**Chapter 5**

**Implementation Methodology**

**5.1 Gathering the data**

Gather the data for training from publicly available twitter datasets which contain positive, negative and neutral tweets.

* 1. **Pre-Processing the data**

The tweets are processed, which firstly involves converting all the textual data tolower case. The tweets also contain URL, hash tags, whitespace, usernames and punctuation marks which don’t provide any information regarding the sentiment. So we filter out that things and we are left with only words for analyzing.

* 1. **Feature Vector**

Feature vector is the most important concept in implementing a classifier. In tweets, we can use the presence/absence of words that appear in tweet as features. In the training data, consisting of positive, negative and neutral tweets, we can split each tweet into words and add each word to the feature vector. Some of the words might not have any say in indicating the sentiment of a tweet and hence we can filter them out. Adding individual (single) words to the feature vector is referred to as 'unigrams' approach. Some of the other feature vectors also add 'bi-grams' in combination with 'unigrams'. For example, 'not good' (bigram) completely changes the sentiment compared to adding 'not' and 'good' individually. Here, for simplicity, we will only consider the unigrams.

**5.4 Filtering Tweet Words**

1. *Stop words* - a, is, the, with etc. These words don't indicate any sentiment and can be removed.
2. *Repeating letters* - if we look at the tweets, sometimes people repeat letters to stress the emotion. E.g. hunggrryyy, huuuuuuungry for 'hungry'. We can look for 2 or more repetitive letters in words and replace them by 2 of the same.
3. *Punctuation* - we can remove punctuation such as comma, single/double quote, question marks at the start and end of each word. E.g. beautiful!!!!!! replaced with beautiful
4. *Words must start with an alphabet* - For simplicity sake, we can remove all those words which don't start with an alphabet. E.g. 15th, 5.34am.

After this words are classified as positive, negative or neutral we add these words to feature vector. In this way we can make a training set and store it in .csv file

* 1. **Feature Extraction & Feature List**

We will use Natural Language Toolkit (NLTK) and implement the two classifiers namely Naive Bayes and Maximum Entropy in Python 2.7. The feature vector obtained from previous steps are labeled and saved in a .csv file. Now we extract the new tweets and labels from the csv file and process it. The feature words extracted from the tweets are processed to convert it into feature vector. The feature vector is then used to train the subsequent classifier. With the help of NLTK we Bulk Extract the Features of the tweets and the respective classifiers.

* 1. **Naive Bayes Classifier**

In machine learning, Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.  A Naive Bayes classifier assumes that the value of a particular feature is unrelated to the presence or absence of any other feature, given the class variable.

* 1. **Retrieving tweets for a particular topic (UI)**

When we build a twitter sentiment analyzer, the input to our system will be a user entered keyword. Hence, one of the building blocks of this system will be to fetch tweets based on the keyword within selected time duration. We use OAuth with Twitter API to fetch tweets from Twitter.

* 1. **Classification of Tweets**

Finally,we pre-process the tweets gathered from the twitter API and classify each tweet as positive, negative or neutral by running our trained classifier on those tweets and show the result graphically (line chart or column chart using Google Charts).

**Code:**

import web

import get\_twitter\_data

import baseline\_classifier, naive\_bayes\_classifier, max\_entropy\_classifier

import json, logging, html\_helper

urls = ( '/', 'index')

class index:

def GET(self):

query = web.ctx.get('query')

html = html\_helper.HTMLHelper()

twitterData = get\_twitter\_data.TwitterData()

if query:

if(query[0] == '?'):

query = query[1:]

arr = query.split('&')

logging.warning(arr)

#default values

time = 'daily'

for item in arr:

if 'keyword' in item:

keyword = item.split('=')[1]

elif 'method' in item:

method = item.split('=')[1]

elif 'time' in item:

time = item.split('=')[1]

#end loop

if(method != 'baseline' and method != 'naivebayes' and method != 'maxentropy'):

returnhtml.getDefaultHTML(error=2)

tweets = twitterData.getTwitterData(keyword, time)

if(tweets):

if(method == 'baseline'):

bc = baseline\_classifier.BaselineClassifier(tweets, keyword, time)

bc.classify()

returnbc.getHTML()

elif(method == 'naivebayes'):

trainingDataFile = 'training\_neatfile\_4.csv'

classifierDumpFile = 'moviereviewtesting-pari.pickle'

#classifierDumpFile = 'naivebayes\_trained\_model.pickle'

trainingRequired = 0

nb = naive\_bayes\_classifier.NaiveBayesClassifier(tweets, keyword, time, \

trainingDataFile, classifierDumpFile, trainingRequired)

nb.classify()

returnnb.getHTML()

elif(method == 'maxentropy'):

trainingDataFile = 'training\_neatfile\_4.csv'

classifierDumpFile = 'maxent\_trained\_model.pickle'

trainingRequired = 0

maxent = max\_entropy\_classifier.MaxEntClassifier(tweets, keyword, time, trainingDataFile, classifierDumpFile, trainingRequired)

maxent.classify()

returnmaxent.getHTML()

else:

returnhtml.getDefaultHTML(error=1)

else:

returnhtml.getDefaultHTML()

if \_\_name\_\_ == "\_\_main\_\_":

app = web.application(urls, globals())

app.run()