Final Report

Social Network data capture and analysis tool

Department of Computer Science

QIANGDE XIANG

Student no: 169042630

Supervisor: YUDONG ZHANG

Date: 18.05.2018

Word count: 10110

Declaration:

I have completed and understood the online plagiarism tutorial and have read and understood the guidance provided on Blackboard regarding plagiarism. I know that the University takes plagiarism seriously and am aware of the University Senate Regulation 11: Regulations governing Student Discipline. I understand the penalty system in place and consequences for my academic performance should I plagiarise in an assignment. I understand that advice on plagiarism issues is available from my tutors and my Programme Director. I declare that all work submitted for the duration of this programme of study will be my own and that I will follow the department guidelines for correctly referencing all of my work.

Signature:

Date:

Table of Contents

1. Abstract

Information about social media users is becoming increasingly important in modern society. A considerable number of works have been engaged in proposing novel methods to predict the user attributes based on large data sets. This paper proposes a novel Logistic Regression based classification that is used to predict the gender of Twitter users. The training is based on four factors that contain hints about the targeted attribute. Comparison with other methods will be given and this work can be extended to predict other attributes of social media users.

Keywords: Twitter, attributes, classification, prediction, gender, machine learning.

2. Introduction

Social media applications such as Twitter and Facebook have become an important tool for human daily interaction. For example, in 2009, the number of Twitter users in the US had reached 8 million and the number for Facebook is 6 million [1]. According to Mislove [2], every second, numerous thoughts and feelings from millions of people all around the world are put forward in the form of tweets. Thus, this kind of rapid growth of social media has led to a large amount of online information related to the users themselves, which stimulate increasing academic interests on the social media data mining and the user demographic feature classification [3].

There is no doubt that the ability to predict a social network user's basic and private information including gender, age, regional origin and political affiliation has a significant influence on the area of advertising, personalization and customization. As Mislove [2] said in his work, Twitter users are very representative of the total population of a country and it is efficient to understand the whole human society by doing research on such a kind of big data. The typical examples include movie box office prediction, elections and stock market.

However, unlike Facebook or other social medias, some basic information such as gender, age and occupation are not available on Twitter. Actually, in recent years, researchers have begun to collect the username, the user profile and the text messages from Twitter and utilize various novel techniques to achieve accurate performance with regard to user attributes.

In my paper, the main task is: 1. To crawler the tweets according to keywords or hashtags. 2. To crawler the tweets according to user screenname and period. 3. To allow users to track real-time tweets according to keywords or hashtags. 4. To allow users to track real-time tweets in a certain amount or time. 5. To provide users the prediction of the tweets that are tracked or searched, including: the gender and sentiment. The structure of this paper is: the first section is Related Work, in which three previous works on Twitter data collection and prediction are illustrated. The second section is Requirements which aims to state the basic knowledge and necessary preparation before the project starts. The third section is Data Analysis. In this section, we will talk about the process of proposing a novel model to predict gender of Twitter users and how to train it with big data to achieve high accuracy. Then it comes to the fourth section Discussion and we focus on explaining why the performance of the model proposed in this paper is accurate and why other models are not used in this work. After that is the App Function section where we will present the whole web application and several figures will be given to show the user-friendly appearance. Next is the Techniques Involved section and in this section we illustrate every key technique that is used in the programming process in detail. Finally, it is the conclusion.

3. Related Work

In [1], they proposed an SVM based classification to predict four user attributes: gender, age, region and political tendency. For the gender prediction, they ignore the user name because a male user might use a female name or some names are difficulty to judge, such as Acadia, Kenyon and Rubens. Instead, they focus on the content and style of users' tweets. With regards to the language content processing, there are many models that are proposed by previous work. One of those models is SocioLinguistic-feature Model. For example, "uh-huh",

“umm” and laughter are more likely to come up from a female speaker. However, tweets are not like speech and the efficient sociolinguistic cues have to be changed. They collect a large number of microblogs and calculate the frequency of every word used by male and female users and finally get the probability of each item. In addition, they preserve emoticons and punctuations which are always ignored by other works. Moreover, it is worth mentioning that during the test process, they found that the follower-following ratio, the follower frequency and the following frequency have nothing to do with the gender attribute. The sample size is 1000 and the number of female user and male user is 500, respectively. At last, SVM is applied to classify the gender based on the parameters mentioned above. The accuracy of this novel method is 72.33%.

In [2], the researcher ignores the content of tweets posted by Twitter users but focus on the first names which they think is the most suggestive factor deciding the gender. They firstly collect 5,836 labelled names and then query the male and female names for babies born in America between 1900 and 2009. After that they calculate the possibility of each name. However, as Mislove discussed in his paper, there are limitations for this method: 1. A user might not write the true name in his/her profile. 2. Some names are quite misleading like the examples in the work by [1]. 3. Different areas have different cultures and the probability of each name might change with the district transferring. Thus, the final accuracy is only 64.2%.

In [3], the author realised the importance of accurate prediction and proposed a novel classifier with high performance. Given the fact that the previous works only achieve the accuracy of lower than 80%, the author used more factors that have relationship with the user gender. Usually, a Twitter user has a screen name (e.g. @uol), full name (e.g. University of Leicester) and profile (e.g. a mom of three cute kids). The full name and profile are optional and can be changed at any time, which means that only depending on one or two factors do not improve the performance significantly even under the condition of a large scale of data. Firstly, the research team collect 184,000 Twitter users with labelled gender. They did not adopt labor-intensive methods. Instead, they track the URL link of each user and get another personal web page of this user, on which the user has provided the gender information. If the user does not provide a URL link in Twitter profile or another web page does not contain the gender hint, then he/she will be ignored. After that, they found that among the 184,000 data, 55% are female and 45% are male, which is in line with the estimation by [6]. Then, they

produced a table in which the features that contain the information about gender are ranked by probability (e.g. $P(Female|features)$). Finally, they adopt fields combination method to find out an input vector with best performance. Depending on one tweet achieves the accuracy of 67.8, while combination of all tweets achieves 75.5. Moreover, the performance of prediction based on name is 77.1% and all those fields combined together is 81.4%.

Given these works above, the conclusion is that: 1. A large set of labelled data should be collected at first. It is because when we construct a new model, the performance of supervised training is much better than semi-supervised and unsupervised training. 2. There are many free APIs on the Internet which help us predict gender according to name (https://gender-api.com/) and text (https://www.uclassify.com/). The former one is based on the public governmental databases which record the names of babies born during the last century in the UK and the latter one is trained upon 11,000 blogs. However, their accuracy will be not higher than 80%, which has been discussed before. Thus, those two factors will become two elements x_1, x_2 of the input vector $\boldsymbol{x}$ in this paper. 3. When training a model and testing the accuracy, try different combinations of fields as many as possible so that the useful fields can be found and the useless fields can be excluded.

4. Requirements

4.1 Basic knowledge about machine learning.

a. What is machine learning: The name of machine learning first come up in 1959 by Arthur Samuel [4]. It is mainly about the study and construction of algorithms from sufficient learning data and prediction of targeted attributes. It does not require us to know exactly how human beings interact with the surroundings, how human recognise objects or how human learn from the nature. What we need to do is just to define a set of functions and collect a large amount of data. Then with the help of computers, we can decide the best function which make the loss function least from the set.

b. How to predict the user attributes based on machine learning: There are many inputs that we can get when we crawl the Twitter with the Twitter Java API. For examples, the frequency that a user post tweets, the total number of tweets from one user, the punctuation the user uses and the key words that occurs in the user's tweets can all be the inputs. Then we transfer the information into quantitative digits and allocate them in a vector $\boldsymbol{x}$. Define a likelihood function $p(c|\boldsymbol{x})$ and label all the data (e.g. if the user is male, then the output is (1,0), else (0,1)). Under the sufficient training data, we can train our function model and find a best set of weights that minimize the loss function.

4.2 Basic knowledge about supervised, semi-supervised and unsupervised data training.

a. Supervised training: All the training data is labelled. For example, every record in database corresponds to a user and the targeted attributes have been obtained. The techniques proposed in previous works are all supervised training. In this paper, 1,000 users are collected from the internet and all the data is labelled. Thus, supervised training is utilized here.

b. Semi-supervised training: Only a small portion of data is labelled. It means we can not directly use the labelled data to train our model and the performance will be disappointed if we do not consider the unlabelled part. Usually we will utilize the self-training and smooth assumption to optimize our model.

c. Unsupervised training: None of the training data is labelled and most cases belong to unsupervised and the performance is usually not as accurate as supervised training.

4.3 Basic knowledge about deep learning.

a. What is deep learning: Deep learning is a subclass of machine learning. It uses multiple layers of nonlinear processing units for the purpose of feature extraction and transformation. Each layer's outputs are the inputs of next layer. The whole model then looks like $f_n(\boldsymbol{w} \cdot f_{n-1}(\dots f_2(\boldsymbol{w} \cdot f_1(\boldsymbol{w} \cdot \boldsymbol{x}))))$ where $f_1, f_2, \dots, f_n$ present the nonlinear function, $\boldsymbol{x}$ presents the input vector and $\boldsymbol{w}$ presents the matrix. The system is also called neuron network because it is similar to our neuron system and stimulus can be transferred and transformed layer by layer. Different levels of neurons correspond to different levels of abstraction.

b. Why deep: Firstly, we hardly know how we learn the objective world when we grew up and it is impossible for us to find out a learning process to model. The only thing we know is that our neurons in the brain are arranged layer by layer and information is transformed and processed through multiple layers of neuron network. Deep learning model is like the structure of human neuron system and with sufficient training data and gradient descent, a model of accurate performance will be generated. Secondly, in the work of conversational speech transcription using deep neural network by Seide [5], the author compared single layer which is fat and short with multiple layers which is thin and tall. When the structure is 2 layers with 2,000 neurons per layer, the word error rate has been decreased to 20.4% while the figure is 22.1% with 1 layer of 16,000 neurons. When the layer size reaches 7, the performance is 17.1%. The well-known AlphaGo is consisted of more than hundreds of layers. In this paper, deep learning will not be utilized because of the fact that the performance is not improved dramatically when utilizing deep learning.

4.4 Basic knowledge about Logistic Regression.

a. Regression model is a popular method to describe the relationship between an output variable and multiple input variables. Regression methods include linear regression and logistic regression [7].

b. Linear Regression is one of the most common examples of modelling whose output is a continuous scalar. A typical case using linear regression is the credit score calculation of bank system. Given the age, income, mortage and other information of a loan applicant, the credit score which indicate the credit rating of this applicant can be obtained through linear regression. Collecting a large set of applicants with the labelled credit score as the output scalar y and putting all the information provided in a vector $\boldsymbol{x}$ as the input, where $y = f(\boldsymbol{x})$, an optimal function f can be found by minimizing the loss function $\sum(y - f(\boldsymbol{x}))^2$.

c. Unlike Linear Regression, the output of Logistic Regression is discrete, taking the gender prediction as an example where the output for male and female is (1,0) and (0,1), respectively. Thus, the process to obtain the best function f is quite different from that in Linear Regression. Similarly, given a large set of Twitter users with the labelled gender (1,0) or (0,1) as $\boldsymbol{y}$' and

putting all the information provided in a vector $\boldsymbol{x}$ as the input, let's assume $\boldsymbol{y} = f(\boldsymbol{x})$, where $\boldsymbol{y}$ is the output possibility (if $\boldsymbol{y}[0] > 0.5$, then the gender is male, otherwise the gender is female). Then the loss function becomes $\sum ln(\boldsymbol{y} \cdot \boldsymbol{y}')$ and the best function f can be obtained by minimizing the loss function.

d. Many distribution functions have been simplified as functions with output of discrete variables. Thus, logistic regression gradually becomes the most popular solution because it is flexible and efficient. In this paper, the Logistic Regression will be utilized for the gender prediction.

4.5 Good Python programming skill.

a. To solve the minimum problem of loss function and pick the best function, we need to utilize gradient descent. We use Python to program.

b. For non-deep learning, we set a reasonable learning rate and the result can be easily obtained. For deep learning we need to solve the complicated function with backpropagation.

c. Fortunately, there are many toolkits that we can utilize to make the progress much easier like Keras 2.0 which is used for deep learning part in this paper. In addition, the PC should have GPU to speed up the calculation process.

4.6 Good Java programming skill.

The final work is a website that allows users to register, log in and use and the web app should be user-friendly.

a. Firstly, the register page should collect detailed user information including birthday, first name, last name, email, password, postcode and so on. Then the register form should validate all the information provided by users before the registering request is sent to the server. After registration, the result should be returned to the client and the page should be redirected to the main page.

b. Secondly, the tweets searching form should be well designed to enable user to search tweets according to several conditions. Then the searching result area should look artistic, including the twitter user picture, the user screenname, the content of tweets, the publish time and the number of retweets. Moreover, the prediction function should be user-friendly, allowing users to know the prediction result of targeted twitter accounts, the sentiment of comments under a selected tweet, the gender percentage of a certain topic and the positive/negative percentage of a certain event.

c. Finally, the most important function is the tracking div area. It should allow users to track real-time tweets according to any topic, any user in a certain amount and during a certain period.

5. Data Analysis

In [8], the author illustrates the gender differences shown by social media statistics. The study illustrates that women are more active than men, which means that the frequency of tweets by female is higher than that by male. Moreover, favourite frequency is another factor that indicates the activeness of Twitter users. Before, the follower/following radio, the follower frequency and the following frequency has been proved to have no relationship with gender. Therefore, in this paper, these three factors will not be taken into account. Instead, the tweet frequency and favourite frequency will become two elements of the input vector $\boldsymbol{x}$. Given the fact that the text content of tweets and the name of user have strong hints of gender, thus the gender probabilities according to text and name will be the other two elements of the input vector $\boldsymbol{x}$. Consequently, there will be four elements named text prediction, name prediction, tweet frequency and favourite frequency in the input vector $\boldsymbol{x}$ in the model proposed in this paper.

In order to achieve high performance, as it is discussed in the requirement part, the gender prediction will apply supervised training. The key step is to collect a large set of labelled data. Fortunately, the csv file recording the information of twitter users including their gender is

available on https://www.kaggle.com/ and the data size is 18,4000. In the csv file, each line records the user screenname which is unique, the gender, the profile, the background colour and so on. We only take two columns: the screenname and the gender. With the help of Twitter4j API which is an interface provided by Twitter to help developers to collect Twitter data, the name of the user, the tweet frequency and favourite frequency are easily obtained.

We firstly consider the accuracy only by name and text. We send the first name of every user to the government name database (https://gender-api.com/) and get the probability of this user being male. If it is larger than 50%, this user is male, otherwise, it is female. Then by comparing the result with the 184,000 labelled data, we get the accuracy of name prediction which is about 68.6%. Similarly, we collect 30 tweets of every user, combine them together and send them to https://www.uclassify.com/ which provide the web service to classify gender according to microblogs. The accuracy of text prediction is 51.3%. Figure 1 is the java program to calculate the accuracy of name prediction and text prediction.

```
//this class is used to test the accuracy of prediction only by name or text

public class DataTest {

    public static void main(String[] args) throws IOException{

        //test the accuracy of name prediction and text prediction
        //accuracy of name prediction is 68.6% in sample size of 1000
        //accuracy of text prediction is 51.3% in sample size of 1000
        //and in sample size of 1500, 2000 or more, the accuracy does not change significantly

        //if name prediction is correct, this variable++
        int name_correct = 0;
        //if text prediction is correct, this variable++
        int text_correct = 0;

        //train1000.csv means in this csv file there are 1000 records.
        File file = new File("C:\\Git\\train1000.csv");
        FileReader reader = new FileReader(file);
        //read in every line
        CSVParser parser = new CSVParser(reader, CSVFormat.EXCEL);
        List<CSVRecord> records = parser.getRecords();
        int count = 1;
        for(;count<records.size();count++){
            CSVRecord record = records.get(count);
            //column 3 is the result by name prediction
            double name_predict = Double.parseDouble(record.get(3));
            //column 1 is the result by text prediction
            double text_predict = Double.parseDouble(record.get(1));
            //column 6 is the labelled gender
            String result = record.get(6);
            //if name prediction is correct
            if((name_predict<0.5&&"female".equals(result))||(name_predict>0.5&&"male".equals(result)))
                name_correct++;
            //if text prediction is correct
            if((text_predict<0.5&&"female".equals(result))||(text_predict>0.5&&"male".equals(result)))
                text_correct++;
        }

        System.out.println(name_correct*1.0/count+"--"+text_correct*1.0/count);
    }

}
```

Figure 1. The code to test the accuracy of name and text prediction

The results also support the view of previous work that predicting gender only upon name or text is beyond enough. Recall the Logistic Regression, it fits the gender prediction because the output is discrete and the input is a vector $\boldsymbol{x}$. Let's suppose that the probability of being a

male upon the input $\boldsymbol{x}$ is $p(male|\boldsymbol{x})$. Then $p(male|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|male)\cdot p(male)}{p(\boldsymbol{x}|male)\cdot p(male)+p(\boldsymbol{x}|female)\cdot p(female)}$. The equation can be simplified like: $f(male|\boldsymbol{x}) = \frac{1}{1+e^{-z}}$, where $z = \ln\frac{p(\boldsymbol{x}|female)\cdot p(female)}{p(\boldsymbol{x}|male)\cdot p(male)}$. Actually, the simplified equation is a sigmoid function and z can be simplified as $\boldsymbol{w}\cdot\boldsymbol{x}$ in [7]. Since all the data have been labelled with male as (1,0) and female as (0,1), the loss function can be written as $\sum \ln(\hat{y}\cdot y)$, where $\hat{y}$ is the prediction result $(p(male|\boldsymbol{x}), p(female|\boldsymbol{x}))$ and y is the true gender vector (1,0) or (0,1). Finally, the best $\boldsymbol{w}$ can be obtained by minimizing the loss function. We use Weka for the Logistic Regression process. Waikato Environment for Knowledge Analysis, known as Weka, is a tool for machine learning in Java. Weka is widely used in many different application areas, including data preprocessing, data analysis, predictive modelling, clustering, classification and regression. The introduction and the process of using Weka will be talked later.

We allocate 500 records of the total data as the test data and name the file as test500.csv. Then we extract another 1000 records of the total data as the training data and name the file as train1000.csv. Firstly, we use logistic regression to train the 1000 records with only two elements in the input vector: the prediction result upon text and the prediction result upon name. The training file is train1000with12.csv and the testing file is test500with12.csv. Figure 2 is the code for training and testing process:

```
//below is to train a model and test the accuracy
String train_path = "src/main/resources/train1000with12.arff";
String test_path = "src/main/resources/test500with12.arff";
//labelled index
int train_class_index = 2;
int test_class_index = 2;
ArffLoader arffLoader = new ArffLoader();
arffLoader.setFile(new File(train_path));
Instances instances = arffLoader.getDataSet();
instances.setClassIndex(train_class_index);
int train_total = instances.numInstances();
int train_right = 0;
Logistic logistic = new Logistic();
logistic.buildClassifier(instances);
//this loop is to calculate the training accuracy
for(int i=0;i<train_total;i++){
    Instance instance = instances.instance(i);
    if((logistic.distributionForInstance(instance)[0]>0.5&&instance.classValue()==0)||(l
        train_right++;
}
System.out.println("Training Accuracy is "+1.0*train_right/train_total);
arffLoader = new ArffLoader();
arffLoader.setFile(new File(test_path));
instances = arffLoader.getDataSet();
instances.setClassIndex(test_class_index);
int total = instances.numInstances();
int right = 0;
//this loop is to calculate the testing accuracy
for(int i=0;i<total;i++){
    Instance instance = instances.instance(i);
    if((logistic.distributionForInstance(instance)[0]>0.5&&instance.classValue()==0)||(l
        right++;
}
System.out.println("Testing Accuracy is "+1.0*right/total);
```

Figure 2. The code to calculate the training and testing accuracy of Logistic Regression

The result then is in Table 1:

Table 1. The accuracy of Logistic Regression with text and name prediction combined

Logistic Regression	Training	Testing
Text and Name	79.20%	78.98%

Then we apply logistic regression for the gender prediction upon tweet frequency and favourite frequency, which means there are two elements in the input vector. The training file is train1000with34.csv and the testing file is test500with34.csv. The result is in Table 2:

Table 2. The accuracy of Logistic Regression with tweet and favourite frequency

Logistic Regression	Training	Testing
Tweet & Favourite Frequency	54.45%	54.2%

After that, we begin to combine all the four factors together and apply logistic regression to train the model, which means there are four elements in the input vector. The training file is train1000.csv and the testing file is test500.csv. The result is in Table 3:

Table 3. The accuracy of Logistic Regression with 4 factors combined together

Logistic Regression	Training	Testing
All the four factors	81.48%	81.60%

The accuracy has been improved above 80 percent. We also use python to apply the deep learning model and the result is not as accurate as expected. The reason is the difficulty of setting the appropriate parameters for every layer. Thus, we discard deep learning in this project. Moreover, the reason why we did not apply Naïve Bayes is that we tried it but the performance is not as good as Logistic Regression.

Table 4. The accuracy of Naïve Bayes with 4 factors combined together

Naïve Bayes	Training	Testing
All the four factors	80.10%	79.40%

As we can see in Table 4, the testing accuracy is not higher than 80 percent. Another question is the sample size. The reason why we choose the 1000 as the training size is that we trained the model with the size of 100, 200, 300, 400, 500, 800, 1000, 1500 and 2000. The accuracy versus the sample size is plotted Figure 3 and thus we choose 1000 samples as the training data.

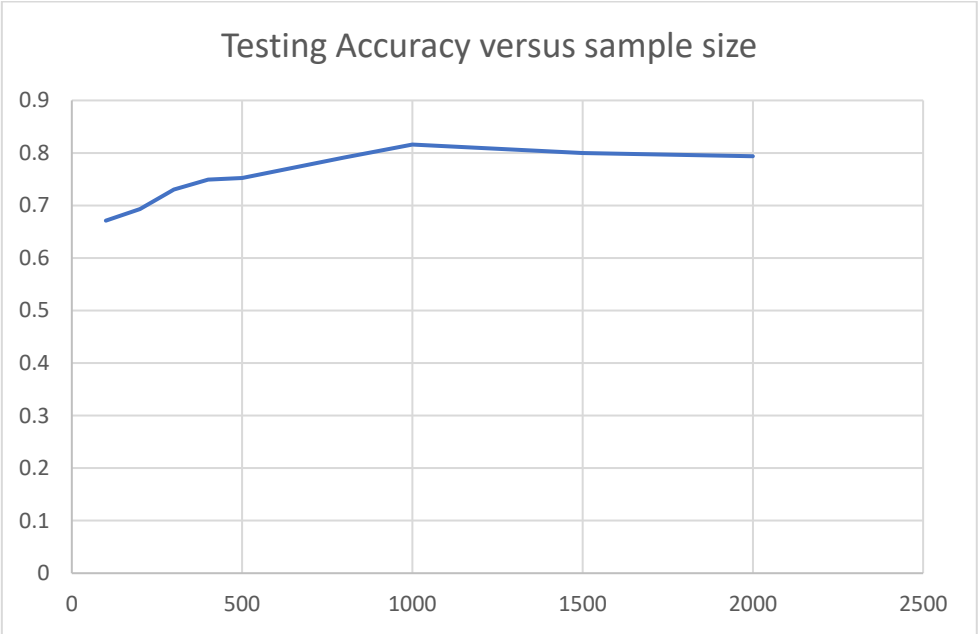

Figure 3. The accuracy of testing data versus the selected sample size

6. Discussion

In this project, we directly apply Logistic Regression $f(male|\boldsymbol{x}) = \frac{1}{1+e^{-z}}$, where $z = \boldsymbol{w} \cdot \boldsymbol{x}$. Then we apply Naïve Bayes for comparison. The performance of the latter one is approaching 80 percent but not as accurate as Logistic Regression. Both of them are used as a classification of data. The Naïve Bayes assumes all the features x_i in $\boldsymbol{x}$ are independent and estimates the joint probability of every element from input vector $f(male|\boldsymbol{x}) = \frac{f(male)\cdot f(\boldsymbol{x}|male)}{f(\boldsymbol{x})}$, where $f(\boldsymbol{x}|male) = \prod f(x_i|male)$. Once the assumption is not true or features are dependent, the performance of Naïve Bayes will not be accurate as expected. Thus, Naïve Bayes is a generative model. Compared with Naïve Bayes, Logistic Regression is a discriminative model

and it directly predicts the result from the training data without any assumption. Therefore, it is reasonable to apply Logistic Regression rather than Naïve Bayes or Gaussian Distribution in this project because we are not able to tell which distribution the features are following or whether they are independent or not.

At first, we calculate the accuracy of the prediction upon text and the prediction upon first name, which are 51.3% and 68.6%, respectively. As previous work illustrated, prediction based on text or name is unpersuadable. Then we combine the two factors together and apply logistic regression to train the model. The training accuracy is above 80 percent and the testing accuracy is approaching 80 percent. Therefore, we believe that the text or name might be confusing but the two factors together are precise to predict gender.

Moreover, we train the model only considering the tweet frequency and favourite frequency. Even though [8] and [10] both point out that female social media users are more active and post more microblogs, this inference is still not proved by Twitter users. During our training process when we combine the tweet frequency and favourite frequency together, the training accuracy is 54.45% and the testing accuracy is only 54.2%. The conclusion about these two factors are: they contain some information about gender but not as much as that by text or name.

At last, we combine the four factors together in the input vector $\boldsymbol{x}$ and the training and testing accuracy are both above 80 percent. Then the expected performance is achieved and we use this model to predict gender in this project. When a prediction request is sent to the server, the server will calculate the tweet frequency and favourite frequency of the Twitter account, collect the name and 30 recent tweets and get the prediction result upon name and text, and finally return a result to client.

7. App Function

Figure 4 is the main page of this app. In the navagation area, the “Home” button makes it convenient for users to return the initial page. The “Twitter” button redirects users to the https://twitter.com and the “Dev” button links to the twitter application website where

twitter developers should apply for their own access key and token. The “Doc” button takes users to the Twitter4j website where they can have access to the tutorials on crawling twitter data in Java. In the “FQ” list, there two items which are “About Us” and “Forgotten Password”. The “About Us” links to the pdf of the project description and the “Forgotten Password” links to the “getback.html” where user can get back their forgotten password through registered email. On the right side of the navigation area is the Log in and Register function. Registered users should use their registered email and password to log in our system so that they can user all the functions properly.

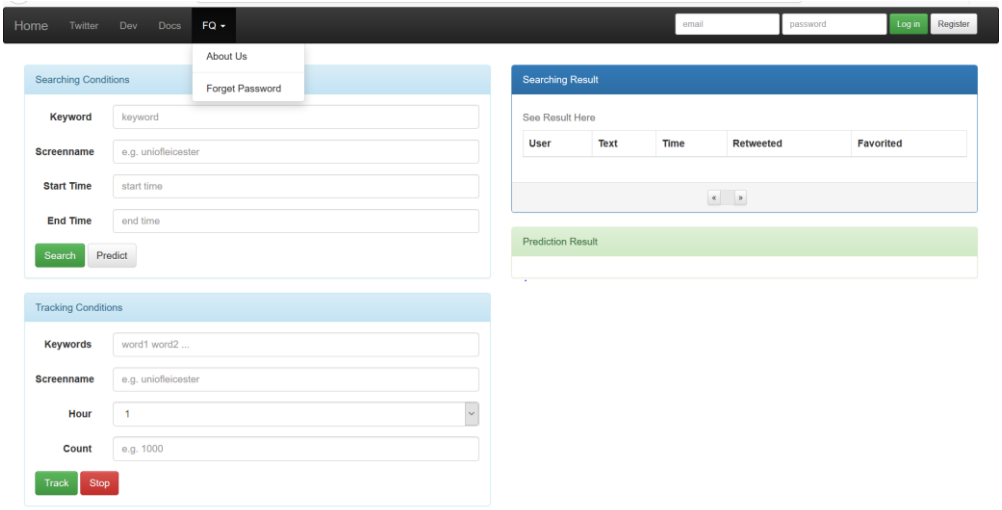

Figure 4. The main page of the project

If the email or password do not match the record stored in database, the error message will be returned to the client page like figure 5.

Figure 5. The error reminder of the log in interface

Figure 6 is the “register.html” page. It allows new users to sign up with personal information such as first name, last name, gender, birthday, email, telephone number, password and postcode of current address. Users have to fill in all the required information correctly and completely. The register page applies jQuery validation function and the request will not be

sent to the server unless the form passes the jQuery validation. For example, any empty input field will make the validation fail, unvalid or repeated email and telephone number will trigger the warning reminder, and the input field of “Password Again” which does not match the password field will also cause the error. When all the input fields validate, clicking “Register” button will send the data to the server and user will be redirected to the log in page if registered successfully.

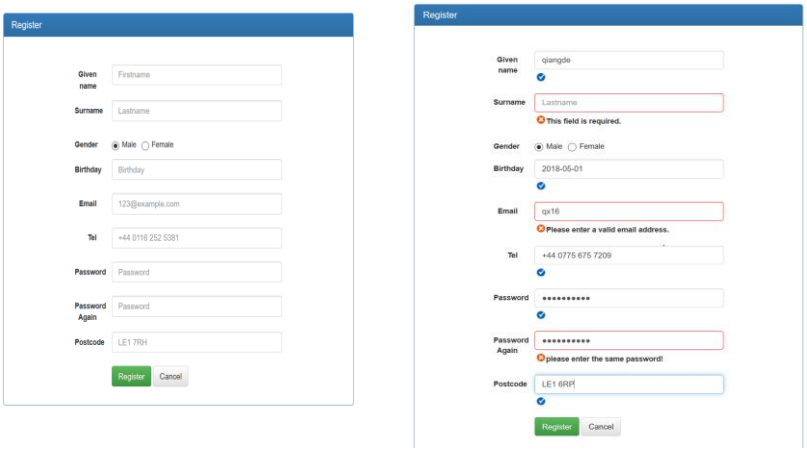

Figure 6. The register page and the jQuery Validation API

After user registers successfully, he/she can log in and the page then looks like figure 7:

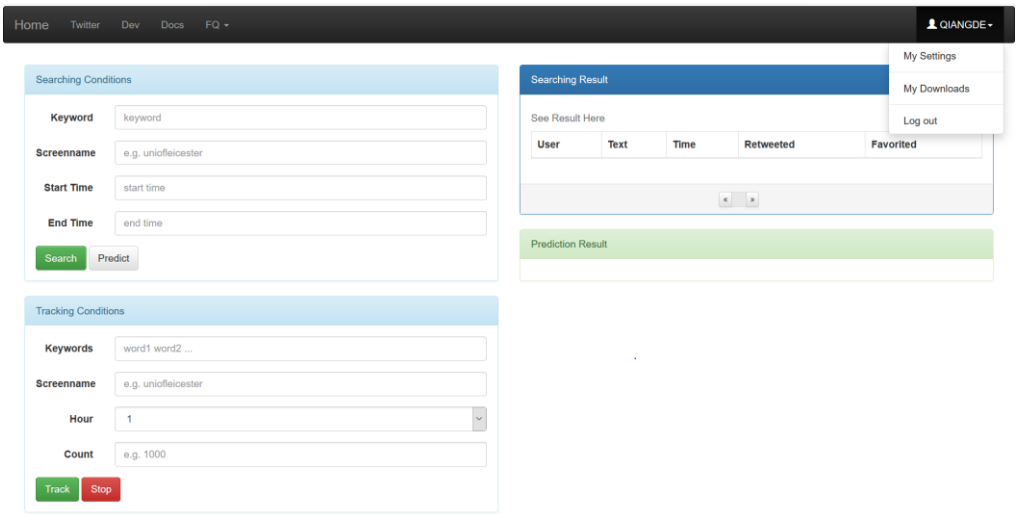

Figure 7. The main page after login

On the right of the navigation area, user can click the “My Settings” button to check and update his/her personal information. The “My Downloads” button is used to download the csv files which contain the tweets that are tracked under the conditions set by user and the “Log out” button is to allow user to log out. Figure 8 is the personal settings page:

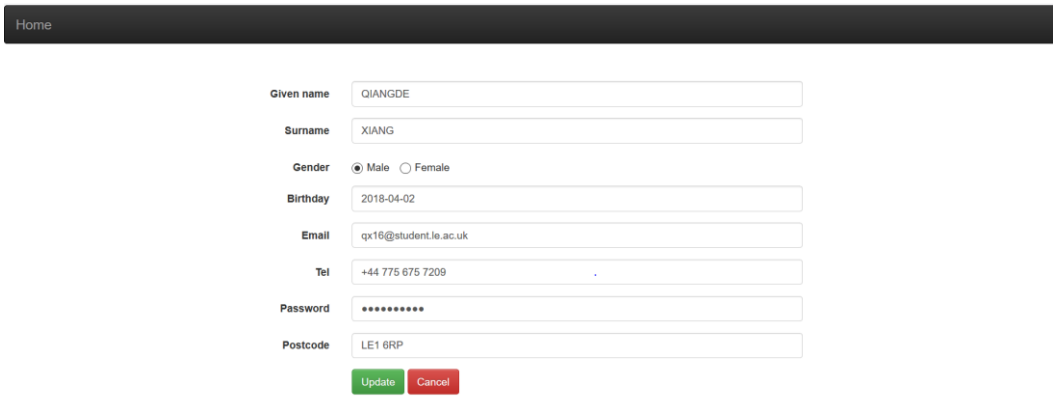

Figure 8. The personal settings page

In the searching area, the input fields include keyword, screenname, start time and end time. Keyword allows users to search tweets that contain the keywords or hashtags, while the screen name provide users the function to get the tweets by targeted Twitter account. The start time and end time enables users to filter the tweets during a specific period. Finally, the searching result will be shown in the “Searching Results” area and looks like figure 9:

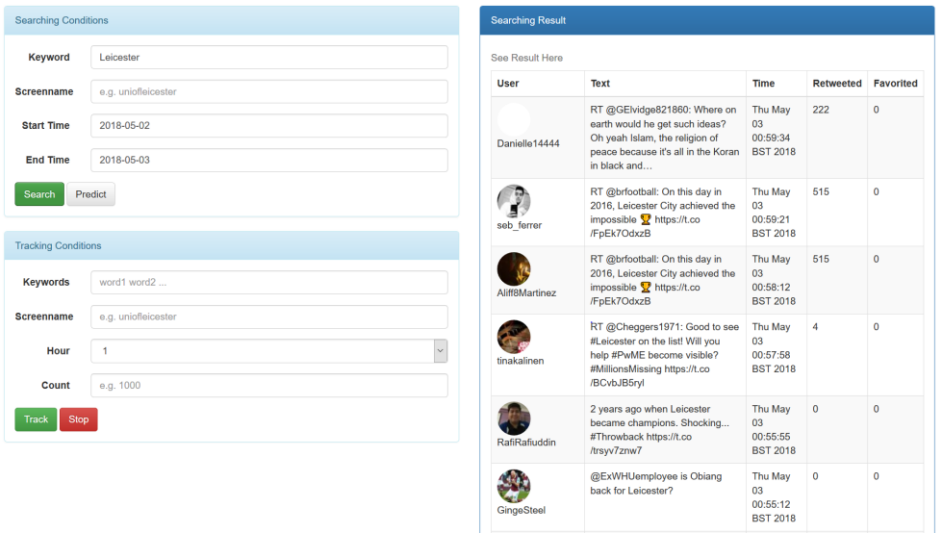

Figure 9. The searching function and result

With respect to the “Predict” button, it is used to predict the gender and sentiment of all the tweets that are searched by user. For a certain searching condition, there might be more than 1000 tweets and how many of them are written by male and how many are by female. Moreover, users might want to know the sentiment analysis result of so many tweets. The prediction result will be shown in the “Prediction Area” like figure 10.

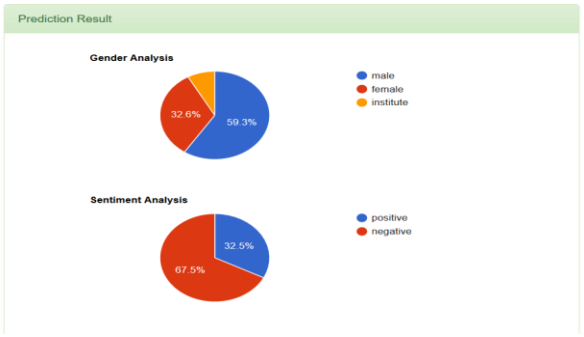

Figure 10. The prediction result

In the tracking area, the input fields include keywords, screenname, hour and count. Keywords refer to the multiple keywords or hashtags that is contained in tweets tracked by user and screenname refers to a specific Twitter account that user wants to track. The hour means how long a user want to track tweets from now and on, while the count means the total number of tweets user needs to collect. If user starts to track, a tracking thread will start and client page will receive the reminder:

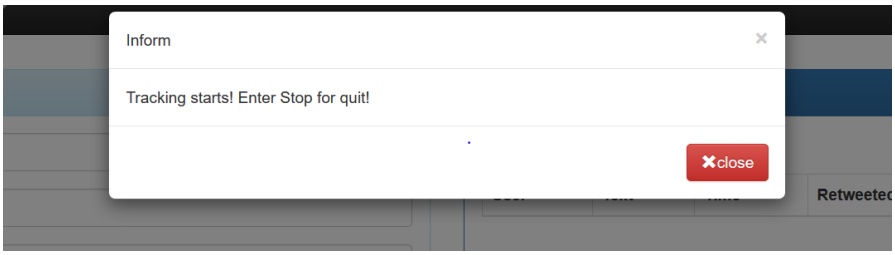

Figure 11. The tracking start reminder

If a tracking thread has started, the system does not allow user to start another tracking task at the same time unless the former task has finished. If user tries to start another tracking, he/she will receive the reminder:

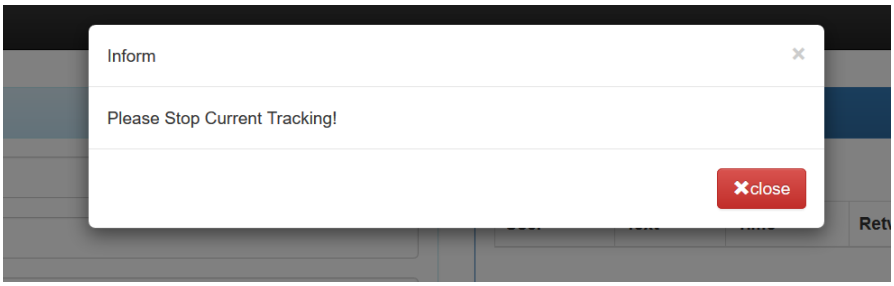

Figure 12. The tracking error reminder

What he/she can do is stopping the current task and then a new tracking task can be executed by server. To stop current tracking thread, user has to click the stop button.

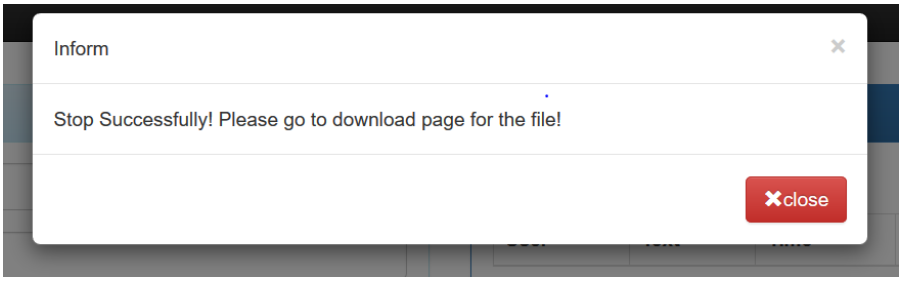

Figure 13. The tracking stop reminder

If a tracking thread finishes itself or user clicks stop button to quit it, a csv file which contain all the tracking information will be generated and user can go to the “My Downloads” page to download it. The “delete” button is the interface for user to delete the tracking record csv file.

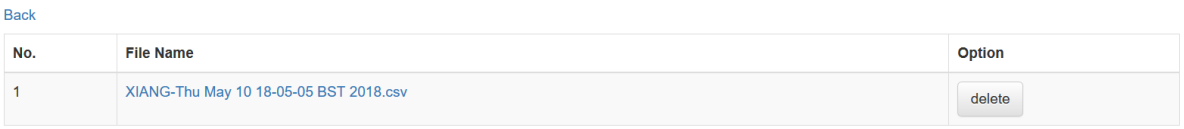

Figure 14. The tracking records page

8. Techniques involved

8.1 Spring MVC framework

Spring MVC is also known as Spring Web MVC, which is built on the Java Servlet API. Before Spring MVC become popular, web developers use Servlet and JSP to develop dynamic web services. Web developers need to configure the web listeners and filters in web.xml and deploy a lot of classes that implements Servlet interface, which makes the job tedious. Thus, several standard frameworks based on Servlet API come up, one of which is Struts. However, Struts can not interact well with Spring Framework which is a popular container to manage instances. In Spring MVC framework, the class DispatcherServlet that implements Servlet interface and contains a Spring container solves the two problems above. It dispatches requests to handlers and then to the corresponding methods. The dispatching and mapping process is based on the @Controller and @RequestMapping annotations. Such a mechanism enables developers to write flexible methods to response the requests from clients and developers do not need to write servlets one by one any more. Moreover, it supports RESTful web style by the use of @PathVariable annotation. The deployment is also very convenient and what developers need to do is to deploy DispatcherServlet in web.xml as shown in figure15.

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    <display-name>personal project</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <!-- listeners -->
    <listener>
        <listener-class>uk.ac.le.qx16.pp.util.MyUtil</listener-class>
    </listener>
    <!-- filters -->
    <filter>
       <filter-name>CharacterEncoding</filter-name>
       <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
       <init-param>
           <param-name>encoding</param-name>
           <param-value>UTF-8</param-value>
       </init-param>
       <init-param>
           <param-name>forceEncoding</param-name>
           <param-value>true</param-value>
       </init-param>
    </filter>
    <filter-mapping>
       <filter-name>CharacterEncoding</filter-name>
       <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!-- servlets -->
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

Figure 15. The web project configuration file web.xml

As DispatcherServlet contains a Spring container which manages instances so that it supports all the functions that come from Spring framework. For instance, it supports IOC (inverse of control) to allow developers to inject an instance managed by Spring container into the attribute of another instance and developers can use @Autowired annotation to implement IOC. It also helps developers with the management of transactions. In the past, developers have to open transaction, commit data, rollback operations, close transaction manually. Thanks to Spring framework, developers can use @Transactional annotation to ask the container to manage the transaction automatically. Figure 16 is the example in this project to deploy a web path controller and interact with database with Spring MVC framework.

```java
@Controller
@RequestMapping(value="/user")
public class UserController {

    @Autowired
    private UserService userService;
    @Autowired
    private TweetsService tweetsService;

    @RequestMapping(value="/login")
    public String login(String email, String pwd, HttpServletRequest req){
        User user = null;
        try{
            user = userService.userLogin(email, pwd);
            if(user!=null){
                req.getSession().setAttribute("user", user);
            }else{
                req.setAttribute("error", "Email or Password incorrect!");
            }
        }catch(RuntimeException e){
            req.setAttribute("error", "Some errors!");
        }
        return "forward:/index.jsp";
    }

    @RequestMapping(value="/register",method=RequestMethod.POST)
    public String register(User user, HttpServletRequest req){
        try{
            userService.userRegister(user);
            req.setAttribute("reg_result", "Successfully Registered!");
        }catch(RuntimeException e){
            req.setAttribute("reg_result", e.getMessage());
        }
        return "register_result";
    }

    @RequestMapping(value="/logout")
    public String register(HttpServletRequest req){
        req.getSession().removeAttribute("user");
        req.getSession().setAttribute("twitter", null);
        req.getSession().setAttribute("query", null);
        return "redirect:/index.jsp";
    }
```

```java
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    UserRepository userRepository;

    public void userRegister(User user) {
        // TODO Auto-generated method stub
        userRepository.save(user);
    }

    public User userLogin(String email, String pwd) {
        // TODO Auto-generated method stub
        return userRepository.findByEmailAndPwd(email, pwd);
    }

    @Override
    @Transactional
    public User updateUser(User user) {
        // TODO Auto-generated method stub
        User u = userRepository.findOne(user.getId());
        u.setBirth(user.getBirth());
        u.setEmail(user.getEmail());
        u.setFirstname(user.getFirstname());
        u.setGender(user.getGender());
        u.setLastname(user.getLastname());
        u.setPostcode(user.getPostcode());
        u.setPwd(user.getPwd());
        u.setTel(user.getTel());
        return u;
    }
}
```

Figure 16. The example of Spring MVC controller and service

8.2 Bootstrap

Nowadays, people use mobile phone more than computers, which means that traditional web page development based on PC size is not considered as the first order any more. In the past, front-end engineers write JavaScript and CSS to make the web page dynamic and vivid. With the help of jQuery which is a fast, small and feature-rich JavaScript library, HTML document elements operation and manipulation becomes much easier than before. Moreover, the way to send AJAX and handle events has been improved to a large degree. Therefore, millions of engineers have changed their way to write JavaScript. Figure 17 is the jQuery code example in this project.

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootst
<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datepick
<script src="https://www.gstatic.com/charts/loader.js"></script>
<script src="js/index.js"></script>
```

```
$(function(){
    $("#register").click(function(){
        window.location.href="pages/register.html";
    });
    $("#start").datepicker({
        format:"yyyy-mm-dd",
        autoclose:true
    });
```

Figure 17. The example of jQuery in the project

Here we import jQuery file at first, then write jQuery code in index.js file. In the index.js file, we see the original JavaScript code “document.onReady = function(){……};” has been simplified as “$(function(){……});” and the event handling like “document.getElementById (‘register’).onclick = function(){……}” has been simplified as “$(‘#register’).click(function{……})”.

However, jQuery does not solve the problem that the page is not fit when the screen size changes and the CSS process is still very labour-intensive and that is the reason why Bootstrap comes up. Bootstrap is a screen size responsive, mobile-first and front-end component library based on jQuery and now it has become the most popular framework in the web industry. Bootstrap is an open source and it simplifies the process of writing JavaScript and CSS. With the help of Bootstrap, front-end developers do not need to write repeated CSS file or basic JavaScript code because Bootstrap supports flexible grid system, nice looking components and powerful plugins based on jQuery. What developers need to do is to define the class name of a component to achieve the proper appearance. Figure 18 is the code example to use Bootstrap in this project.

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3:3.7/css/bootstrap.min.css" integr
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"

            <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
            <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"

        <nav class="navbar navbar-inverse">
            <div class="container-fluid">
                <!-- Brand and toggle get grouped for better mobile display -->
                <div class="navbar-header">
                  <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                  </button>
                  <a class="navbar-brand" href="http://localhost:8080/pp">Home</a>
                </div>
```

Figure 18. The example of Bootstrap framework in the project

Here to use Bootstrap, firstly we need to import Bootstrap CSS file. Since it is based on jQuery, we import jQuery file and Bootstrap js file. Then we do not need to write CSS or JavaScript related to the style of page any more. Defining the class name of the related component is enough. The page is fit to mobile phone size automatically.

8.3 Spring Data JPA

JPA is the abbreviation of Java Persistence API. It is developed by EJB expert group and is used widely in Java EE applications and Java SE applications. Before JPA become popular, there had been many ORM frameworks. ORM is the abbreviation of Object-Relational Mapping and the purpose of ORM frameworks is to simplify the way for programmers to interact with database. Java developers are quite familiar with JDBC and they have to write a large amount of SQL sentences to persist data into database or query data from database. Moreover, different databases have different SQL grammar, which increases the programming difficulty when the database has to be changed in a project. To simplify this process, ORM requires developers to write Java POJO class and then maps the class to the corresponding table in database and maps the attributes of the class to the columns of the table. With ORM frameworks, developers do not need to write any SQL sentence. Instead, they only need to write codes to operate Java objects to interact with database and can ignore which database it is or the SQL grammar. One of the most famous ORM framework is Hibernate and some examples of Hibernate is in Figure 19.

```java
@Entity
@Table(name = "EMPLOYEE")
public class Employee {
   @Id @GeneratedValue
   @Column(name = "id")
   private int id;

   @Column(name = "first_name")
   private String firstName;

   @Column(name = "last_name")
   private String lastName;

   @Column(name = "salary")
   private int salary;
```

```java
Session session = factory.openSession();
Transaction tx = null;
Integer employeeID = null;

try {
   tx = session.beginTransaction();
   Employee employee = new Employee();
   employee.setFirstName(fname);
   employee.setLastName(lname);
   employee.setSalary(salary);
   employeeID = (Integer) session.save(employee);
   tx.commit();
} catch (HibernateException e) {
   if (tx!=null) tx.rollback();
   e.printStackTrace();
} finally {
   session.close();
}
```

Figure 19. The example of Entity Annotation of JPA

Here, Hibernate requires developers to use annotations to tell the session (the connection to database) which table this class presents and which column this attribute refers to. The operation of Hibernate is simple: create a session, open a transaction, commit and close the session.

Although the ORM frameworks simplify the operation for developers to interact with database, there are several competitive ORM frameworks like TopLink and MyBatis, which increases the coupling of codes. Therefore, JPA comes up and solves the problem. JPA also provides the object-oriented method to persist, update, delete and query data in database and the process is similar with Hibernate: write annotations on POJO classes, create an entity manager (connection to database), query or operate Java objects and close the entity manager. However, JPA only provides the interface of the functions above and does not provide the classes that implemented the interfaces. For Hibernate developers, they have to write classes to implement the JPA interfaces. Actually, the Hibernate EntityManager class contains the Hibernate Session and all the interface methods are implemented by session. Other ORM frameworks also need to write classes that contain their own sessions to implement JPA interfaces. With JPA, the coupling caused by different ORM frameworks is decreased and the whole project becomes more robust.

Spring framework also provides data access interfaces for different ORM frameworks. Based on JPA developed by EJB expert group, Spring group develop Spring Data JPA and the function is more powerful and the process is much simpler. Next, the process of constructing data access layer in this project will be illustrated as the code example of Spring Data JPA.

Firstly, we import the jar by maven and then configure the database information in spring.xml. We use MySql as the database and c3p0 as the database connection pool. In addition, we use Hibernate as the ORM framework and use annotation driven method to manage transactions as shown in figure20.

```
<!-- database connection pool -->
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" value="com.mysql.jdbc.Driver"/>

    <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/test?usel
    <property name="user" value="root"/>
    <property name="password" value="123456"/>
    <property name="maxPoolSize" value="30"/>
    <property name="minPoolSize" value="10"/>
    <property name="checkoutTimeout" value="1000"/>
    <property name="acquireRetryAttempts" value="2"/>
    <property name="maxIdleTime" value="1800"></property>
</bean>
```

```
<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="jpaVendorAdapter" ref="hibernateJpaVendorAdapter"/>
    <property name="packagesToScan" >
        <array>
            <value>uk.ac.le.qx16.pp.entities</value>
        </array>
    </property>
    <property name="jpaProperties">
        <props>
            <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
            <prop key="hibernate.ejb.naming_strategy">org.hibernate.cfg.ImprovedNamingStrategy</prop>
            <prop key="hibernate.cache.provider_class">org.hibernate.cache.NoCacheProvider</prop>
            <prop key="hibernate.show_sql">false</prop>
            <prop key="hibernate.format_sql">false</prop>
            <prop key="hibernate.hbm2ddl.auto">update</prop>
        </props>
    </property>
</bean>

<bean id="hibernateJpaVendorAdapter" class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"/>

<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="entityManagerFactory"/>
</bean>

<tx:annotation-driven transaction-manager="transactionManager" proxy-target-class="true" />

<jpa:repositories base-package="uk.ac.le.qx16.pp.repository"  transaction-manager-ref="transactionManager"
```

Figure 20. The example of Spring Data JPA configuration in the project

Here, HibernateJpaVendorAdapter is the class implementing the interfaces provided by Spring Data JPA. With HibernateJpaVendorAdapter, developers can use JPA backed up by Hibernate in Spring MVC development. Since we want the Spring container to help us manage transaction, we set the transactionManager bean and inject the entityManagerFactory. At the same time, the “tx:annotation-driven” tag opens the tag scanner. Once the container finds a method with “@Transanctional”, the container will automatically open and close the transaction for developers as shown in figure 21.

```
@Entity
public class User {
    private int id;
    private String firstname;
    private String lastname;
    @Id@GeneratedValue
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    @Column(unique=true)
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    private List<TrackingRecord> trackingRecords;
    @OneToMany(fetch=FetchType.LAZY,cascade={CascadeType.REMOVE},mappedBy="user")
    public List<TrackingRecord> getTrackingRecords() {
        return trackingRecords;
    }
    public void setTrackingRecords(List<TrackingRecord> trackingRecords) {
        this.trackingRecords = trackingRecords;
    }
```

Figure 21. The example of Entity Annotation of JPA in the project

For a POJO class, there should be a “@Entity” annotation above the class name so that entityManagerFactory can recognise the class as a tale in database and map the attributes to

the correct columns. For id column, we need to write the “@Id@GeneratedValue” to inform entityManagerFactory that this attribute presents the primary key. With “@Column(unique = true)”, entityManagerFactory is able to set this column “unique value” in database. Moreover, the annotations “@OneToOne”, “@OneToMany”, “@ManyToMany” and “@ManyToOne” together explain the mapping relations and the foreign key between different tables.

```
@Repository
public interface UserRepository extends JpaRepository<User, Integer>{
    public User findByEmailAndPwd(String email, String pwd);
}
```

Figure 22. The Example of Repository of Spring Data JPA in the project

With “@Repository” annotation, Spring container is able to recognise the interface as a data access interface. Then container utilize the Proxy mechanism of Java to generate a Proxy class that can interact with database. The interface with “@Repository” annotation is also known as the data access layer and it must extends the JpaRepository interface. The first element of generics should be the POJO class and the second element should be the id attribute of the POJO class. When developers want to find a user that matches the email and password from the log in request, they need to follow the grammar of Spring Data JPA and the sentence is like “findByEmailAndPwd”. All the Spring Data JPA grammar is referring to https://projects.spring.io/spring-data-jpa/.

8.4 Twitter4J

In order to allow developers to search, track and collect Twitter data, Twitter offers several APIs. For example, with the get request https://api.twitter.com/1.1/account/settings.json, developers can get the returned json that contain all the information of a certain Twitter account. To learn all the API, the website is https://developer.twitter.com/en.html. Although the function of Twitter API is powerful and feature-rich, developers have to write a large amount of codes to send Http Request to the Twitter server and get back the returned json to analyse. There is no doubt that such a process is a drawback when developing a web service. Fortunately, Twitter4j, a Java library for Twitter API, comes up. It is an unofficial Java library and allow developers to embed the Twitter service easily into their projects. With Twitter4J,

Java developers do not need to write HttpConnection codes any more. It is worth mentioning that the Twitter4J library is every easy to learn and the document is available online (http://twitter4j.org/en/index.html).

Before using Twitter4J, we still need three steps: Firstly, we have to possess a Twitter account. If you have not got one, then go to https://twitter.com/. Secondly, we have to apply for the access key and secret token. Twitter API require us to put the personal key and token in the request head. If you have not got the key and token, then go to https://apps.twitter.com/. Lastly, we need to add the Twitter4J dependency into our project. In this project, we use Maven to download the Twitter4J jar.

```
<dependency>
    <groupId>org.twitter4j</groupId>
    <artifactId>twitter4j-core</artifactId>
    <version>4.0.3</version>
</dependency>

<dependency>
    <groupId>org.twitter4j</groupId>
    <artifactId>twitter4j-stream</artifactId>
    <version>4.0.3</version>
</dependency>
```

Figure 23. The example of using maven to import Twitter4J API

The Twitter4J includes Twitter4J core and Twitter4j stream. Before using Twitter4J core or Twitter4J stream, we need to create a factory instance and set the access key and secret token so that we can access the Twitter data successfully.

```
TwitterFactory tf = new TwitterFactory();
Twitter twitter = tf.getInstance();
twitter.setOAuthConsumer(API_KEY, API_SECRET);
twitter.setOAuthAccessToken(new AccessToken(ACCESS_KEY, ACCESS_SECRET));
```

Figure 24. The example of creating Twitter instance in the project

With Twitter4J core, developers can easily get the Twitter user information, search the tweets that contain keywords or hashtags during a certain period in the past, judge if a tweet is retweeted, collect all the comments of a tweet and so on. All the functions above are contained in the methods of twitter4j.User, twitter4j.Twitter, twitter4j.Status and twitter4j.Paging. Figure 25 is the code example to use Twitter4J core to search the tweets in this project.

```
//Twitter user
User user = null;
//list of tweets under some conditions
List<Status> statuses = new ArrayList<Status>();
try {
    //get page 1 and 30 tweets per page
    Paging paging = new Paging(1,30);
    statuses = twitter.getUserTimeline(screenname, paging);
} catch (TwitterException e) {
}
StringBuffer sb = new StringBuffer();
//loop ths statuses
for(Status status:statuses){
    //get the user information from tweet
    if(null==user) user = status.getUser();
    sb.append(TwitterUtil.filterTweet(status.getText()));
}
```

Figure 25. The code to use Twitter API in the project

With Twitter4J stream, developers can easily track tweets from now and on. Developers can set the total number of tweets that they want to collect, set the time how long they want the stream to track and write a class to implement StatusListener to take some actions when a tweet is tracked. Moreover, Twitter4J stream allows developers to override the methods such as onCleanUp and onConnect of the interface ConnectionLifeCycleListener to take some actions when a stream ends and starts. It is used in this project for the function that when a user stops the streaming thread, the system saves all the tweets into the database automatically because we override the onCleanUp method of ConnectionLifeCycleListener.

8.5 Concurrency Thread

Concurrency refers to the phenomenon that multiple tasks run simultaneously in one computer no matter how many core processors this computer has. In [9], all modern operating systems allow concurrent tasks and concurrence is also supported in JVM. In Java, developers can execute multiple threads parallelly in a single-core processor, which improve the efficiency of the application. As we know, I/O processes always occupy a large amount of time in a task, thus concurrency threads are often used in applications where there are frequent I/O operations required. Normally, there are two ways to execute a thread. One is directly through Thread instance and the other one is through the class that implements Runnable interface. Figure 26 are two examples to start a thread.

```java
class Test extends Thread
{
    public void run()
    {
        System.out.println("Run method executed by child Thread");
    }
    public static void main(String[] args)
    {
        Test t = new Test();
        t.start();
        System.out.println("Main method executed by main thread");
    }
}
```

```java
class Geeks {
    public static void m1()
    {
        System.out.println("Hello Visitors");
    }
}

// Here we can extends any other class
class Test extends Geeks implements Runnable {
    public void run()
    {
        System.out.println("Run method executed by child Thread");
    }
    public static void main(String[] args)
    {
        Test t = new Test();
        t.m1();
        Thread t1 = new Thread(t);
        t1.start();
        System.out.println("Main method executed by main thread");
    }
}
```

Figure 26. The two examples of creating a new thread

The advantage of using the first method is that the process is simple. However, the second method is usually recommended. Through implementing Runnable interface, a class can extend other classes while a class that extends Thread can no more extend any other classes. Another advantage of the second method is that implementing Runnable is much more flexible and some advanced APIs for multiple threads require developers to write class that implements Runnable. With Runnable interface, developers can implement some simple functions that run under the multi-threading condition. However, if developers want to monitor and control the process of multiple threads, execute some codes after all the threads finish and take actions when some threads throw exception or interrupt, they have to turn to the thread pool. Java supports thread pool and figure27 is the picture that illustrates the principle of thread pool [11].

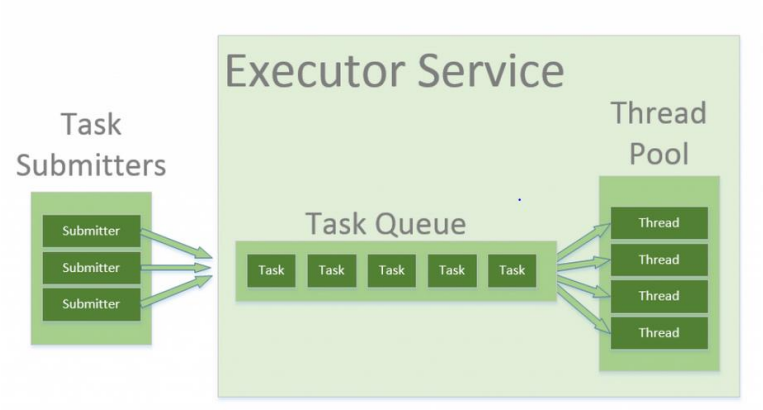

Figure 27. The principle of thread pool

The Executors class contains several static methods that helps us create thread pool instances. The instances created by Executors have already been configured and include SingleThreadExecutor, FixedThreadPool and CachedThreadPool. Normally, we use FixedThreadPool (FTP) and FTP is also used in this project. FTP is a thread pool with fixed length, which means that this pool can execute a fixed number of thread simultaneously. Figure 28 is the example of creating FTP in this project.

```
ExecutorService threadPool = Executors.newFixedThreadPool(3);
for(Status status:statuses){
    final String text = TwitterUtil.filterTweet(status.getText());
    System.out.println(text);
    //thread to get 3 sentiment analysis result
    threadPool.execute(new Runnable() {
        @Override
        public void run() {
            // TODO Auto-generated method stub
            HttpURLConnection conn = null;
```

Figure 28. The example of using concurrency threads in the project (1)

This is the process of predict the sentiment of the tweets. For every tweet, the text has to be sent to the remote server to get the result of sentiment analysis. If we do not apply thread pool technique here, the process might take more than 1 minute because every task might take one second and more than one thousand tweets are analysed one by one. We create an FTP, for every tweet, we create a Runnable class to send it to remote server and all the tasks are executed at the same time.

After all the tasks have been submitted to the FTP, they will wait at the task queue and be sent to the pool to be executed when FTP is available. Then how can developers monitor the alive threads, how can they prevent the exceptions that are thrown during the execution, or how can they stop the FTP if the time is out. FTP provides several APIs to enable developers to write flexible and powerful functions. The method “getPoolSize()” of FTP returns the fixed size of FTP, “getQueue()” returns the list of thread that are waiting in the queue, “shutdown()” makes the FTP stop to accept new submissions and “awaitTermination()” tells the FTP to quit all the tasks when the time is due. Figure 29 is the example of controlling FTP in this project.

```
threadPool.shutdown();
try {
    threadPool.awaitTermination(10, TimeUnit.SECONDS);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Figure 29. The example of using concurrency threads in the project (2)

After all the tasks are submitted, we invoke the “shutdown” method. If the “shutdown” is not invoked, the “awaitTermination” method will throw exception. Once it is invoked, all the new tasks will not be accepted. Then we invoke the “awaitTermination” method and the parameter 10 means that if all the tasks are not finished in 10 seconds, the pool quit and all the threads interrupt. With this operation, we save the time, improve the efficiency of system and manage the concurrency threads successfully.

8.6 Weka for Java

We have mentioned that in this project logistic regression is used to predict gender and the toolkit is Weka. However, the function of Weka is more than logistic regression. In order to learn Weka, we firstly import Weka library in the project with Maven.

```
<dependency>
    <groupId>nz.ac.waikato.cms.weka</groupId>
    <artifactId>weka-dev</artifactId>
    <version>3.9.2</version>
</dependency>
```

Figure 30. The example of using maven to import Weka API

Weka supports supervised training and therefore developers should have collected a large set of labelled data before they use Weka. In addition, Weka requires the data in the format of Arrf which is the unique file format in Weka. Usually, the collected data is in the csv or excel file and developers need to write additional codes to transform the file format. Fortunately, Weka provides powerful API to enable developers to finish the transformation fast and stably. Figure 31 is the code example of using Weka to transform csv files:

```
InputStream in = DataTrain.class.getResourceAsStream("/train1000with12.csv");
OutputStream out = new FileOutputStream("src/main/resources/train1000with12.arff");
CSVLoader loader = new CSVLoader();
loader.setSource(in);
Instances instances = loader.getDataSet();
ArffSaver saver = new ArffSaver();
saver.setInstances(instances);
saver.setDestination(out);
saver.writeBatch();
in.close();
out.close();
```

Figure 31. The example of transforming big data into arff file

The CSVLoader in Weka is used to read in the target csv file and then the records will be extracted from csv file through invoking the "getDataSet()" method of CSVLoader. After that developers need to create an ArffSaver instance to save the records in Arff format. At last, closing the input and output stream should never be forgotten.

With Arff files, we start to use Weka to train the model. Weka provides several classifiers such as Naïve Bayes, Gaussian Distribution and Logistic Regression. As talked before, in this project, Logistic Regression is applied and the code example is in figure 32:

```
//below is to train a model and test the accuracy
String train_path = "src/main/resources/train800.arff";
String test_path = "src/main/resources/test500.arff";
//labelled index
int train_class_index = 4;
int test_class_index = 4;
ArffLoader arffLoader = new ArffLoader();
arffLoader.setFile(new File(train_path));
Instances instances = arffLoader.getDataSet();
instances.setClassIndex(train_class_index);
int train_total = instances.numInstances();
int train_right = 0;
Logistic logistic = new Logistic();
logistic.buildClassifier(instances);
//this loop is to calculate the training accuracy
for(int i=0;i<train_total;i++){
    Instance instance = instances.instance(i);
    if((logistic.distributionForInstance(instance)[0]>0.5&&instance.classVal
        train_right++;
}
System.out.println("Training Accuracy is "+1.0*train_right/train_total);
```

Figure 32. The example of using Weka for Logistic Regression Training

ArffLoader allows developers to read in data from arff files. The Instances class is the set of all the data to be trained. The "setClassIndex()" method of Instances is used to set the column index of labelled result. Then we create a Logistic instance which is a classification for Logistic Regression. If developers want to use Naïve Bayes, they should create a NaiveBayes instance, while the SMO class is for SVM (support vector machine). Other classifiers and the training details can be seen at https://weka.wikispaces.com/. The "buildClassifier()" method is to build the model according to the set of training data. After invoking this method, the logistic instance then becomes the Logistic Regression classifier. The last step is to test the training

and testing accuracy. The “distributionForInstance()” method of Logistic class is to test the probability of a given record. We assume that if the return double digit is larger than 0.5, the result is male, otherwise it is female.

8.7 Google Chart API

In the web development, developers might need to present some data in the format of charts to make the result much more visualized. In this project, Google Chart API (GCA) is used because GCA provides a concise and artistic way to visualize results on websites. GCA provides a variety of chart types from simple line charts to complicated radar charts. In this project, we mainly present the data in Google Pie Chart. When the user searches a topic, a large set of related tweets will be returned. If the user wants to know the sentiment analysis like the positive percent or the gender percent of the tweets, pie chart is the most suitable chart to present. The way to use GCA is quite simple and what developers need to do is to include the google chart js file, list the data to be present, define a div tag with id and display the chart upon the data in the div area. Figure 33 is the code example of using Google Chart API to plot pie chart in this project:

```
$.ajax({
    type:"post",
    url:"tweets/predict",
    data:params,
    dataType:"json",
    success:function(data){
        google.charts.load('current', {'packages':['corechart']});
        google.charts.setOnLoadCallback(function(){
            var pieldata = google.visualization.arrayToDataTable([
                                ['Task','None'],
                                ['male',data.male],
                                ['female',data.female],
                                ['institute',data.totalNames-data.male-data.female]
                                ]);
            var pieloptions = {'title':'Gender Analysis'};
            var chart1 = new google.visualization.PieChart(document.getElementById('chart1'));
            chart1.draw(pieldata, pieloptions);
```

Figure 33. The example of using Google Chart API in the project

After sending an AJAX request to the server and receiving the prediction result successfully, we start to draw pie chat in the success function. “google.charts.load” means loading the visualization API and packages. “google.charts.setOnLoadCallback” means running the callback function when the API and packages are completely loaded. In the callback function, we define a data that is to be present, after which we define the title of the chart. Then we

create a Google Pie Chart object and link it to the div tag whose id is “chart1”. Finally the method “draw()” is invoked and the pie chart is plotted.

8.8 jQuery Validation API

Validation problem appears in web developers’ daily work. When a user logs in, signs up or fills in a form, developers need to write JavaScript codes to validate the content that is provided by user. As it is common for users to type in wrong information or text without correct format, the importance of validation should be in mind of every developer. There is no doubt that developers can easily write their own codes to validate the form information if they are familiar with JavaScript. However, it will waste a large amount of time and the performance is not as good as the Validation API provided by jQuery team. Therefore, in this project, jQuery Validation API is used and see https://jqueryvalidation.org/ for more details. The form code is shown in figure 34:

```
<form id="reg_form" action="user/register" class="form-horizontal col-sm-8 col-sm-offset-2" role="form" method="post"
    <div class="form-group">
        <label for="firstname" class="col-sm-2 control-label">Given name</label>
        <div class="col-sm-10">
            <input type="text" class="form-control" name="firstname" placeholder="Firstname" id="firstname">
        </div>
    </div>
    <div class="form-group">
        <label for="lastname" class="col-sm-2 control-label">Surname</label>
        <div class="col-sm-10">
            <input type="text" class="form-control" name="lastname" placeholder="Lastname" id="lastname">
        </div>
    </div>
    <div class="form-group">
        <label class="col-sm-2 control-label">Gender</label>
        <div class="col-sm-10">
            <label class="radio-inline"><input type="radio" name="gender" value=1 checked>Male</label>
            <label class="radio-inline"><input type="radio" name="gender" value=0>Female</label>
        </div>
    </div>
    <div class="form-group">
        <label for="birth" class="col-sm-2 control-label">Birthday</label>
        <div class="col-sm-10">
            <input type="text" name="birth" class="form-control" placeholder="Birthday" id="birth">
        </div>
    </div>
    <div class="form-group">
        <label for="email" class="col-sm-2 control-label">Email</label>
        <div class="col-sm-10">
            <input type="text" class="form-control" name="email" placeholder="123@example.com" id="email">
        </div>
    </div>
```

Figure 34. The example of register form in the project

It is a normal form without validation. Now we import jQuery Validation CSS and js files and the jQuery js file should be included before. Then we write the validation function on the js file.

```
<!-- below is validation css for jquery -->
<link rel="stylesheet" href="https://jqueryvalidation.org/files/demo/site-demos.css">

<!-- below two are validation js for jquery -->
<script src="https://cdn.jsdelivr.net/jquery.validation/1.16.0/jquery.validate.min.js"></script>
<script src="https://cdn.jsdelivr.net/jquery.validation/1.16.0/additional-methods.min.js"></script>
```

Figure 35. The code to include jQuery Validation CSS and js files

```
$("#reg_form").validate({
    rules:{
        firstname:{
            required:true
        },
        lastname:{
            required:true
        },
        birth:{
            required:true
        },
        email:{
            required:true,
            email:true
        },
        tel:{
            required:true,
        },
        pwd:{
            required:true
        },
        pwdConf:{
            required:true,
            equalTo:"#pwd"
        },
        postcode:{
            required:true
        }
    },
    messages:{
        pwdConf:{
            equalTo:"please enter the same password!"
        }
    },
    success:"valid"
});
```

Figure 36. The example of jQuery Validation code

jQuery Validation API provides the “validate()” method to enable developers to add validation function to a normal form. In this method, the “rules” attribute requires developers to set the validation item of a certain input component. For example, the “required” attribute means this input should not be empty, “email” attribute means this input should be in line with the email format and “equalTo” attribute means the value of this input should be the same as that in another input. Moreover, the “messages” attribute provides developers the interface to set the reminder for the error. Here we set the “Please enter the same password” message for the case that the user does not confirm the password correctly. The validation function has been present in chapter of App Function.

8.9 Bootstrap Datepicker API

It is normal for web developers to design time field for users to type in their birthday, start date or expire date. Developers can implement such a field easily with CSS and JavaScript. However, to achieve fast deployment, stable performance and artistic appearance, the Bootstrap Datepicker is strongly recommended in this paper. The advantages of Bootstrap have been illustrated before and Bootstrap also provides various plugins that are widely used in web projects. In the first step, we need to include the CSS file and js file that can be downloaded in the official website of Bootstrap.

```
<!-- below is datepicker css for bootstrap -->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.4.1/css/bootstrap-datepicker3.

<!-- below is datepicker js for bootstrap -->
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.4.1/js/bootstrap-datepick
```

Figure 37. The example of including Bootstrap Datepicker CSS and js

Then we set the id of the “<input type=’text’>” which we want to add the datepicker function to. After that we need to write JavaScript codes to implement that function.

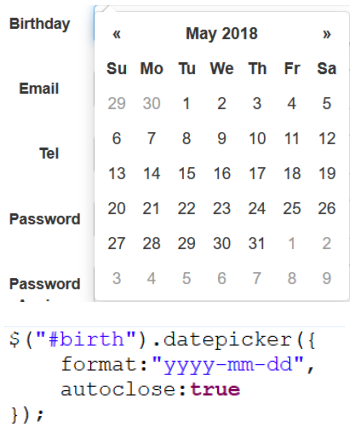

Figure 38. The code of implementing the datepicker function

Bootstrap Datepicker API provides developers the “datepicker()” method. In this method, the “format” attribute enables developers to set the format of date string. Here we set the format “yyyy-mm-dd” and the result will be “2018-01-01”. The “autoclose” attribute will close the datepicker window automatically when user selects a date.

8.10 Java Email API

In web projects, sometimes the server has to send an activation email or confirmation email to users. Java EE also provides such APIs to make it easy for developers to implement this function. In this project, if users forget their password, they should get back their password back through the getBack.html page. They have to provide the registered email and get back through email. Thus, in this project, we use javaMail API which is a platform-independent framework to send and receive emails. The same as other APIs in this project, we firstly import the packages through maven.

```
<!-- javaMail -->
<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>mail</artifactId>
    <version>1.4</version>
</dependency>
<dependency>
    <groupId>javax.activation</groupId>
    <artifactId>activation</artifactId>
    <version>1.1.1</version>
</dependency>
```

Figure 39. The example of using Maven to import javaMail

```
static{
    Properties properties = new Properties();
    properties.setProperty("mail.smtp.auth", "true");
    properties.setProperty("mail.transport.protocol", "smtp");
    properties.setProperty("mail.smtp.host", HOST);
    Session session = Session.getInstance(properties);
    Transport transport = null;
    try {
        transport = session.getTransport();
        transport.connect(SENDER_ACCOUNT, SENDER_PWD);
        System.out.println("Connect Successfully!");
    } catch (Exception e) {
        // TODO Auto-generated catch block
        System.err.println("Connect Error!");
    }
    SESSION = session;
    TRANSPORT = transport;
}

public static void main(String[] args) throws Exception{
    MimeMessage message = new MimeMessage(SESSION);
    message.setFrom(new InternetAddress(SENDER_ADDRESS));
    message.setRecipient(MimeMessage.RecipientType.TO,new InternetAddress("qiangde_xiang@163.com"));
    message.setSubject("Twitter Data: Get Password Back", "UTF-8");
    message.setContent("IMPORTANT", "text/html;charset=UTF-8");
    message.setSentDate(new Date());
    TRANSPORT.sendMessage(message, message.getAllRecipients());
}
```

Figure 40. The codes to send email through javaMail API in the project

In the first step, we create a Properties instance to set the properties. The protocol we applied is smtp. Then we need to create a Session object which is the channel for sending and receiving emails. After that, Transport object is obtained and we have to authenticate the username and password. Once the authentication is passed, the Transport object can be used to send emails. When we start to send an email, we should create the MimiMessage instance.

Then the “From” and “Recipient” attributes should be set and they present the sender address and receiver address respectively. The “Subject” and “Content” attributes are also compulsory to set and the charset should be set as “UTF-8”. Finally, we can send a message that contain the password to the registered users who forget their passwords.

9. Conclusion

In this paper, we propose a novel method to predict the gender of Twitter users according to their texts, names, tweet frequency and favourite frequency. Compared with Naïve Bayes classifier, we apply Logistic Regression to train the model. All the data is labelled and the training and testing data size is 1000 and 500 respectively. We use Weka to implement the training process and the training and testing accuracy are both above 80 percent, which is higher than that in previous work mentioned at beginning. We also developed a web application to allow users to search and track tweets. As illustrated in the App Function section, the app is user-friendly and fit to mobile size. Users can search tweets according to keywords and hashtags during a certain period. For the searching result, users can see the gender prediction and recent sentiment analysis result of a certain tweet author. Moreover, with respect to a certain topic, there might be thousands of related tweets and users are able to get the analysis result about the gender and sentiment percentage. All the results are shown in Google Pie Chart, which seems very visualized. For the purpose of collecting big data, this app allows registered users to track real-time tweets according to multiple keywords in a certain time. Users can download the csv files that contain tracking results in their personal page. Other functions of this app include log in, register with validation, get back password, update personal settings, delete tracking records, access Twitter.com, download tutorials about Twitter4J and so on. The techniques used in this project include Spring MVC framework, Spring Data JPA, JavaScript, jQuery framework, jQuery validation plugin, Bootstrap framework, Bootstrap datepicker plugin, Weka, Google Chart, javaMail, Twitter4J, concurrency thread and so on. All the techniques have been illustrated in detail in Techniques Involved section.

References:

[1] D. Rao, D. Yarowsky, A. Shreevats, M. Gupta. Classifying Latent User Attributes in Twitter. SMUC'10 pp.37-44, October 30, 2010.

[2] A. Mislove, S. Lehmann, Y. Ahn, J. Onnela, J. Rosenquist. Understanding the Demographics of Twitter Users. AAAI pp.554-557, 2011.

[3] J. Burger, J. Henderson, G. Kim, G. Zarrella. Discriminating Gender on Twitter. EMNLP pp. 1301-1309.

[4] S. Arthur. Computer Games I. Springer, New York, NY. pp. 335-365.

[5] F. Seide, G. Li, D. Yu. Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. ISCA. 28-31, August, 2011.

[6] B. Heil, M.J. Piskorski. New Twitter research: Men follow men and nobody tweets. Harvard Business Review, June, 2009.

[7] D.W. Hosme, S. Lemeshow, R.X. Sturdivant. Applied Logistic Regression, John Wiley & Sons, Incorporated, 2013. ProQuest Ebook Central,

https://ebookcentral-proquest-com.ezproxy4.lib.le.ac.uk/lib/leicester/detail.action?docID=1138225.

[8] S.J. Sin. Social Media Problematic Everyday Life Information-Seeking Outcomes: Differences Across Use Frequency, Gender, and Problem-Solving Styles. Journal of The Association for Information Science and Technology, pp.1793-1807. May 13, 2015.

[9] F. Javier. Java 7 Concurrency Cookbook, Packt Publishing, 2012. ProQuest Ebook Central,

https://ebookcentral-proquest-com.ezproxy3.lib.le.ac.uk/lib/leicester/detail.action?docID=1057944.

[10] X. Chen, S.J. Sin, Y.L. Theng, C.S. Lee. Why Students Share Misinformation on Social Media: Motivation, Gender, and Study-level Differences. The Journal of Academic Librarianship, pp.583-592. September, 2015.

[11] E. Paraschiv. Introduction to Thread Pools in Java. August 27, 2017,

http://www.baeldung.com/thread-pool-java-and-guava.