Software Engineering I Group II

Health Monitoring Analytics 2014

Technical Documentation

D. Yao, W. Fang, W. Zhang, Y. Sun, Y. Wu, Z. Zheng

TABLE OF CONTENTS

Overview

The purpose of this documentation is to better explain how our system work from a programmer perspective. Since there are multiple programming languages that we used to develop our own system, it is easily to take a look at the codes by separating them into corresponding functions and stages.

The system is separating in to three part: Server-side, Website, IOS device. This document would give some concise explanations about how different parts of the system work.

1.Server-side

For several specific reasons, the back-end of our system is constructed by Mysql, PHP, and Apache. All codes that used in the server to retrieve tweets and analyze data are coding by PHP. The figure following shows the structure of the file of the server.

Index of /demo1_code(no_ios)

Name	Last modified	Size	Description
Parent Directory		-	
database/	2014-10-29 17:58	-	
database structure/	2014-10-29 17:58	-	
features/	2014-10-29 17:58	-	
jsonsender/	2014-10-29 17:58	-	
web/	2014-10-29 17:58	-	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 1-1 Structure of Server files

Several folders are appeared. The database folder holds the files that used to retrieve the raw data and classify them. The database_structure folder just stores a SQL structure file that you could use to reconstruct a database has the same structure as our database. The features folder contains all analysis codes that could achieve the system's goals. The jsonsender folder owns the program files that take the responsibilities to provide the communication between front-end and back-end of the system. Lastly, all website files (including html, JS, CSS) that available now are put in the web folder.

1.1. Tweet retrieve

Index of /demo1_code(no_ios)/database

Name	Last modified	Size	Description
Parent Directory		-	
140dev_config.php	2014-10-12 20:32	2.0K	
db_config.php	2014-09-12 01:48	602	
db_lib.php	2014-10-12 19:59	3.7K	
dbct.php	2014-09-12 03:53	103	
error_log.txt	2014-10-27 04:17	599M	
get_tweets.php	2014-10-06 15:28	3.7K	
monitor_tweets.php	2014-09-12 02:49	1.6K	
mysql_database_schem..>	2013-12-14 21:41	2.9K	
parse_tweets.php	2014-09-16 20:47	5.2K	
phirehose/	2014-10-29 17:58	-	
twitter_auth.txt	2014-09-08 01:28	229	
twitteroauth/	2014-10-29 17:58	-	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 1-2 Database folder

Let's step into the database folder. Such files are inherited from Group #1 of 2013FALL.

The main file we use to retrieve tweets from twitter through the streaming API named get_tweets.php. While running it, 140_dev_config.php, db_lib.php, db_config.php, and all the files inside the phirehose are also called. 140_dev_config.php provides the needed keys for Twitter API connection and db_lib.php constructs the connection between the server and the database. db_config.php sets up the options (like the password and the user name) that used by the server to connect the database. The files inside the phirehose folder is shown below. They are derived from an Open Source Project available on Github and could help us use Twitter API in an easier way.

Index of /demo1_code(no_ios)/database/phirehose

Name	Last modified	Size	Description
Parent Directory		-	
OauthPhirehose.php	2014-04-22 16:13	4.1K	
Phirehose.php	2014-04-22 16:13	33K	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 1-3 Phirehose folder

Another essential files are the two php files inside the twitteroauth folder. They are used for retrieving the tweets data through Rest API and also are derived from the internet.

Index of /demo1_code(no_ios)/database/twitteroauth

Name	Last modified	Size	Description
Parent Directory		-	
OAuth.php	2014-05-22 15:53	26K	
twitteroauth.php	2014-05-22 15:53	7.5K	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 1-4 Twitteroauth folder

After the system retrieves the raw data (in json_cache format) and stores them in to its own database, it should start the parse_tweets.php file in the database folder to parse the raw data into different parts and restores them into several new tables. All codes inside this PHP file is easy to understand and would not be elaborated.
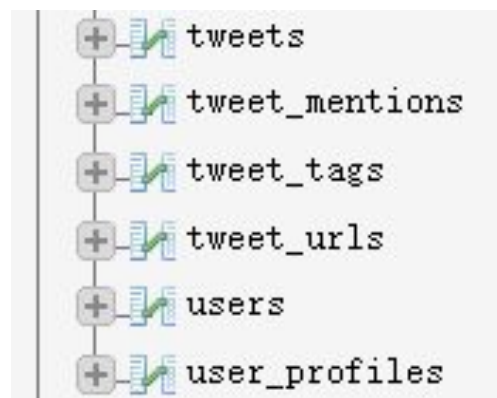
Figure 1-5 Tables created by parse_tweets.php

Above basically explain how the system retrieves and stores the necessary information into its own database.

1.2. Data analysis

Following is the folders inside the feature folder. All of them are used for analyzing the data retrieved from twitter.

Figure 1-6 Features folder

Each folder includes the PHP files that could achieve the function corresponding to their name. All feature tables that we obtain after all analysis is shown in the figure 1-7.

Figure 1-7 Features tables

1.2.1. Demographic analysis

The files related to demographic analysis are shown below.

Name	Last modified	Size	Description
Parent Directory		-	
EI_area_age.php	2014-10-14 10:30	1.1K	
EI_area_gender.php	2014-10-14 10:30	1.1K	
EI_time_age.php	2014-10-14 10:46	929	
EI_time_gender.php	2014-10-14 10:40	1.1K	
EI_type_age.php	2014-10-14 10:46	929	
EI_type_gender.php	2014-10-14 10:49	920	
demography.php	2014-10-28 12:44	3.4K	
demography_overall.php	2014-10-28 17:17	1.2K	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 1-8 Demography folder

The leading file of this part is named demography.php. By running this file, we could access the API from the third party and sequentially update the gender, age, and type attributes in the user table. Several key codes are displayed below.

```
function getGender($user, $name, $description) {
$api = 'http://textalytics.com/core/userdemographics-1.0';
$key = '0e70ff2a84eab3fda9019ab2202cd91e';
// We make the request and parse the response to an array
$response = sendPost($api, $key, $user, $name, $description);
$json = json_decode($response, true);
$gd = '';
if(isset($json['gender'])) {
   if($json['gender'] == 'M')
      $gd = 'MALE';
   else if($json['gender'] == 'F')
      $gd = 'FEMALE';
}
return $gd;
}


// Auxiliary function to make a post request
function sendPost($api, $key, $user, $name, $description) {
   $data = http_build_query(array('key'=>$key,
                  'user'=>$user,
                  'name'=>$name,
                  'desc'=>$description,
                  'src'=>'sdk-php-1.0')); // management internal parameter
   $context = stream_context_create(array('http'=>array(
         'method'=>'POST',
         'header'=>
            'Content-type: application/x-www-form-urlencoded'."\r\n".
            'Content-Length: '.strlen($data)."\r\n",
```

7

```
                    'content'=>$data)));
    $fd = fopen($api, 'r', false, $context);
    $response = stream_get_contents($fd);
    fclose($fd);
    return $response;
  } // sendPost
```

It is easy for us to understand. By connecting the URL provided by textalytics.com through the existing key, after providing the name, screen_name, and description of user's account we could get the speculative gender information (in JSON format) of this user. We could use the same method to infer other demography information (like age, account type) about the user. The result of demography.php is shown blow. We could notice that several columns are inserted into the user table.

gender ▲ 1	age	type
MALE	15-24	ORGANIZATION
MALE	25-34	PERSON
MALE	15-24	ORGANIZATION
MALE	<15	PERSON
MALE	15-24	ORGANIZATION
MALE	>65	ORGANIZATION
MALE	15-24	
MALE	25-34	ORGANIZATION

Figure 1-9 User table after running demography.php

Other files in the demography folder are used to calculate the corresponding number of tweets posted by specific ages and genders (basically separated by different exercise types, states, and times). Also several of them can do the statistics about the distribution of age and gender. All of them are running base on the information inferred by demography.php and the codes inside them directly explain what they are doing, hence they would not be elaborated here.

1.2.2. Sentiment value

Since the mathematical method to calculate the sentiment value is introduced in the full report #1, we focus on how to achieve such function here.

As what is mentioned in the report, we need the ANEW list to calculate the overall score of each valid tweets. Hence, the ANEW list is imported in the system's database and named ANEW table, as shown below.

description	Valence_Mean	Valence_SD	Arousal_Mean	Arousal_SD	Word_Frequency
abduction	2.76	2.06	5.53	2.43	1
abortion	3.5	2.3	5.39	2.8	6
absurd	4.26	1.82	4.36	2.2	17
abundance	6.59	2.01	5.51	2.63	13
abuse	1.8	1.23	6.83	2.7	18
acceptance	7.98	1.42	5.4	2.7	49
accident	2.05	1.19	6.26	2.87	33
ace	6.88	1.93	5.5	2.66	15
ache	2.46	1.52	5	2.45	4
achievement	7.89	1.38	5.53	2.81	65
activate	5.46	0.98	4.86	2.56	2
addict	2.48	2.08	5.66	2.26	1
addicted	2.51	1.42	4.81	2.46	3
admired	7.74	1.84	6.11	2.36	17
adorable	7.81	1.24	5.12	2.71	3
adult	6.49	1.5	4.76	1.95	25
advantage	6.95	1.85	4.76	2.18	73
adventure	7.6	1.5	6.98	2.15	14

Figure 1-10 ANEW Table

The files related to sentiment value is also listed here.

Index of /demo1_code(no_ios)/features/exercise_sentiment

Name	Last modified	Size	Description
Parent Directory		-	
ANEW.csv	2014-10-27 14:41	31K	
EI_state_mood.php	2014-10-25 11:45	3.7K	
EI_time_mood.php	2014-10-27 15:32	3.7K	
EI_time_type_mood.php	2014-10-27 15:32	4.0K	
EI_type_mood.php	2014-10-27 17:31	3.3K	
EI_type_state_mood.php	2014-10-16 20:56	4.1K	
read_me.txt	2014-10-25 10:38	422	
table_data/	2014-10-29 17:58	-	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 1-11 Sentiment folder

The process to calculate the mood_value of different type and area is similar, let's take EI_state_mood.php for example here. Following is several important codes

inside this file.

```
for($p=0;$p<count($state_name);$p++){   //loop state
    echo $state_name[$p]."<br>";
    $k_tweet = 0;
    $sql_tweet = "SELECT tweet_text FROM tweets LEFT JOIN users on tweets.user_id =
                    users.user_id WHERE location LIKE BINARY '%".$state_name[$p]."%'";
    $result_tweet = mysqli_query($con,$sql_tweet);

    while($array_related = mysqli_fetch_array($result_tweet)){
        $tweet_related[$k_tweet] = $array_related[0];
        $k_tweet++;
    }
    $valid_count = 0;
    $score_single = 0;
    $score_all = 0;
    for($i=0;$i<count($tweet_related);$i++){
        //preparing to start keyword match for the first tweet
        $k_match = 0;
        //$test_match=array();
        for($j=0;$j<count($desc);$j++){
            if(strpos(".$tweet_related[$i].",$desc[$j])){
                $test_match[$k_match]= $j;
                $k_match++;
            }
        }
        if($k_match > 1){
            $valid_count++;
            $s_scoreall = 0;
            $s_weightall = 0;
            //We calculate the score of previous tweet
            for($t=0;$t<count($test_match);$t++){
                $s_scoreall    += $val_mean[$test_match[$t]]/ $val_sd[$test_match[$t]];
                $s_weightall   += 1 / $val_sd[$test_match[$t]];
                $score_single = $s_scoreall / $s_weightall;
            }
        }
        else{
            $score_single = 0;
        }
            //$score_single is the score of a single tweet
            //$score_all is the total score of a type
            $score_all = $score_single + $score_all;

    }
```

```
$final = $score_all / $valid_count;
echo $final."<br>";
echo $valid_count."<br>";
$sql = "INSERT INTO EI_state_mood(State_Name,Mood_Value) VALUES
        ('".$state_name[$p]."',$final)";
mysqli_query($con,$sql);
unset($tweet_related);
}       //ALL states' mood_value are available
```

In these codes, we take out the related tweets (according to the state they belong to) from the databse. For each tweets, we match every word inside it with the 1000+ existing word that are stored in the ANEW table, calculate the corresponding score of this tweet. And after, we calculate the mean score of all tweets belong to the state and store the score in a new table (shown in figure 1-12), and then move on for another state until we get the score of every state.

State_Name	Mood_Value
AK	6.3158
AL	6.53492
AR	6.68003
AZ	6.36641
CA	5.90376
CO	5.77029
CT	5.74512
DE	5.76196
FL	6.2218
GA	5.9987

Figure 1-12 ei_state_mood table

We could calculate the mood_value of the other kinds (like exercise type and time) in the same way.

1.2.3.   Exercise duration

Following is the files inside the exercise_duration folder.

Index of /demo1_code(no_ios)/features/exercise_duration

Name	Last modified	Size	Description
Parent Directory		-	
Etime_state.php	2014-11-01 19:21	2.5K	
Etime_state_mean.php	2014-10-27 13:39	1.6K	
Etime_type.php	2014-10-12 15:57	2.3K	
Etime_type_mean.php	2014-10-09 00:51	1.3K	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 1-13 Exercise_duration folder

The methods they use to achieve the function (extracting the duration time from tweets) are similar, hence we will take one of them for explanation. Let's have a look at the Etime_state.php file. Below is two key functions of it.

```php
function number($str){
    return preg_replace ('/\D/s', '', $str);
}

function extractnum($string){
  //calculate mins
    $pos_min = strpos("$string","min");
    $str_min = substr($string, $pos_min-4, 4);
    $num_min = number ($str_min);
    $num_min_int = (int)$num_min;
  //caculate hours
    $pos_hr = strpos("$string","hr");
    $str_hr = substr($string, $pos_hr-4, 4);
    $num_hr = number($str_hr);
    $num_hr_int = (int)$num_hr;
  // judge if hours exsist
    if ($num_hr_int != 0){
        $num_hr_min = $num_hr_int*60;
    }else{$num_hr_min = 0;}

    $num_min_sum = $num_min_int + $num_hr_min;
    return $num_min_sum;
}
```

The first function number() is used to return the number inside a given string. In the second function extractnum(), we first get the position of 'min' or 'hr'. Then we extract three words exactly before it and send them as a whole string to the number() function to extract the number in it (if there is any). At last the program would calculate the total time of the single tweet. Hence, if the tweet mention about exercise duration, we could extract it and store the data into our database. The result of Etime_state.php is listed after.

state_name	exercise_time
AK	20
AK	180
AK	180
AK	3
AK	3
AK	90
AK	90
AK	45

Figure 1-14 Etime_state table

1.2.4. Exercise frequency

Following is the files inside the exercise_frequency folder.

Index of /demo1_code(no_ios)/features/exercise_frequency

Name	Last modified	Size	Description
Parent Directory		-	
freq_state.php	2014-10-28 00:58	1.0K	
freq_state_count.php	2014-10-28 00:52	1.3K	
freq_state_mean.php	2014-10-28 00:42	1.5K	
freq_type.php	2014-10-28 21:45	741	
freq_type_count.php	2014-10-28 21:45	1.1K	
freq_type_mean.php	2014-10-28 21:46	1.3K	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 1-15 exercise_frequency folder

In order to calculate how frequently people exercise. Those files are needed to be run in a specific sequence. Take freq_state.php, freq_state_count.php and freq_state_mean.php for instance. The freq_state.php file is used to extract and record the state name, screen name and created data of every exercise-related tweet. Hence, the system should launch it first. The corresponding codes are listed here.

```
for($j = 0; $j < count($state_name); $j++){
    $sql = "SELECT tweets.screen_name, tweets.created_at FROM tweets LEFT JOIN
        users on tweets.user_id = users.user_id WHERE location LIKE
        '%".$state_name[$j]."%' and tweets.created_at not like '2014-10-12%' and
        tweets.created_at not like '2014-10-20%'";
    $result = mysqli_query($con,$sql);
    while($array = mysqli_fetch_array($result)){
        $sql = "INSERT INTO freq_state(state_name, screen_name, created_at) Values
            ('".$state_name[$j]."','".$array[0]."','".$array[1]."')";
        mysqli_query($con,$sql);
    }
}
```

All of the information extracted are stored in the freq_state table as below.

state_name	screen_name	created_at
AK	lexica	2014-09-15
AK	CintyWinty	2014-09-15
AK	IFollowPeter	2014-09-15
AK	tmj_OKA_health	2014-09-15
AK	shaunandrewsnet	2014-09-15
AK	AradiaOakville	2014-09-15
AK	T4PNicosia	2014-09-15
AK	HealthCatalyst	2014-09-15
AK	vic23	2014-09-15

Figure 1-16 freq_state table

13

Then, we should run the freq_state_count.php, which could count how many exercise-related tweets that were sent by the same user. The crucial part of the codes and the result are shown below.

$sql = "SELECT screen_name, count(screen_name) FROM freq_state WHERE state_name
    = '".$state_name[$j]."' group by screen_name";

state_name	screen_name	state_count ▼ 1
AK	AKJobConnecter	2
AK	denise1321	2
AK	dobly79	2
AK	FJMaris	2
AK	GCFitnessClub	2
AK	IFollowPeter	2
AK	IHealthcareJobs	2
AK	jokoutomo9	2
AK	MollyMaloofMD	2
AK	refrigeration	2

Figure 1-17 freq_state_count table

At last, the freq_state_mean.php could simply calculate the average, maximum number and the standard deviation of the count number in specific states by using following codes.

$sql1 = "SELECT AVG(state_count) FROM freq_state_count WHERE state_name LIKE
    '%".$state_name[$i]."%'";
$sql2 = "SELECT MAX(state_count) FROM freq_state_count WHERE state_name LIKE
    '%".$state_name[$i]."%'";
$sql3 = "SELECT STDDEV(state_count) FROM freq_state_count WHERE state_name
    LIKE   '%".$state_name[$i]."%'";

The eventual table is presented here.

state_name	mean	max	std
AK	1	2	0.192945
AL	1	2	0.21505
AR	1	2	0.207749
AZ	1	2	0.182934
CA	1	2	0.226407
CO	1	2	0.218355
CT	1	2	0.238768
DE	1	2	0.200829
FL	1	2	0.229455
GA	1	2	0.2285
HI	1	2	0.189903
IA	1	2	0.206427

Figure 1-18 freq_state_mean table

We do exactly the same thing for calculating the frequency of specific types of exercise.

1.2.5. Others

Some analysis programs are not appeared above including counting the total number of tweets that were sent from a specific state and calculating the amount of tweet related to a given type of exercise. We could achieve those functions by simply counting and filtering, hence there is no need to elaborate them.

1.3. Communication

Considering we have both the IOS device and the website, it is necessary for the server to identify them since what information they need could be very different. For communication, we use both GET and POST methods according to which platform is requesting this data. Several rows of the codes that enable the server to identify what information is needed to send and who is asking for these information is shown below.

```
$who=$_GET['who'];

if($who==1) $id=$_GET['id'];
else{
    $id=$_POST['id'];
    $usr=$_POST['usr'];
    $gender=$_POST['gender'];
    $age=$_POST['age'];
    $state=$_POST['state'];
    $exercise=$_POST['exercise'];
}
…
…
switch($id){
case 0:
    $sql = "SELECT * FROM EI_area_overall";break;
case 1:
    $sql = "SELECT * FROM EI_type_overall";break;
case 2:
    $sql = "SELECT * FROM EI_time_type";break;
…
…
if ($result = mysqli_query($con, $sql)){
    $resultArray = array();
    $tempArray = array();
    while($row = $result->fetch_object()){
        $tempArray = $row;
```

```
        array_push($resultArray, $tempArray);
      }
      echo json_encode($resultArray);
  }
  …
  …
```

In these codes, when who = 1, it means that it is the website that requesting the data. Otherwise it should be the IOS device. The program and then would send corresponding data according to the id value it get from the front-end. Such code is a part of jsonsender.php, which is located in the jsonsender folder as below.

Index of /demo1_code(no_ios)/jsonsender

Name	Last modified	Size	Description
Parent Directory		-	
jsonsender.php	2014-10-28 21:53	5.3K	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 1-19 jsonsender folder

2. Website

Following is the files existing in the web folder. All of them is usable, however, only a few of them are used in this demo.

Index of /demo1_code(no_ios)/web

Name	Last modified	Size	Description
Parent Directory		-	
HeatMap.html	2014-10-11 10:06	1.7K	
Markers_Tweets.html	2014-10-10 15:59	3.4K	
StateMap.html	2014-10-10 15:59	6.4K	
StateMap_geochart.html	2014-10-10 15:59	6.4K	
StateMap_mood.html	2014-10-27 22:18	6.4K	
backup/	2014-10-29 17:58	-	
dEI_time_overall.html	2014-10-25 22:15	1.2K	
dEI_time_overall_lin..>	2014-10-25 22:15	1.2K	
dEI_type_overall_col..>	2014-10-25 21:21	1.3K	
dEI_type_overall_pie..>	2014-10-10 16:23	1.3K	
dcorrelation_he.html	2014-10-26 10:13	1.3K	
dleaderboard_overall..>	2014-10-25 21:21	1.4K	
dleaderboard_overall..>	2014-10-25 21:21	1.4K	
jquery-1.11.1.min.js	2014-05-01 13:59	94K	
jsonsender.php	2014-10-27 20:57	3.1K	
statesmap.html	2014-10-25 14:53	455	
statesmap.js	2014-10-16 22:28	5.9K	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

Figure 2-1 web folder

Although the complete website is not well constructed yet, there are several functions we achieved in the mobile platform that are based on the website (such as the state map and heat map). We simply offer the web view for the user to browse them through the IOS device.

2.1. Receive data

For receiving the JSON format data sent by the server, we get assistant from the jquery-1.11.1.min.js source, which is an Open Source Project available online. Multiple useful functions for parsing the JSON data is provided by it. What we need to do is just add such a row of code inside the html file.

```
<script src="jquery-1.11.1.min.js"></script>
```

The JS codes we used to received data and how the front-end program stores the parsed data is shown below.

```
function getdata() {
```

17

```
$.getJSON(
    "../jsonsender/jsonsender.php", // The server URL
    {id: 0, who: 1},
    showdata // The function to call on completion.
);
}
function showdata(json) {
    var tweetData = [];
    for (var i = 0; i < eval(json).length; i++){
        tweetData[i] = parseInt(json[i].state_count);
    }
…
…
```

How it work is basically explained in the communication section before.

2.2. Information Display

We will only discuss about the state map and heat map this time since only these two functions are used in this demo.
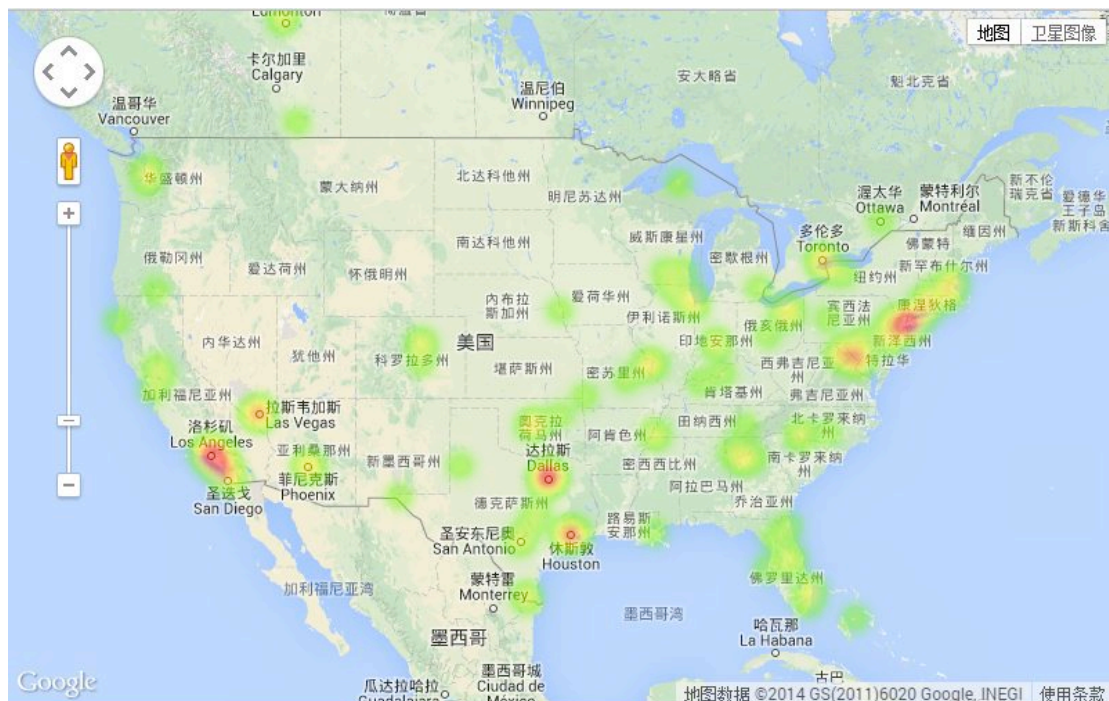
Figure 2-2 Heat map

The heat map is created through Google Map API. It is very easy to use. What we should do first is to include such code inside the html file.

```
<script
src="http://maps.googleapis.com/maps/api/js?key=XX&libraries=visualization"></script>
```

The available key requested from the Google account should be replace the XX of the code above.

The rest of the code is used to translate our raw data's format into what the Google API is asking for, build up the map option we want to use, and set up the event to draw the map. Whole program is shown below.

```
google.maps.event.addDomListener(window, 'load', getdata);
function getdata() {
  $.getJSON(
    "../jsonsender/jsonsender.php", // The server URL
    {id: 3, who: 1},
    showdata // The function to call on completion.
  );
}
function showdata(json){
  // Adding Data Points From database
  var tweetData = [];
  for (var i = 0; i < eval(json).length; i++){
    var addData = new google.maps.LatLng(json[i].geo_lat, json[i].geo_long);
      tweetData.splice(i, 0, addData);
  }
  var mapOptions = {
    zoom: 4,
    center: new google.maps.LatLng(39, -96),
    mapTypeId: google.maps.MapTypeId.ROADMAP
    // mapTypeId: google.maps.MapTypeId.SATELLITE
  };
  var map = new google.maps.Map(document.getElementById('map-canvas'),
      mapOptions);
  var pointArray = new google.maps.MVCArray(tweetData);
  var heatmap = new google.maps.visualization.HeatmapLayer({
    data: pointArray
  });
  heatmap.setMap(map);
  heatmap.set('radius', 20);
  heatmap.set('opacity', 0.5);
}
```
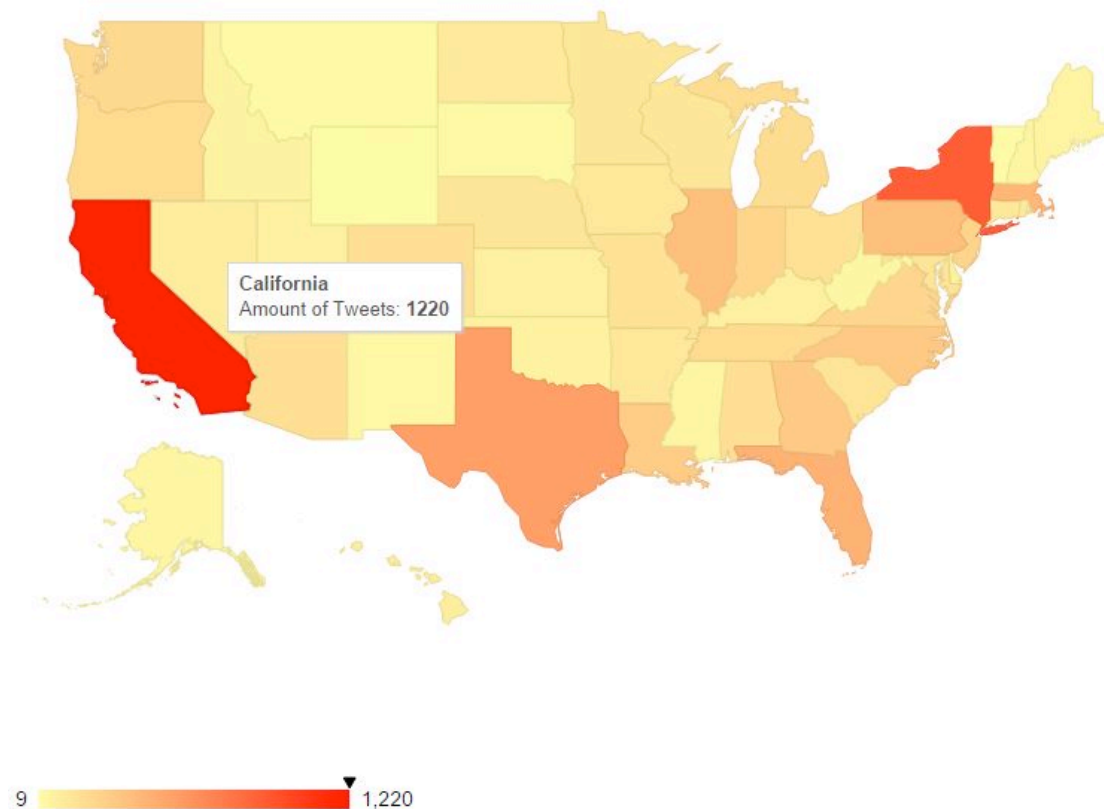
Figure 2-3 State map

The state map is created through Google Chart API (Geochart), which is also very easy to use. What we should do first is to include such code inside our html file.

```
<script src='http://www.google.com/jsapi'></script>
```

The whole process to draw this state map is quite similar as how we draw the heat map: we first load the geochart source from Google. And then translate our raw data's format into what the Google API is asking for, build up the chart option we want to use, and finally set up the event to draw the chart. The key part of the program is shown below.

```
google.load('visualization', '1', {'packages': ['geochart']});
google.setOnLoadCallback(getdata);
function getdata() {
    $.getJSON(
        "../jsonsender/jsonsender.php", // The server URL
        {id: 0, who: 1},
        showdata // The function to call on completion.
    );
}
function showdata(json) {
    var tweetData = [];
    for (var i = 0; i < eval(json).length; i++){
```

```
        tweetData[i] = parseInt(json[i].state_count);
    }
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Country');
    data.addColumn('number', 'Amount of Tweets');
    data.addRows([[{v:'US-AK',f:'Alaska'},tweetData[0]]]);
    data.addRows([[{v:'US-AL',f:'Alabama'},tweetData[1]]])
    data.addRows([[{v:'US-AR',f:'Arkansas'},tweetData[2]]]);
    …
    …
    var options = {
        backgroundColor: {fill:'#FFFFFF',stroke:'#FFFFFF' ,strokeWidth:0 },
        colorAxis: {colors: ['#FFFF99', 'FF0000']},
        datalessRegionColor: '#f5f5f5',
        displayMode: 'regions',
        enableRegionInteractivity: 'true',
        resolution: 'provinces',
        region:'US',
        keepAspectRatio: true,
        width:800,
        height:700,
        tooltip: {textStyle: {color: '#444444'}, trigger:'hover'}
    };
    var chart = new google.visualization.GeoChart(document.getElementById('visual_geochart'));
    chart.draw(data, options);
}
```

Inside the option codes, the most important thing we should remember is that we have to set the resolution option to 'provinces', otherwise the map would display in the world wide mode.

3. IOS device

Most of the information after analysis is displayed through IOS device in this demo.

3.1. File structure

In the Figure 3-1, the main design files are under the hma2014_v2 folder. Because of the MVC mechanism in IOS, it is convenience to separate the static UI drawing and dynamic events response. UI designs are implemented in the Main.storyboard file which is shown as Figure 3-2. Each IOS page is corresponding to a class file in the fold and the UI elements in the page are linked to the code in the class file. PNChart

fold contains the code for drawing charts, e.g., bar chart, pie chart and line chart.
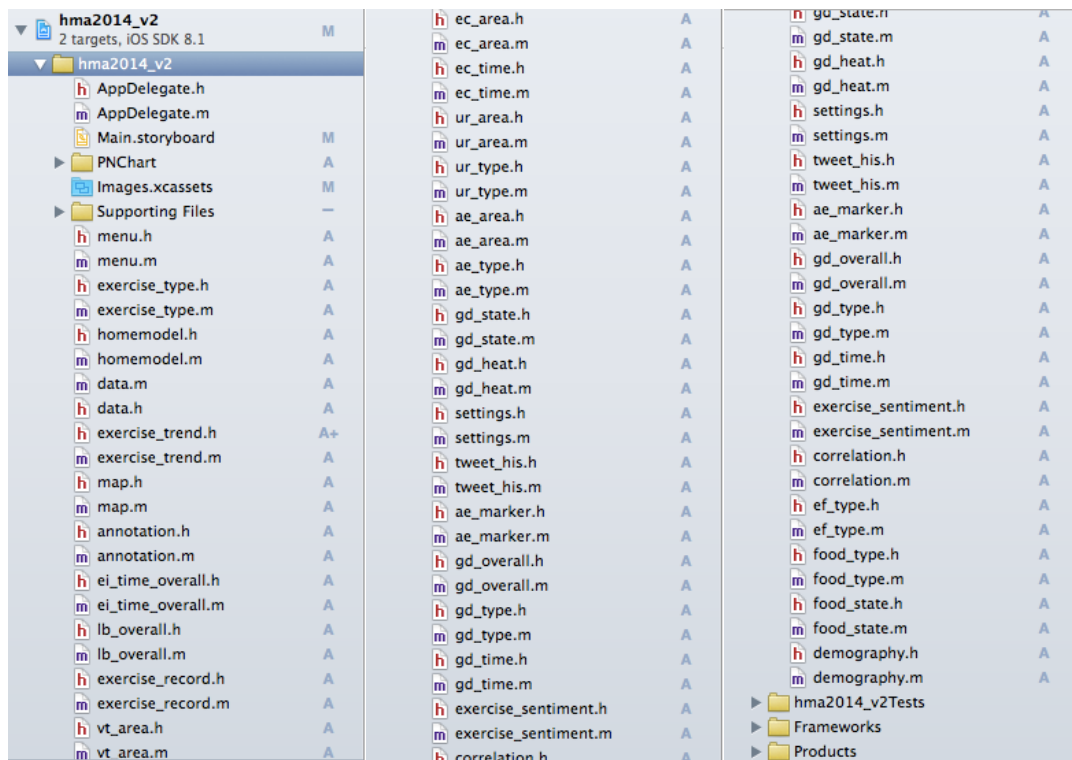
Figure 3-1 IOS folder
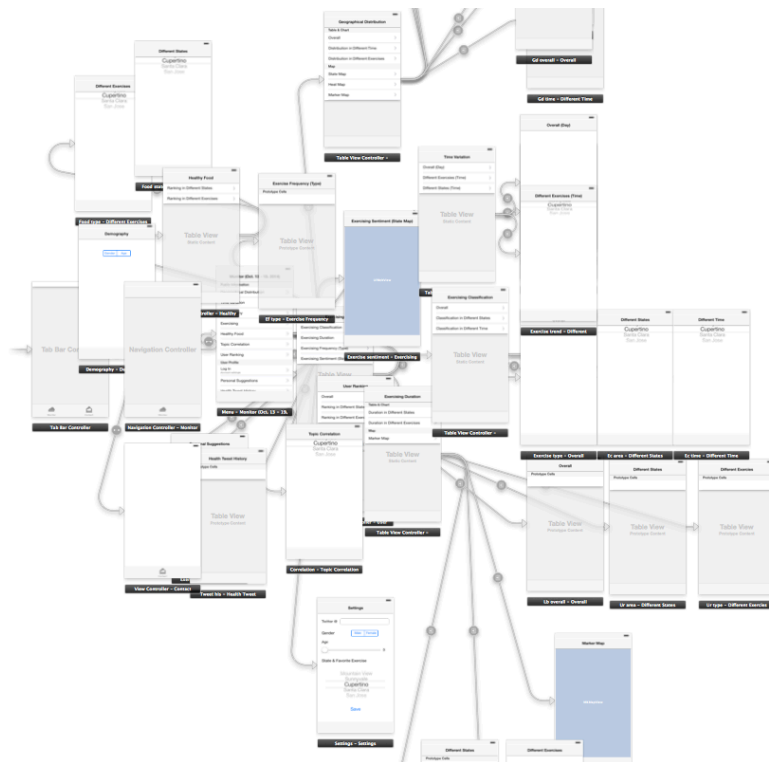
Figure 3-2 UI design

3.2. Receive data

The files called data describe the data structure or members when communicating to the server, and the homemodel is a protocol, delegated by other classes for the communication purpose. The code in the data.h file is shown as below:

```objc
@interface data : NSObject
@property (nonatomic, strong) NSString *screen_name;
@property (nonatomic, strong) NSString *tweet_text;
@property (nonatomic, strong) NSString *profile_img_url;
@property (nonatomic, strong) NSString *exercise_count;
@property (nonatomic, strong) NSString *exercise_type;
@property (nonatomic, strong) NSString *date_count;
@property (nonatomic, strong) NSString *geo_lat;
@property (nonatomic, strong) NSString *geo_long;
@property (nonatomic, strong) NSString *state_name;
@property (nonatomic, strong) NSString *state_count;
@property (nonatomic, strong) NSString *mean;
@property (nonatomic, strong) NSString *max;
@property (nonatomic, strong) NSString *std;
@property (nonatomic, strong) NSString *created_at;
@property (nonatomic, strong) NSString *exercise_time;
@property (nonatomic, strong) NSString *health_count;
@property (nonatomic, strong) NSString *fruit_count;
@property (nonatomic, strong) NSString *gender_age;
@property (nonatomic, strong) NSString *demography_count;
@property (nonatomic, strong) NSString *exercise_time_max;
@property (nonatomic, strong) NSString *exercise_freq;
@property (nonatomic, strong) NSString *exercise_freq_max;
@property (nonatomic, strong) NSString *food_state;
@property (nonatomic, strong) NSString *food_exercise;

@end
```

In the homemodel.m, the data requests, i.e. feature tables in the mysql database, are distinguish by the index argument, and the mutable array q is the data will be sent to the server.

```objc
- (void)downloaditems: (int) index post:(NSMutableArray *)p
{
    NSString *myRequestString;
    // Download the json file
    if (index == 6 || index == 13) {
        myRequestString = [NSString
stringWithFormat:@"id=%d&usr=%@&gender=%@&age=%@&state=%@&exercise=%@",index,p[0],p[1],p[2],p[3],p[4]];
```

```
    }
    else if(index == 14){
        myRequestString = [NSString stringWithFormat:@"id=%d&usr=%@",index,p[0]];
    }
    else{
        myRequestString = [NSString stringWithFormat:@"id=%d",index];
    }


    NSData *myRequestData = [NSData dataWithBytes: [myRequestString UTF8String] length:
[myRequestString length]];
    NSMutableURLRequest *urlRequest = [[NSMutableURLRequest alloc] initWithURL:
[NSURL URLWithString: @"http://172.20.10.2/~yaodongyang/jsonsender/jsonsender.php"]];
    [urlRequest setHTTPMethod: @"POST"];
    [urlRequest setValue:@"application/x-www-form-urlencoded"
forHTTPHeaderField:@"content-type"];
    [urlRequest setHTTPBody: myRequestData];
    [NSURLConnection connectionWithRequest:urlRequest delegate:self];

}
```

This is a post method and it will communicate with the jsonsender.php in the server file fold. 172.20.10.2 is the private IP of the server. Other codes in the homemodel parse the JSON data coming from the server (code fraction 1) and call the itemdownloaded function in the delegating classes (code fraction 2).

Code fraction 1:

```
- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
    // Create an array to store the locations
    NSMutableArray *_data = [[NSMutableArray alloc] init];

    // Parse the JSON that came in
    NSError *error;
    NSArray *jsonArray = [NSJSONSerialization JSONObjectWithData:_downloadedData
options:NSJSONReadingAllowFragments error:&error];

    // Loop through Json objects, create question objects and add them to our questions array
    for (int i = 0; i < jsonArray.count; i++)
    {
        NSDictionary *jsonElement = jsonArray[i];

        // Create a new location object and set its props to JsonElement properties
        data *newdata = [[data alloc] init];
        newdata.tweet_text = jsonElement[@"tweet_text"];
        newdata.profile_img_url = jsonElement[@"profile_image_url"];
```

```objectivec
        newdata.screen_name = jsonElement[@"screen_name"];
        newdata.exercise_count = jsonElement[@"exercise_count"];
        newdata.date_count = jsonElement[@"time_count"];
        newdata.geo_lat = jsonElement[@"geo_lat"];
        newdata.geo_long = jsonElement[@"geo_long"];
        newdata.state_name = jsonElement[@"state_name"];
        newdata.state_count = jsonElement[@"state_count"];
        newdata.exercise_type = jsonElement[@"exercise_type"];
        newdata.mean = jsonElement[@"mean"];
        newdata.max = jsonElement[@"max"];
        newdata.std = jsonElement[@"std"];
        newdata.created_at = jsonElement[@"created_at"];
        newdata.exercise_time = jsonElement[@"exercise_time"];
        newdata.health_count = jsonElement[@"health_count"];
        newdata.fruit_count = jsonElement[@"fruit_count"];
        newdata.demography_count = jsonElement[@"demography_count"];
        newdata.gender_age = jsonElement[@"gender_age"];
        newdata.exercise_time_max = jsonElement[@"exercise_time_max"];
        newdata.exercise_freq = jsonElement[@"exercise_freq"];
        newdata.exercise_freq_max = jsonElement[@"exercise_freq_max"];
        newdata.food_state = jsonElement[@"food_state"];
        newdata.food_exercise = jsonElement[@"food_exercise"];

        // Add this question to the locations array
        [_data addObject:newdata];
    }

    // Ready to notify delegate that data is ready and pass back items
    if (self.delegate)
    {
        [self.delegate itemsdownloaded:_data];
    }
}
```

Code fraction 2:

```objectivec
- (void)itemsdownloaded:(NSArray *)items
{
    _feeditems = items;
    feature = [[NSMutableArray alloc] init];

    for (int i=0; i < [_feeditems count]; i++) {
        data *item = _feeditems[i];
        [feature addObject:item.state_count];
    }
```

```
NSString* string0 = [NSString stringWithFormat:@"running: "];
ec_state_label.text = [string0 stringByAppendingString:feature[0*state_num+indexx]];

    [self display: 0];

}
```

3.3. Display

3.3.1.   Bar chart

Take ec_area.m for example. The index here is the index in the picker in order to
change the display when selecting different states.

```
- (void)display: (int) index {

    indexx = index;

    self.barChart = [[PNBarChart alloc] initWithFrame:CGRectMake(5, 160.0,
SCREEN_WIDTH, 300.0)];
    self.barChart.backgroundColor = [UIColor clearColor];
    self.barChart.yLabelFormatter = ^(CGFloat yValue){
        CGFloat yValueParsed = yValue;
        NSString * labelText = [NSString stringWithFormat:@"%1.f",yValueParsed];
        return labelText;
    };
    self.barChart.labelMarginTop = 5.0;
    [self.barChart
setXLabels:@[@"Run",@"Cycle",@"Swim",@"Basket",@"Volley",@"Tennis",@"FootB"]];

    [self.barChart
setYValues:@[feature[0*state_num+index],feature[1*state_num+index],feature[2*state_num+ind
ex],feature[3*state_num+index],feature[4*state_num+index],feature[5*state_num+index],feature[
6*state_num+index]]];
    [self.barChart
setStrokeColors:@[PNTwitterColor,PNTwitterColor,PNTwitterColor,PNTwitterColor,PNTwitter
Color,PNTwitterColor,PNTwitterColor]];

    [self.barChart strokeChart];
    self.barChart.delegate = self;
    [self.view addSubview:self.barChart];
}
```

3.3.2.   Line chart

Take vt_area.m for example.
- (void)display_f2: (int) index p2:(int)index2{

```
    /*
    //Add LineChart
    lineChartLabel = [[UILabel alloc] initWithFrame:CGRectMake(0, 90, SCREEN_WIDTH,
30)];
    lineChartLabel.text = @"Exercise Trend";
    lineChartLabel.textColor = PNMauve;
    lineChartLabel.font = [UIFont fontWithName:@"Avenir-Medium" size:23.0];
    lineChartLabel.textAlignment = NSTextAlignmentCenter;
    */


    lineChart = [[PNLineChart alloc] initWithFrame:CGRectMake(5, 190.0, SCREEN_WIDTH,
300.0)];


    lineChart.yLabelFormat = @"%1.1f";
    lineChart.backgroundColor = [UIColor clearColor];
    [lineChart
setXLabels:@[@"Morning",@"Afternoon",@"Evening",@"Night",@"Overnight"]];
    lineChart.showCoordinateAxis = YES;


    // Line Chart Nr.1
    NSArray * data01Array;


    data01Array = @[morning[index], afternonn[index], evening[index],night[index],
overnight[index]];


    PNLineChartData *data01 = [PNLineChartData new];
    data01.color = PNTwitterColor;
    data01.itemCount = lineChart.xLabels.count;
    data01.inflexionPointStyle = PNLineChartPointStyleCycle;
    data01.getData = ^(NSUInteger index) {
        CGFloat yValue = [data01Array[index] floatValue];
        return [PNLineChartDataItem dataItemWithY:yValue];
    };


    // Line Chart Nr.2
    NSArray * data02Array;


    data02Array = @[morning[index2], afternonn[index2], evening[index2],night[index2],
```

overnight[index2]];


```
    PNLineChartData *data02 = [PNLineChartData new];
    data02.color = PNFreshGreen;
    data02.itemCount = lineChart.xLabels.count;
    data02.inflexionPointStyle = PNLineChartPointStyleSquare;
    data02.getData = ^(NSUInteger index) {
        CGFloat yValue = [data02Array[index] floatValue];
        return [PNLineChartDataItem dataItemWithY:yValue];
    };


    if (index2 == 0) {
        lineChart.chartData = @[data01];
    }
    else{
        lineChart.chartData = @[data01, data02];
    }

    [lineChart strokeChart];

    lineChart.delegate = self;

    //[self.view addSubview:lineChartLabel];
    [self.view addSubview:lineChart];

}
```

3.3.3.   Pie chart

Take demography.m for example.

```
MALE.text = @"MALE";
        FEMALE.text = @"FEMALE";
        NSArray *items;
        items = @[
                    [PNPieChartDataItem dataItemWithValue:[demography_count[0] intValue]
color:PNTwitterColor],
                    [PNPieChartDataItem dataItemWithValue:[demography_count[1] intValue]
color:PNRed],
                    ];


        pieChart = [[PNPieChart alloc] initWithFrame:CGRectMake(40.0, 155.0, 240.0, 240.0)
```

items:items];

```
        pieChart.descriptionTextColor = [UIColor whiteColor];
        pieChart.descriptionTextFont   = [UIFont fontWithName:@"Avenir-Medium"
size:14.0];
        pieChart.descriptionTextShadowColor = [UIColor clearColor];
        [pieChart strokeChart];


        [self.view addSubview:pieChart];
```

3.3.4.   Table

When using table to display large amount of data by scrolling the screen, it is needed to inherit the UITableViewController father class. Then override the functions to set the table's attributes. Take ef_type.m for example.

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return _feeditems.count;
}


- (NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section
{
    return [sections objectAtIndex:section];
}


- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{

    // Return the number of rows in the section.
    return 3;
}


- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    // Retrieve cell
    NSString *cellIdentifier = @"ef_type_cell";
    UITableViewCell *myCell = [tableView dequeueReusableCellWithIdentifier:cellIdentifier];

    // Get the location to be shown
    data *item = _feeditems[indexPath.section];
```

```
    switch (indexPath.row) {
        case 0:
        myCell.textLabel.text = [NSString stringWithFormat:@"Average in a week: %@
days",item.mean];
        break;
        case 1:
        myCell.textLabel.text = [NSString stringWithFormat:@"Maximun in a week: %@
days",item.max];
        break;
        case 2:
        myCell.textLabel.text = [NSString stringWithFormat:@"Individual difference:
%@",item.std];
        break;

        default:
        break;
    }

    return myCell;
}
```

3.3.5.   Map

Except for the marker map which is implemented by the inner MapKit framework
(code fraction 1), the state map and heat map are implemented by the webview (code
fraction 2). The webview is a browser in the IOS app, so it loads the html file in the
server.

Code fraction 1:

```
- (void)locationManager:(CLLocationManager *)manager didUpdateToLocation:(CLLocation
*)newLocation fromLocation:(CLLocation *)oldLocation {

    [locationManager stopUpdatingLocation];

    NSString *strLat = [NSString stringWithFormat:@"%.4f",newLocation.coordinate.latitude];
    NSString *strLng = [NSString
stringWithFormat:@"%.4f",newLocation.coordinate.longitude];
    NSLog(@"Lat: %@    Lng: %@", strLat, strLng);

    CLLocationCoordinate2D coords =
CLLocationCoordinate2DMake(newLocation.coordinate.latitude,newLocation.coordinate.longitu
de);
    float zoomLevel = 1;
```

```
    MKCoordinateRegion region =
MKCoordinateRegionMake(coords,MKCoordinateSpanMake(zoomLevel, zoomLevel));
    [mapview setRegion:[mapview regionThatFits:region] animated:YES];
}
```

Code fraction 2:

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    NSString *fullurl = @"http://172.20.10.2/~yaodongyang/web/StateMap.html";
    NSURL *url = [NSURL URLWithString:fullurl];
    NSURLRequest *request = [NSURLRequest requestWithURL:url];
    [gd_webview loadRequest:request];
}
```

4. Android application

4.1. File structure

The figure 4-1 shows all the java classes that are used in this application. It includes three parts. The first part is the Activity files, which contains all the activities of every user interface. The second part is the JSONSender class, which is used for getting JSON data from the database through the server. The third part is all the other classes which are used to store information of twitter users.
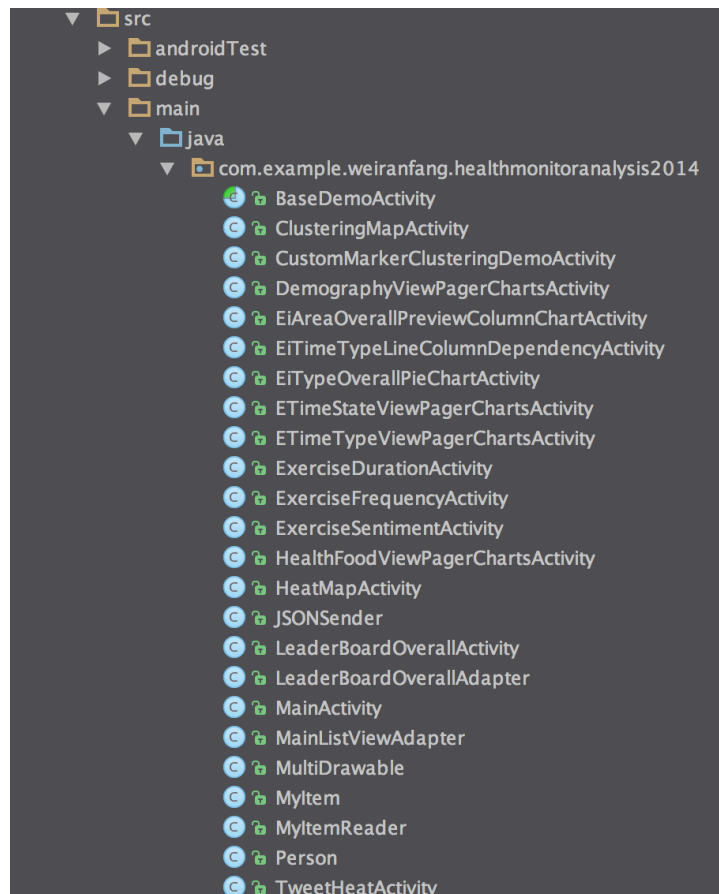
Figure 4-1 Java Class files

The Figure 4-2 shows all the layout files for every activity class and all fragment files that are used for UI design.
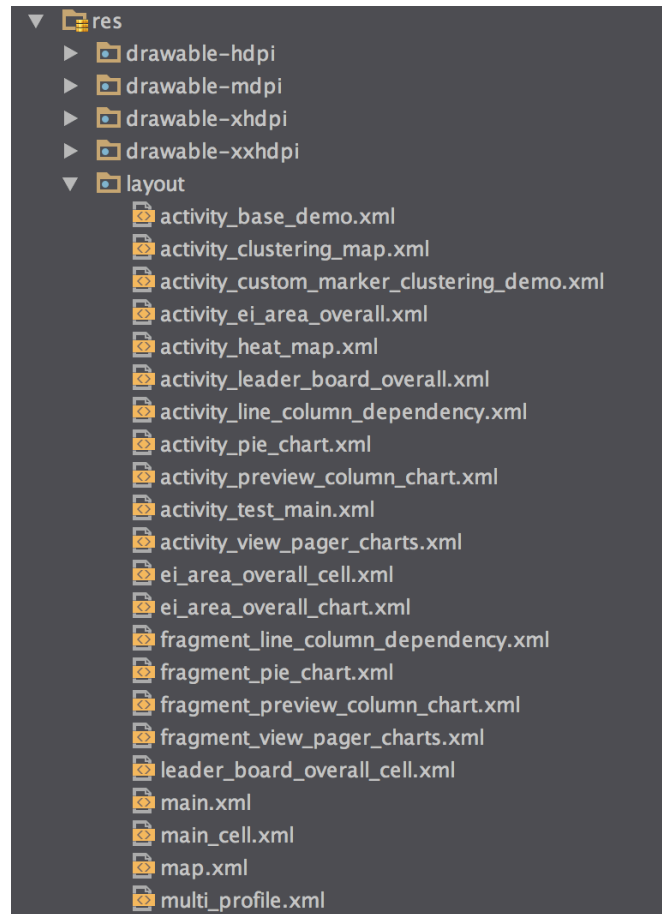

Figure 4-2 Layout files

4.2. Receive data

The JSONSender.java is used to get the data from the database and store and pass them in JSONArray. The main method in this class is GetArrayByID(int id), where id is the id number of our features specified in our jsonsender.php file.

public class JSONSender {

```java
public JSONArray GetArrayByID(int id) {

    // URL for getting all customers
    String url = "http://192.168.1.10/hma2014/jsonsender/jsonsender.php?id=" +
id;

    // Get HttpResponse Object from url.
    // Get HttpEntity from Http Response Object
    HttpEntity httpEntity = null;

    try {
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpGet httpGet = new HttpGet(url);
        HttpResponse httpResponse = httpClient.execute(httpGet);
        httpEntity = httpResponse.getEntity();
    } catch (ClientProtocolException e) {
        // Signals error in http protocol
        e.printStackTrace();
        //Log Errors Here
    } catch (IOException e) {
        e.printStackTrace();
    }

    // Convert HttpEntity into JSON Array
    JSONArray jsonArray = null;

    if (httpEntity != null) {
        try {
            String entityResponse = EntityUtils.toString(httpEntity);
            Log.e("Entity Response    : ", entityResponse);
```

```
                    jsonArray = new JSONArray(entityResponse);

              } catch (JSONException e) {

                    e.printStackTrace();

              } catch (IOException e) {

                    e.printStackTrace();

              }

        }

        return jsonArray;

    }

}
```

4.3. Display

4.3.1. Pie chart

Fields:

```
private PieChartView chart;
private PieChartData data;
private boolean hasLabels = true;
private boolean hasLabelsOutside = true;
private boolean hasCenterCircle = false;
private boolean hasCenterText1 = false;
private boolean hasCenterText2 = false;
private boolean isExploaded = false;
private boolean hasArcSeparated = false;
private boolean hasLabelForSelected = false;
```

Methods:

```java
public boolean onOptionsItemSelected(MenuItem item)

private void reset()

private void generateData(JSONArray jsonArray)

private void explodeChart()

private void separateSingleArc()

private void toggleLabelsOutside()

private void toggleLabels()

private void toggleLabelForSelected()

private void prepareDataAnimation()
```

Inner classes:

```java
private class ValueTouchListener implements PieChartOnValueTouchListener {

    public void onValueTouched(int selectedArc, ArcValue arcValue)

    public void onNothingTouched()

}


private class GenerateDataTask extends AsyncTask<JSONSender,Long,JSONArray> {

    protected JSONArray doInBackground(JSONSender... params)

    protected void onPostExecute(JSONArray jsonArray)

}
```

4.3.2. Bar chart

Inner Classes:

```java
public static class PlaceholderFragment extends Fragment {

    public final static String[] states = new String[] { "AK", "AL", "AR", "AZ", "CA",
```

```
"CO", "CT", "DE", "FL", "GA", "HI", "IA", "ID", "IL", "IN", "KS", "KY", "LA", "MA",
"MD", "ME", "MI", "MN", "MO", "MS", "MT", "NC", "ND", "NE", "NH", "NJ", "NM",
"NV", "NY", "OH", "OK", "OR", "PA", "RI", "SC", "SD", "TN", "TX", "UT", "VA",
"VT", "WA", "WI", "WV", "WY"};
    private ColumnChartView chart;
    private PreviewColumnChartView previewChart;
    private ColumnChartData data;
    private ColumnChartData previewData;

    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState)
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)
    public boolean onOptionsItemSelected(MenuItem item)
    private void generateData(JSONArray jsonArray)
    private void previewY()
    private void previewX(boolean animate)
    private void previewXY()
}

private class ViewportListener implements ViewportChangeListener {
    public void onViewportChanged(Viewport newViewport)
}

private class GenerateDataTask extends AsyncTask<JSONSender,Long,JSONArray> {
    protected JSONArray doInBackground(JSONSender... params)
    protected void onPostExecute(JSONArray jsonArray)
}
```

4.3.3. Line & Column Dependency chart

Inner classes:

```java
public static class PlaceholderFragment extends Fragment {
    public final static String[] types = new String[] { "Running", "Cycling", "Swimming",
"Basketball", "Volleyball", "Tennis", "Football",};
    public final static String[] clocks = new String[] { "0:00", "1:00", "2:00", "3:00",
"4:00", "5:00", "6:00", "7:00", "8:00", "9:00", "10:00", "11:00", "12:00", "13:00",
"14:00", "15:00", "16:00", "17:00", "18:00", "19:00", "20:00", "21:00", "22:00",
"23:00", };
    private LineChartView chartTop;
    private ColumnChartView chartBottom;
    private LineChartData lineData;
    private ColumnChartData columnData;

    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState)
    private void generateColumnData(JSONArray jsonArray)
    private void generateInitialLineData()
    private void generateLineData(int color, float range)
    private void generateRunningLineData(JSONArray jsonArray)
    private void generateCyclingLineData(JSONArray jsonArray)
    private void generateSwimmingLineData(JSONArray jsonArray)
    private void generateBasketballLineData(JSONArray jsonArray)
    private void generateVolleyballLineData(JSONArray jsonArray)
    private void generateTennisLineData(JSONArray jsonArray)
    private void generateFootballLineData(JSONArray jsonArray)
    private class ValueTouchListener implements
ColumnChartView.ColumnChartOnValueTouchListener{}}
```

```
}

private class GenerateDataTask extends AsyncTask<JSONSender,Long,JSONArray> {
    protected JSONArray doInBackground(JSONSender... params)
    protected void onPostExecute(JSONArray jsonArray)
}
```

4.3.4. Heat Map

Fields:

```
private static final int ALT_HEATMAP_RADIUS = 10;
private static final double ALT_HEATMAP_OPACITY = 0.4;
private static final int[] ALT_HEATMAP_GRADIENT_COLORS = {
    Color.argb(0, 0, 255, 255),// transparent
    Color.argb(255 / 3 * 2, 0, 255, 255),
    Color.rgb(0, 191, 255),
    Color.rgb(0, 0, 127),
    Color.rgb(255, 0, 0)
};
public static final float[] ALT_HEATMAP_GRADIENT_START_POINTS = {
    0.0f, 0.10f, 0.20f, 0.60f, 1.0f
};

public static final Gradient ALT_HEATMAP_GRADIENT = new
Gradient(ALT_HEATMAP_GRADIENT_COLORS,
            ALT_HEATMAP_GRADIENT_START_POINTS);
```

private HeatmapTileProvider mProvider;

private TileOverlay mOverlay;

private boolean mDefaultGradient = true;

private boolean mDefaultRadius = true;

private boolean mDefaultOpacity = true;

private HashMap<String, DataSet> mLists = new HashMap<String, DataSet>();

Methods:

protected int getLayoutId()

protected void startDemo()

public void changeRadius(View view)

public void changeGradient(View view)

public void changeOpacity(View view)

private ArrayList<LatLng> readItems(int resource)

private ArrayList<LatLng> readItems(JSONArray jsonArray)

Inner classes:

public class SpinnerActivity implements AdapterView.OnItemSelectedListener {

    public void onItemSelected(AdapterView<?> parent, View view,

                    int pos, long id)

    public void onNothingSelected(AdapterView<?> parent)

}

private class DataSet {

    private ArrayList<LatLng> mDataset;

    private String mUrl;

    public DataSet(ArrayList<LatLng> dataSet, String url)

    public ArrayList<LatLng> getData()

```
    public String getUrl()
}
```

```
private class GenerateJSONArrayTask extends
AsyncTask<JSONSender,Long,JSONArray> {
    protected JSONArray doInBackground(JSONSender... params)
}
```

4.3.5. Marker Map

Field:

```
private ClusterManager<MyItem> mClusterManager;
```

Methods:

```
protected void startDemo()
private void readItems()
```

Inner Classes:

```
private class GenerateJSONArrayTask extends
AsyncTask<JSONSender,Long,JSONArray> {
        @Override
        protected JSONArray doInBackground(JSONSender... params) {
            // it is executed on Background thread
            return params[0].GetArrayByID(15);
        }
}
```