tweet.lastIndexOf(s)) {

				if (Tweet.indexOf(s) > 0

				&& Tweet.indexOf(s) < Tweet.length() - 1) {

					String s1 = removeJunk(Tweet

					.charAt(Tweet.indexOf(s) - 1) + "");

					String s2 = removeJunk(Tweet

					.charAt(Tweet.indexOf(s) + 1) + "");

					if (s1.length() == 1 && s2.length() == 1) {

						Tweet = Tweet.replace(s,
					maps.getSpecialCharsMap()

					.get(s));

					}

					else{

						Tweet = Tweet.replace(s, "");

					}

				}

				else{

					Tweet = Tweet.replace(s, "");

				}

			}

		if (Tweet.indexOf(s) > 0

		&& Tweet.indexOf(s) < Tweet.length() - 1) {

			String s1 = removeJunk(Tweet
```

```
                .charAt(Tweet.indexOf(s) - 1) + "");

                String s2 = removeJunk(Tweet

                .charAt(Tweet.indexOf(s) + 1) + "");

                if (s1.length() == 1 && s2.length() == 1) {

                        Tweet = Tweet.replace(s, maps.getSpecialCharsMap()

                                .get(s));

                }

                else{

                        Tweet = Tweet.replace(s, "");

                }

        }

    }

}
```

The above code takes the tweet and then checks if they contain special chars, and if so, they would also check if they are the part of a word or not, and replaces only if they are a part of the word.

3.5 Remove Junk characters

Also, the tweet can have lot of junk characters, which will be not relevant to the context there, say "I am getting,,,,,.//.../../.. bored$£%*(((don't know--- why^&", so to express the tweet, a person could type a lot of other characters which is not related to the tweet, they should be removed, also these days, people are used for hashtags, which start with # and also to refer others, people started using @, they should be removed while analysing the tweet. They can be removed by the following code. The main point to notice here is this method should be done after going through all the above methods, the order should be followed, because emoticons and special characters contain the characters whcih are not alphabets, so if this method is done before, the above methods will be of no use.

```
    public String removeJunk(String Tweet) {

        StringBuilder sb = new StringBuilder(0);

        char[] words = Tweet.toCharArray();

        for (char letter : words) {

            if ((letter > 64 && letter < 91) || (letter > 96 && letter<
        123) || (letter == ' ')) {

                sb.append(letter);

            }
```

```
		}

		String s = sb.toString();

		return s;

	}
```

The above code, takes a tweet in the form of a String as an input and then takes away all the characters which not alphabets, because all the words what I have in my files are alphabets.

3.6 Remove repeated characters

People these days are using the letters inside the words which repeats more than once, to show that they are more happy people are using "happyyyyyyyyyyy" instead of "happy", so to solve these kind of problems, I have made a method which takes the word and then makes the letter which is repeated for more than two times continuesly, will be replaced with only one character of it, as in any of the english words, there is no word that has more than two letters which repeats consecutively, so the following code is based on that.

```
public String removeRepeats(String word) {

	int count = 0;

	StringBuilder sb = new StringBuilder(0);

	for (int i = 0; i < word.length() - 1; i++) {

		if (word.charAt(i) == word.charAt(i + 1)) {

			count++;

		} else {

			if (count > 1)

				sb.deleteCharAt(sb.length() - 1);

			count = 0;

		}

		if (count <= 1) {

			sb.append(word.charAt(i));

		}

	}

	if (!(word.charAt(word.length() - 2) == word.charAt(word.length() - 1) &&
word.charAt(word.length() - 3) == word.charAt(word.length() - 1)))

		sb.append(word.charAt(word.length() - 1));

		String s = sb.toString();
```

```
            return s;

        }
```

3.7 Scoring the words in the Tweet :

Now the tweet is passed in for analysis which has a series of if and else conditions, so if the twee is found in a map, it doesn't enter in to any of the other blocks, the code is written below,

```
public double process() throws IOException {

        StringTokenizer word = new StringTokenizer(Tweet);

        int count = 0;

        double intensity = 0.0;

        int polarity = 1;

        StringBuilder str = new StringBuilder(0);

        int flag = 0;

        while (word.hasMoreElements()) {

                String nextword = word.nextToken();

                if (maps.getIntensifierMap().containsKey(nextword)) {

                        intensity             =             intensity             +
                maps.getIntensifierMap().get(nextword);

                        count++;

                } else if (maps.getPosAndNegMap().containsKey(nextword)) {

                        if (count == 0) {

                                score = score + maps.getPosAndNegMap().get(nextword)

                                * polarity;

                                polarity = 1;

                        } else {

                                score = score

                                + (intensity * maps.getPosAndNegMap().get(nextword));

                }

                } else if (maps.getNegationList().contains(nextword)) {

                        polarity = polarity * -1;

                } else if (maps.getAbbreviationsMap().containsKey(nextword)) {
```

```
                        if (flag == 0) {

                                str.append(maps.getAbbreviationsMap().get(nextword) +
                        " ");

                                if (word.countTokens() == 0) {

                                        word = new StringTokenizer(str.toString());

                                        flag = 1;

                                }

                        }

                } else if (nextword.equals("but")) {

                        score = 0.0;

                }

        }

        return score;

}
```

Now the above code analyses each word, gives their score, changes their **polarity** if there is a **negation** word next to it, multiplies with the intensifier if there is a **multiplier** word next to it, also makes the score to 0 when it finds a **but** and then analyses the rest of the words after "but".

"BUT" word is the special case for analysing the words, "but" in english sentence normally changes the meaning of the word till then and starts a new meaning to it, "I thought he is an idiot but, he looked so honest today ." so if that sentence is sent for analysis, all the score made till but is made 0 and then new score starts from the words after "but", thus makes the tweet positive.

3.8 Fetching tweets from Twitter

So, with those above methods, files, and maps my algorithm is finished, now I need to get tweets from the twitter and then those should be sent to this algorithm for analysis. To get Tweets from Twitter, I used an api called Twitter4j, after importing the Twitter4j library by downloading .jar files from internet, I made the following code to get the tweets from Twitter.

```
Twitter twitter = tf.getInstance();

if (keyword.length() < 1) {

        System.out.println("Please enter your tweet");

        System.exit(-1);

}

try {

        Query query = new Query(keyword + " -RT");
```

```
        query.setLang("en");

        if(latitude != 1111111111.0 && longitude != 1111111111.0)

               query.setGeoCode(new   GeoLocation(latitude,   longitude),   1000,
        "km");

        QueryResult result;

        query.setCount(100);

        result = twitter.search(query);

        List<Status> tweets = result.getTweets();

        return tweets;

}
```

After authorising with the twitter, by filling all the credentials which will be sent to our mail when we authorise through twitter developers, the above code is written, which gets the keyword from the search bar in .jsp file and is sent for analysing, **.setcount()** method decides the number of tweets to be retrieved, my default st of tweets are 100, we can change it to our required number and then the Tweets will be sent in to an ArrayList and is sent to the above algorith for analysis. There is another method called **.setGeoCode()** which gets the tweets based on the longitude and latitude, as one of my features fetches the tweets based on a country, that method is called only if that featured is accessed, if not the tweets are got from all over the world. There is another method called **.setLang()** which sets the language to English, because my algorithm can only analyse English words.

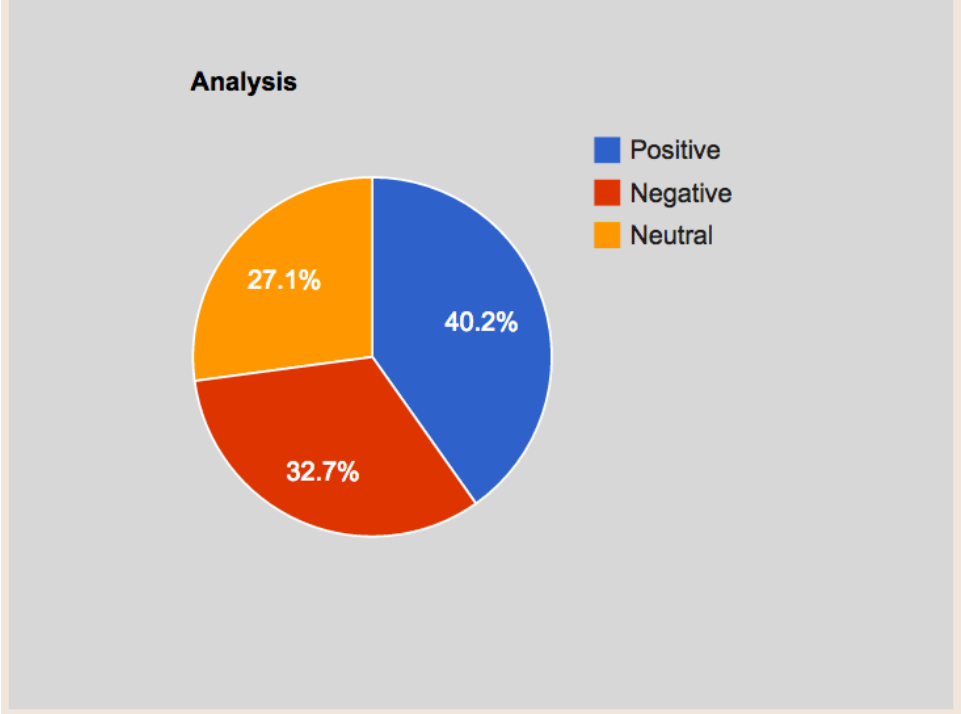
3.9 Creating a Pie Chart :

Once the analysis is done for the tweets, their count of sentiments say, positive, negative and neutral counts are sent in to a java script code which is called **Google Charts,** which makes a pie chart with the set of parameters passed. The code is given below.

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>

    <script type="text/javascript">

       google.load('visualization', '1.0', {'packages':['corechart']});

       google.setOnLoadCallback(drawChart);

       function drawChart() {

              var data = new google.visualization.DataTable();

              data.addColumn('string', 'Topping');

              data.addColumn('number', 'Slices');

              data.addRows([

              ['Positive', <%=pos%>],
```

```
                ['Negative', <%=neg%>],

                 ['Neutral', <%=neu%>]]);

                var options = {'title':'Analysis',

                'width':400,

                'height':300,

                backgroundColor: '#D8D8D8',

        };

 var                    chart                    =                    new
google.visualization.PieChart(document.getElementById('chart_div'));

          chart.draw(data, options);

        }

    </script>
```

So the above code makes a pie chart based on the values, sample figure of a pie chart when analysed looks like this,



3.10 Storing in a Database :

After the tweets are evaluated, they are sent in to java script code to make pie charts, and also the scores of the keyword are stored in the database, the values are further used for getting history of the sentiments of the tweets. To establish a database connection, I have made a class called **DatabaseConnection.java** and made all the processing there, that class has two methods which can store the data in to the table and one more method which gets the history of the tweets which will be explained later.

```
myStatement.executeUpdate("INSERT INTO keywordInfo values('" + keyword + "','"+
dateFormat.format(date)    +  "','"  +  positive  +  "','"  +  neutral  +  "','"  +
negative + "');");
```

Current date of the day can be got by importing Date class in java and then executing the above statement puts in the postive, negative and neutral scores of a keyword in to the database.

3.11 Implementing the code on server :

To call the above java files, on to the server, I have used **.JSP** files (Java Server Page files), which is has java code between HTML and CSS code, all the CSS styles are written in file called styles.css and then used for all the JSP files, the server used is Apache Tom Cat 7.0.

The homepage is named TwitterEvaluation.jsp, which has a serach bar which takes in the keyword and gets the relavent tweets with the pie chart based on their scores. On the left hand side, we also have some extra features which can be used through this page. The page when run on the server looks like this

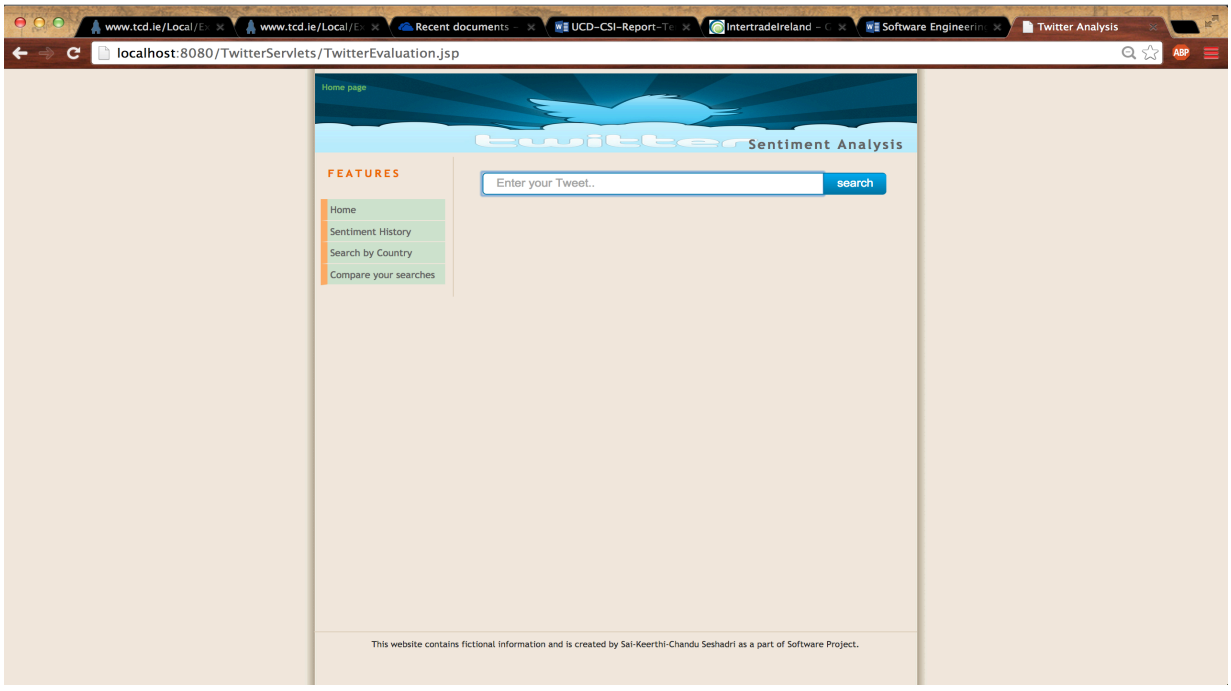


Figure 1. The home page of the Twitter analysis

So, now the user should enter the tweet and the hit the search button, once the search button is pressed, the java files which are explained above are run in the background and then sent the

scores are then sent to the java script file which then shows the pie chart of the sentiments and also the tweets are displayed with different colours, the purpose being, the positive tweets are displayed blue in colour, where are the negative tweets are displayed in red colour and the neutral tweets are displayed in orange colour. The below is the example when the keyword "macbook" is searched.

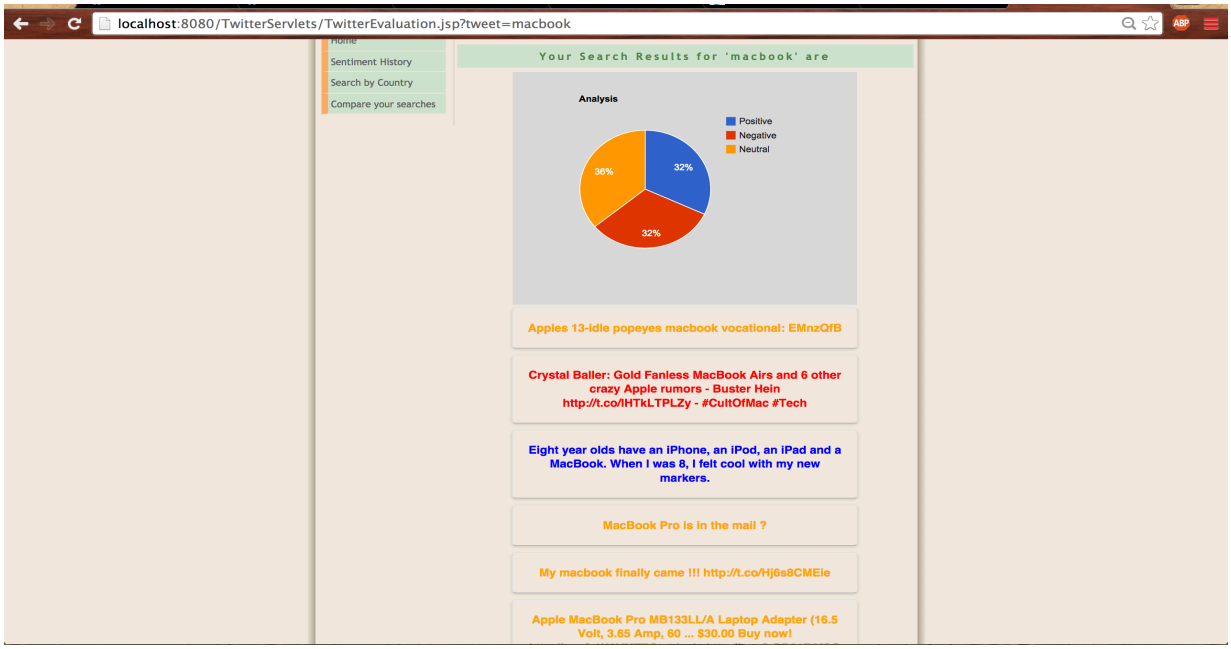


Figure 2. Search results for "macbook"

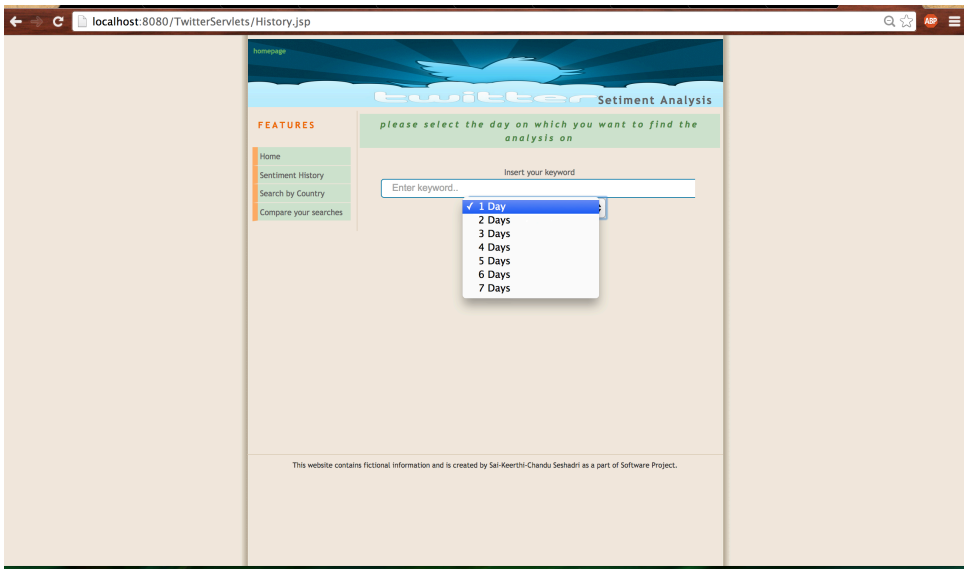
4 Extra Features Added

In addition to the functions asked to perform, I have additionally added three more useful features which I thought could be handy to the users, all the features can be accessed on the left hand side of the page which are under the column called "Features", the features I have added are

- **Sentiment History**
- **Search by Country**
- **Comparison of Searches**

4.1 Sentiment History

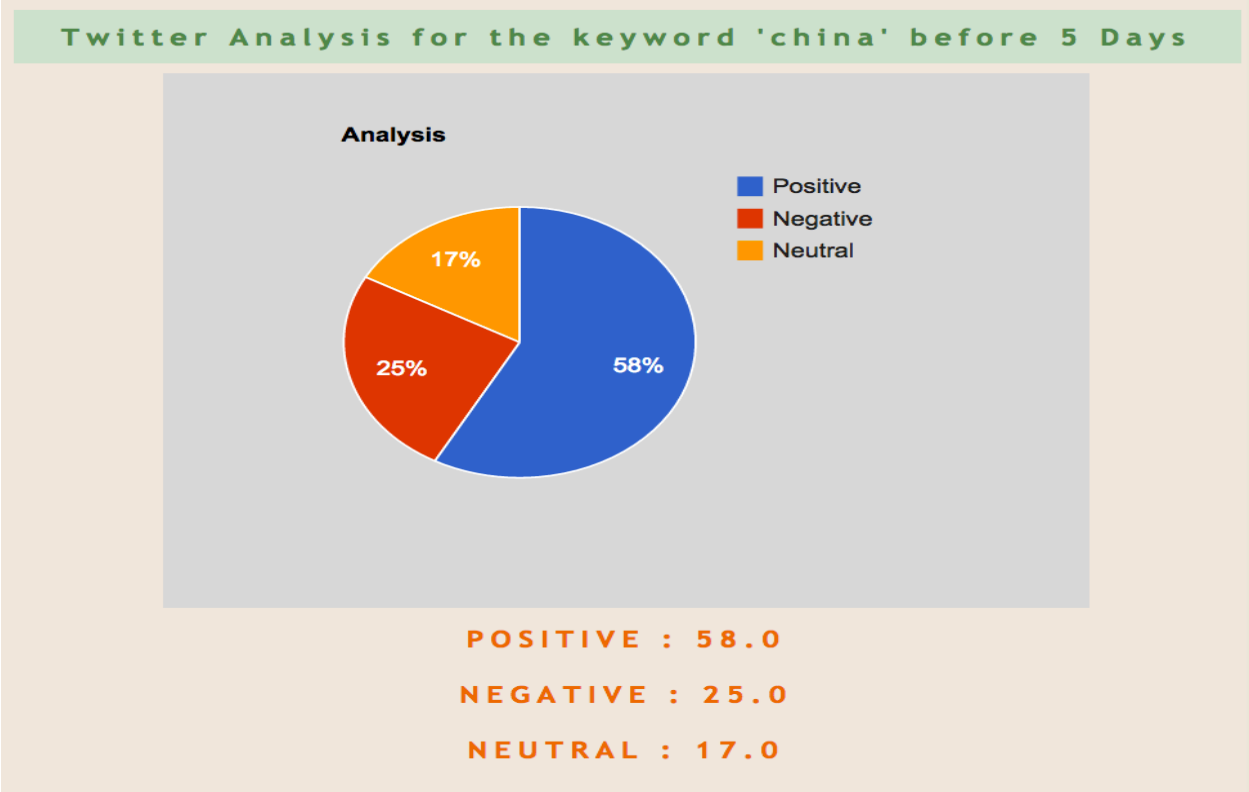
Sentiment History is a feature which gets the sentiments of the tweets before the amount of days asked by the user, as Twitter allows to store the tweets only for 7 days legally, the maximum days they can go back and search for is 7 days (I could have showed for more than 7 days before because I am not actually storing the tweets, just their sentiments, but for legal purposes, I am displaying it 7), so in this feature, the user selects a day from the drop down list provided which gets him the sentiments of the keyword entered before the number of days selected



so we have to enter the keyword and select the number of days as shown above, and hit submit, then the database history method will be called which gets the average of the positives, negatives and also neutral tweets. Average is found because if the user searches for a keyword for more than once in a day, then to display the two set of tweets, I am finding average of the two sets, as I have made the count to be 100, I will be finding average to the same number of tweets.

```
int today = cal.get(Calendar.DAY_OF_MONTH);

day = today - day;

String sql = "SELECT AVG(Positive),AVG(Negative),AVG(Neutral) FROM keywordInfo
WHERE Keyword='" + keyword.toLowerCase() + "' AND DateOfSearch='" +
dateFormat.format(date) + "" + day + "';";

myResultSet = myStatement.executeQuery(sql);

int i = 0;

while(myResultSet.next()){

	scores[i++] = myResultSet.getDouble("AVG(Positive)");

	scores[i++] = myResultSet.getDouble("AVG(Negative)");

	scores[i] = myResultSet.getDouble("AVG(Neutral)");

}

myConnection.close();
```

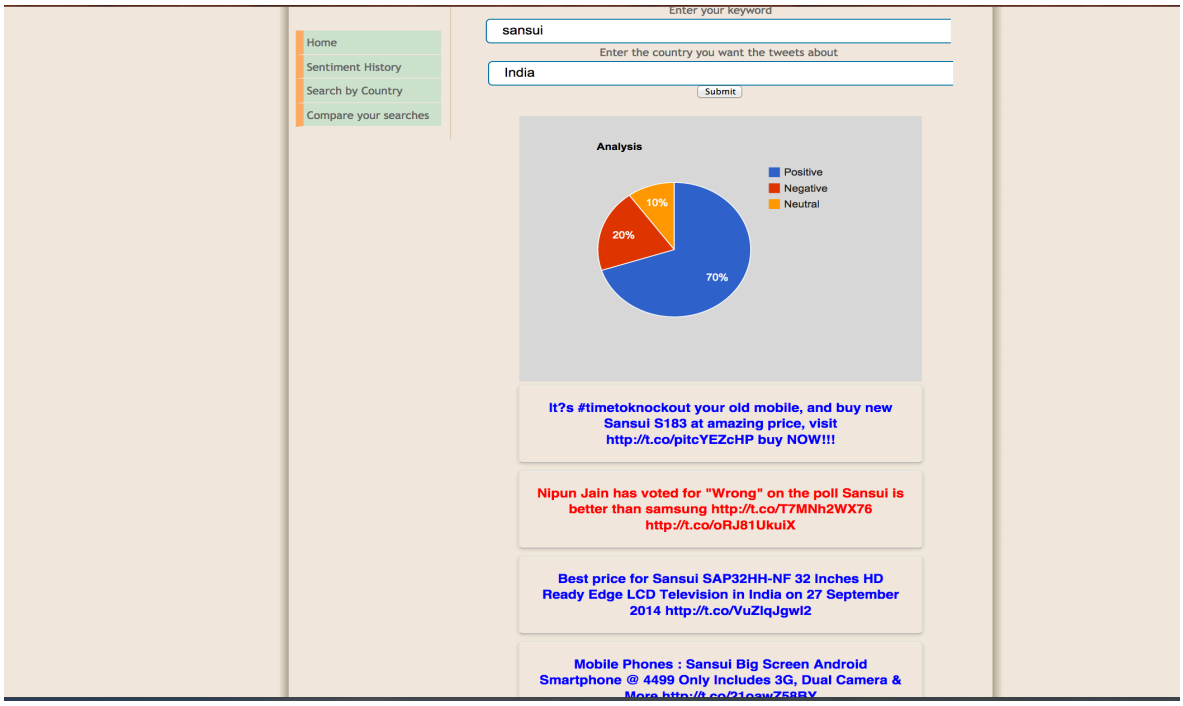
So the above code takes the day and also fetches the average of the scores of the given keyword asked for a particular date. If it doesn't find any keyword in the list, it gives an error message saying "sorry we don't have any searches for the keyword on the date asked", if it finds a keyword in the database, then it shoes a pie chart showing the results before the days asked.



So, the above is the analysis for the keyword China before 5 days. This feature could be more handy for the people to know about the people's sentiments on some events, say **Iphone 6** has got a lot of **positive** tweets till it got a negative review that is getting **bent** in pockets, so this would be handy to search how are the sentiments varying for the people **before** it got bent and after the news is out.

4.2 Search by Country

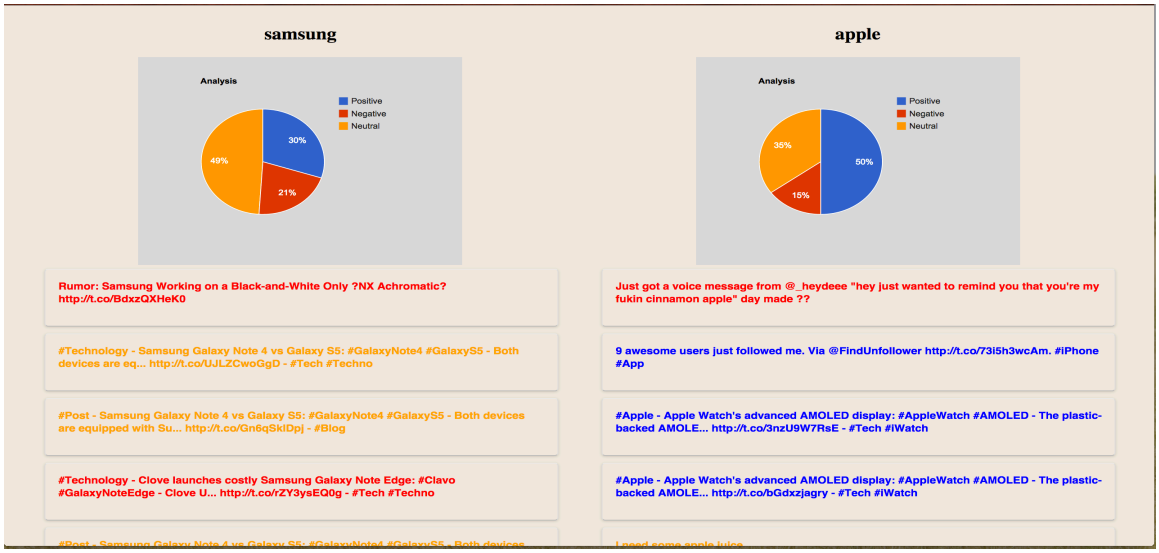
Search by country asks the user to enter the country what they want to get the tweets related to, and also the keyword which they want the tweets on, so after entering those values, once hitting the submit button, tweets related to the entered country will be retrieved, this is done with a file with the lsit of countries and their longitudes and latitudes, so they will be passed to the in built method in Twitter4j api called seGeoCode() and then gets the tweets based on the country.



The above results are for a electronic brand named **Sansui** which is popular only in India, so the items which are popular at a certain place can be searched from this also this is useful to determine the sentiments of the people about the particular product say Iphone, it could get different sentiments from France when compared to China, so that it could improve the company's focus on them

4.3 Comparison of Searches

This feature compares the tweets of the two keywords entered, there will be two search bars provided to enter the keywords to compare and the compared results are shown in a new jsp file, with their pie charts and the tweets under them.



So, this feature is very useful when a user wants to search the sentiments of the two different tweets, the major mobile companies Samsung and Apple were searched where apple has got more positive tweets in the above example. So this helps the users and also to the companies which makes use this project to compare their products with several other products.

The validation of the results for this sentiment analysis tool has been done by hand, but only a small set of data has been validated, and it would represent

The accuracy rate of this sentiment analysis tool has not been calculated, but the developed sentiment analysis tool shows similar results when compared to other sentiment analysis online tools, such as sentiment140.

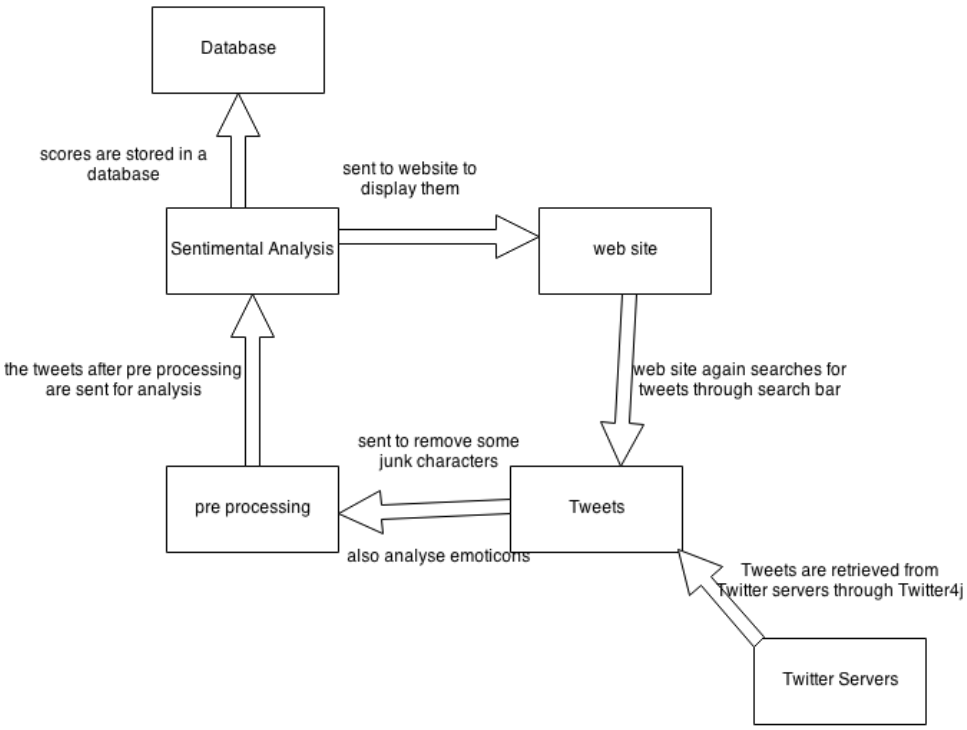
Introduction/background - e.g. background on twitter, sentiment analysis, text analysis, graphing of data, etc. Similar to your inital report.

Design and Implementation - self descriptive.

Results - Any analysis of the output of your project such as number of false negative sentiment, speed and efficiency, etc

5. System Architecture

The below diagram represents the system architecture of the sentimental analysis project, so the input is taken from the user through the website and then calls the java classes which gets the tweets from the Twitter servers using an api called Twitter4j and then after the tweets are retrieved, the tweets are then sent for preprocessing where in all the emoticons, special characters, repeated characters etc problems are solved and also the junk characters are removed. After the pre processing step, the tweets are then sent for analysis, after analysis, they are sent to the website to display the user and also simultaneously stored in a database with the date of search.



6. Technology used

The only language used for the whole project is JAVA on both front end and also on the back end of the project, which made the code work very **fast** when **compared** to the other programming languages like **php**, **python** where apis are used.

So, all the algorithm was written in **.java** class files, and all the server side scriptinhg is written in **.jsp** files (Java Server Page), in which java code is written between html code, and a lot of CSS is also used, which is used from a separate file called styles.css. The server used to run the .jsp files is Apache Tomcat 7.0.

To store the positive , negative, and neutral scores of every keyword, the database is used, it was MYSQL database server which is linked in to .java files using JDBC connection.

The pie charts are generated using java script code which is written inside .jsp files, which are popularly known as Google charts.

Problems Encountered

7. Problems encountered

There are a lot of problems encountered while working on this, but they rather looked as challenges, the main problem which had a lot of time of mine was that the Eclipse can't recognize the jar files until you manually right click on a folder and paste them there, and also it took some time to import them.

There were some problems with the html and css, the alignment of the tweets were a problem as they were generated dynamically and also placing the pie chart with the generated dynamic data.

Another problem was there is a limit for the twitter to fetch tweets, and you need a paid account to get more number of tweets, which makes the analysis very narrow, so if the number of tweets fetched increases, the accuracy of the analysis would have increased more.

8. Conclusions

This analysis is really useful for all the organizations who wants to know the user attitude towards the product, and this is more challenging, lot more interesting with a lot of research work needed.

With this project I have made my own algorithm which can fetch the tweets, filter them, and analyse them to get the sentiments of them in three forms. The approach followed was Lexicon which was proved that works faster when compared to other apis and also accurate.

I have also provided three more features that could be very handy and useful for the users while dealing with this project

9. Future Work

The future work on this project could be increasing accuracy, although Lexicon methods work well with the analysis, they are not as good as Naïve bayes approach in terms of accuracy, so I would start searching for some more slang words which differ from region to region, example "bollox" is a regional word for Ireland, but that word can't be found in the dictionary.

When coming to countries, I gave the radius of all the countries as 1000, which differs from country to country, so I could make it better to have some countries with groups based on radius, say sets of 10,000 , 5000 etc. Also this project could analyze only tweets from English language, so this could be improved by putting different sets of files from other languages and improving the analysis tool.

The code I used could be made more efficiently than I made there, there is some redundancy in the code with maps initialization, CSS etc, that should taken care, which improves the speed of the code.

10. Bibilography

Maite Tabaoda, Julian Brooke, Milan Tofiloski, Kimberly Voll, Manfred Stede. 2014.

Lexicon Based Methods for Sentiment Analysis

http://www.sfu.ca/~mtaboada/docs/Taboada_etal_SO-CAL.pdf

Prabu Palanisamy, Vineet Yadav, Harsha Elchuri. 2013.

Simple and Practical Lexicon Based Approach to Sentiment Analysis

http://www.cs.york.ac.uk/semeval-2013/accepted/99_Paper.pdf

Michael Clark. 2014.

An Introduction to Machine Learning with Applications in R

http://www3.nd.edu/~mclark19/learn/ML.pdf

Blinov, Klelovkina, Kotelnikov, Pestov. 2012.

Research of Lexical Approach and Machine Learning Methods for Sentiment Analysis

http://www.dialog-21.ru/digests/dialog2013/materials/pdf/BlinovPD.pdf

Kang, H., Joon Yoo, S. and Han, D. (2012) 'Senti-lexicon and improved Naïve Bayes algorithms for

http://www.valpolife.com/business/technology-tools/45464-pnc-social-media-bootcamp-proves-the-importance-of-social-media

sentiment analysis of restaurant reviews' Expert Systems with Applications, 39 (2012).

Liu, B. (2010) 'Sentiment Analysis and Subjectivity' Handbook of Natural Language Processing (2010).

Medhat, W., Hassan, A. and Korashy, H. (2014) ‘Sentiment analysis algorithms and applications: A survey’

Ain Shams Engineering Journal (2014).