**CS6320, Fall 2015 Project Report**

Group: Daybreak

Hengxiu Gao 2021229554

**1. Problem Description**

Train a Naïve Bayes classifier to perform the text classification tasks like sentiment analysis on social media content.

**2. Proposed Solution**

Collect enough Twitter data using twitter API, labeled these data with sentiment (positive, negative, neutral) by hand, using these data as training data, extract some features include the most positive word, the most negative word and so on from the data, to train a Naïve Bayes classifier. Use this classifier to test the accuracy.

**3. Implementation Details**

**· Baseline system**

Given several different types of sequences from the social media (twitter), use a Bag of Words Model, assume all positions in the given sequence have the same distribution, and ignore the order of words, use the overall sentiment score computed by AFINN model (Nielsen, 2011) as the feature, train a classifier using some training data and predict the sentiment on the test data.

In order to view the result of the test, a UI is built as shown in Fig.1, it allows users to type in the sentence they want to classify by the baseline system, the result of the sentiment will be viewed in the UI.

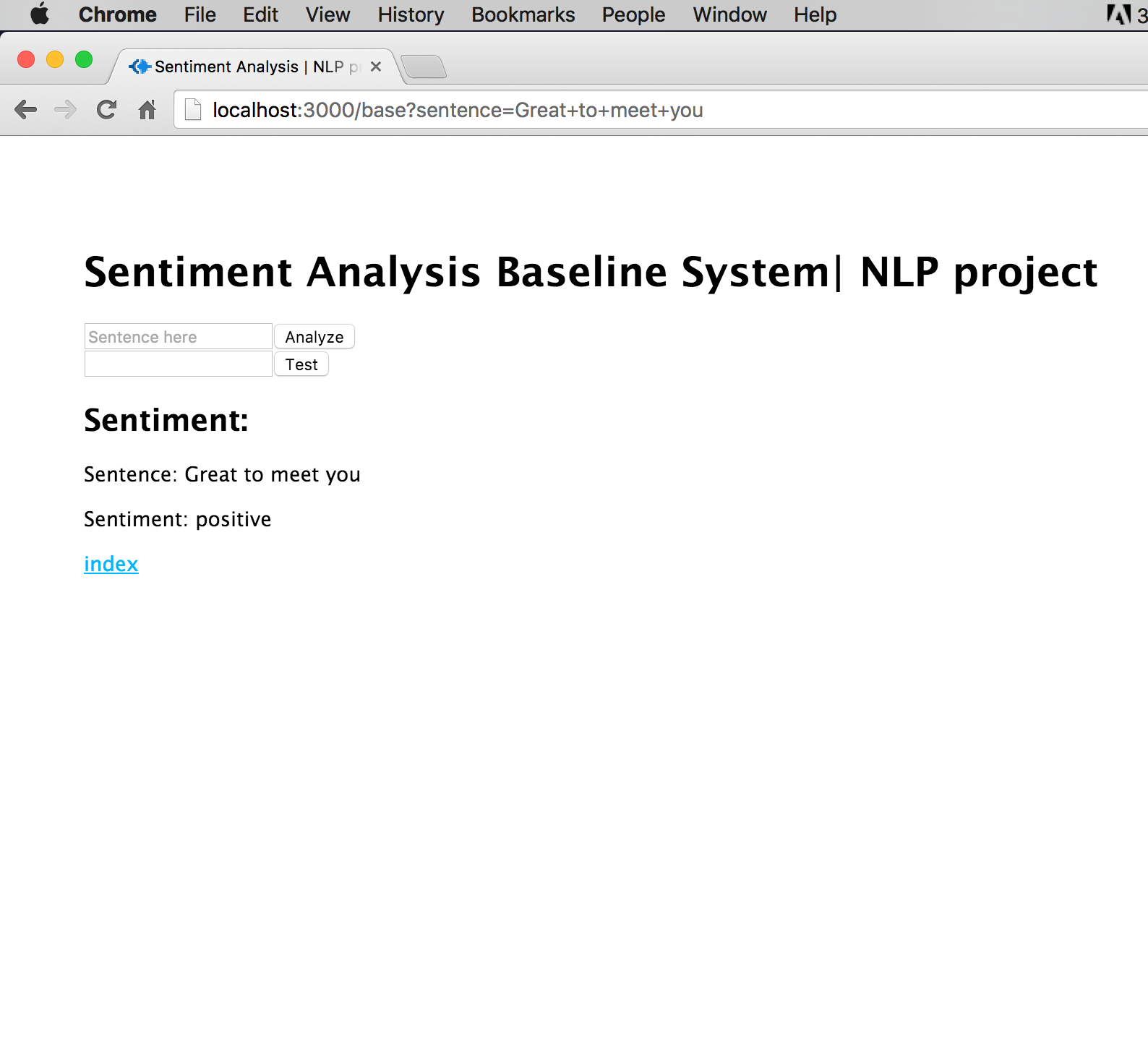


Fig. 1 The user interface of baseline system.

**· Improvement strategy**

1). User Interface

Besides the page for the baseline system, the index page and the final system page also need to be built.

The index page is shown in Fig. 2.

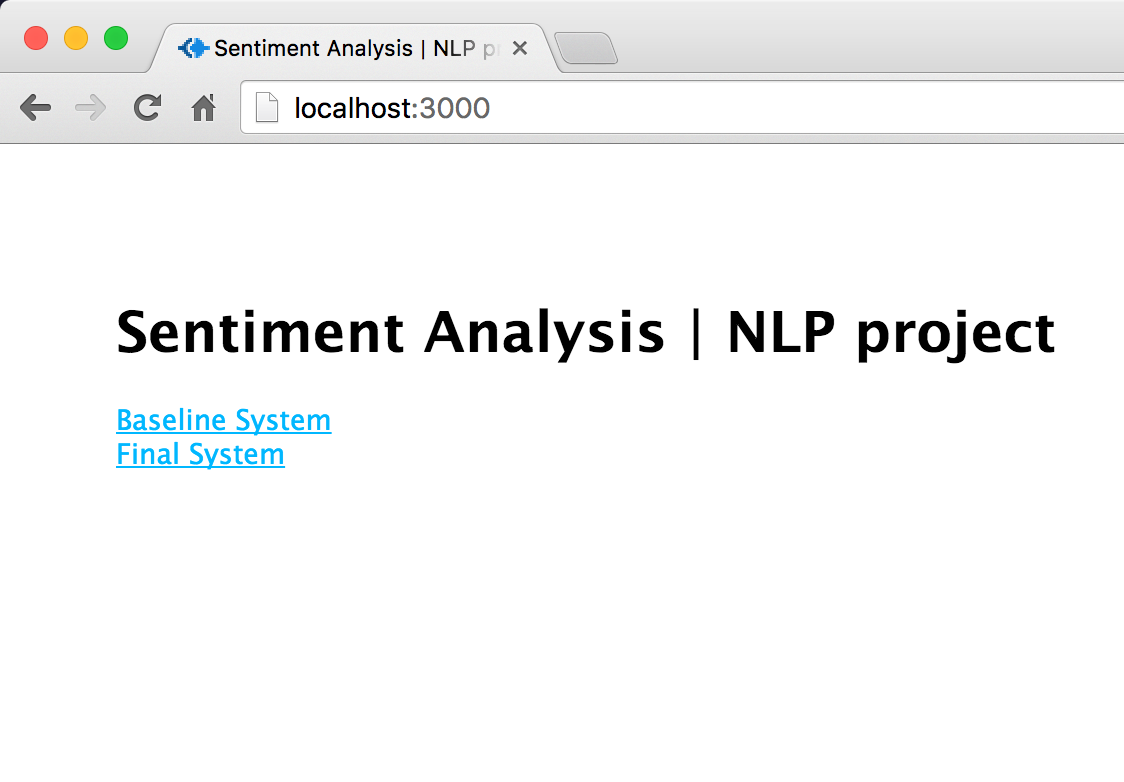
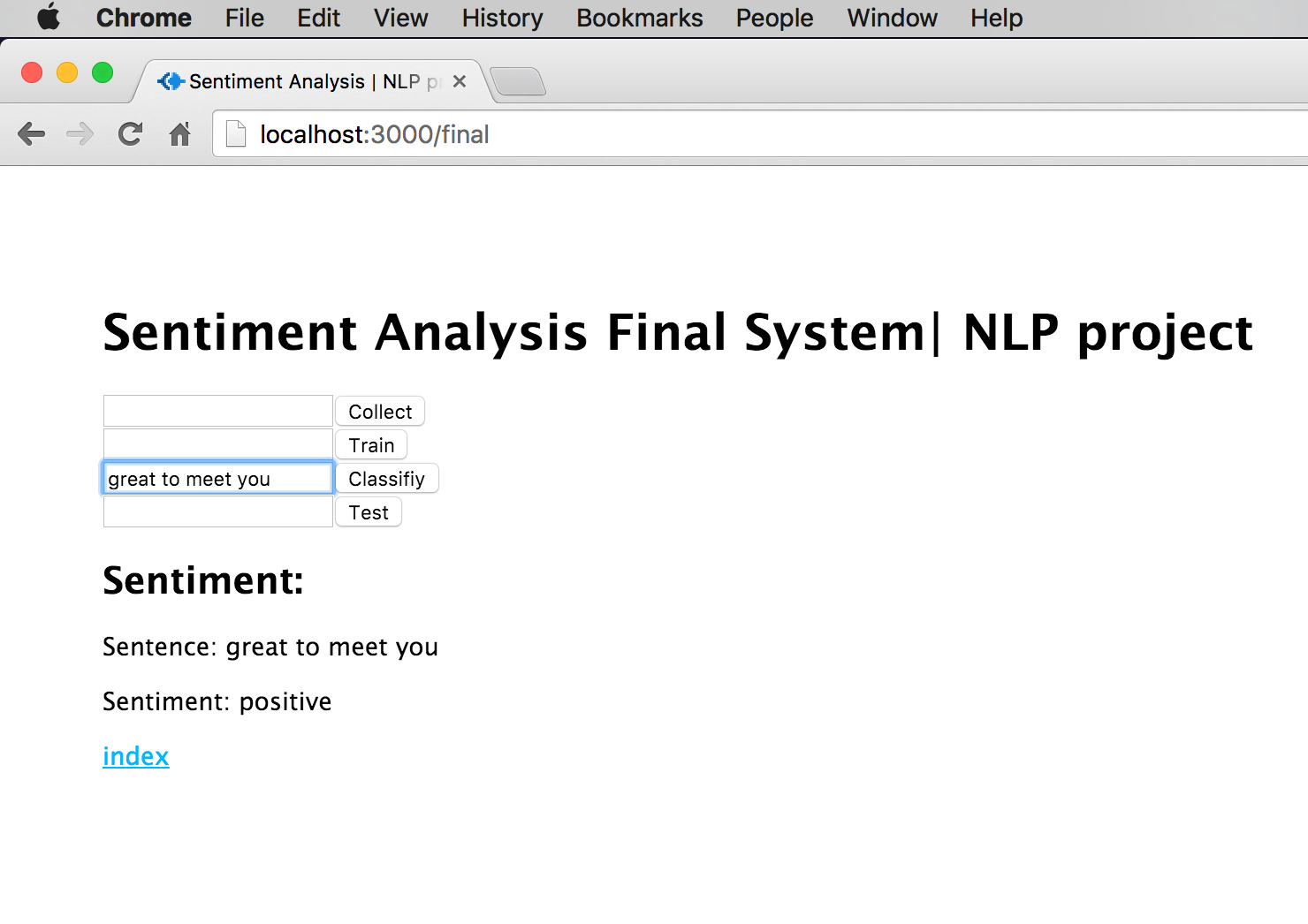


Fig. 2 The index page.

The final system page is shown in Fig. 3



In the page of final system, users can collect data from twitter, store these data in a file named “train\_data.txt” file, then train the collected data by the Naïve Bayes classifier, and output the trained classifier in the file “classifier.txt”. After generate the classifier, the users can type in the sentence they want to classify and click classify to see the result using the trained classifier. There is also a test button that allows users to test the accuracy using the pre-trained classifier.

2). Data collection

First of all, twitter data need to be collected before performing the training and testing, the data can be collected through twitter app, which provides the API to search and collect the twitters.

Once the twitter data has been collected, the raw twitter data must be processed to remove the special characters, including ‘@’, ‘#’, emoji and so on, such filtrations can be performed using regular expression.

For example, the raw twitter data as shown in fig 1:

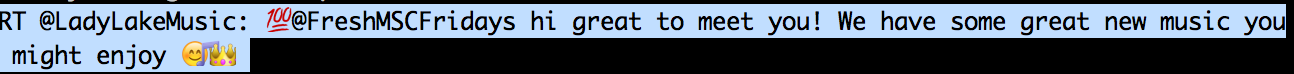




Fig. 1. Twitter raw data

After removing the special characters and strings, the processed data as well as the labeled sentiment as shown in fig 2.



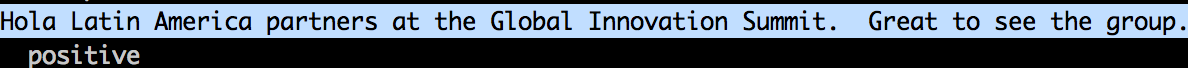


Fig. 2. Processed Twitter data with labels.

When finishing the data collection and filtration, it is necessary to label the data with its sentiment (positive, neutral or negative). Ideally the label should be hand labelled to insure the correctness, but given the tight timeframe, another method has been taken here, an online API from monkey learn (<http://monkeylearn.com/)> is used to perform the data label job, it is assumed that the label given by the monkey learn is always correct. The train data consists of the processed data from twitter and its sentiment labels.

3). Feature extraction

There are too many features can be used from the twitter data but we can’t use all of them, so the feature extraction need to be performed to pick the appropriate features for training. There are mainly two types of features are extracted here:

(1). Word Unigram

(2). Word Tags

For extracting the unigram information, there is another model AFINN (Nielsen, 2011) being used, to get the sentiment score for each word. A POS Tagging model is also used to get the tag for each word.

The exacted feature looks like:

features = {overall score,

best score, best word tag, pre best score, pre best word tag,

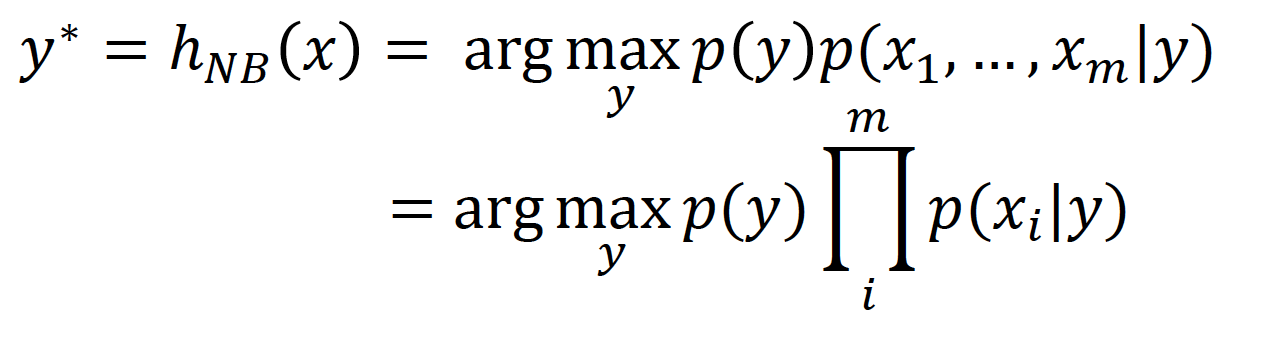
worst score, worst word tag, pre worst score, pre worst word tag,

# of RB, # of MD, # of WRB};

Here the score means the sentiment score, the best score is the score of the highest positive score word, best word tag is the POS tag of that word, pre best score is the word before the best score word, the worst score is the score of lowest negative score word. The # of RB is the number of words tagged as RB, same for the # of MD, # of WRB.

4). Naïve Bayes

Apply Naïve Bayes assumption as shown in the equation 1, train a Naïve Bayes Classifier using the feature exacted by the previous step and the labels.



Equation 1

**· Examples**

Given the input texts: (comes from Twitter)

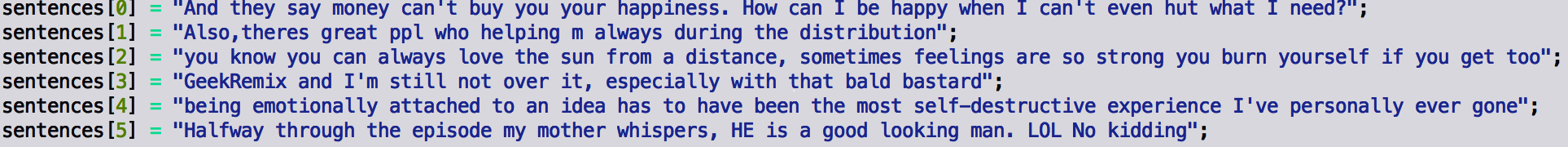


Fig. 3 Inputs

Output their sentiments: negative, positive, positive, neutral, negative, positive.

**· Programming tools (including third party software tools to be used)**

Programming tools:

Node.Js

3-party software tools:

Twit (<https://www.npmjs.com/package/twit)>, for collecting twitter data.

Monkey learn (<http://monkeylearn.com/)>, for labelling the data.

Sentiment (<https://github.com/thisandagain/sentiment)>, AFINN-based sentiment analysis for Node.js, slightly modified by me.

Natural (<https://www.npmjs.com/package/natural)>, tokenizing the sentence.

Pos (<https://www.npmjs.com/package/pos)>, POS Tagging for the sentence.

**· Architectural diagram**

The architectural diagram as shown in Fig. 4, note that the test set also need to be processed by the POS Tagging, Word frequency and feature extraction to get the features.

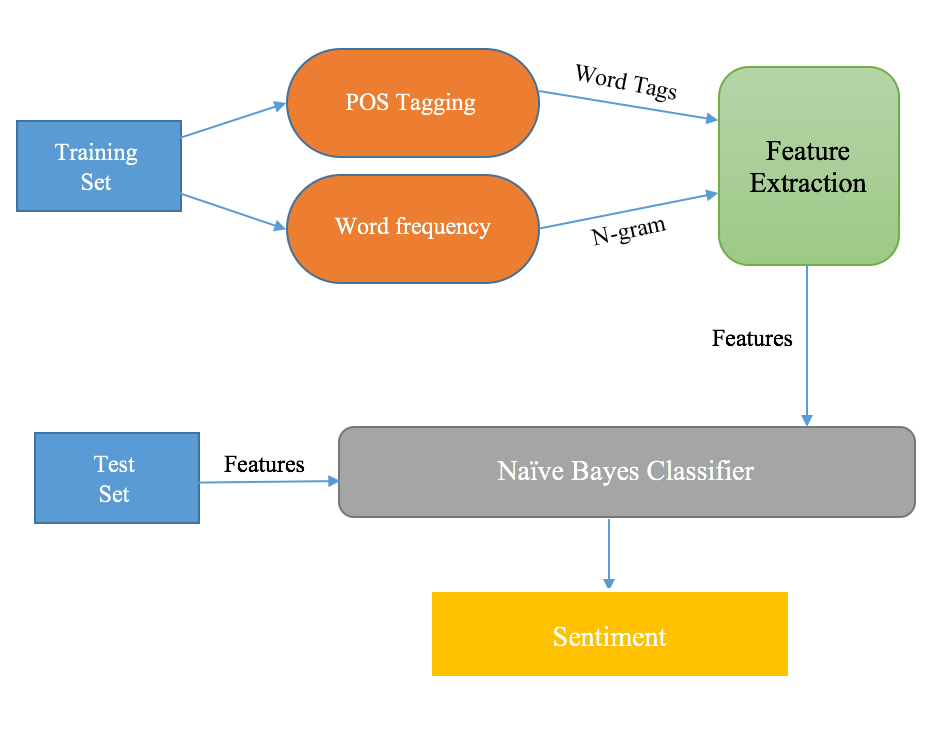


Fig. 4 architectural diagram

**· Results**

The size of the training data is 98650, the size of the test data is 82400.

Base System:

Accuracy is 0.5408 over the test set.

Final System:

The accuracy of train set using the trained classifier is 0.62198

The accuracy of test set using the trained classifier is 0.6214

**· A summary of the problems encountered during the project and how these issues were resolved**

Problems: how to decide the features used to train the classifier.

How to resolve: Use different sets of features to train the classifier, compare the accuracy result and choose one of the best set.

**· Pending issues**

1). Some collected twitters need to be filtered, for instance, non-English twitter, repeat content twitter, non-sense twitter and so on.

2). UI could be improved.

3). The Stream file write/read functions have bugs.

**· Potential improvements**

1). Processed the twitter data to remove non-English twitter, repeat content twitter, non-sense twitter and so on, to improve the test accuracy.

2). Find a more appropriate set of feature to train the classifier to improve the test accuracy.

3). Use AngularJS to re-write the front-end of the web-site

# Bibliography

Nielsen, F. Å. (2011). *A new ANEW: Evaluation of a word list for sentiment analysis in microblogs.* Copenhegan: Information Retrieval (cs.IR); Computation and Language (cs.CL).