Second Progress Report

Final Year Project (2017/2018)

Sentiment-Aware POI search

Bachelor of Science in Computer Science

Hong Kong Baptist University

Cheung Siu Sin 16214684

Ma Yu Kan 16228073

Table Of Contents

1 Abstract

This proposal will discuss the problem, proposed solution, the role of the user, language used, hardware and software requirement and finally the distribution of the workload.

This project, Sentiment-Aware POI search, will separate into two parts, first of all, the part of searching the location required by the user based on the category of the buildings. This part is similar to the Google Map's searching part which also providing results based on user's location and the keywords given by the user.

So, what makes this project special? Besides of just searching the location just like the Google Map, we tend to offer a result which gives a more accurate and optimized result by making the searching become sentiment awareness and we will use Twitter as the source of the comments, which we will cover it up in the later passage.

2 The object of second progress report

By using this report, it will present the progress between the initial of the project to the second progress report.

As it is a group project, it will separate into two parts, one of the parts will mention the progress on the server and the construction of the website.

For the other part, it will mention about the sentiment analysis part of the project, which includes the reason for choosing Twitter, machine learning approach, the workflow of the program and the feature vector.

After that, we will talk about what are we going to develop after the second progress presentation and an abstract on how to execute it.

3 Progress made on the server and website

This part will discuss the progress made on the server side and the website side.

Server side

Setup the environment

The server is borrowed from the Computer Science Department which comes with the Linux version I chose before, but it only comes with a blank Ubuntu with no software install so that I managed to install the software needed for the development, for example, apache2, NodeJS and python3.

As we decided to develop the sentiment analysis program in the use of the python program language, python is installed for it. However, by using the traditional ways to start the program, the user need to keep the SSH connecting which is not practical and time wasting so that I chose the forever.js (https://github.com/foreverjs/forever) to keep the program from running and provide some additional functionality such auto retry when error occurred and collecting all the logs and output together for better debugging and less resource wasting.

Besides, Let's Encrypt being used to sign the SSL certificate and enable the server by using the secure connection (HTTPS). Instead of using HTTP, HTTPS can provide additional security while using the website.

Website side

As the program still not develop the communication between the program and the website so that the website will not provide any function with the program at this time. The website is to provide the prototype of the website design and give out a concept of how the website is going to work.

Google Map API

The **Google Map API** is used in the middle of the website will provide the map function of the website.

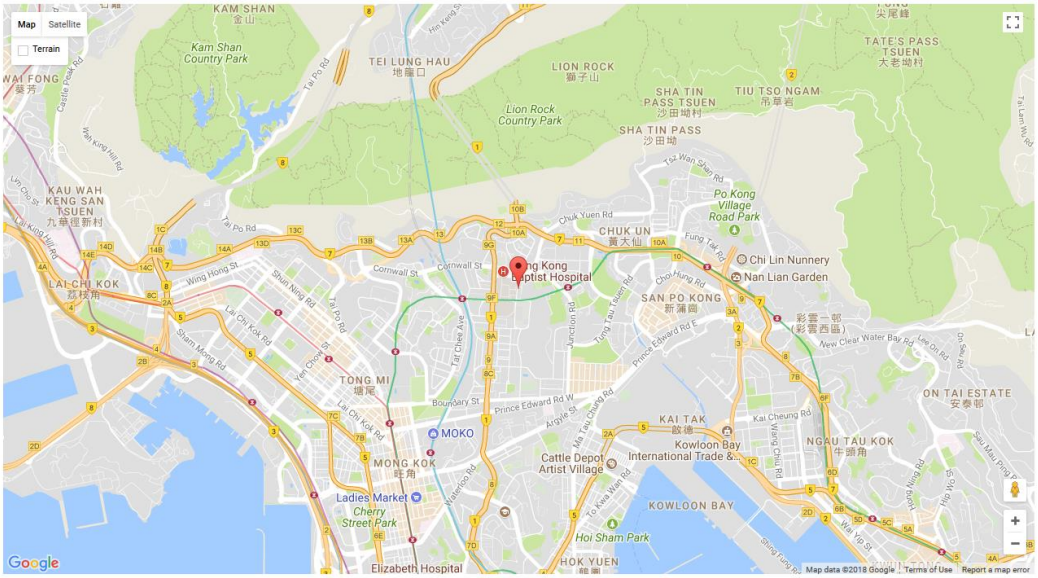


Figure 1 The Google Map spawn by using Google Map API

```
function initMap() {

    var uluru = {
        lat: 22.337877,
        lng: 114.181800
    };
    map = new google.maps.Map(document.getElementById('map'), {
        zoom: 14,
        center: uluru
    });
    var marker = new google.maps.Marker({
        position: uluru,
        map: map
    });
}
```

Figure 2 Code using to initialization the map

Beside of displaying the map on the website, the Google Map API can also be used to draw a circle on the map which indicates the radius of the searching.

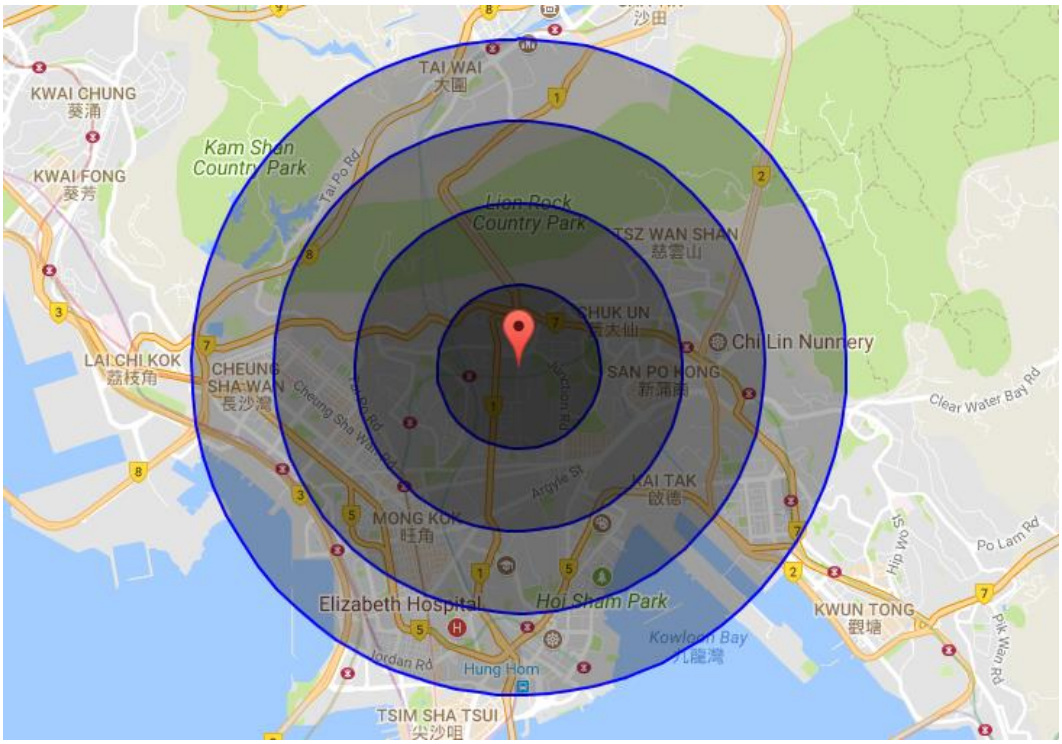


Figure 3 Screenshot showing the map drawing circle by using the Google Map API

```
function submitquery() {

    var lat = parseFloat(document.getElementById('lat').value);
    var lng = parseFloat(document.getElementById('lng').value);
    if(isNaN(lat) || isNaN(lng)){
        lat = 22.337877;
        lng = 114.181800;
        console.log("lat = "+lat+"\n"+"lng = "+lng);

    }
    var radius = document.querySelector('input[name="radius"]:checked').value;
    circle = new google.maps.Circle({
        strokeColor: '#0000FF',
        strokeOpacity: 1.0,
        strokeWeight: 2,
        map: map,
        center: {
            lat: lat,
            lng: lng
        },
        radius: radius * 1000
    });
    map.fitBounds(circle.getBounds());
```

Figure 4 Code using to draw the circle on the map

Bootstrap

Apart from using Google Map API to display the map on the website, Bootstrap also be used to build a responsive website. As its responsive grid system, the website can show responsively.

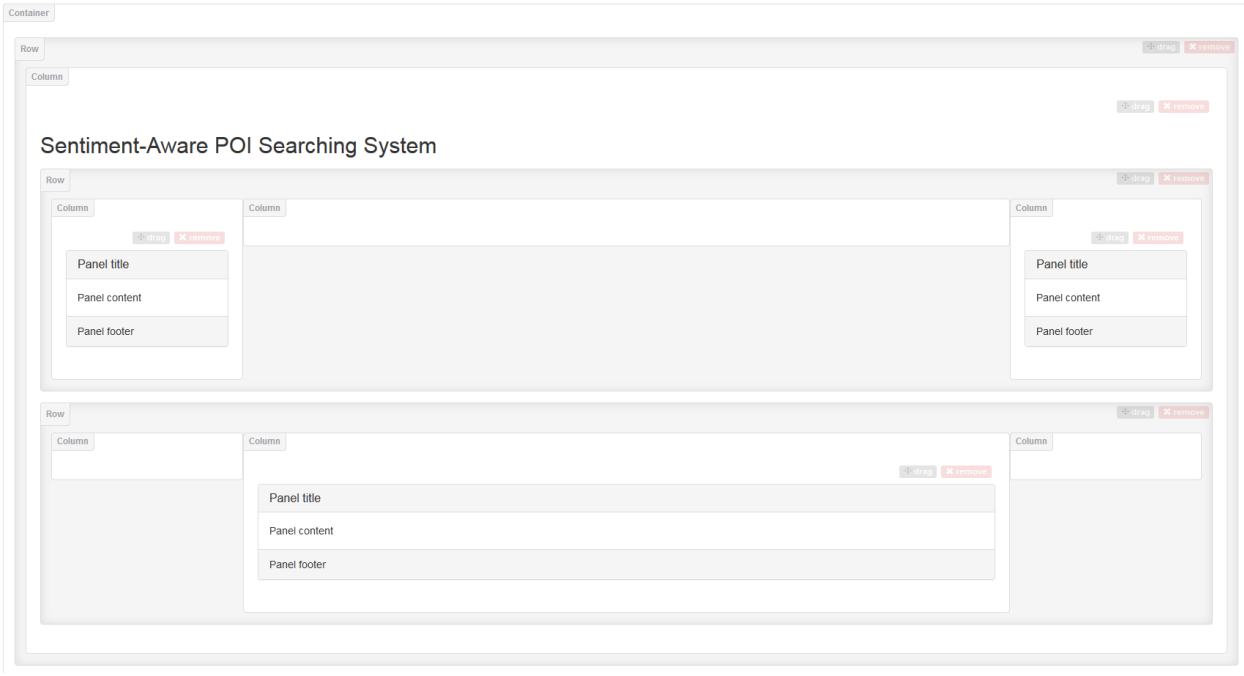


Figure 5 Screenshot of the bootstrap design of the website

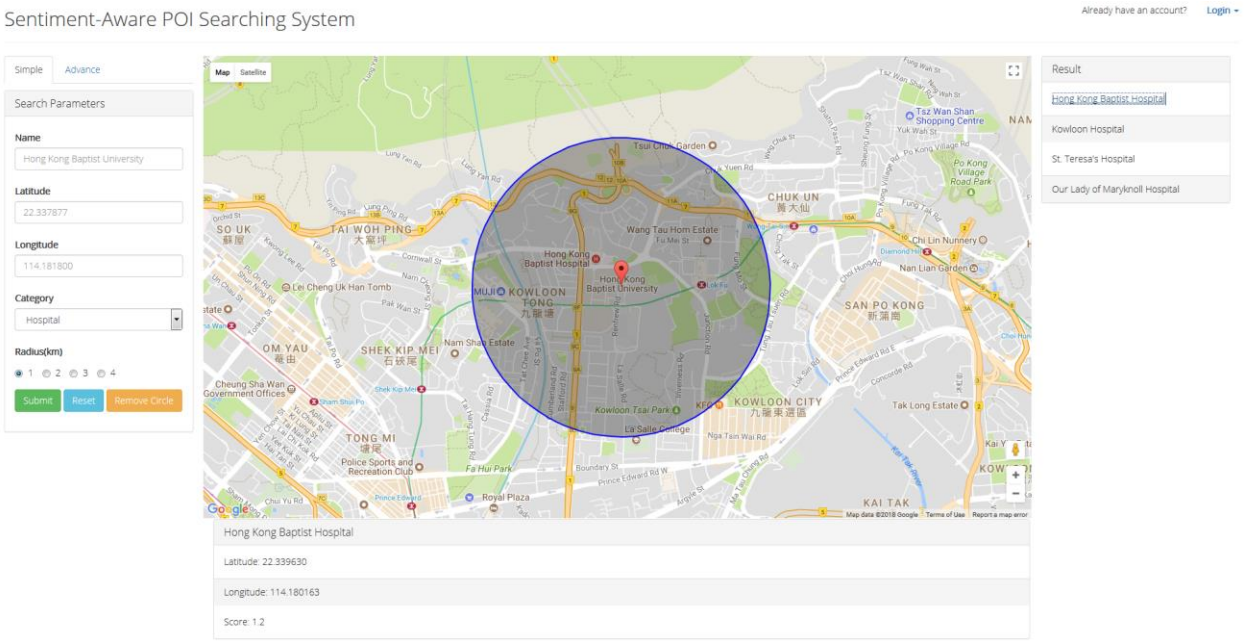


Figure 6 Screenshot of the current website

Google Font

Google font also being used to set the font face, Google Font provide a reliable source of a different kind of font

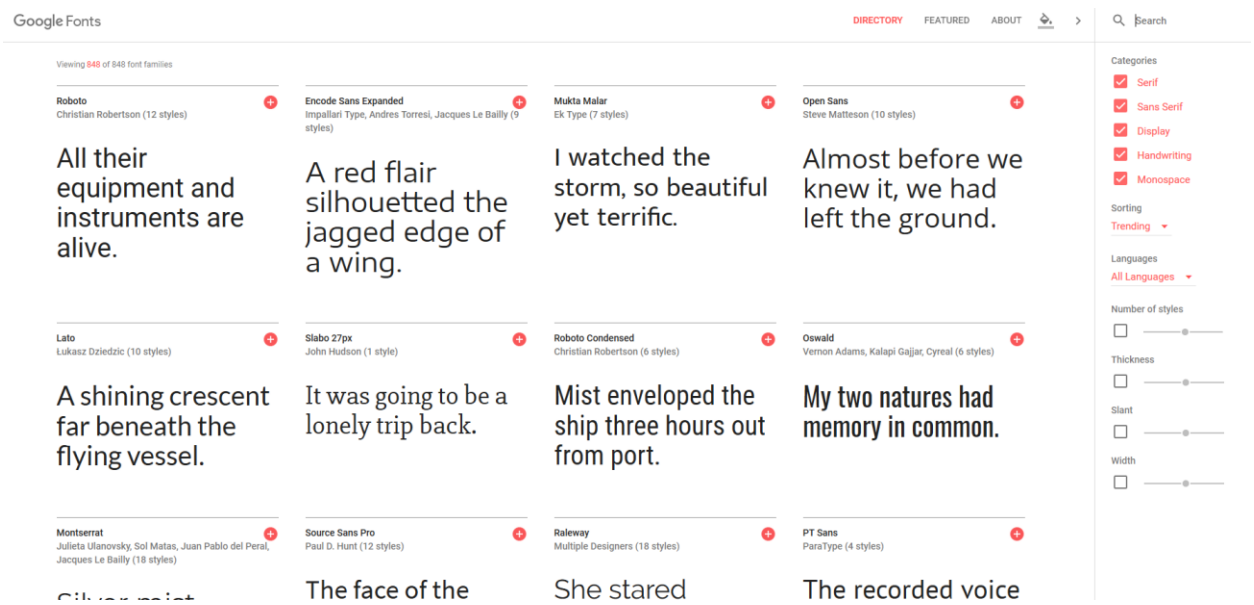


Figure 7 Screenshot of the Google Fonts

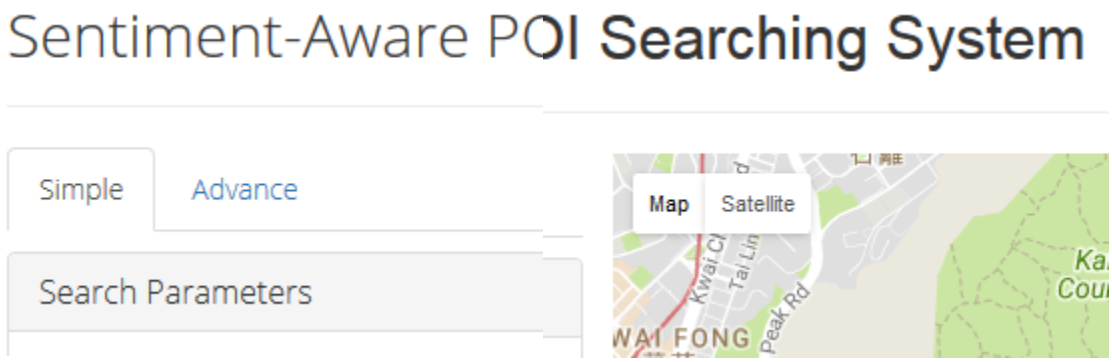


Figure 8 Example of the changes of the font face

The example above shows the difference between setting a custom font face and using the default font face. The screenshot on the left is set with custom font face and the screenshot on the right is using the default font face. By using the different font face, various style of the website can establish.

4 Progress made on the sentiment analysis

This part will discuss on why choosing Twitter, machine learning approach, the workflow of the program and the feature vector.

Reason for choosing Twitter

Nowadays, there are so many social platforms in our life, such as Facebook, Weibo and Twitter. From the perspective of Sentiment Analysis, we discuss a few characteristics of Twitter:

Length of a Tweet –A Twitter post is up to 140 characteristics. This means that a tweet is a single sentence practically, void of complex grammatical constructs. This is a difference from movie reviews of Sentiment Analysis.

The language used – People use Twitter via a variety of media, such includes mobile phone apps, website and SMS. Because of this and the 140-character limit, the language used in Tweets tend to be more colloquial and filled with slang and misspellings. And the hashtags are popularity gained on Twitter, and it is a primary feature in any way given a tweet. Each tweet is approximately 1-2 hashtags, as shown in below:

Figure 9 Example of a tweet

Data availability – For the Twitter API, we can be easy to collect millions of tweets for training. And there also provides a few labelled datasets that it is manually or automatically labelled tweets.

With Sentiment Analysis, our goal is to build an accurate Twitter Sentiment Analyzer for getting the top 10 of tweets on the user's needs topic. That is, the Sentiment Analyzer can determine the opinion of ***tweeples*** (people who use Twitter.com) on this topic as positive, neutral or negative, then the scoring function will calculate the strength sentiment by a sentence on this topic as a numeric range of -2(highly negative) to 2(highly positive).

Machine Learning Approach

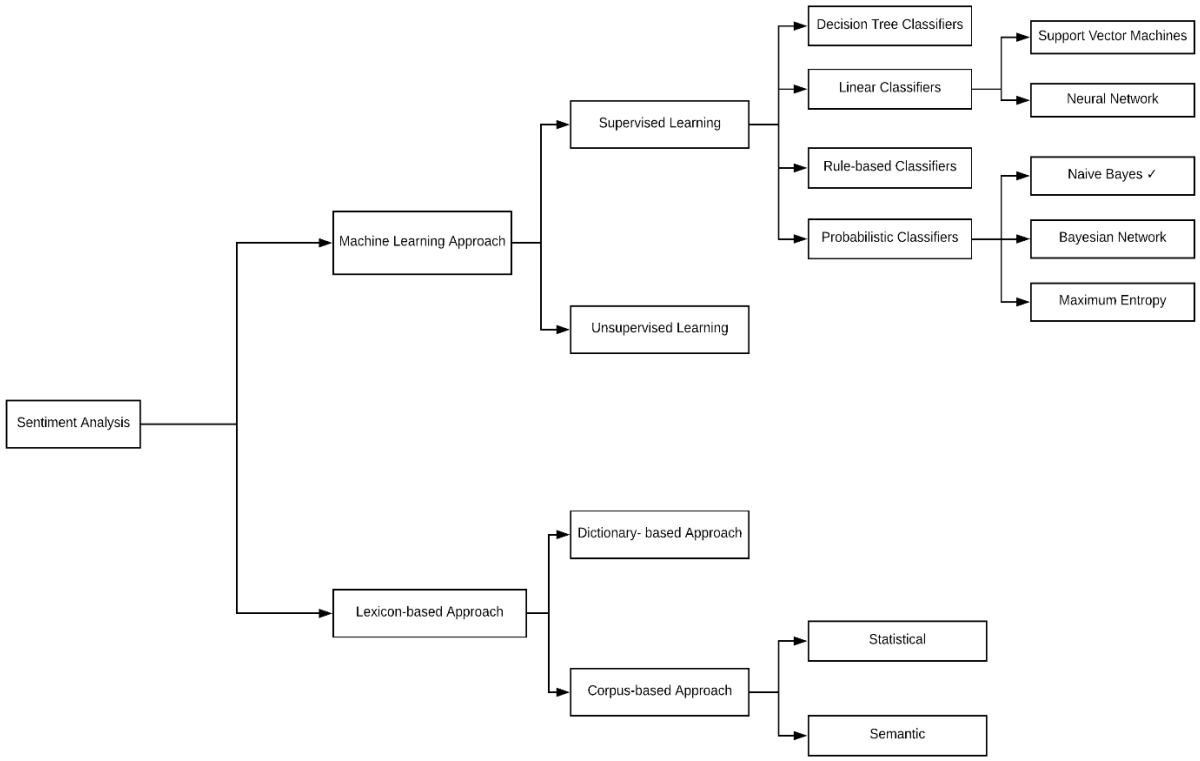


Figure 10 Types of Sentiment Analysis

In this project, I will adopt ***Naïve Bayes of Supervised Learning*** to achieve Sentiment Analyzer because most of the researchers claim to have gotten best results by using Naïve Bayes classifier, and it is the faster and simplest classifier.

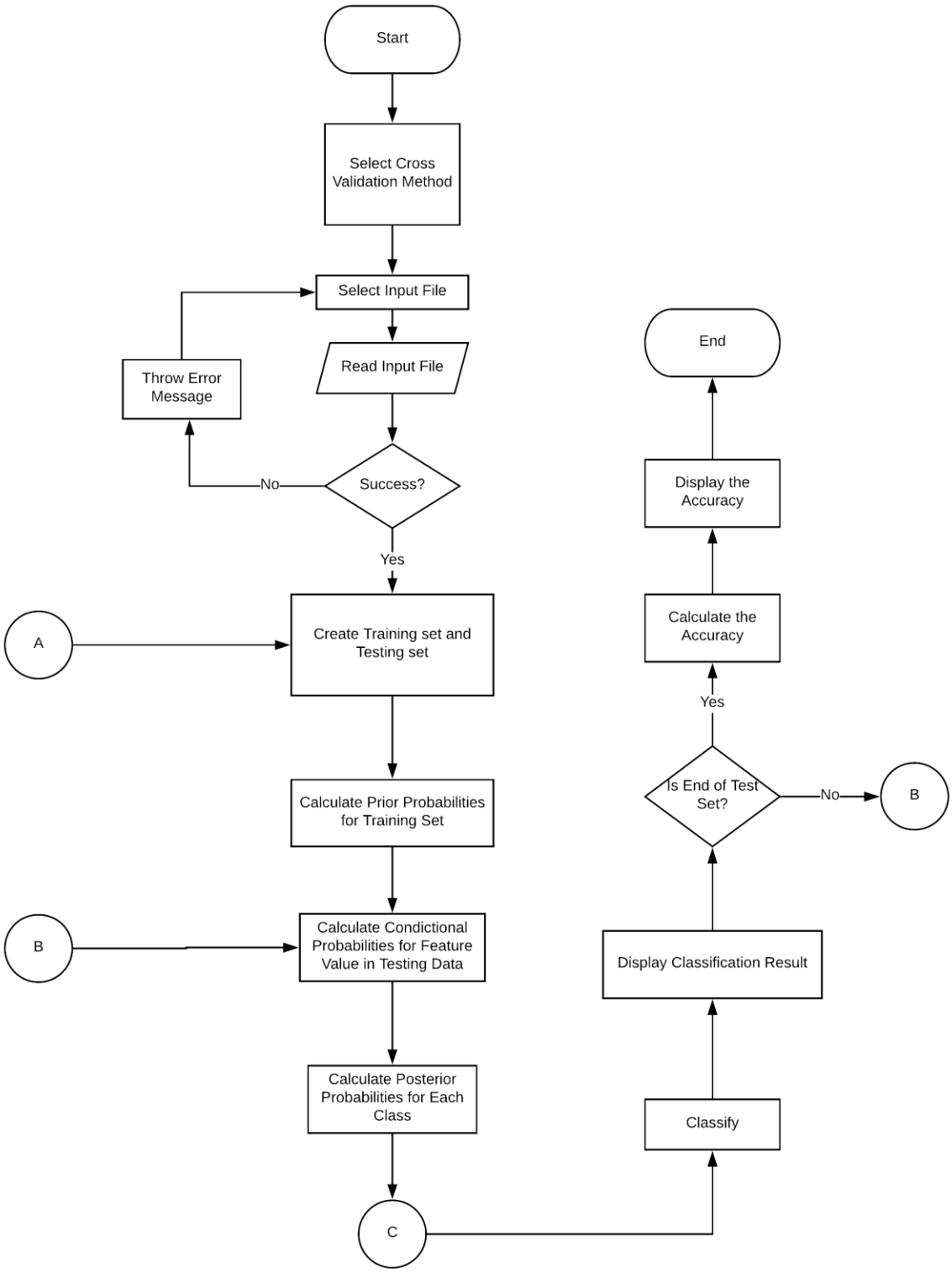


Figure 11 Flowchart for Naïve Bayes Classifier

I have studied to get that Naïve Bayes Classifier is based on the bag-of-words model, but there is the difference between them:

With the Bag-of-Words model, it checks the word of the text that whether it appears in a positive-words-list or a negative-words-list. If the word is on the positive list, the total score of the text will update +1, rather the total score will update -1. Thus, at the end, the total score is more than 0(positive), the text is classified as positive, if it is less than 0(negative), the text is classified as negative.

With the Naïve Bayes Classifier, it does not determine only for a small set of positive and negative words into account, but all words in a text will be trained with Naïve Bayes Classifier. The entire feature vector is the most important for Naïve Bayes Classifier, which is to combine each of the feature words of texts if a feature word is present, there will mark as 1, else marked as 0. So, we can think of each text as a bunch of 1s and 0s for computer reading, learning and classifying based on Naïve Bayes Model. And then Naïve Bayes model will build three Classifier of ***Positive Classifier***, ***Neutral Classifier*** and ***Negative Classifier*** based on the labelled dataset, each text as a bunch of 1s and 0s will be classified with this three Classifier respectively. Next, the program will compare the probabilities of them to get the maximum value as the sentiment.

Workflow of Programme

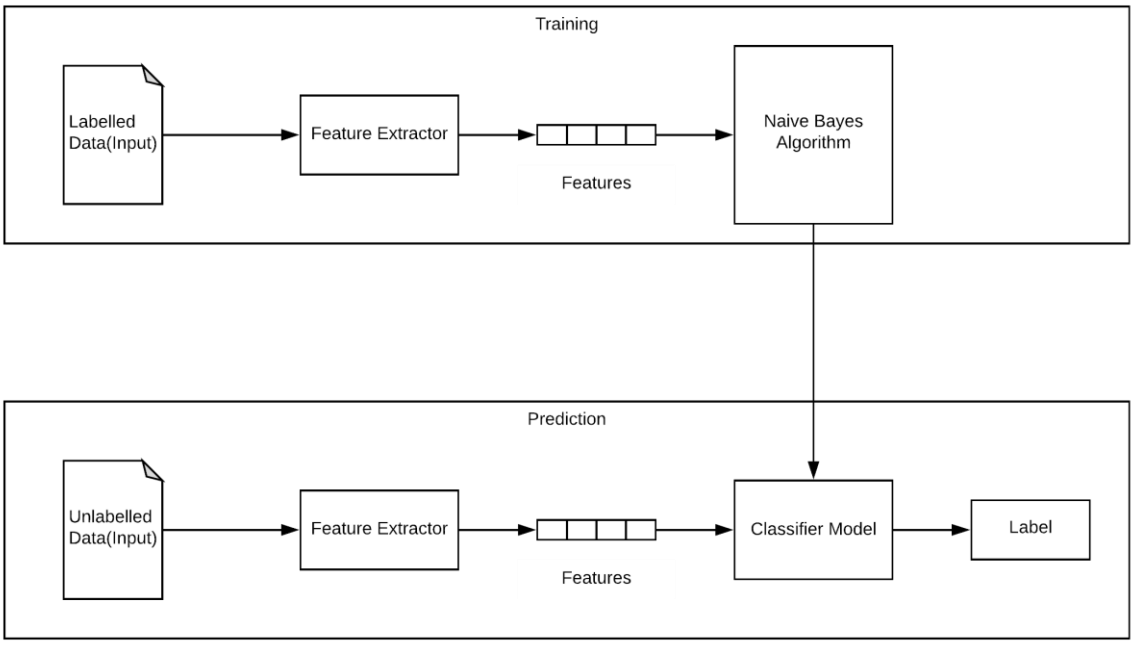


Figure 12 Workflow of the program

The Naïve Bayes Classifier must be trained with a labelled dataset, this dataset is manually made. In this project, I have collected a total of 21,603 tweets, such including 9667 tweets as ***positive***, 9666 tweets as ***negative*** and 2770 tweets as ***neutral***. These tweets are downloaded from a few public source, and then I was to format these tweets as the same file by programming. The topic of these tweets includes ***Apple***, ***Google***, ***Microsoft*** and ***Twitter***.

Pre-process Tweets

No matter training data or classifying tweets, there must pre-process the tweets, which aims to:

1. Convert the tweets to lower case.
2. Convert the hyperlink to a "URL" string because the content of the site does not be intended to follow and determined.
3. Convert the "@username" tag to AT_USER, the reason is similar to point 2.
4. Replace the "#hashtag" hashtag with the word because we can get the useful information by the hashtags, so we need to replace the useful tag with the exact same word without the hash, for example, #bitcoin replaced with "bitcoin".
5. Remove punctuation and the additional white space, e.g.: "it is a wonderful day!!!" replaced with "it is a wonderful day".

The code like:

```
#start process_tweet
def process_tweet(self, tweet):
    #Conver to lower case
    tweet = tweet.lower()
    #Convert https?://* to URL
    tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)
    #Convert @username to AT_USER
    tweet = re.sub('@[^\s]+','AT_USER',tweet)
    #Remove additional white spaces
    tweet = re.sub('[\s]+', ' ', tweet)
    #Replace #word with word
    tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
    #trim
    tweet = tweet.strip()
    #remove first/last " or 'at string end
    tweet = tweet.rstrip('\'"')
    tweet = tweet.lstrip('\'"')
    return tweet
#end
```

Figure 13 Codes for processing the tweets

Feature Vector

After processing, we need to find feature vector of tweets. But what is it? Feature vector is the most important concept in implementing a classifier, which directly determines how successful the classifier will be. The feature vector is used to build a model which the classifier learns from the training data and further can be used to classify previously unseen data.

I take a simple example to explain this ---- "Gender Identification". We often can distinguish others' gender by their names since there are some distinctive characteristics between Male and Female names, for example, names ending in *t*, *s*, *r*, *o* and *k* that are likely to be male, while names ending in *i*, *e* and *a* that are likely to be female. So, we can build a classifier based on this model by checking the ending letter of the names as a feature.

Similarly, and with above explanation, there can be used the present(marked as 1) or absence(marked as 0) of words that appear in a tweet as features. Thus, I will split each tweet into words and add these words to the feature vector, as well as I will remove some words that it may not indicate any sentiment in a tweet.

Thus, the filtering step aims to:

1. **Remove the unnecessary words** – These words have not any indicating sentiment and will be removed, for example, a, the, with, it etc. The StopWordList file is found in http://xpo6.com/list-of-english-stop-words/.
2. **Replace the repeating letters** – Sometimes we look at the tweets that the tweeples repeat some words to stress their emotion, like, so happyyyyyyyyy, haaaaaaappy for "happy". So, we will look for other repetitive letters in words and replace them.
3. **Remove punctuation** – this step is similar as task 5 of Pre-processing, but this step focus removing on single word, which is to remove the punctuation such as single/double quote, question marks comma and son on at the beginning and ending of each word, for example, "wonderful!!!!!" replaced with "wonderful".
4. **Make word start with an alphabet** – for example, "2nd", "11:30am", these words do not have any indicating sentiment, we will directly remove all those words.

The code of Filtering tweet words for Feature vector

```
#start replaceTwoOrMore
def replaceTwoOrMore(s):
    #look for 2 or more repetitions of character
    pattern = re.compile(r"(.)\1{1,}", re.DOTALL)
    return pattern.sub(r"\1\1", s)
#end
```

Figure 14 Codes on filtering tweet words

```
#start getStopWordList
def getStopWordList(stopWordListFileName):
    #read the stopwords
    stopWords = []
    stopWords.append('AT_USER')
    stopWords.append('URL')

    fp = open(stopWordListFileName, 'r')
    line = fp.readline()
    while line:
        word = line.strip()
        stopWords.append(word)
        line = fp.readline()
    fp.close()
    return stopWords
#end
```

Figure 15 Codes on filtering tweet words

```
#start getfeatureVector
def getFeatureVector(tweet, stopWords):
    featureVector = []
    words = tweet.split()
    for w in words:
        #replace two or more with two occurrences
        w = replaceTwoOrMore(w)
        #strip punctuation
        w = w.strip('\'"?,.')
        #check if it consists of only words
        val = re.search(r"^[a-zA-Z][a-zA-Z0-9]*[a-zA-Z]+[a-zA-Z0-9]*$", w)
        #ignore if it is a stopWord
        if(w in stopWords or val is None):
            continue
        else:
            featureVector.append(w.lower())
    return featureVector
#end
```

Figure 16 Codes on filtering tweet words

```
#start extract_features
def extract_features(tweet):
    tweet_words = set(tweet)
    features = {}
    for word in featureList:
        features['contains(%s)' % word] = (word in tweet_words)
    return features
#end
```

Figure 17 Codes on filtering tweet words

The Accuracy

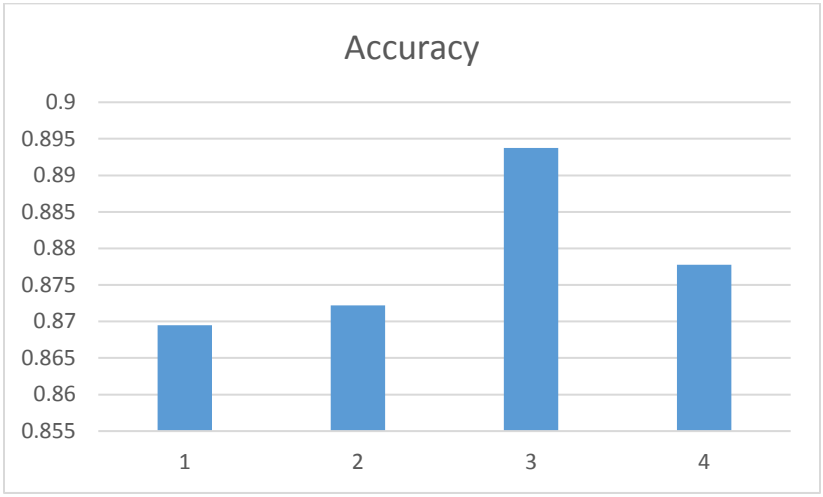


Figure 18 Graph of the accuracy

correct	18784	9637	7687	8221	
wrong	2820	1412	914	1145	
total	21604	11049	8601	9366	
accuracy	0.86946862	0.87220563	0.89373329	0.87774931	Average 0.87828921

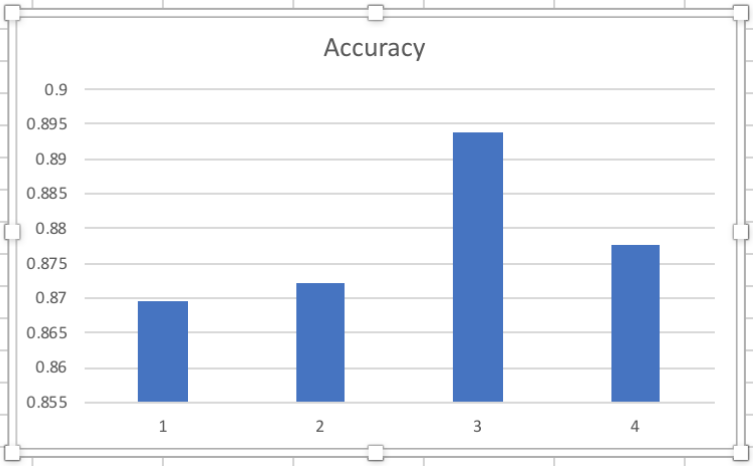


Figure 19 Average of the accuracy

As shown in figure 18 and figure 19, the accuracy is about classifying a few tweets based on Naïve Bayes Classifier, we can see that the average of accuracy is up to **87.83%**.

Sentiment Scoring

With the sentiment scoring, this part aims to let the user know that the strength sentiment in a tweet on a topic. I will adopt ***Joint Distribution*** to calculate a numeric range of -2 to 2. In the program, I will implement Natural Language Tool Kit(nltk) to count the speech tag as a corpus set, then find the probabilities of positive and negative of each tagging word on a topic. The formula is:

$$S = [P(\text{words}, \text{positive}) - P(\text{words}, \text{negative})] \times 2$$

P(words, positive) means a sum of the probability of each tagging word of positive, the probability of a tagging word = number of this tagging word in a corpus set of positive / amount of a corpus set of a positive * number of this tagging word in a tweet.

P(words, negative) means a sum of the probability of each tagging word of negative, the probability of a tagging word = number of this tagging word in a corpus set of negative/amount of a corpus set of negative * number of this tagging word in a tweet.

Universal Part-of-Speech Tagset

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Figure 20 Form of the universal Part-of-Speech Tagset

```
def countTag(self, para):
    sentense = word_tokenize(para)
    for i,j in nltk.pos_tag(sentense):
        if j in ['JJ', 'JJR', 'JJS', 'RB', 'RBR', 'RBS']:
            if(i in self.word_feature_tag):
                self.feature_word_tag_count[i] += 1
            else:
                self.word_feature_tag.append(i)
                self.feature_word_tag_count[i] = 1

    print "Speech tag: %s \nTag count: %s" % (self.word_feature_tag, self.feature_word_tag_count)
# end
```

Figure 21 Code of tagging

```
def SentimentScoring(self):
    f = open("bitcoin_tagging_words.txt","r")
    total_word_positive = []
    total_positive_tag_count = {}
    corpus_set_pos = 0

    total_word_negative = []
    total_negative_tag_count = {}
    corpus_set_neg = 0
    fl = f.readlines()

    for line in fl:
        word = line.split(":")
        if(word[0] == 'positive'):
            total_word_positive.append(word[1].strip())
            total_positive_tag_count[word[1]] = int(word[2])
            corpus_set_pos += int(word[2])
        else:
            total_word_negative.append(word[1].strip())
            total_negative_tag_count[word[1]] = int(word[2])
            corpus_set_neg += int(word[2])

    p_pos = 0.00 #probability of positive of word
    p_neg = 0.00 #probability of negative of word
    for w in self.word_tagging:
        if(w.strip() in total_word_positive):
            p_pos += float(total_positive_tag_count[w])/corpus_set_pos*float(self.word_tagging_count[w])

        if(w.strip() in total_word_negative):
            p_neg += float(total_negative_tag_count[w])/corpus_set_neg*float(self.word_tagging_count[w])


    score = (p_pos-p_neg)*2
    return score
```

Figure 22 Code of the sentiment scoring

Picture 14

```
Arons-MBP:Sentiment Analysis Aron$ python input_demo.py
Please enter a sentence for classifying : bitcoin debuted on this day in 2009. a year &amp;amp; a half later, a man named laszlo hanyecz is 1st tos
pend bitcoins in… URL
Started to instantiate Naive Bayes Classifier
Classifying naive bayes
Speech tag: ['later', 'laszlo', 'url']
Tag count: {'url': 1, 'later': 1, 'laszlo': 1}
negative | bitcoin debuted on this day in 2009. a year &amp;amp; a half later, a man named laszlo hanyecz is 1st tospend bitcoins in… url
score: -0.20
Arons-MBP:Sentiment Analysis Aron$
```

Figure 23 Result of tagging word and sentiment scoring on Bitcoin topic

And we have made a website to display the result after classifying with https://fypvm.problem.express:6143/FYP/result.html (shorten url: https://goo.gl/awBXje), the users can visit this link to see the result, and they can send feedback(s) to us that tell us which tweet is wrong by classifying. This can help us to collect the needs data conveniently and quickly.

Some pictures like below:

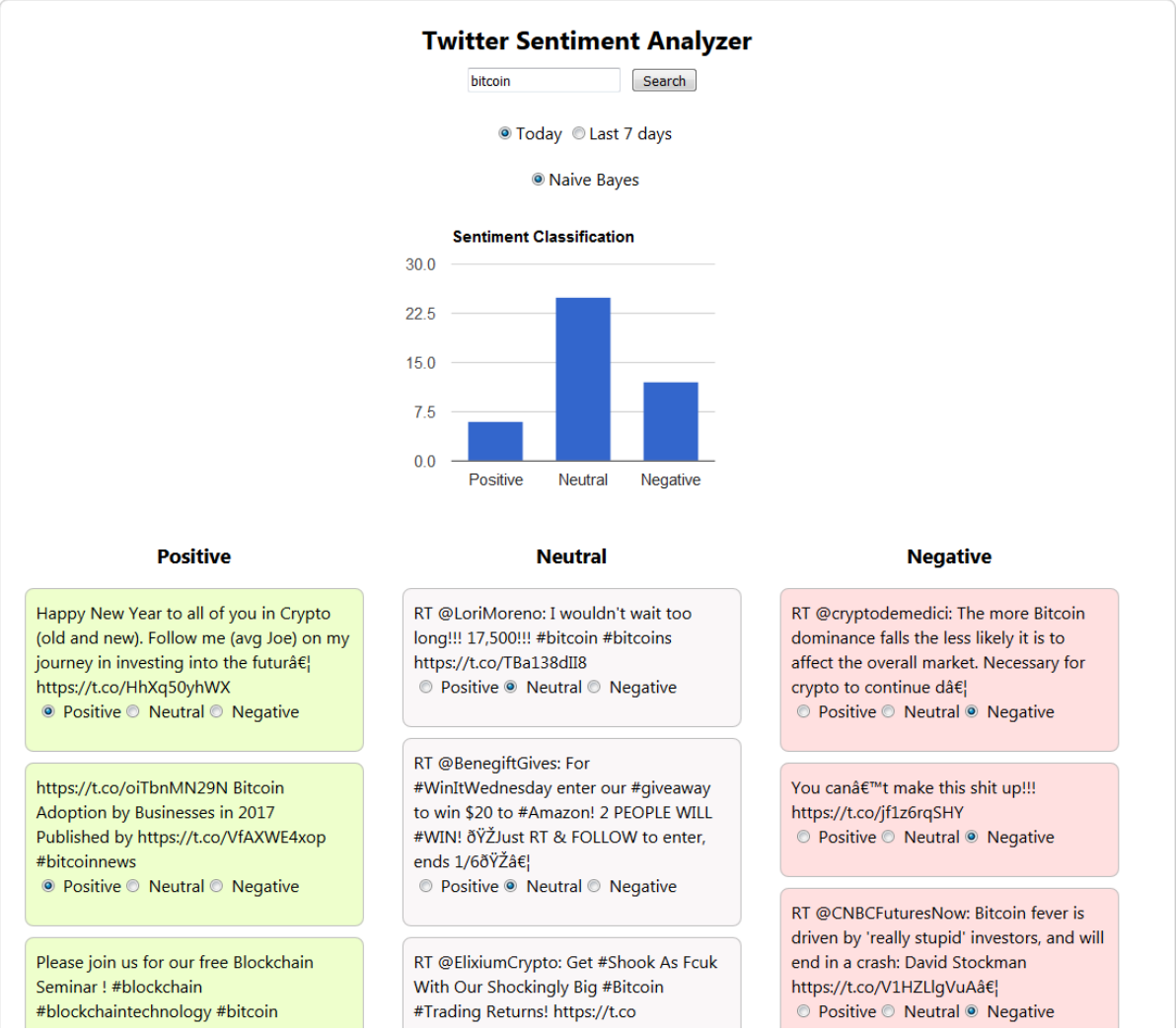


Figure 24 Screenshot of the Twitter Sentiment Analyzer

Figure 25 Screenshot of the Twitter Sentiment Analyzer

5 Future Planning

Website Part

More mobile responsive

As the current design of the website only works well on the computer, which will become broken layout when a show on the smartphone, one of the aims of the future development of the website is to make the separate design of the computer and the mobile.

It can be achieved by adding the side navigation menu which gives out some more space to the website while giving out the same amount of functions.

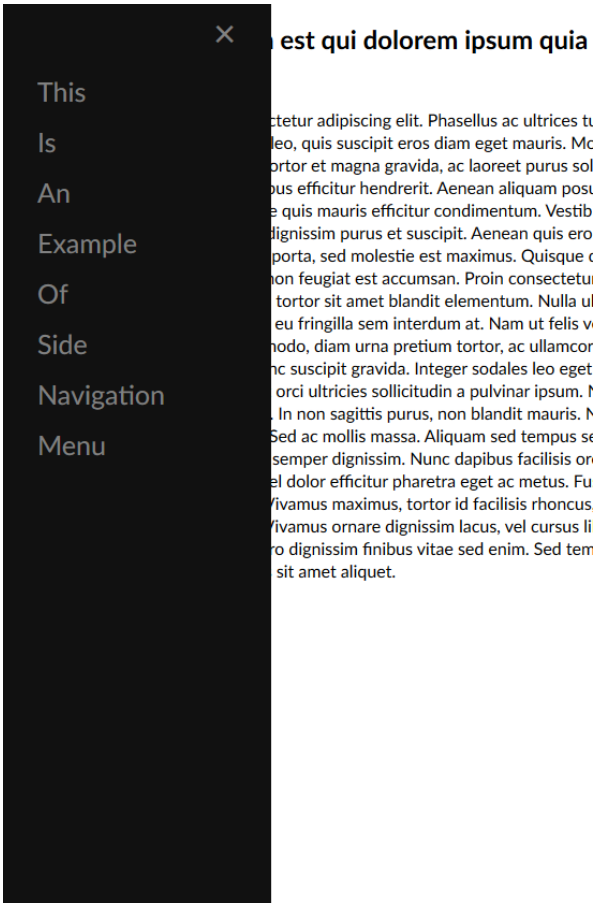


Figure 26 Example of side navigation menu

Implement the website with NodeJS MVC framework

Besides of making the website more mobile responsive, the website will also implement the use of NodeJS MVC framework into the website, by using the MVC framework, for example, Sails JS. By using Sails JS, it can automatically generate REST APIs and it can significantly decrease the time of the development.

Figure 27 Icon of Sails JS

Integration with the sentiment analysis part

As the current development is separated into two parts so that we will integration into a product and let it work together, the objective of the product is to search for the location and looking for the place at the view of the Twitter. JSON will be used to communicate with the program which serves as an APIs.

Provide login function

Login function will also provide the final product, by providing the login function, the system will able to record the searching of the user for them to search again easily, also it will enable the system to collect the data of the user behaviour and analysis it in order to give out more personalized user interface.

Sentiment Analysis Part

Collect the data on different category

As there are many types of location on the map, so we need to collect each type of data separately

Unsupervised Learning

As the current stage is using Naïve Bayes of Supervised Learning which is the simplest method to achieve it. Supervised learning needs the manual input data into the classifier to train in order to get a higher accuracy classifier model. However unsupervised didn't need manually labelled data to get a higher accuracy classifier model which will not limit the data inputted, it will keep brute force the answer to get a higher accuracy. It is a challenging task and it must increase the understanding of machine learning.

The requirement of the server

CPU: 4 Core CPU
RAM: 8 GB RAM
Hard Disk: 500G Disk Space
Operation System: Ubuntu Server 16.04.3 LTS
With Network Connection

The requirement of the client

Supporting Web Browser

Apple Safari 10 or above
Google Chrome 56 or above
Mozilla Firefox 55 or above

Support Smart Device

Apple iPhone with iOS 9 or above
Google Android 6 or above

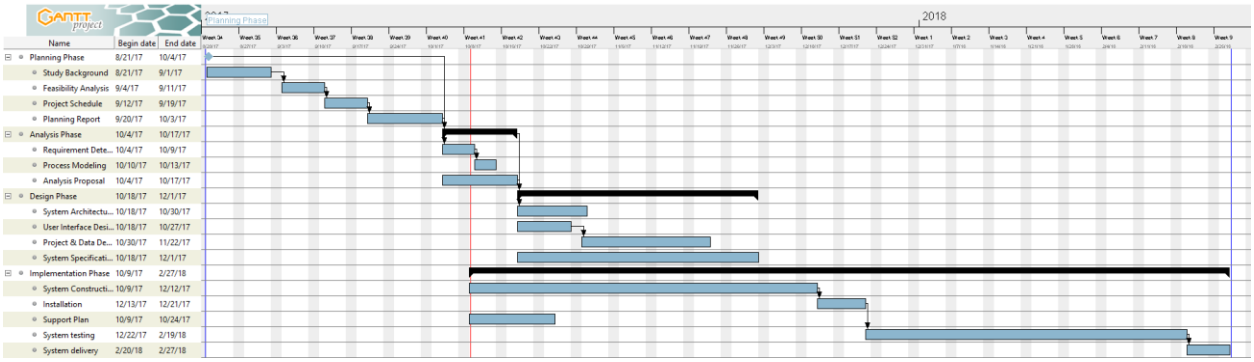
6 Detailed Project Plan

Task List

Task ID	**Planning**	**Duration (days)**	**Dependency**
A1	Study Background	10	
A2	Feasibility Analysis	6	(A1) Study Background
A3	Project Schedule	6	(A2) Feasibility Analysis
A4	Planning Report	10	
Task ID	**Analysis**	**Duration (days)**	**Dependency**
B1	Requirement Determination	4	
B2	Process Modeling	4	(B1)Requirement Determination
B3	Analysis Proposal	10	
Task ID	**Design**	**Duration (days)**	**Dependency**
C1	System Architecture Design	9	
C2	User Interface Design	8	(C1)System Architecture Design
C3	Project & Data Design	18	(C2)UI Design
C4	System Specification	33	

Task ID	**Implementation**	**Duration (days)**	**Dependency**
D1	System Construction	47	
D2	Installation	7	(D1)System Construction
D3	Support Plan	12	
D4	System testing	42	(D2) Installation
D5	System delivery	6	(D4) System testing

Gantt chart



Full-Size Photo: https://goo.gl/KQBMZR

7 Distribution of the workload

Cheung Siu Sin will responsible for the website part and the server side, for example, the user interface of the web application and the server structure.

Ma Yu Kan will responsible for the sentiment analysis part of this application, for example, the design of the algorithm, the implementation of the algorithm and the optimization of the algorithm.