Spring 2012

BLOG INFORMATION CLASSIFICATION

Nishant Patel
San Jose State University

BLOG INFORMATION CLASSIFICATION

A Project Report

Presented to

The faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Masters of Computer Science

SAN JOSE STATE UNIVERSITY

By

Nishant Patel

(SJSU ID: 007483099)

May 2012

The Undersigned Project Committee Approves The Project Titled

Blog Information Classification

by

Nishant Patel

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Chris Tseng,	Department of Computer Science	Date
Dr. T. Y. Lin,	Department of Computer Science	Date
Mr. Vineet Kothari,	Sr. Software Engineer, eBay	Date

APPROVED FOR THE UNIVERSITY

Associate Dean,	Office of Graduate Studies and Research	Date

ABSTRACT

Blog Information Classification

by Nishant Patel

Information Classification is the categorization of the huge amount of data in an efficient and useful way. In the current scenario data is growing exponentially due to the rise of internet rich applications. One such source of information is the blogs. Blogs are web logs maintained by their authors that contain information related to a certain topic and also contain authors view about that topic. Micro blogs, on the other hands, are variations of blogs that contain smaller data as compared to blogs. Nevertheless, it also contains rich information.

In this project, Twitter, a micro blogging website has been targeted to gather information on certain trending topics. The information is in the form of tweets. A tweet is a post or an update on status on the Twitter website. These tweets are extracted using Twitter Search APIs. This data is then classified into different classes based on its content. Using the classified data, features are extracted from the tweets and suggestions are given to the users based on the trending topics.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my project advisor Dr. Chris Tseng for his unparalleled guidance, insights and motivation throughout the project. I would also like to thank my committee members – Dr. T. Y. Lin and Mr. Vineet Kothari for providing me their valuable feedback. I would also like to thank my parents and friends for their support and encouragement throughout the Masters program.

Table of Contents

List of Figures

1. Introduction

In information technology, data sets are growing at a very pace. So much so, that the current magnitude of online data is of the order of petabytes, exabytes and zetabytes. Classification of such unstructured data is the need of the hour as it is a very rich source of information. In this project, blogs have been targeted as the source of information as they contain data pertaining to trending topics. More specifically, twitter micro-blogging site is targeted that contains micro-blogs. It is called as a micro-blogging site because there is a limit to number of characters that one can use to tweet. This limit is 140 characters per tweet.

Blog is a collection of articles arranged in reverse chronological order. Blog, in technical term is a collection of text entries but in real world, it means much more than that. People like to communicate and express themselves, but not everyone gets an opportunity and a platform to express them. Blogs came into existence with web 2.0 which gave everyone a chance to put their thoughts online for everyone to read. This not only lets everyone share their thoughts and experience but also plays a very important role in setting up trends. In the last ten years, blogs have played a big role in politics. Several candidates have used blogs to express their views on certain issues and this has had a huge influence on their campaign. "I saw the influence of blogs on the campaign of Howard Dean for president," David Perlmutter, associate dean and professor of journalism at KU, writes on his own blog [1]. Weblogs or blogs started as a place for individuals to put on their personal thoughts or we can say the blogs were a new form of personal journals for people but gradually blogs started influencing people's opinion as well. Because of the Internet, these personal journals or thoughts can reach a large number of people, to which other people react and post comments or start a new thread. The number of readers and writers of the blogs keep on multiplying and during that time, a not very well known topic reaches out to

the whole world. People who are not really fond of writing big articles or maintaining a blog too play an important role. They read blogs and form their own opinions accordingly and hence everyone is influenced in some way. Micro-blogs like Twitter posts or Facebook status messages have brought blogging to a whole new level where any news travels faster than any other medium. Celebrities like Hollywood actors post micro-blogs on Twitter and are followed by millions of fans. A single twitter post about a product or a trend can influence those millions of fans in hours.

Twitter micro-blogs capture information every second (even less). Everyone is using Twitter these days and they seem to quite active on updating the statuses that contain news, trending topics etc. You know, sharing little messages with each other on phones and computers. It turns out that all these little messages, if you look at them all at once, become an easy way to find people, news and trends. If these tweets are captured and structured in totality they serve as an amazing pool of information to deal with. The town would have a resource for knowing what's interesting and who's talking about it. This is what Twitter Search has done for Twitter - it captures and organizes every tweet so that it's possible to find people and specific information in real time [2]. Twitter data keeps on changing at a moment-by-moment rate. It reflects consciousness of the public. This data is an ocean of opinions, emotions, sentiments and other information that is difficult to synthesize in a raw data format. For example, if one uses Twitter to search for ‘iPhone’ then he will get a whole lot of information regarding its usage, features, recommendations, likes, dislikes etc. But this raw tweet data is unstructured and does not provide any direct way of analysis or does not provide any statistics that a user can use to make an opinion about the current trends. To address these limitations of the raw data synthesis, we have

developed an application that classifies the information in the form of tweet data and provides the end user with meaningful and classified data. It also provides trending topics statistically that helps user to make choices between different features.

Blog Information Classification starts with user entering a search keyword in the browser. That keyword is taken as an input on client side and sent to the web service that generates a file of tweets. This web service is written in PHP. It uses Twitter Search API to get all the information the tweets. The Twitter Search API is a dedicated API for running searches against the real-time index of recent Tweets. This information is received as a json object. This json object is parsed to get the textual information of the tweets. Once this information is parsed, it is saved into a file. The file format of the tweet file is not specific. Rather flat files are used to save the tweets onto to persistent medium. Tweets are saved as one tweet per line i.e. all the tweets are newline separated. This tweet file is served as an input to the classifier code.

The classifier module keeps listening to the input file and regularly fetches data for classification. These tweets are then synthesized by the classifier module. They are first broken into tokens and these tokens are processed against the data dictionaries that are specific to the classes. Data dictionaries contain class-specific keywords that tells whether the tokens obtained from the tweets are either objective, positive, subjective negative etc. The classifier module is scalable in terms classes. Any number of classes can be added as data dictionaries that will give the user an advantage of further classifying the data. The dictionaries used in the classifier are in the form of text files that contains machine-learned or trained data sets of the specific class. For example, dictionary for positive class will contain a large number positive keywords or words that have positive sense or sentiment. The baseline classifier used is Naïve Bayes classifier. It

will be further explained in detail in subsequent sections. It classifies the tweets into respective classes and writes it into a comma-separated-value file.

The library used for Naïve Bayes classifier is NLTK. Natural Language Toolkit (NLTK) is an open source program, modules and problem-sets that provide computational linguistics. NLTK covers symbolic and statistical natural language processing, and is interfaced to annotated corpora [3][14]. The prominent features of the NLTK module used are *nltk.classifier* and *nltk.classifier.feature.* The first feature defines the basic data types and interfaces that are used to define the text classification task and the classifier training task. The second one defines data types, classes, and interfaces used for feature-based classification. Features provide a standard way of encoding the information used to make classification decisions. This standard encoding allows the same classifier to be used to solve many different problems.[4] Other features of NLTK are spam-filtering, plagiarism detection/ document similarity, document categorization/ topic detection, phase extraction, smarter search, sentence/word tokenization, text classification etc.

The web service that is present at the middle ware of the application keeps listening to the csv file and once it finds the end of file, it takes all the rows of the csv file and converts it into a json object. This json object is sent to the client side that is received by the client code through AJAX request. The client uses data charting libraries for processing the data and presenting it in user friendly form. These libraries present a very effective and efficient medium of analyzing data.

2. Theoretical Concepts

The underlying purpose of this project is to classify blog data. Classification of the unstructured data can be done with help of a classifier. There are many classifiers available for example, Maximum Entropy Classifier; Kernel based Semi-Naïve Classifier among others. For this project, I have chosen Naïve Bayes Classifier because of its ease of use, simple design and ability to solve complex problems. In this section I will explain in detail about the theoretical concepts and methodology involved in Naïve Bayes Classification. Then, I will illustrate the application of this classifier with a simple example.

Naïve Bayes Classifier

A Naïve Bayes Classifier is based on the concept that the presence of a feature in a class is independent of the existence of any other feature in that class [15]. For example, an eatable orange in color, having round shape and citrus in nature can be considered as orange. It may happen that these features are dependent on each other or dependent on the other features of the same class. But in naïve Bayes classifier all these features independently contribute to the probability that the eatable is orange.

Naïve Bayes classifier can be trained very effectively using proper learning methodologies. It depends on the kind of probabilistic model used. The design of Naïve Bayes model is very simple in terms of complexity. Despite that, it has been proved that it is able to solve an array of complex real world problems. In 2004, analysis of the Bayesian classification problem has shown that there are some theoretical reasons for the apparently unreasonable efficacy of naive Bayes classifiers [5]. For estimating the parameters (variances and means of the variables) only small amount of training data is required. For each class it requires only the variance of the

variables to be determined because independent variables are assumed. It does not require the entire covariance matrix.

The Naïve Bayes probabilistic model of classifier is a conditional model:

$$p(C|F_1,\ldots,F_n)$$

Figure 1 Probability Model for Naïve Bayes Classifier

(source**:** http://en.wikipedia.org/wiki/Naive_Bayes_classifier**)**

where, C = number of classes or outcomes

F_1, $F_{2,\ldots\ldots,}F_n$ = variables for the Features

A large number of features or very large value of the feature makes this probabilistic model infeasible. Therefore, this formula is remodeled into the following:

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\ p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}.$$

Figure 2 Reformulated Probability Model for Naïve Bayes Classifier

(Source**:** http://en.wikipedia.org/wiki/Naive_Bayes_classifier**)**

In layman terms, it can be explained as:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

Figure 3 Reformulated Probability Model for Naïve Bayes Classifier (in simple terms)

Well, the main goal of the classifier is to find the number of classes represented ***C***. Therefore, the denominator of the above formula is not of much interest to us and is mainly constant. We concentrate on the numerator. It can be re-written as the following:

$$p(C, F_1, \ldots, F_n)$$

$$\propto p(C)\ p(F_1, \ldots, F_n|C)$$

$$\propto p(C)\ p(F_1|C)\ p(F_2, \ldots, F_n|C, F_1)$$

$$\propto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3, \ldots, F_n|C, F_1, F_2)$$

$$\propto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ p(F_4, \ldots, F_n|C, F_1, F_2, F_3)$$

$$\propto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ \ldots p(F_n|C, F_1, F_2, F_3, \ldots, F_{n-1}).$$

Figure 4 Applying Chain-Rule on the Probability Model for Naïve Bayes Classifier

(Source**:** http://en.wikipedia.org/wiki/Naive_Bayes_classifier**)**

Now, assuming every feature F_i is independent (conditionally) of all the other features in the classifier. This means that for F_i and F_j, ***j*** is not equal to ***i***.

Also,

$$p(F_i|C, F_j) = p(F_i|C)$$

Figure 5 Probabilities of Fi and Fj are independent

(Source**:** http://en.wikipedia.org/wiki/Naive_Bayes_classifier**)**

Therefore, the joint probabilistic model becomes:

$$p(C, F_1, \ldots, F_n) \propto p(C)\ p(F_1|C)\ p(F_2|C)\ p(F_3|C)\ \cdots$$
$$\propto p(C) \prod_{i=1}^{n} p(F_i|C).$$

$$p(C, F_1, \ldots, F_n) \propto p(C)\ p(F_1|C)\ p(F_2|C)\ p(F_3|C)\ \cdots$$
$$\propto p(C) \prod_{i=1}^{n} p(F_i|C).$$

Figure 6 Joint Probabilistic Model

(Source**:** http://en.wikipedia.org/wiki/Naive_Bayes_classifier**)**

Using the above mentioned assumption of mutual independence of features for class C, the conditional distribution becomes:

$$p(C|F_1, \ldots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C)$$

Figure 7 Conditional Distribution for class C

(Source: http://en.wikipedia.org/wiki/Naive_Bayes_classifier)

In the above formula,

Z= a constant scaling factor called evidence. It depends on features and if features are known it becomes constant.

Explanation of Naïve Bayes Classifier with example:

Bayesian Theorem is the underlying principle of the Naïve Bayes Classifier. It is aptly suited for a very large input data set. As mentioned above, it can outperform most of the complex classifier present, despite its simplistic design.

Let’s demonstrate the application of Naïve Bayes Classifier with an example. Consider a graph/plot of objects marked as red and yellow. They form a cluster as shown in the figure. Now the task of the classifier is to properly assign the incoming new object to its proper class. This task is based on the information of existing objects.

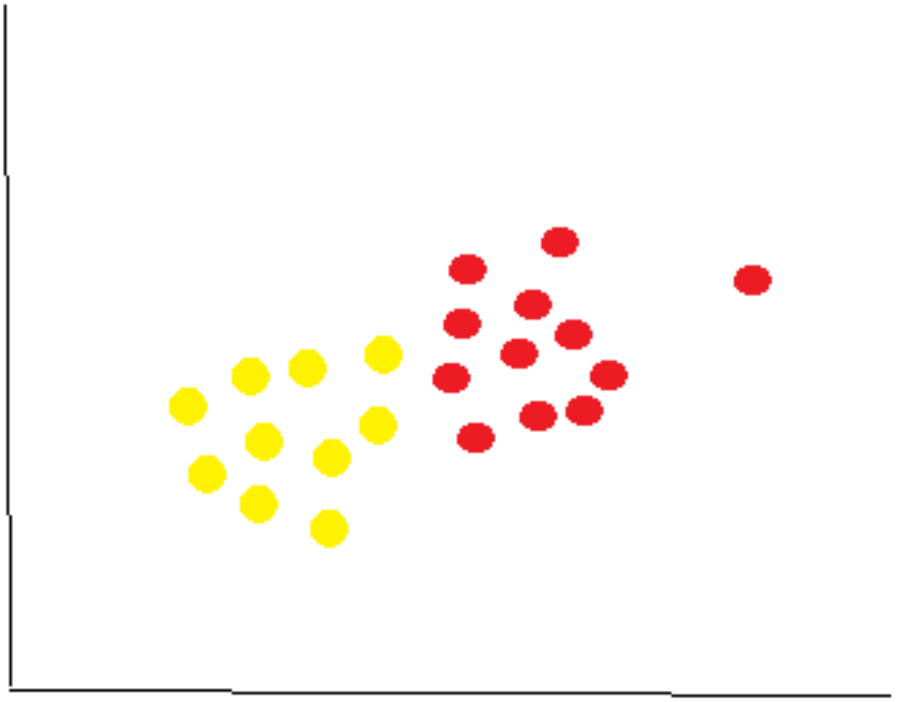


Figure 8 Plot showing clusters of objects of two classes: yellow and red

Looking at the figure we get an idea that the number of yellow objects is approximately equal to that of the red objects. This implies that the probability of the new object to be arrived in either of the classes is equal. For example, if there had been thrice as many yellow objects as red objects then the probability of the new object to be a part of the yellow class would have been thrice that of red class. In Bayesian classification this probability is known as ***Prior Probability.*** These are based on previous experience and the numbers of existing objects.

Using the above notion we can write that:

Prior Probability of Yellow objects = No. of Yellow objects / Total no. of objects

Prior Probability of Red objects = No. of Red objects / Total no. of objects

If we consider that we a total of 60 objects of which 40 are yellow and 20 are red.

Using the above formulae:

Prior Probability of Yellow objects = 40 / 60 = 2/3

Prior Probability of Red objects = 20/ 60 = 1/3

After calculating the prior probabilities, we can now classify the new incoming object which is colored as blue.

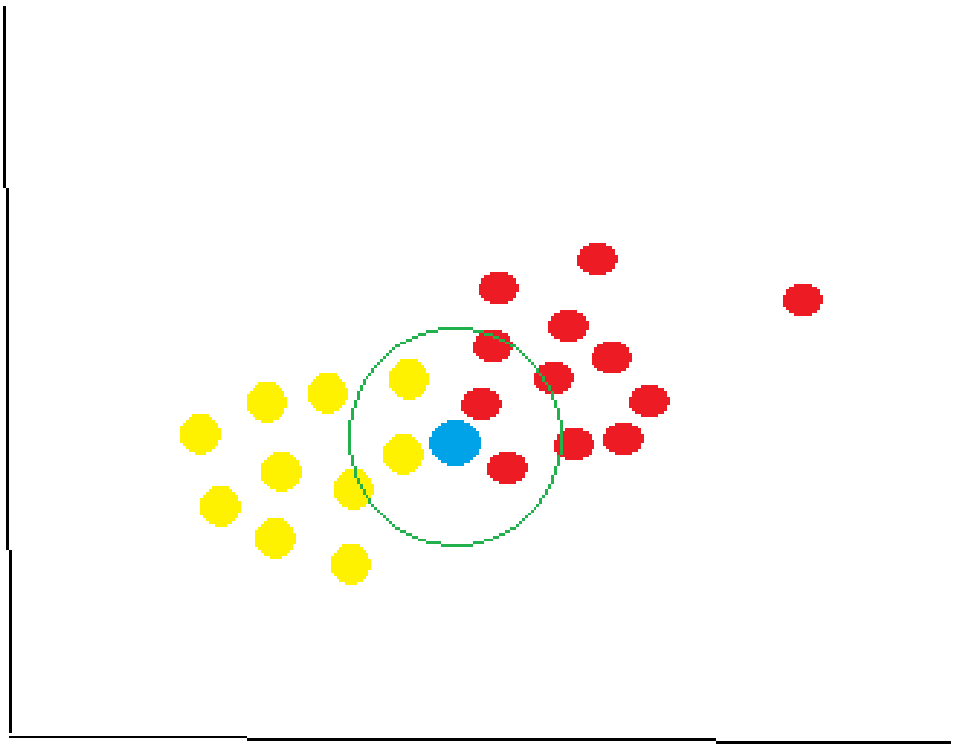


Figure 9 New object (colored in blue) to be classified

As seen from the diagram, the objects form a cluster. A small cluster of these objects is taken into consideration that lies in the vicinity of the new object (say T). It's quite likely that more the number of objects of a particular class in the sample cluster, more will be the probability of the new object T to belong to that class. To calculate this we count the number of objects of each class in the sample circle (marked by green circle).

Likelihood of T given Yellow = No. of Yellow in the vicinity / Total no. of Yellow cases

Likelihood of T given Red = No. of Red in the vicinity / Total no. of Red cases

In the sample Cluster, there are approximately 3 Red objects and 2 Yellow objects. Therefore,

Likelihood of T given Yellow = 2 / 40

Likelihood of T given Red = 3 / 20

Prior Probability results show that T should belong to yellow class but Likelihood of T being in class Red is greater. In Bayesian classification both the probabilities are taken into consideration. This combined probability is called ***Posterior Probability.***

Hence,

Posterior Probability of T being in Yellow =

*Prior Probability of Yellow objects * Likelihood of T given Yellow*

Posterior Probability of T being in Red =

*Prior Probability of Red objects * Likelihood of T given Red*

Therefore, we get the final posterior probabilities as:

*Posterior Probability of T being in Yellow = (2/3) * (2/40) = 1/30*

*Posterior Probability of T being in Red = (1/3) * (3/20) = 1/20*

Conclusively, the new object T belongs to Red class because it has a higher Posterior Probability than other classes. The same concept is used for all the new objects for their classification [6].

3. Technology Stack Used

For implementation of this project there were several options available for frameworks, languages, packages, APIs etc. I did some preliminary analysis before jumping on to the final set of technology stack used.

The major concern was the implementation of the Classifier. For that I tried a couple of software and frameworks. First, I started with IBM Unstructured Information (UI) Modeler [7]. It is a tool developed by IBM to classify unstructured data sets into classes. These classes are based on keywords, their size, cohesion and distinctiveness. But my purpose was to classify data differently. Then, I tried Splunk [8] that classifies the IT data from servers and networks for the analysis, search, monitoring and reporting. Hence, again it was not the required tool that I needed. Finally, I explored NLTK library for classification the micro-blogging tweet data. NLTK was originally created in 2001 as part of a computational linguistics course in the Department of Computer and Information Science at the University of Pennsylvania [9]. I used *nltk.classify* of the library that does classification. It uses Naïve Bayes classification algorithm which is very strong algorithm known for its simplicity.

For the middleware, I implemented web-services in PHP. Other options available were Java Apache Axis 2, Python etc. But I chose PHP as it is considered one of the best server-side languages for Web Development. This REST Web Service will get input keyword from user and pass the tweets related to that keyword to the classifier. Then it passes back the classified data to the client.

The goal of the project is to provide user with the classified data. For that efficient UI technologies should be used. The data should be easy process on the client. At the same time it is

required to be visually attractive and presentable. There are a bunch of technologies that I used for this purpose. JavaScript, AJAX, jQuery for processing the json object on the client and constantly fetching data from the server. HTML and CSS were used to make UI visually appealing. As the data was show for the purpose of analysis, there was a scope of implementing charts. For charting there are many options available these days. Some of the most commonly used tools are Google Charting APIs, Scalable Vector Graphics, and HTML5 Canvas Solutions. I am using HighCharts in this project for visualization of data.

4. Design and Implementation Details

The project is designed to separate the implementation on client-side and server-side. The client-side handles the user queries and displays the result of classification back to the user on the browser. The server receives the data from the client through the web-service and processes it against the classifier and sends the result to the web service which in turn passes it to the client.

4.1 System Architecture

The workflow of the application is shown below:

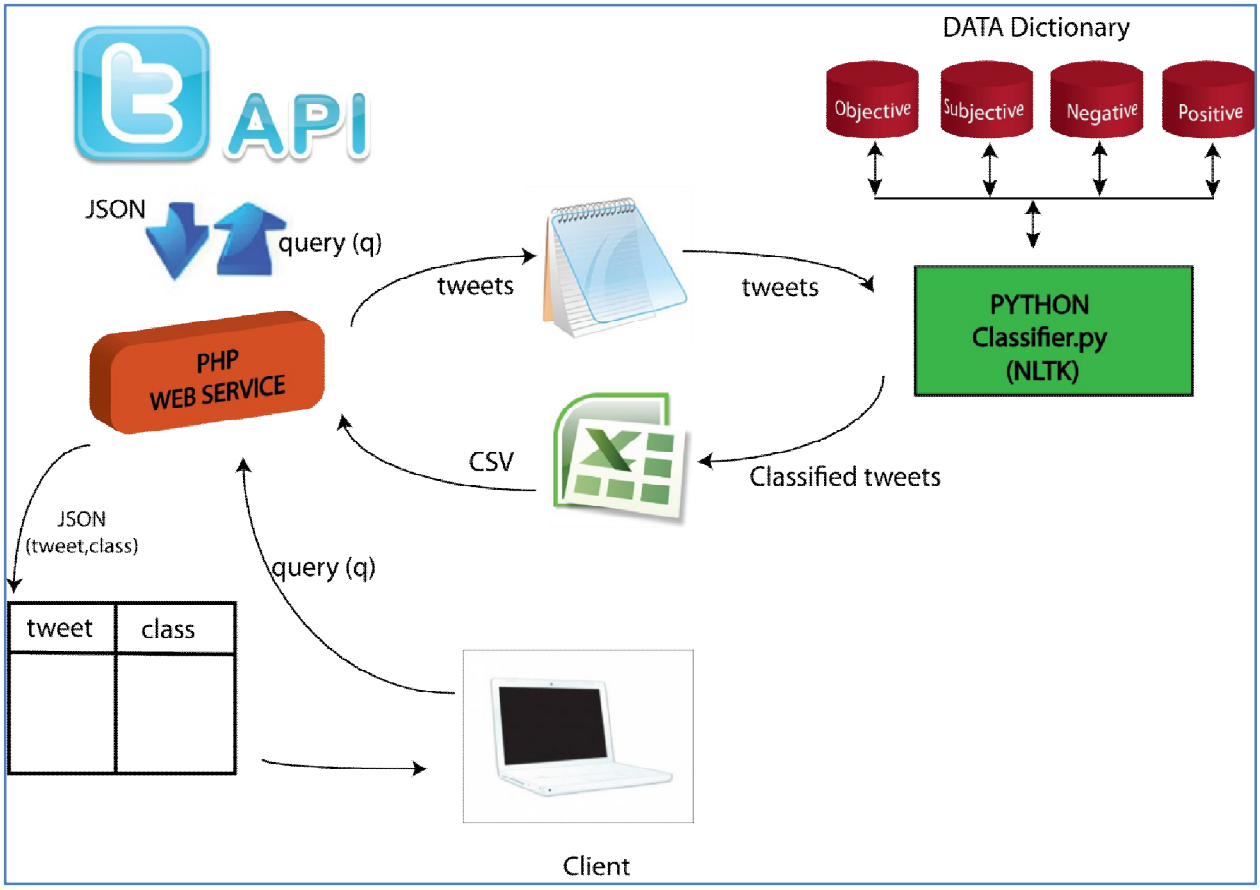


Figure 10 System Architecture

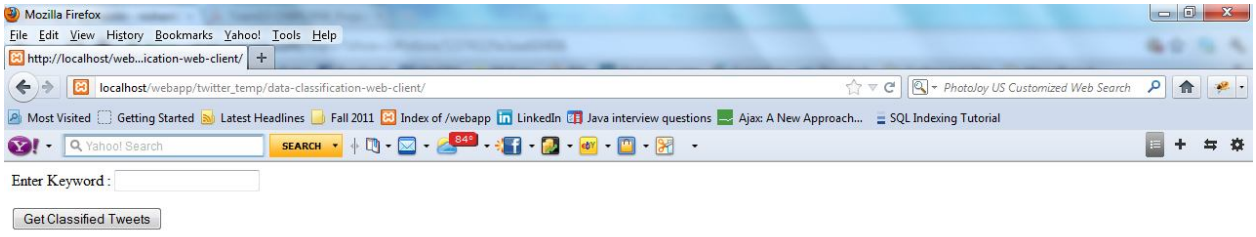
4.2 Client - Side Implementation

As already mentioned in the above section, for implementing User Interface frontend technologies used are HTML, CSS, JavaScript, AJAX and jQuery. The frontend makes server side calls through jQuery and AJAX and receives the response back from the server.

This screenshot shows the code for Search input box on the User Interface followed by its UI:

```
<body>
<form action="">
    Enter Keyword : <input type='text' />
</form>
<button>Get Classified Tweets</button>
<div id="parent"></div>
<div id="chart"></div>
</body>
```

Figure 11 Code snippet for Search Text-box



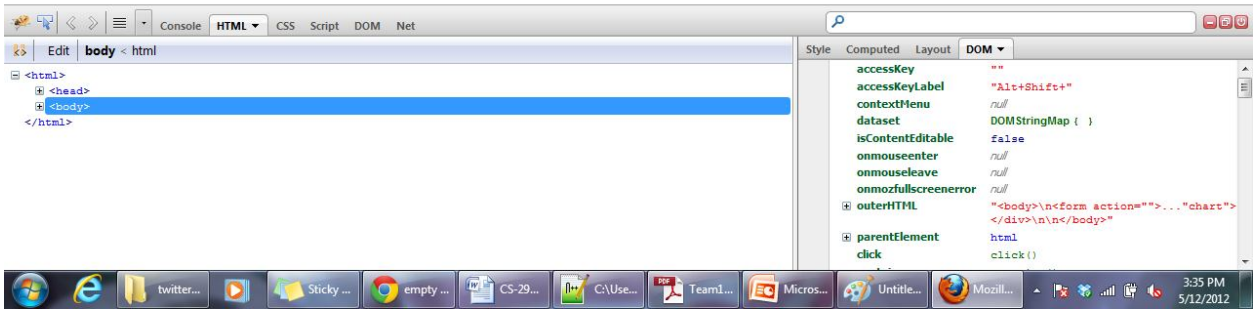


Figure 12 User Interface for Search Text Box

Once the Client code receives the json object from the Web Service, it parses that data and presents it a user friendly manner. On the top, it displays the tweet-class key-value pair in tabular format. Then it breaks down the distribution of the classes into a pie-chart. This statistics makes it easy for the user to do analysis for the query that he/she entered. It helps the user in making an informed decision about any trending topic.

Following screenshots show the classified data in tabular manner:

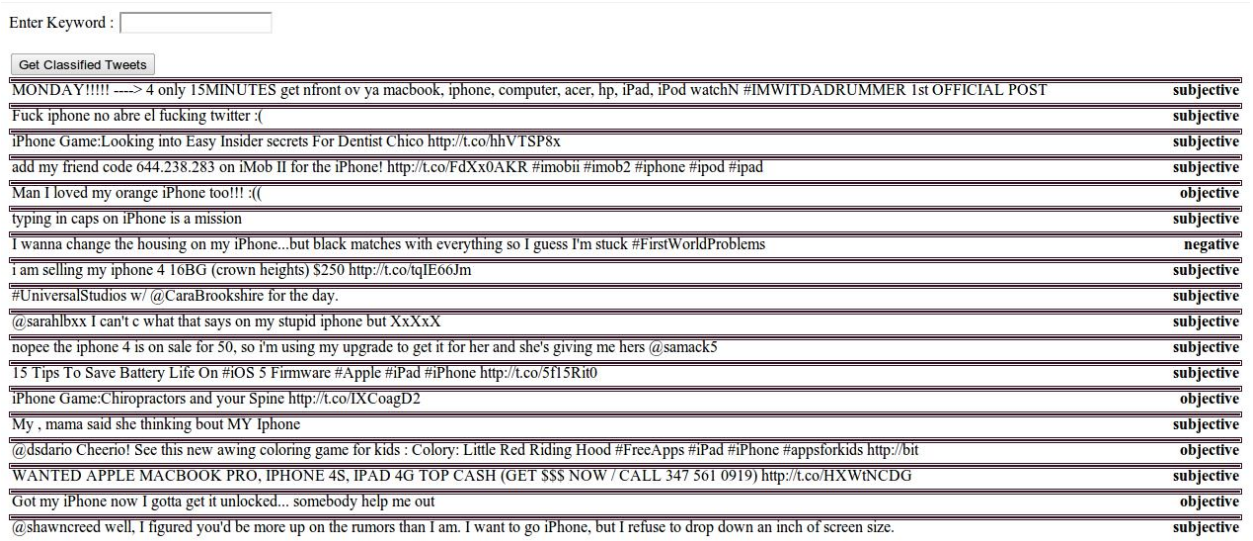


Figure 13 User Interface showing the classified data (tweets)

Following screenshot shows distribution of classes in a pie-chart

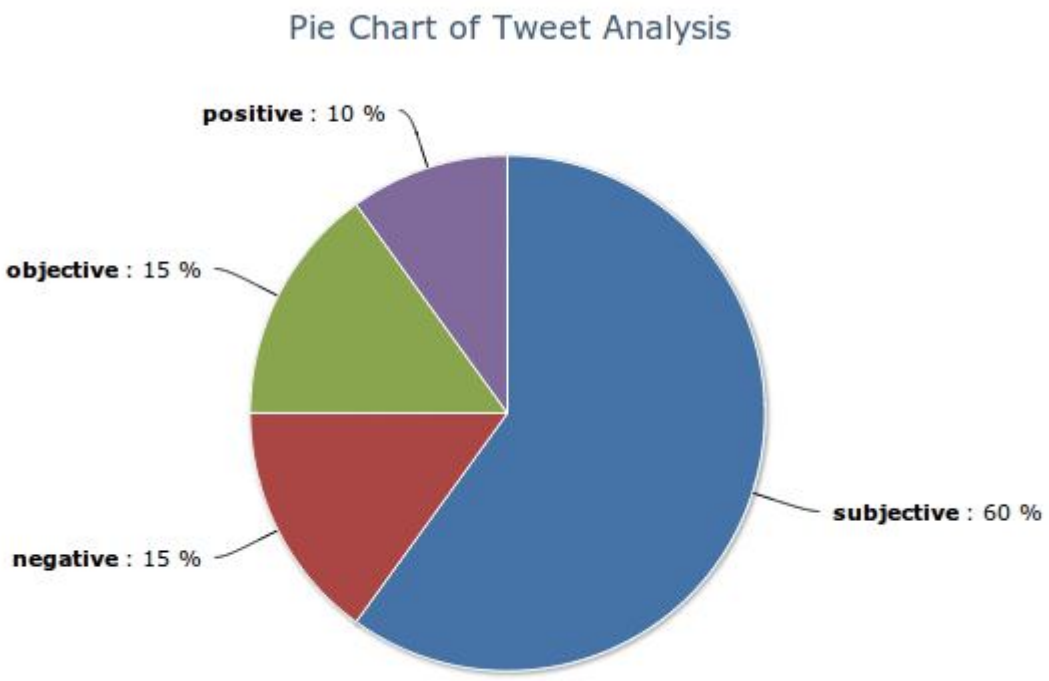


Figure 14 User Interface showing class distribution in pie-chart

4.3 Web Service Implementation

In this project I have implemented a Web Service in PHP. The advantages of implementing the web service are:

1. **Interoperability**: It is one of the major advantages of Web Services. Services can be developed in any programming languages and it uses standard internet protocols.
2. **Usability**: It exposes business logic of many different systems over the internet. Hence the consumer of the service is free to choose the best among all. The consumer only needs to change the client code for its own application.
3. **Reusability**: Web services provide the closest thing possible to zero-coding deployment of such services. Hence its components are easy to be reused on the consumer's application appropriately.

4. **Ease of Deployment**: Web Services are deployed over standard Internet technologies. Hence it can be deployed over the firewalls to some other servers as well. [10]

Components of the Web Service implemented:

1. **Fetching Tweets from Twitter based on keywords:**

   This component is responsible for fetching the tweets from the Twitter based on the search keywords entered by the user. For this purpose Twitter Search APIs are used. The Twitter Search API is a dedicated API for running searches against the real-time index of recent Tweets [11]

   There are some limitations of these APIs:

   - It does not index all the tweets but only the recent ones.
   - Tweets older than a week cannot be found.
   - It does not support complex queries.
   - No authentication is used. Hence queries are anonymous by nature.
   - Focus of the search results is on relevance not on the completeness of the results.

While using the Twitter Search APIs best practices should be followed and URL should be formed properly in order to make it work efficiently.

The table below lists some useful parameters in Search APIs that we will be using during implementation [12]:

Parameters

q required	Search query. Should be URL encoded. Queries will be limited by complexity. **Example Values**: `@noradio`
callback optional	**Only available for JSON format**. If supplied, the response will use the JSONP format with a callback of the given name.
geocode optional	Returns tweets by users located within a given radius of the given latitude/longitude. The location is preferentially taking from the Geotagging API, but will fall back to their Twitter profile. The parameter value is specified by "`latitude,longitude,radius`", where radius units must be specified as either "`mi`" (miles) or "`km`" (kilometers). Note that you cannot use the near operator via the API to geocode arbitrary locations; however you can use this `geocode` parameter to search near geocodes directly. **Example Values**: `37.781157,-122.398720,1mi`
lang optional	Restricts tweets to the given language, given by an ISO 639-1 code.
locale optional	Specify the language of the query you are sending (only `ja` is currently effective). This is intended for language-specific clients and the default should work in the majority of cases. **Example Values**: `ja`
page optional	The page number (starting at 1) to return, up to a max of roughly 1500 results (based on `rpp * page`). **Example Values**: `10`

Figure 15 Twitter GET Search Parameters list –Part1

(Source: https://dev.twitter.com/docs/api/1/get/search)

result_type optional	Optional. Specifies what type of search results you would prefer to receive. The current default is "mixed." Valid values include: • `mixed`: Include both popular and real time results in the response. • `recent`: return only the most recent results in the response • `popular`: return only the most popular results in the response. **Example Values**: `mixed`, `recent`, `popular`
rpp optional	The number of tweets to return per page, up to a max of 100. **Example Values**: `100`
show_user optional	When `true`, prepends ":" to the beginning of the tweet. This is useful for readers that do not display Atom's author field. The default is `false`.
until optional	Optional. Returns tweets generated before the given date. Date should be formatted as YYYY-MM-DD. **Example Values**: `2010-03-28`
since_id optional	Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occured since the since_id, the since_id will be forced to the oldest ID available.

Figure 16 Twitter GET Search Parameters list – Part2

(Source: https://dev.twitter.com/docs/api/1/get/search)

The result of these APIs is a json object containing information about the tweets. For our purpose we need mainly the text information of the tweet that is stored in the ***text*** parameter inside the json object. The code snippet that does this task of parsing the json object is shown below:

```php
/* method to get tweets searched for keyword $q */
function getTweets($q){
    // getting data from twitter
    $twitter_url="http://search.twitter.com/search.json?q=$q&rpp=100&lang=en";

    $data = file_get_contents($twitter_url);

    $obj = json_decode($data);

    $results_arr = $obj->{'results'};

    return $results_arr; // array containing tweets information
}
```

Figure 17 Code snippet to get tweet text from json object

2. **Storing the tweet text on a persistent medium**

The getTweets() method returns an array containing tweet texts. This component of the web service writes the data into flat files (persistent medium). Following is the code snippet for file writing:

```php
/* method to get tweets in a flat file */
function writetoFile($arr){
    $tweetFile = "data.txt";
    for($i=0; $i < sizeof($arr); $i++){
        $text .= $arr[$i]->text."\n";
        //echo $text;
    }
    file_put_contents($tweetFile, $text, FILE_APPEND | LOCK_EX);
}
```

Figure 18 Code snippet to write tweet data in text file

3. Converting classifier response(csv) in json

The file generated above “data.txt” is the input to the classifier. And the classifier in turn writes the classified data into results.csv. This file has to be parsed and convert into json object that client side javascript can understand. The code snippet for this functionality is shown below:

```
/* method to convert classifier response(xls) in json*/
function csvToJSON(){
    $file = fopen("results","r");
    $result_array = array();
    while(! feof($file))
    {
        $data = fgetcsv($file,1000,"|");
        $tweet = $data[0];
        $class = $data[1];
        $tweet_class_arr["tweet"] = $tweet;
        $tweet_class_arr["class"] = $class;
        array_push($result_array ,$tweet_class_arr );
    }
    fclose($file);
    $jsonobj = json_encode($result_array);
    echo $jsonobj;
    return jsonobj;
}
```

Figure 19 Code snippet for converting classifier response in json

4.4 Server-Side Implementation

The central part of the server implementation is the classifier. For classification I have used NLTK Library. It is a very powerful Python based library for Natural Language Processing. There are several modules under this toolkit that used for variety of purposes like String

Processing, Collocation Discovery, Part of Speech tagging etc. I have used the Classification module of this toolkit named *nltk.classify*.

In this subsection I will explain the working of the classifier module.

1. The python classifier keeps listening to *inputdata* file. Once it gets all the tweets it passes them to classify method. And then writes the results into '*results*' file.

2. This method is used to get classes from the specified path for the document containing data. This routine first opens the file from the path. Then it reads the file line by line and parses it to get the class. These classes are appended in the class array defined at the start of the routine. Once it reaches the end of the file, it closes the file object and returns the array of classes.

```
def getClassificationClasses(path):
    classes = []
    listing = os.listdir(path)
    for infile in listing:
        f = open(path + infile)
        lines = f.readlines()
        classes.append((lines,infile))
        f.close()
    return classes
```

Figure 20 Code snippet to get classification classes.

3. This method gets all the classes from the previous method puts them in array. This data structure holds the information in the form of line (tweet) and its respective

class. After parsing the entire class array it returns the training data in an array. The code for this routine is shown below:

```
def parseTrainingData(classes):
    all_training_data = []
    for (lines, aclass) in classes:
        for aline in lines:
            all_training_data.append((aline,aclass))
    return all_training_data
```

Figure 21 Code snippet for parsing training data

4. This method gets an input of tweet array that holds all the word tokens along with their classes (or sentiments). The intent of this routine is to get all the words from the tweets and put them in an array. This array is returned to the caller method. The code for this routine is shown below:

```
def get_words_in_tweets(tweets):
    all_words = []
    for (words, sentiment) in tweets:
      all_words.extend(words)
    return all_words
```

Figure 22 Code snippet for extracting words from the tweets

5. This routine uses NLTK API '***nltk.FreqDist ( )'*** to get the features of the words. This API takes a list of words as input argument and returns words along with their frequencies in sorted order. Words which are in the form of keys are returned. The code for this routine is shown below:

```
def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    word_features = wordlist.keys()
    return word_features
```

Figure 23 Code snippet for getting word features

6. This method takes tweet data as input argument and forms a hash of words and their presence in the classifier. For example: [hate, true]. The code is shown below:

```
def extract_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains(%s)' % word] = (word in document_words)
    return features
```

Figure 24 Code snippet for getting features and their existence in the classifier

7. We need a training data to evaluate our words for their classes. For this purpose we train our data dictionaries. If the pickle file for classifier is not present the code trains the input data and puts it in classifier.pickle file else it uses the existing pickle file to load the data. In python, pickle library is used to serialize and de-serialize a Python object structure [13]. The code to handle this situation is shown below:

```
if os.path.isfile('my_classifier.pickle'):
    f= open('my_classifier.pickle',"rb")
    classifier = pickle.load(f)
    f.close()
    f= open('word_features.pickle',"rb")
    word_features = pickle.load(f)
    f.close()

else:
    all_training_data = parseTrainingData(getClassificationClasses('classification/'));
    tweets = []
    for (words, sentiment) in all_training_data:
        words_filtered = [e.lower() for e in words.split() if len(e) >= 3]
        tweets.append((words_filtered, sentiment))

    word_features = get_word_features(get_words_in_tweets(tweets))

    training_set = nltk.classify.apply_features(extract_features, tweets)

    classifier = nltk.NaiveBayesClassifier.train(training_set)

    f = open('my_classifier.pickle', 'wb')
    pickle.dump(classifier, f,1)
    f.close()
    f = open('word_features.pickle', 'wb')
    pickle.dump(word_features, f,1)
    f.close()
```

Figure 25 Code snippet to load the data into pickle file

The entire server-side implementation process for the Classifier can be summarized in the figure below:

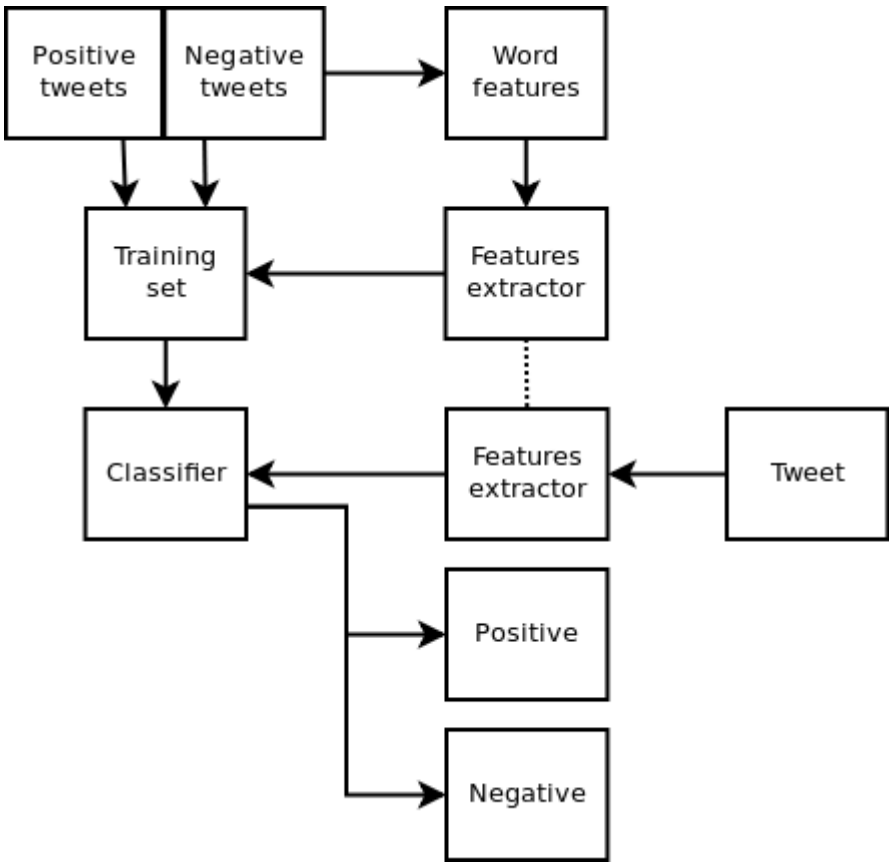


Figure 26 Workflow of Classifier

(Source: http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/)

4.5 Debugging Methodology:

In the server-side implementation as well as in the web-service implementation proper debugging strategy has been used. Proper debug messages are placed at all the critical points in the flow of execution. While developing such a complex application it is often required to use proper tracing mechanism to keep a track of the flow. These messages are written in the log/debug files that can be viewed later to determine the operation of the application which is not possible to track through User Interface alone. Log files are often useful and come in handy when there are errors in the code. Apart from that, debug messages are outputted on the console as well in order to aid the developer to check the sequence of operations.

In the web service, debug messages are used when the request is received from the user. Then debug message is put after all the tweets are extracted for the queried keyword. And lastly, a debug message is put when tweets are written into the file.

The following screenshots show the application of debug messages in the web service code:

```
if(isset($_GET['q']) ) {
    file_put_contents("debug", "DEBUG 001: request received: \n",FILE_APPEND | LOCK_EX);
    $format = 'json';
    $q = $_GET['q'];
    $q_explode = explode(" ", $q);
    foreach($q_explode as $q_each){
        $x++;
        if($x==1)
            $param .= "$q_each";
        else
            $param .= "%20$q_each";

    }
    $q=$param;
    file_put_contents("debug", "DEBUG 002: request keyword: ".$q."\n",FILE_APPEND | LOCK_EX);


    $tweetArray = getTweets($q);
    file_put_contents("debug", "DEBUG 003: tweets received \n",FILE_APPEND | LOCK_EX);
    writetoFile($tweetArray);
    file_put_contents("results", ""); //flush results
    file_put_contents("debug", "DEBUG 004: file written\n",FILE_APPEND | LOCK_EX);
    while(file_get_contents("results")==""){
        if(file_get_contents("results")!="")
            break;
    }
```

Figure 27 Debug messages in Web Service code marked in Red

In the server code, first debug is used as long the classifier code keeps on listening to the input data file. Then it is used when the tweets are extracted by classifier code. And lastly, it is used when results are written into the results file.

The following screenshots show the application of debug messages in the classifier code:

```
while(True):
    print "DEBUG 001: LISTENING"
    all_tweets = []
    while(True):
        all_tweets = open("inputdata").read().splitlines()
        if len(all_tweets)>0:
            if all_tweets[len(all_tweets)-1]=="THISISLASTTWEET":
                break
    print "DEBUG 001: REQUEST RECEIVED"
    tweet_results = ""
    del all_tweets[-1]
#     all_tweets = open("inputdata").read().splitlines()
    print "DEBUG 002: TWEETS EXTRACTED"
    for tweet in all_tweets:
        classification_result = classifier.classify(extract_features(tweet.split()))
        tweet_results+= tweet + "||" + str(classification_result)+"\n"
    print "DEBUG 003: RESULTS READY"
    f = open("results","w")
#    f.write(tweet + "||" + str(classification_result) + "\n")
    f.write(tweet_results)
    f.close()
    print "DEBUG 004: TWEETS ANALYSED : " + str(len(all_tweets))
    open("inputdata", 'w').close()
```

Figure 28 Debug messages in Classifier code marked in Red

The following screen shot shows the flow debug messages in the debug file while the application is running:

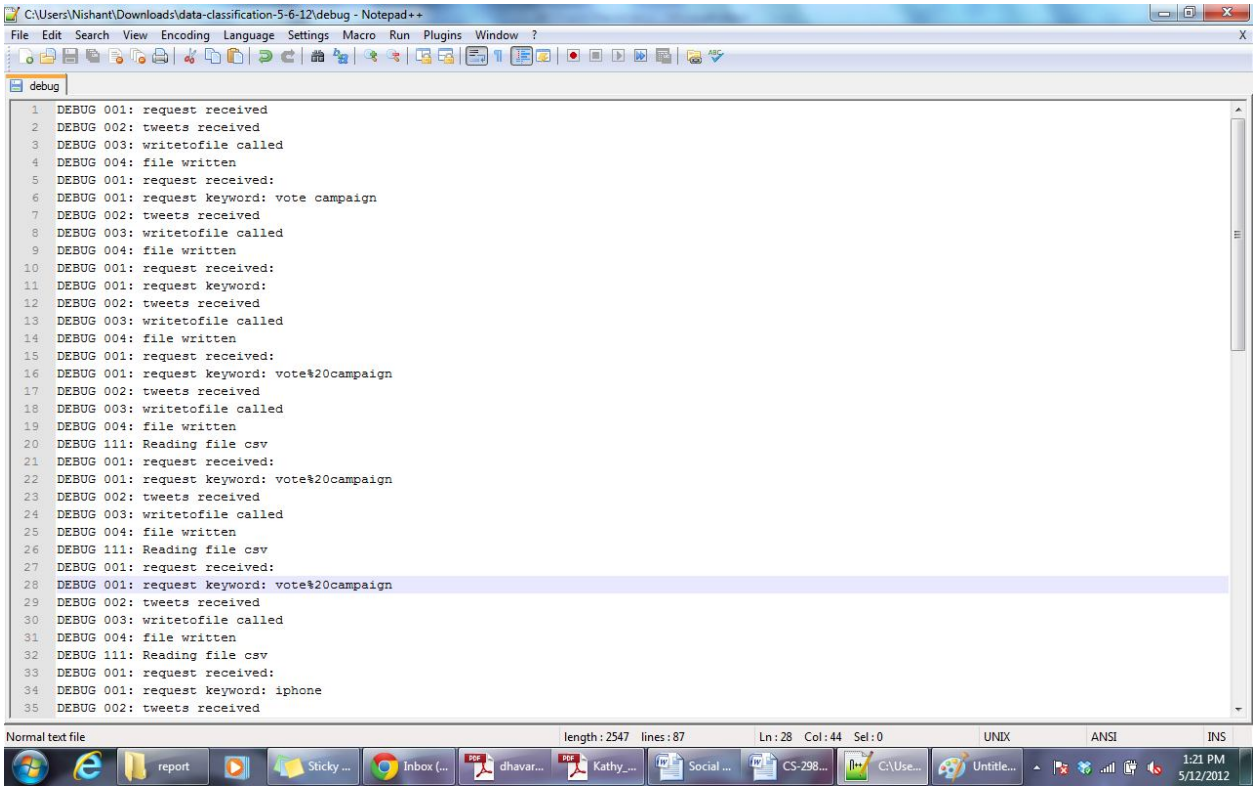


Figure 29 Debug file during execution

5. Summarization

5.1 Summary

Blog Information Classification is a very useful application that helps the user to analyze the current trends for a particular keyword. This can be achieved by classifying the Twitter blogs into different category classes that are based on certain sentiments, opinions etc. If a user wants to see how an android phone is trending in the current scenario, then this application will come in handy. Twitter micro-blogs are targeted for this purpose because nowadays people are much more active on Twitter and express their opinions freely on this micro-blogging website. Hence, it serves a good and authentic source to measure the trends. Therefore, once the user enters android, he/she will be provided a data in form of table that contains tweet and their class. Also, a pie-chart will be shown that contains distribution of the classes. If, for example, positive percentage of tweets containing “battery life” has more positive percentage then the user will come to know that people who have used Android are happy or have positive opinion about its battery life. This kind of analysis becomes easy with the help of Blog Information Classification

5.2 Conclusion

I completed this project as a student of San Jose State University, Computer Science Department. This project helped me learn and understand many technologies and concepts that were alien to me before the project. I learned how to develop REST Web Service APIs and understood their importance in today’s world of Internet and HTTP protocols. I also took a small step towards information classification concentrating on a relatively small source of input data called micro-blogs as compared to the entire blogosphere. So looking at the current trends of the growing data this application will be very helpful for users who want to get quick analysis of tons of data within seconds.

5.3 Recommendations for Future Research

This project is developed keeping in mind the scalability perspective. For basic parsing of data and handling the request-response mechanism, a Web Service has been used. Also, any number of classes can be added to the classifier in order to get more classified information. This will not change the code of the classifier as it addition of classes is easily configurable. Also, one major point that needs to be addressed is the need of having best possible trained data for the accuracy of the results. On the client side, option for more and more forms of data visualization techniques should be made available for the user so that they can view/analyze the data in a manner they want to do.

6. References

[1] Prof explores effects of blogs on politics. Retrieved May 12, 2012, from http://www.oread.ku.edu/2006/november/20/blogs.shtml

[2] Twitter Search. Retrieved May 12, 2012, from http://www.commoncraft.com/video/twitter-search

[3] Learning to Classify Text. Retrieved May 12, 2012, from http://nltk.googlecode.com/svn/trunk/doc/book/ch06.html

[4] NLTK: Text Classification. Retrieved May 12, 2012, from http://nltk.googlecode.com/svn/trunk/nltk-old/doc/technical/classification/classification.tex

[5] Zhang, Harry. *The Optimality of Naive Bayes.* Retrieved May 12, 2012, from http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf

[6] Naive Bayes Classifier. Retrieved May 12, 2012, from http://www.statsoft.com/textbook/naive-bayes-classifier/

[7] IBM Unstructured Information Modeler. Retrieved May 12, 2012, from https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=a2e02728-f640-4e45-a627-7e1f5e030fac

[8] Splunk. Retrieved May 12, 2012, from http://www.splunk.com/

[9] Natural Language Processing. Retrieved May 12, 2012, from http://nltk.googlecode.com/svn/trunk/doc/book/ch00.html

[10] Advantages & Disadvantages of Web services. Retrieved May 12, 2012, from http://social.msdn.microsoft.com/Forums/en-US/asmxandxml/thread/435f43a9-ee17-4700-8c9d-d9c3ba57b5ef/

[11] GET Search. Retrieved May 12, 2012, from https://dev.twitter.com/docs/api/1/get/search

[12] Using the Twitter Search API. Retrieved May 12, 2012, from https://dev.twitter.com/docs/using-search

[13] pickle — Python object serialization. Retrieved May 12, 2012, from http://docs.python.org/library/pickle.html

[14] Looper, E; Bird, S. *NLTK: The Natural Language Toolkit*. Retrieved May 12, 2012, from http://acl.ldc.upenn.edu/acl2002/TNLP/pdfs/TNLP005.pdf

[15] Naive Bayes Classifier. Retrieved May 12, 2012, from http://en.wikipedia.org/wiki/Naive_Bayes_classifier