

Software Engineering

# **Health Monitoring Analytics**

The Report #2 of Group #1

Junjie Feng, Kai Kang, Ruotian Zhang  
Tianzhe Wang, Weidi Zhang, Zhijie Zhang

<https://sites.google.com/a/scarletmail.rutgers.edu/health-monitoring-analytics/?pli=1>

Nov 9, 2015

# **Individual Contributions Breakdown**

“All team members contributed equally”

# Table of Contents

Individual Contributions Breakdown.....	1
1. Interaction Diagrams.....	4
1.1 Use Case 1: Data Collecting and Classify.....	4
1.2 Use Case 2: User Registration.....	5
1.3 Use Case 3: Data Classifying.....	6
1.4 Use Case 5: Heat Map.....	7
1.5 Use Case 13 & 20: Search for Information & Display.....	8
2 Class Diagram and Interface Specification.....	9
2.1 Class Diagram.....	9
2.2. Data Types and Operation Signatures.....	10
2.2.1. Database.....	10
2.2.2. TweetHeat.....	11
2.2.3. FrequencyAnalysis.....	12
2.2.4. DurationAnalysis.....	13
2.2.5. SentimentAnalysis.....	13
2.2.6. CorrelationCalculate.....	14
2.2.7. PersonalAnalysis.....	14
2.2.8. DemographyAnalysis.....	15
2.2.9. Controller.....	16
2.2.10. Communicator.....	16
2.3. Traceability Matrix.....	17
3. System Architecture and System Design.....	20
3.1 Architectural Style.....	20
3.2 Identifying Subsystems.....	20
3.3 Mapping System to Hardware.....	21
3.4. Persistent Data Storage.....	22
3.4.1 Database improvement.....	22
3.4.2 Registered users.....	22
3.5 Network Protocol.....	23
3.6. Global Control Flow & Hardware Requirements.....	24
4. Algorithms and Data Structures.....	25
4.1 KNN.....	25
4.1.1 Basic Concept of KNN.....	26
4.1.2 KNN Implementation.....	27
4.2 NLTK TextBlob.....	29
4.3 SVM.....	30
4.4 Data Structure.....	31
5. User Interface Design and Implementation.....	33
5.1 Website.....	33
5.1.1 Homepage.....	33
5.1.2 Login and Signup page.....	34
5.1.3 Sports page.....	35

5.2 Android Application.....	36
6. Design of Tests.....	40
6.1 Overall Description.....	40
6.2 Function unit test.....	40
6.2.1 Test Unit: Twitter retrieve.....	40
6.2.2 Test Unit: Data base.....	41
6.2.3 Test Unit: Useful tweets and advertising.....	41
6.2.4 Test Unit: Self-involve and “audience” tweets.....	42
6.2.5 Test Unit: Heat map and Sports intension & involvement.....	43
6.2.6 Test Unit: Exercise history.....	43
6.2.8 Test Unit: Android application registration.....	45
6.2.9 Test Unit: Android application display page.....	45
6.3 Test Coverage.....	45
6.4 Integration testing strategy.....	46
7 Plan of Work.....	48
7.1 Merging the Contributions from Individual Team Members.....	48
7.2 Project Coordination and Progress Report.....	48
7.3 Plan of Work.....	49
7.4. Breakdown of Responsibilities.....	50
7.4.1. Project basic structure work.....	50
7.4.2 Product ownership (Features).....	51
7.4.3 Breakdown of responsibilities (Report 2).....	52
8 References.....	54

# 1. Interaction Diagrams

## 1.1 Use Case 1: Data Collecting and Classify

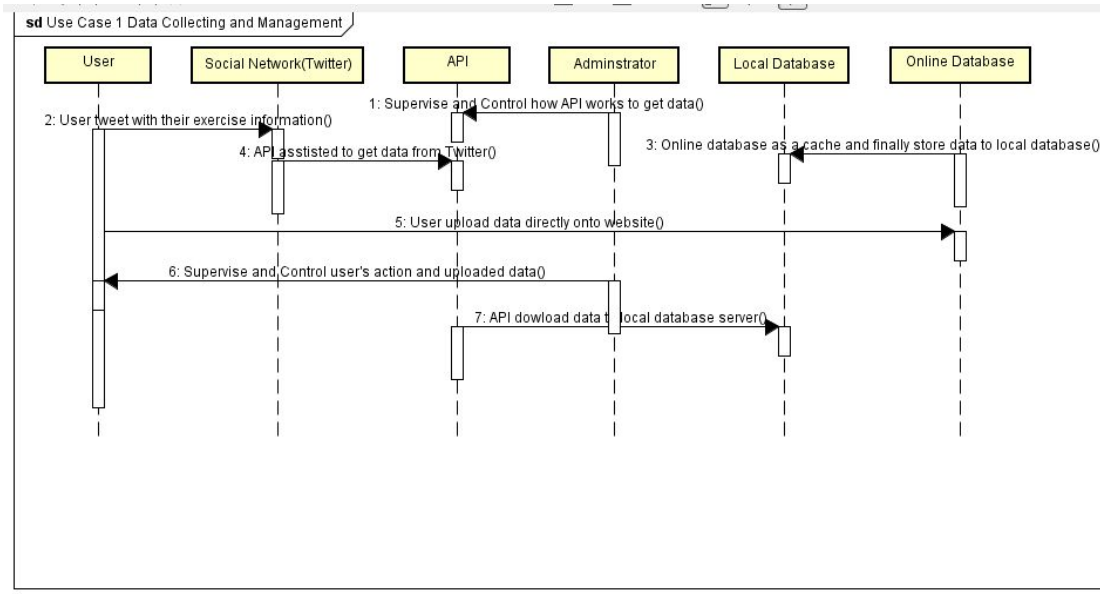


Figure 1-1 Use Case 1: Data Collecting and Classify

The figure above comes from the use case 1: Data Collection and Management. First the administrator control how the system get tweeter from API from tweeter web: the mining speed, frequency, or other parameter. At mean time, all the user publish their tweeters on line, we get relative ones via certain words. The next stop, we store the data into 'online-database' and the online database will communicate with our local database. Next, user upload their data or request like searching certain activities on line. Controller or supervise need to check if their actions is malicious. Finally, we use API to download data into our database.

## 1.2 Use Case 2: User Registration

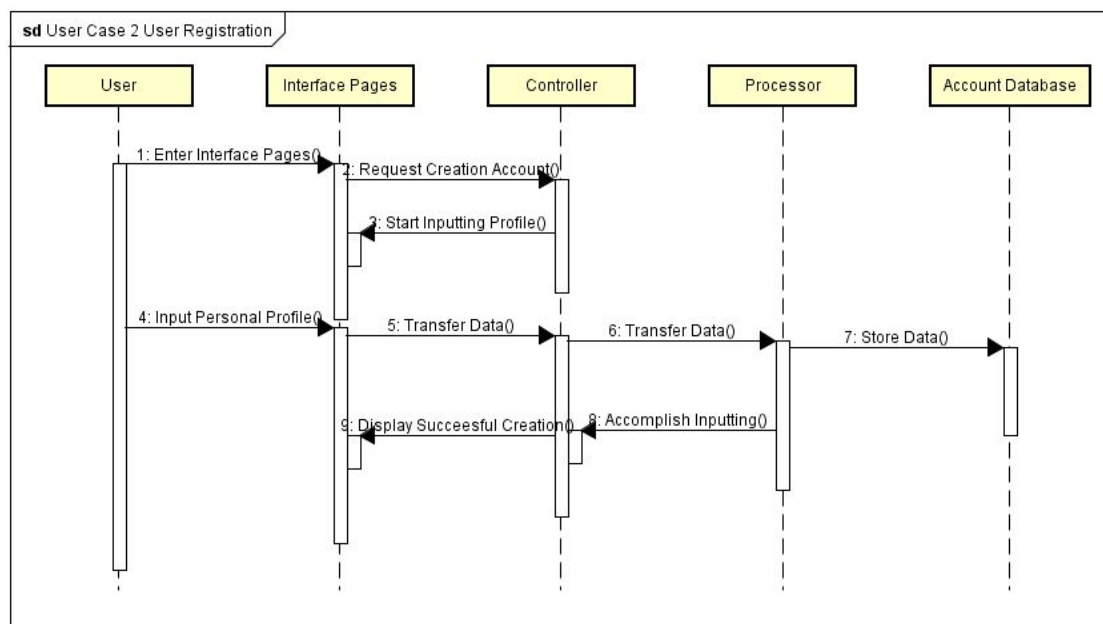


Figure 1-2 Use Case 2 User Registration

The figure above comes from user case 2: User Registration. First, user enter our web site and select “Create a new account”. Then system will receive this request and let user input personal profile on the web site. Next all profile data transfer to account database and the data is stored there. While system check out that all the blanks in profile have received the data, the interface page will show “Successful creating your account” on the screen.

## 1.3 Use Case 3: Data Classifying

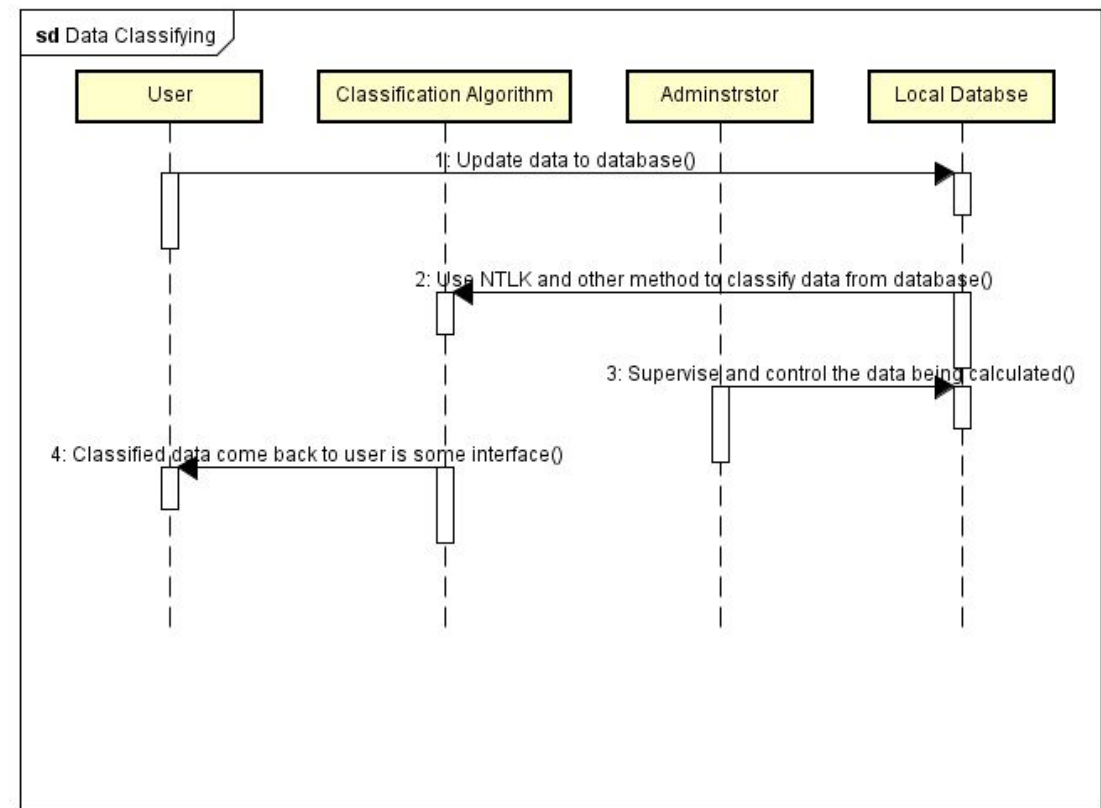


Figure 1-3 Use Case 3: Data Classifying

The figure above comes from user case 3: Data Classifying. The users' tweeter will be sorted and stored into our database. Then the classifying service will classify the data using NLTK which is a lib for Python. After classification, the supervisor will use another advanced method to put them into more specific categories. And the last step is to give back all the information to application to display.

## 1.4 Use Case 5: Heat Map

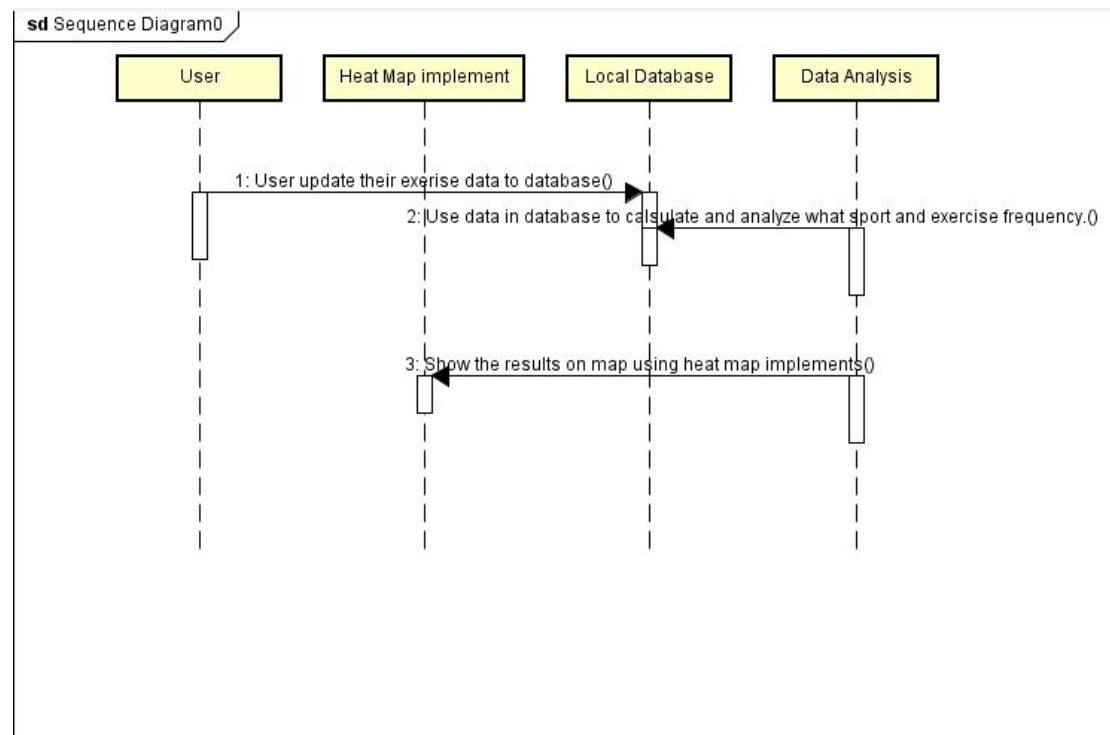


Figure 1-4 Use Case 5: Heat Map

The figure above comes from user case5: Heat Map. The first step is all the users data will be filtered, sorted, classified and stored into the database. Then all these valid data will be calculated and generated useful data for analyze. The last step users can see many kinds of data through heat map to get a general impression about the activities trend.



## 1.5 Use Case 13 & 20: Search for Information & Display

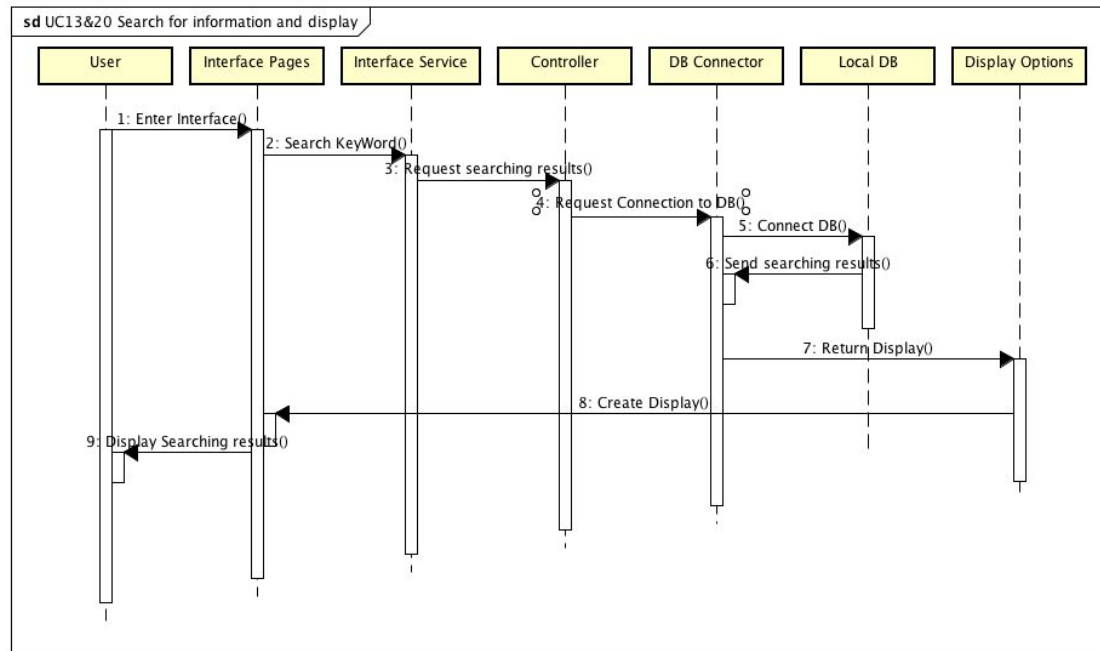


Figure 1-5 Use Case 13 & 20 Search for Information and Display

The function of Use Case 13 & 20 is displaying the information of searching request made by the user. First, any user opened the website, simply type in the key words they want to search and hit enter on the keyboard or click on the search button. Then the Interface Service emits the request to Controller for getting searching results. Since the controller could not get touch with the database directly, it should send the request to the DB Connector for connection to the local DB. After Local DB returns the related information to the DB Connector, the specific data for user is sent to the Display Options for proper ways to display. Correlate display ways for searching results are created by Display Options and should be posted to Interface Pages. At last the Interface Pages display the searching results for user on the screen.

# 2 Class Diagram and Interface Specification

## 2.1 Class Diagram

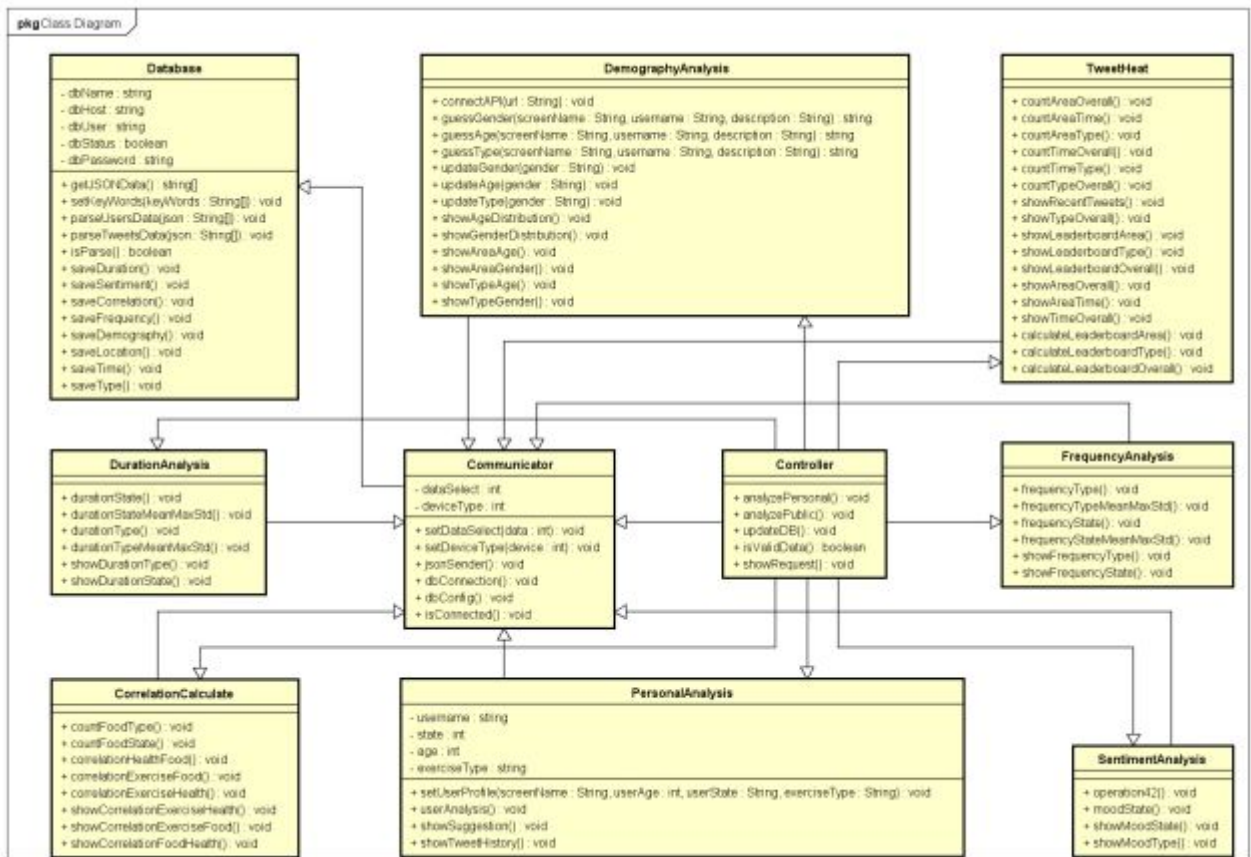


Figure 2-1 Class Diagram

This figure shows up the class diagram of our system. We distribute our system functions into ten classes.

Before other classes could accomplish their functions, the “database” class will be called first in order to collect tweets data, analyze the data, parse the data in JASON format, and store the data after analysis.

“Communicator” class is the class which deals with the work between showing function and the database. It could control what data should be transferred from the database and should be shown up on which screen.

“Controller” class controls all the system include user’s demand and system control. From this class, other classes function would know when to run their own analysis functions.

## 2.2. Data Types and Operation Signatures

Note that, since we are building our system based on the previous groups' work, some basic classes and functions remains the same as previous groups' because they are already well implemented.

### 2.2.1. Database

The "Database" class contains five variables as follows that stores the basic information of the database used for the connection and coordination.

***dbName*** String variable storing the name of the database.

***dbHost*** String variable storing the host of the database.

***dbUser*** String variable storing the username of the database.

***dbStatus*** Boolean variable indicating the status of the database, either open or closed.

***dbPassword*** String variable storing the password corresponding to the username.

The functions of this class are listed below. Functions include several get functions that are used to obtain data from Twitter. Also they contain several save functions that are used for storing analytical data into database.

***getJSONData()*** This function is used for obtaining raw tweets data from Twitter Streaming API and returns an array of JSON value, each element represents one piece of tweet data.

***setKeywords(keyWords : String[])*** This function is used for setting the keywords that are needed for filtering the tweets. As for our project, all the key words should be about health and exercise.

***parseUsersData(json : String[])*** This function is used for parsing the collected JSON data containing user information into readable user data and storing them in the users table of our database.

***parseTweetsData(json : String[])*** This function is used for parsing the collected JSON data which contains tweets information into readable tweet data and store them in different tables.

***isParse()*** This function is used for check the parse result of each piece of JSON data. If data has been parsed, return true. Else, return false.

***saveDuration()*** This function is used for saving the analytical data related to the exercise duration time into new feature tables of the database.

***saveSentiment()*** This function is used for saving the analytical data related to the sentiment computing into new feature tables of the database.

***saveCorrelation()*** This function is used for saving the analytical data related to the correlation computing into new feature tables of the database.

***saveFrequency()*** This function is used for saving the analytical data related to the keyword frequencies into new feature tables of the database.

***saveDemography()*** This function is used for saving the analytical data related to the demography information into new feature tables of the database.

***saveLocation()*** This function is used for saving the analytical data related to different locations into new feature tables of the database.

***saveTime()*** This function is used for saving the analytical data related to different time of the day into new feature tables of the database.

***saveType()*** This function is used for saving the analytical data related to the exercise types into new feature tables of the database.

## 2.2.2. TweetHeat

The “TweetHeat” class contains mostly count functions. They are used to count the number of tweets in different categories. This class also contains several calculate functions to sort the count result to make the leaderboard, and also some show functions to display all the responding charts onto the user interface.

*Count functions:*

***countAreaOverall()*** This function is used for counting the number of tweets in different areas.

***countAreaTime()*** This function is used for counting the number of occurrences of different areas in different time periods.

***countAreaType()*** This function is used for counting the number of occurrences of different exercise types in different areas.

***countTimeOverall()*** This function is used for counting the number of occurrences of different time periods.

***countTimeType()*** This function is used for counting the number of occurrences of different time periods corresponding to different exercise types.

***countTypeOverall()*** This function is used for counting the number of occurrences of different exercise types.

*Show functions:*

***showRecentTweets()*** This function is used for showing on map the most recently posted tweet and its user.

***showTypeOverall()*** This function is used for displaying the analytical chart showing the number of tweets concerning different exercise types.

***showLeaderboardArea()*** This function is used for displaying the leader board ranked by the number of tweets concerning different areas.

***showLeaderboardType()*** This function is used for displaying the leader board ranked by the number of tweets concerning different exercise types.

***showLeaderboardOverall()*** This function is used for displaying the leader board ranked by the number of tweets concerning exercise and health.

***showAreaOverall()*** This function is used for displaying the analytical chart showing the number of tweets in different states.

***showAreaTime()*** This function is used for displaying the analytical chart showing the number of tweets in different states in different time periods.

***showTimeOverall()*** This function is used for displaying the analytical chart showing the number of tweets in different time periods.

*Calculation functions:*

***calculateLeaderboardArea()*** This function is used for sorting the count result of different areas and storing them in decreasing order.

***calculateLeaderboardType()*** This function is used for sorting the count result of different exercise types and storing them in decreasing order.

***calculateLeaderboardOverall()*** This function is used for sorting the count result of all tweets concerning exercise and health tweeted by each user and storing them in decreasing order.

### 2.2.3. FrequencyAnalysis

The “FrequencyAnalysis” class contains several functions. They are used to calculate the different frequencies corresponding to different states and types.

There are also some functions that used for calculating the mean, max and standard deviation of such frequencies. In addition, there are some show functions to display all the responding charts onto the user interface.

*Frequency Calculation functions:*

***frequencyType()*** This function is used for calculating the frequency of exercise in different exercise types according to user’s tweets.

***frequencyTypeMeanMaxStd()*** This function is used for counting the mean, maximum and standard deviation of those frequency calculated in ***frequencyType()***.

***frequencyState()*** This function is used for calculating the frequency of exercise in different states according to user’s tweets.

***frequencyStateMeanMaxStd()*** This function is used for counting the mean, maximum and standard deviation of those frequency calculated in ***frequencyState()***.

*Show functions:*

***showFrequencyType()*** This function is used for displaying the analytical chart that would show the frequency values of exercise corresponding to different exercise types.

***showFrequencyState()*** This function is used for displaying the analytical chart that would show the frequency values of exercise corresponding to different states.

## 2.2.4. DurationAnalysis

The “DurationAnalysis” class contains several functions that are used to calculate the duration of exercise corresponding to different states and types. There are also some methods that could calculate the mean, max and standard deviation of such durations. In addition, there are some show functions to display all the responding charts onto the user interface.

*Duration calculation functions:*

***durationState()*** This function is used for calculating the Duration of exercise in different states according to user’s tweets.

***durationStateMeanMaxStd()*** This function is used for counting the mean, maximum and standard deviation of those duration calculated in ***durationState()***.

***durationType()*** This function is used for calculating the duration of exercise in different exercise types according to user’s tweets.

***durationTypeMeanMaxStd()*** This function is used for counting the mean, maximum and standard deviation of those duration calculated in ***durationType()***.

*Show functions:*

***showDurationType()*** This function is used for displaying the analytical chart that would show the duration values of exercise corresponding to different exercise types.

***showDurationState()*** This function is used for displaying the analytical chart that would show the duration values of exercise corresponding to different states.

## 2.2.5. SentimentAnalysis

The “SentimentAnalysis” class contains several functions that are used to calculate the sentiment value of the tweet text corresponding to different states and types. In addition, there are some show functions to display all the related charts on the user interface.

*Mood calculation functions:*

***moodType()*** This function is used for calculating the mood value of tweets for different exercise types according to user’s tweet text.

***moodState()*** This function is used for calculating the mood value of tweets in different states according to user’s tweet text.

*Show functions:*

***showMoodState()*** This function is used for displaying the state map that would show the happiness degree corresponding to different states.

***showMoodType()*** This function is used for displaying the analytical chart that would show the mood values corresponding to different exercise types.

### 2.2.6. CorrelationCalculate

The “CorrelationCalculate” class contains several functions that count the number of the tweets that refer to food corresponding to different states and exercise types. There are also several functions that would calculate the linear regression value among the tweets count of health, exercise and food. Besides, there are show functions to display all the relationship in charts on the user interface.

*Food calculation functions:*

**countFoodType()** This function is used for counting the number of tweets related to different kinds of food according to multiple types of exercise.

**countFoodState()** This function is used for counting the number of tweets related to different kinds of food according to different states.

*Correlation functions:*

**correlationHealthFood ()** This function is used for calculating the linear regression value between the counts of tweets related to health and food.

**correlationExerciseFood ()** This function is used for calculating the linear regression value between the counts of tweets related to exercise and food.

**correlationExerciseHealth()** This function is used for calculating the linear regression value between the counts of tweets related to exercise and health.

*Show functions:*

**showCorrelationExerciseHealth()** This function is used for showing two line charts that would indicate the count number of tweets related to exercise and health separately.

**showCorrelationExerciseFood()** This function is used for showing two line charts that would indicate the count number of tweets related to exercise and food separately.

**showCorrelationFoodHealth()** This function is used for showing two line charts that would indicate the count number of tweets related to food and health separately.

### 2.2.7. PersonalAnalysis

The “PersonalAnalysis” class is consisted of the following four variables that store the user’s personal information.

**username** String variable storing the user’s name.

**state** String variable storing the user’s current location.

**age** Integer variable storing the user’s age.

**exerciseType** String variable storing the user’s favorite exercise type.

This class include functions that are used to give out the personal suggestions based

on the information the users provide.

**setUserProfile** (screenName : String, userAge : int, userState : String, exerciseType : String) This function is used for setting the basic information that is needed for the analysis.

**userAnalysis()** This function is used for making analysis on the proper healthy food and exercise duration time based on the information the users provide.

**showSuggestion()** This function is used for displaying the proper personal suggestions based on the analysis.

**showTweetHistory()** This function is used for displaying the past tweets the user posted and showing the time when the tweet has been created.

## 2.2.8. DemographyAnalysis

The “DemographyAnalysis” class has some speculations functions that use third party’s API to make speculations on the demographical information of the Twitter users based on their tweets, some update functions that update the speculation results into the existing table, and some show functions that display the analytical results.

**connectAPI(url : String)** This function is used for connecting our database to the third party’s API. The parameter “url” is the website of this API.

**guessGender(screenName : String, username : String, description : String)** This function is used for making a guess on the gender of a certain user based on his/her screen name, user name and the user description.

**guessAge(screenName : String, username : String, description : String)** This function is used for making a guess on the age of a certain user based on his/her screen name, user name and the user description.

**guessType(screenName : String, username : String, description : String)** This function is used for making a guess on the user type, organization or person, of a certain user based on his/her screen name, user name and the user description.

**updateGender(gender : String)** This function is used for updating the guess result of the user gender in the existing users table.

**updateAge(gender : String)** This function is used for updating the guess result of the user age in the existing users table.

**updateType(gender : String)** This function is used for updating the guess result of the user type in the existing users table.

**showAgeDistribution()** This function is used for displaying the age distribution among all the users who have posted exercise-related tweets.

**showGenderDistribution()** This function is used for displaying the gender distribution among all the users who have posted exercise-related tweets.

**showAreaAge()** This function is used for displaying the age distribution among users who have posted exercise-related tweets in different states.

**showAreaGender()** This function is used for displaying the gender distribution



among users who have posted exercise-related tweets in different states.

**showTypeAge()** This function is used for displaying the distribution of different exercise types in different age groups.

**showTypeGender()** This function is used for displaying the distribution of different exercise types in different gender groups.

## 2.2.9. Controller

The “Controller” class represents as a vital part in the system-to-be. All the executions of its functions are linked to some other classes. The user utilize these functions and receive the analytical results corresponding to his/her choice.

**analyzePersonal()** This function is used for making analysis based on personal information we obtained from the user.

**analyzePublic()** This function is used for making analysis based on public information we obtained from the Twitter.

**updateDB()** This function is used for making updates on the existing database if some changes need to be made.

**isValidData()** This function is used for verifying whether a certain piece of data is valid. If valid, return true, else, return false.

**showRequest()** This function is used for displaying corresponding analytical charts or tables based on the request from the user.

## 2.2.10. Communicator

The “Communicator” class is linked to all the other classes that command to get data from the database. It has two variables storing the type of the platform and which feature table we need to use.

**deviceType** Integer variable storing the type of the platform the JSON data need to be sent to. We choose 0 representing the web platform, and 1 representing the IOS platform.

**dataSelect** Integer variable storing which feature tables we need to use for analysis. Different numbers represent different feature tables.

This class contains a `jsonSender()` function which is essential to send the database data to other platforms in JSON formation. Other functions are used for connecting the database.

**setDataSelect(data : int)** This function is used for setting which feature table we need to use for our analysis.

**setDeviceType(device : int)** This function is used for setting the type of platform we use to display the analytical results.

**jsonSender()** This function is used for packaging the data we need in certain tables and sending them to the platform we choose.

**dbConnection()** This function is used for making connection with the database.

**dbConfig()** This function is used for initializing the database.

**isConnected()** This function is used for testing the connection to the database. If connected, return true. Otherwise, return false.

## 2.3. Traceability Matrix

	controller	communicator	Database	Personal Analysis	Demography analysis	Duration Analysis	Frequency analysis	Tweet heat count	Correlation calculate	Sentiment analysis
Domain concepts										
Controller	X									
Profile Storage				X	X					
Interface Services				X	X	X	X	X	X	X
Interface Pages				X	X	X	X	X	X	X
Display Options				X	X	X	X	X	X	X
Treated information			X							
DB connector		X	X							
Data Analyst		X	X	X	X	X	X	X	X	X

Third part connector			X	X	X	X	X	X	X	X
Domain concepts										
Controller	X									
Profile Storage				X	X					
Interface Services				X	X	X	X	X	X	X
Interface Pages				X	X	X	X	X	X	X
Display Options				X	X	X	X	X	X	X
Treated information			X							
DB connector		X	X							
Data Analyst		X	X	X	X	X	X	X	X	X
Third part connector			X	X	X	X	X	X	X	X

Table 2-1 Traceability matrix

Apparently, there are many classes evolved from the same Domain Concept except the Controller, which is basically achieved by the single class that also named Controller. Concept Profile Storage include user's personal data (like demography information). These information could derived from Classes Personal Analysis and Demography Analysis since both of the classes contain related methods to infer and output such message.

Interface Services, Interface Pages and Display Options construct the user interface part of the system, hence the last seven classes derived from them all.

Because the treated information is basically stored in the database of our system, there are not many classes involved with this concept. Only the class Database is needed to get such information.

Both of the two classes ---Communicator and Database ---take the responsibility of Concept DB Connector. Due to some specific reasons, we separate them by different functions. Database takes charges of the data management within the data base. Communicator transfers the data between other classes and the database.

Data Analyst is separated into seven classes that related to specific types' analysis and two classes that enable them to do communication with database.

Because the treated information is basically stored in the database of our system, there are not many classes involved with this concept. Only the class Database is needed to get such information.

Both of the two classes ---Communicator and Database ---take the responsibility of Concept DB Connector. Due to some specific reasons, we separate them by different functions. Database takes charges of the data management within the database. Communicator transfers the data between other classes and the database.

Data Analyst is separated into seven classes that related to specific types' analysis and two classes that enable them to do communication with database.

## **3. System Architecture and System Design**

### **3.1 Architectural Style**

Software architecture refers to the high level structures of a software system, the discipline of creating such structures, and the documentation of these structures. It is the set of structures needed to reason about the software system. Let's begin description from how our system works. Our system loads and stores data from Twitter to data base, analyzes the data for different business features using python implements, finally presents the results to users on websites and apps. The users should be able to access our system to find what they want and log in their own account, but only administer can conduct the data processing job. At first phases, our system uses API get data from Twitter, store them on local database, and use python language packages to analyze these data. Then, allow user to login in our own website and upload their exercise or health data, this website can also be shared to Twitter, which can attract more users. So there will be two major architectures, one is the database, including data from API Twitter and analysis implement, the other is presenting applications such as website and cell-phone apps, which can show our work and provide interface to users. There will be two parts involved, users login in, do exercises, update data and tweet, while administer runs database, website and analyzing data form API.

### **3.2 Identifying Subsystems**

As discussed above, we can define our subsystems as four parts to achieve basic requirement, we can add more features later to this structure.

The collection subsystem is responsible for scrawling data from Twitter to our database. The data will be in .sql format and stored directly in a table in our database. The store subsystem will parse (extract) the properties such as screen name, date, tweet from API and insert the properties into other tables. The analytics subsystem will compute results from the data stored in the tables mentioned above in order to fulfill the different business goals, and insert the results in feature tables. The presentation subsystem simply structures the user interface using a website which is now already been built up. We still use .sql format for transmission, because the html can be accessed with sql through php tools. The presentation will not get all the feature data at the initialization stage since data is large and loading is slow. The presentation will only select what is wanted to be shown.

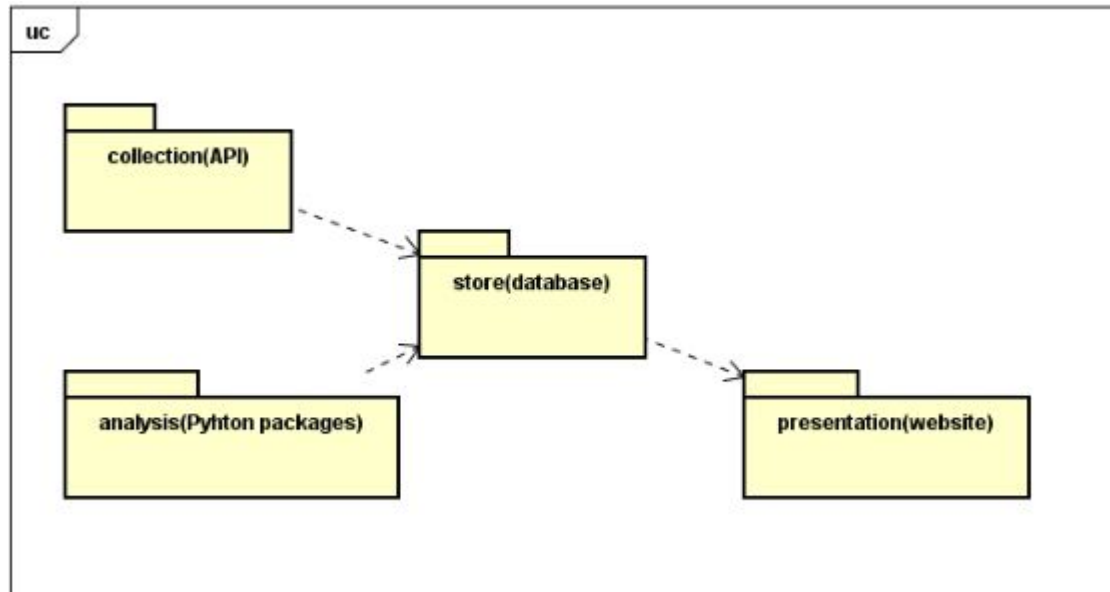


Figure 3-1

### 3.3 Mapping System to Hardware



Figure 3-2

In our system, database and server should be on one computer, while website can be accessed from any computer. Server means the data processing and analyzing tools and execution.

What is the relationship between the subsystems and the hardware? The collection, store, analytics and transmission subsystems will be deployed on the server. The computer for database only stores the data, and will be accessed by the server. If it is a web application, the presentation and plot subsystems will be also deployed on the computer for server. The user interface structure and the data for plot will be downloaded by the browser. While accessing the website, the computer in which database and server should be open and connected to internet. Further technical discussions about host accessing websites will appear later.

## 3.4. Persistent Data Storage

### 3.4.1 Database improvement

Since our project is based on the previous project, so we decided to modify and improve their database rather than creating a brand new database. So far, as we designed, we want to separate every different sport and classify them into two different categories. So in figure 3-4-1, we create tables according to the design.

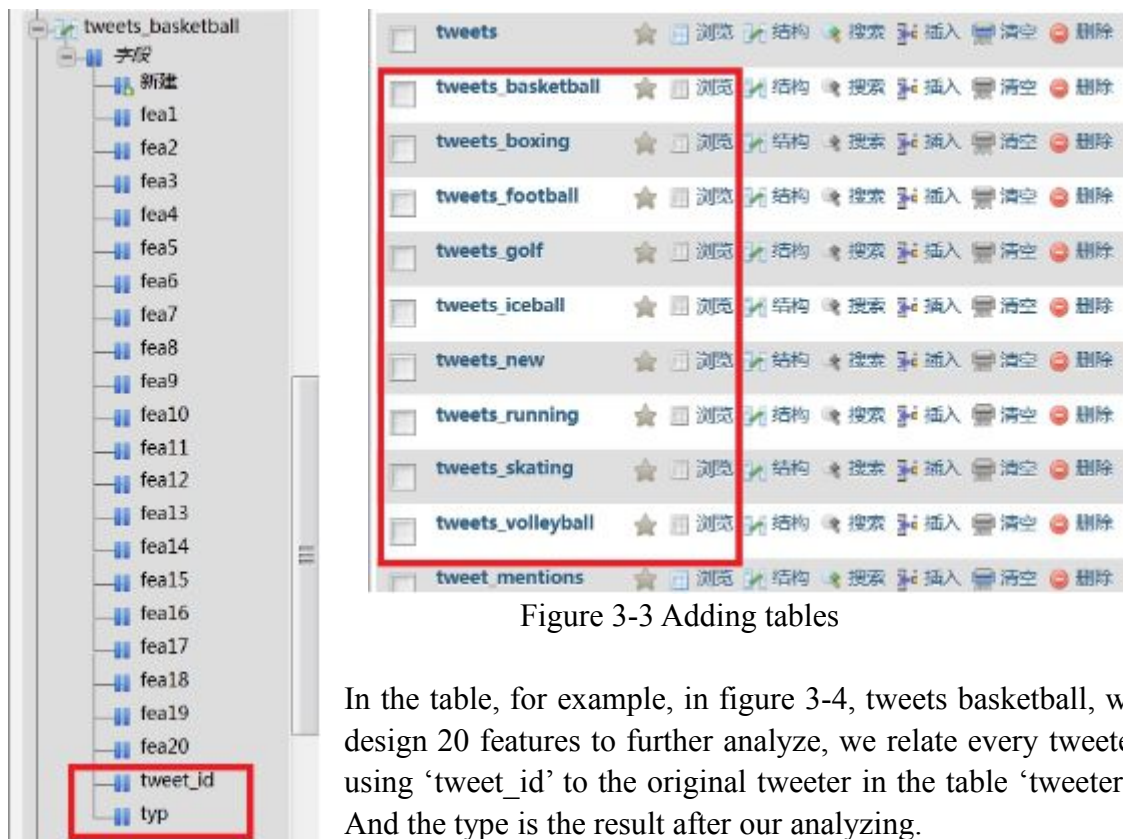


Figure 3-3 Adding tables

In the table, for example, in figure 3-4, tweets basketball, we design 20 features to further analyze, we relate every tweeter using 'tweet\_id' to the original tweeter in the table 'tweeter'. And the type is the result after our analyzing.

Figure 3-4 Table Value

### 3.4.2 Registered users

Also, we decided to add a table to record the registered users, although we didn't give the very detail information about the difference between signed users and unsigned users, but we all think we it is necessary, so we design the database structure in advanced.

In the figure 3-5, we create a table named "registered\_users" to record every info for user data.

id	name	password	age	Location	email	creat_time
1	tianzhe	cf79ac6addba60ad018347359bd144d2	18	NJ	tianzhewang@126.com	2015-10-28 20:46:10

Figure 3-5 Registered users.

### 3.5 Network Protocol

Since our aim of project is to give a better analyze of data which comes from the GET method in HTTP protocol. So we distribute the responsibilities for every group member. So we defined two modules for our primary system. Server for display 1), server for functionality 2). In the display server, we create a website for users (registered or not) to browse, and finding information. In the functionality server, our analyze tool will be implemented on that. Of course these two servers should be in the LAN (local area network), while we think there is no need for both our them to connect to the internet for the safety concern. The separate server is our ideal design (database + operation), but for our real implementation and time limitation, we probably merge them into one.

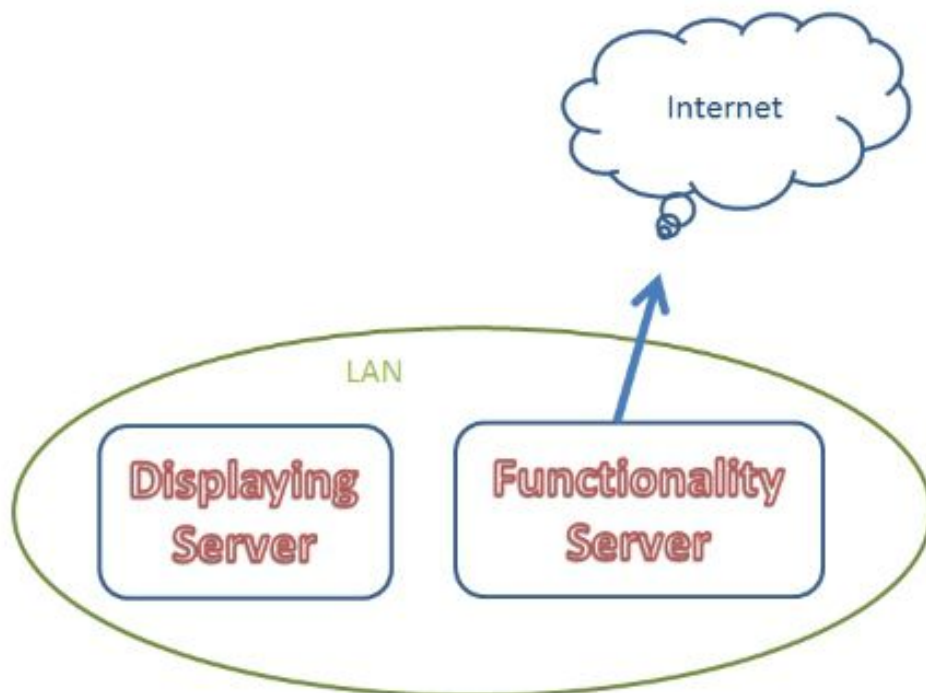


Figure 3-6 Networks



### 3.6. Global Control Flow & Hardware Requirements

For the data management and analyze, window service is our first choice. Since it can start with every time server power on, and can hide from users, easy to control. The service include: data mining, data analyze, database action.

As for hardware, we set the same aim as last project (project 2, 2014): 5 GB. But, we respectively disagree the 5G parameter set by them can be achieved. Since they obtain every tweeter in the database without any clearing schedule, we think the database will be large than our expectation and the system will be slow. So after our considering, we decide not to obtain every tweeter, only record the result we wanted, to simply the database like figure 3-7.

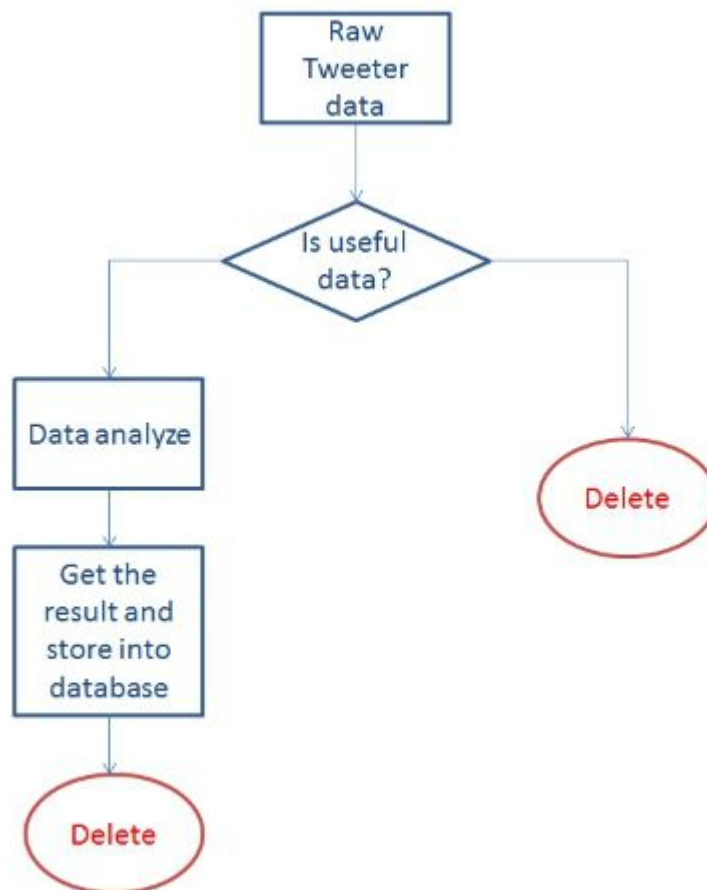


Figure 3-7 Database Simplification

## 4. Algorithms and Data Structures

Since we try our best to level up the accuracy of the tweeter content, we implement KNN algorithm to filter the raw tweeter data and NLTK testBlob to analyze the emotional trend, also we use SVM to classify each tweeter into two categories.

In pattern recognition, the k-Nearest Neighbors algorithm (KNN) is a non-parametric method used for classification and regression. In our project, we use KNN algorithm for tweets classification that is to determine every tweet is related to exercise and sports or not. We use training set to train our KNN algorithm. Then we run the PHP program to fetch tweets from our local database, filter them and then insert relative tweets into a new table in our local database.

We use the TextBlob to implement the emotional analysis. TextBlob is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more. TextBlob stands on the giant shoulders of NLTK and pattern, and plays nicely with both.

In our project, we use TextBlob to do the emotional analysis on every exercise and sports related tweets and classify them into either a positive tweet or a negative one. By doing so, we are able to see how many of people talking specific sports are having negative feelings.

The code below is a simple Python example program using TextBlob API to get emotional analysis on tweets.

We use SVM (support vector machine) to categorize every tweeter into two categories. SVM is an advanced technology using training data to build model to predict the incoming data (more information about SVM, you can refer other books about it ). In this early stage of our project, we just test the sexuality according to the height and weight (since they are all numeric character).

There are many SVM tools online. We choose 'libSVM' created by Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, Department of Computer Science, National Taiwan University. We choose it since its easy-using and nice guide manual. There are many version of it, we choose c++ version since it is more familiar with knowledge scope.

### 4.1 KNN

KNN is a very important algorithm in our project. We use KNN classification to filter

out all the nonrelative tweets we do not need.

### 4.1.1 Basic Concept of KNN

In k-NN classification, the output is a class membership. An object is classified by a majority of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small).

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. The training examples are vectors in a 3 dimensional feature space, each with a class label. In this project, the labels are only two: exercise relevant and exercise irrelevant. Relevant tweets shall be stored in the database while the irrelevant ones are discarded.

The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. This is done by manually reading the tweets and labeling them with “True” or “False” class.

In the classification phase, k is a pre-defined constant. The k for text identification in this project should be chosen as 3. As mentioned in reference from TAMU [[http://courses.cs.tamu.edu/rgutier/cs790\\_w02/l8.pdf](http://courses.cs.tamu.edu/rgutier/cs790_w02/l8.pdf)],  $k = n/2$ , for which n means the number of features.

In this project, there are 3 features: sports type, quantity and adjective list. The reason to choose these three types of keywords stated as:

1. Sports type is necessary since the tweet should be about health and exercise, any keywords that have direct relevance to sports and exercise should be considered first.
2. When examining the tweets on Twitter that are talking about health, it is more likely to have not only sports type keyword alone but also description about how much work or exercise the user has made. So the keyword of quantity will help classifying the tweets.
3. Adjective list is a supplement to give tweets that contain human’s emotion and feeling’s words.

An unlabeled tweet is classified by assigning the label that is most frequent among the k training samples nearest to that query point. Distance measurement shall be calculated as Euclidean distance:

$$D_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}$$

The distance means how close a new tweet is related to the training sets, in this project, the distance is the measurement on probability similarity.

In the project, a 3D characteristic vector is extracted from every tweet using keywords finding. X, Y and Z stand for the keywords' scores.

The score is calculated based on the probability of a tweet relevant to exercise by containing this keyword alone. To demonstrate this, take one tweet "I run 5 miles, tired." for example. If in 100 tweets that contain keyword "run" and there are actually 10 tweets talking about exercise, the probability of "run" will be 0.1. And we re-arrange the probability into a range of 0~40 score, thus the score of "run" will be  $40 \times 0.1 = 4$ . Apply these procedures to other keywords, saying "miles" is score 10 and "tired" is score 15.

In the system, the probability table of keywords and training sets must be set preliminarily. This procedure is done manually by reading the tweets. Then the following steps shall do the data sifting procedure:

1. When the system crawls a new tweet, search for the keyword in it first and extract the three types of keywords.
2. Score the keywords using established probability table.
3. Calculate the Euclidean distance of the newly crawled tweet's characteristic vector to the training sets using X Y Z coordinates.
4. Sort the distance and choose the top k points as the k-NN.
5. Find the majority in k-NN and identify the newly crawled tweet as the same label as the majority.

The classification can be explained through the chart above. There are some pre-set training sets in the coordinates with certain location. And the new tweet, for example, is "I run 5 miles, tired". It is scored by the three filters and locates in the coordinates. We have not decided whether the new tweet is true (health related) or false (health unrelated).

The next step is calculating the distance between the new tweet's coordinates to those of training sets. The training sets are also located in the coordinates with the same standard for keywords scoring, while their classifications of true and false are labeled manually. Using Euclidean distance for distance calculation.

Then sort the training sets by their distance to the new tweet, select top 3 training sets with minimum distance. And check their classification, if 2 or 3 of them are true, and then classify the new tweet as true, vice versa.

### **4.1.2 KNN Implementation**

Our implementation is based on the former group's implementation.

We draw the training sets our project in this three dimensional chart. In the chart above, the blue points refer to training sets that are health related (true), and black points refer to health unrelated training sets (false). A new tweet is labeled as green,

and the implementation is presented below.

#### **4.1.2.1 Setting Training Set**

One improvement we made is to enlarge the training set. The training set provided by the former group has a relatively small amount of training examples leading to low accuracy of the KNN filter.

We choose three types of keywords: sports type, quantity and adjective list as our 3D coordinates.

The axis for sports type, quantity and adjective list are scale from 0~40. This is calculated by the following step:

1. Calculate the probability of tweets that containing the keywords of sports type, quantity and adjective list those are health related separately. The result ranges from 0~1.0 (0~100%).
2. Times 40 to all the probability result and then the result ranges from 0~40. Because the minimum difference of probability of different keywords is 0.025 in our project, so to distinguish these two values of probability we times 40 to make the difference equal or greater than 1.
3. Round the result to the nearest integer to meet codes' int calculation. And the integer shall be the score of the keyword.

After setting up the axis, training sets are also located by calculating the keywords' coordinates (or score) in the training sets' text. X, Y and Z are acknowledged as well as location.

This method is to distinguish keywords of different health related tweets. If a new tweet saying "I do 40 squads" (new in the chart), it should be compared with a tweet that contains the same or close weighted keywords, like "I do 50 push ups" (Point 2), and "I have a great day doing push ups"(Point 3) instead of comparing to tweet that contains much different weighted keywords as "I run 5 miles".

After this procedure, the new tweets' score coordinates are scattered in the coordinates and ready to be compared with similar weighted tweets, which are used as training sets.

#### **4.1.2.2 Classify new tweets**

Then we calculate the Euclidean distance of new tweet and training sets. The new tweet's coordinates are calculated in the same way as the training sets.

Then sort the new tweet's neighbor training sets by the distances, choose the top k training sets. The k's value is 3 in our project.

The reason is that in the classification phase,  $k$  is a pre-defined constant. The  $k$  for text identification in this project should be chosen as  $k = n/2$ , for  $n$  means the number of features. The reference is from TAMU.

[[http://courses.cs.tamu.edu/rgutier/cs790\\_w02/l8.pdf](http://courses.cs.tamu.edu/rgutier/cs790_w02/l8.pdf)]

In this project, there are 3 features: sports type, quantity and adjective list. Also it is better to set  $k$  as an odd number to avoid a tie of two different classifications. Thus we choose  $k = 3$  as our parameter. The 3 training sets are circled in a red circle in the chart above.

After choosing the top 3 nearest training sets, we examine the classification of the training sets, if 2 or 3 of them are true, and then classify the new tweet as true, vice versa.

In the chart, the Point 1 is “She does her sit-ups to get out of bed”. Though “sit-ups” is weighted near the “push ups”, the whole meaning of the tweet is not about health or exercise. And after sorting the top 3 nearest training sets, the new tweet has two true nearest training sets and one false. Thus the new tweet is classified as true.

We do this sift procedure to each tweet we extracted from Twitter and only the true tweets are used for all other analysis while the false ones are discarded. We have collected 200 training sets and apply the algorithm to over 100000 tweets in our database.

## 4.2 NLTK TextBlob

TextBlob is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more. TextBlob stands on the giant shoulders of NLTK and pattern, and plays nicely with both.

In our project, we implemented the TextBlob API to get sentiment analysis on each tweets. The TextBlob provides multiple features for developers to use. So far we are using Naive Bayes Classification to classify each tweets into two section: positive tweets and negative tweets.

As long as we classified all sports and exercise related tweets, we could have a full sentiment analysis towards each sport or exercise activity. Government or other companies will be interested in these sentiment analysis results because they can figure out what problems or improvements exist so that they can take advantage of it.

## 4.3 SVM

In the SVM algorithm, we treat every tweet as a vector. Every word in the sentence (tweeter) can have a weight. We use the weight calculation formula (figure 4.1) to calculate each weight<sup>1</sup>

$$\phi_i(x) = \frac{tf_i \log(idf_i)}{\kappa}$$

Figure 4-1

Tf<sub>i</sub> is the number how many times certain words happened in file x (we take x as each tweeter content), idf<sub>i</sub> is the ratio of all the files (tweeters) to the number how many files contain certain words.

We don't try to eliminate the stop words or other function word like 'the'. The reason is, suppose two tweeter is all "the the the the the" which has no meaning. They will make these two testing point stay at the 'dividing line' like figure 4.2.

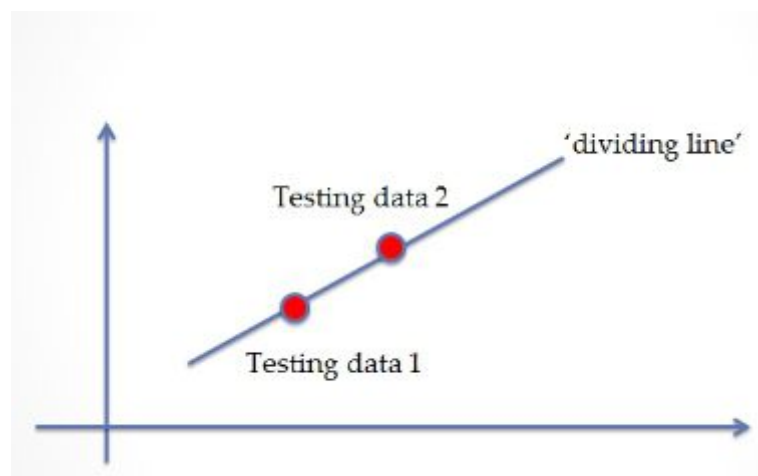


Figure 4-2

But in the real world, it will not happen. There will be some key word which has high weight to indicate which category the tweet should be in. just like show in the figure 4.3.

---

<sup>1</sup> An Introduction to Support Vector Machine and Other Kernel-based Learning' Nello Cristianini and John Shawe-Taylor

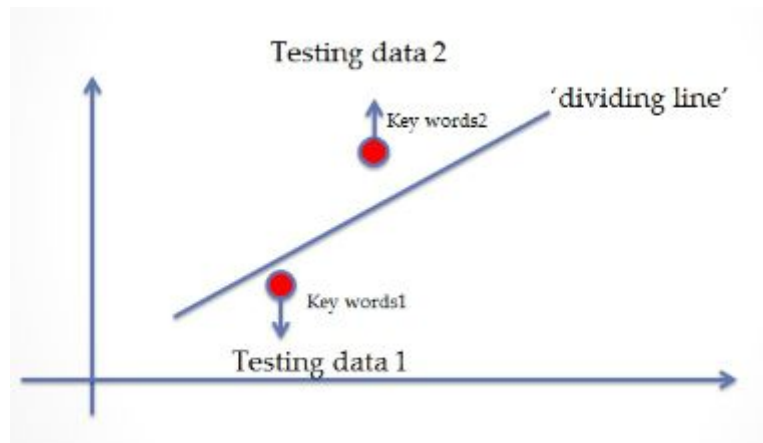


Figure 4-3

Since the algorithm behind SVM is very complicated, so we don't want to illustrate them here (it may need a book to tell them clearly). For more information and knowledge about SVM, please refer to 'An Introduction to Support Vector Machine and Other Kernel-based Learning' Nello Cristianini and John Shawe-Taylor, or other material.

## 4.4 Data Structure

Our database is based on the previous database, but we do some improvement. We create tables for each specific sport, and tables for storing the feature weight.



Figure 4-4

As shown in figure 4.4, we have the table for raw data from Tweets, store into tweet tables.



tweet_id	tweet_text	entities
653926589974102017	RT @_dajjal: Coming up as a child I walked some lo...	Tzo4OUzdGRD0Gfzcy6NDp7czo4OUoY
653926844421554179	my car is home with 0 miles to empty 8amp; I walke...	Izo4OUzdGRD0Gfzcy6NDp7czo4OUoY
653927176287451376	...OFTHEMATCH...BEN...PAUL...RAN...123...MILES...IN...1...HOU...	Tzo4OUzdGRD0Gfzcy6NDp7czo4OUoY
653927205693747200	RT @Cross_Prob: The Men at the Chicago marathon we...	Tzo4OUzdGRD0Gfzcy6NDp7czo4OUoY

Figure 4-5

As shown on figure 4.5, after the KNN filter, we filter the useless tweeter and store the valuable ones into separate tables. Then we use NTLK to calculate the weight for each word using the formula in the section 4.3. the whole data processing can be seen like in figure 4.6.

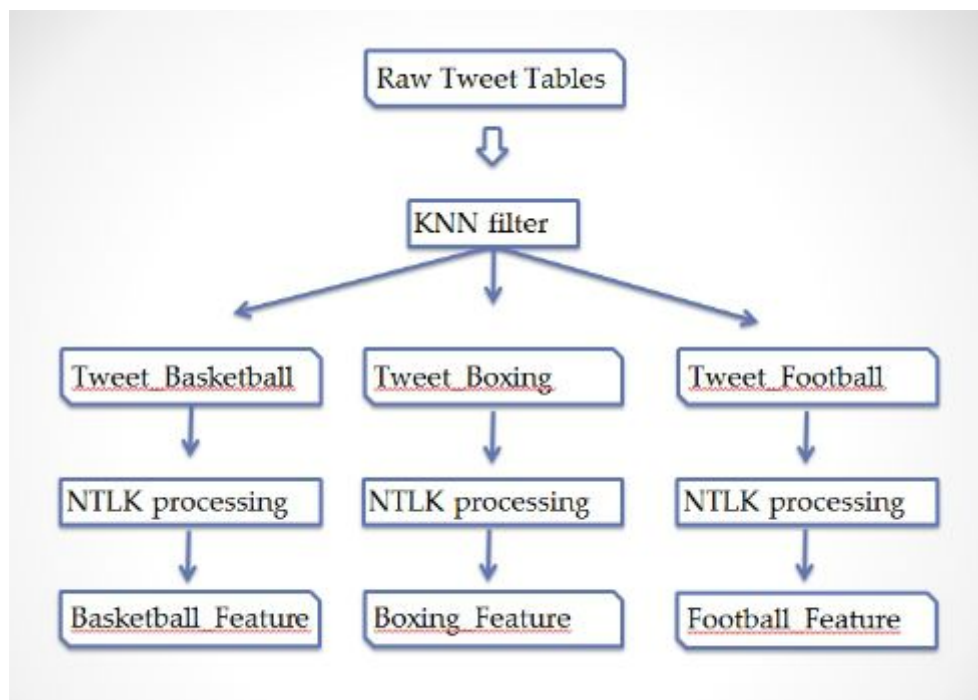


Figure 4-6

After we get the result about which category each tweeter belongs to, we can calculate the sum for each category for further data analyze and display.

## 5. User Interface Design and Implementation

For now most of our job is displayed on the basis of website. In this part we will explain how to use our website, step by step. Our goal is to help user to use this health monitor system. Since this is the very first version, so we can just provide some really basic functions, but more and more new features will be added, as well as more accurate data will be implemented.

### 5.1 Website

#### 5.1.1 Homepage

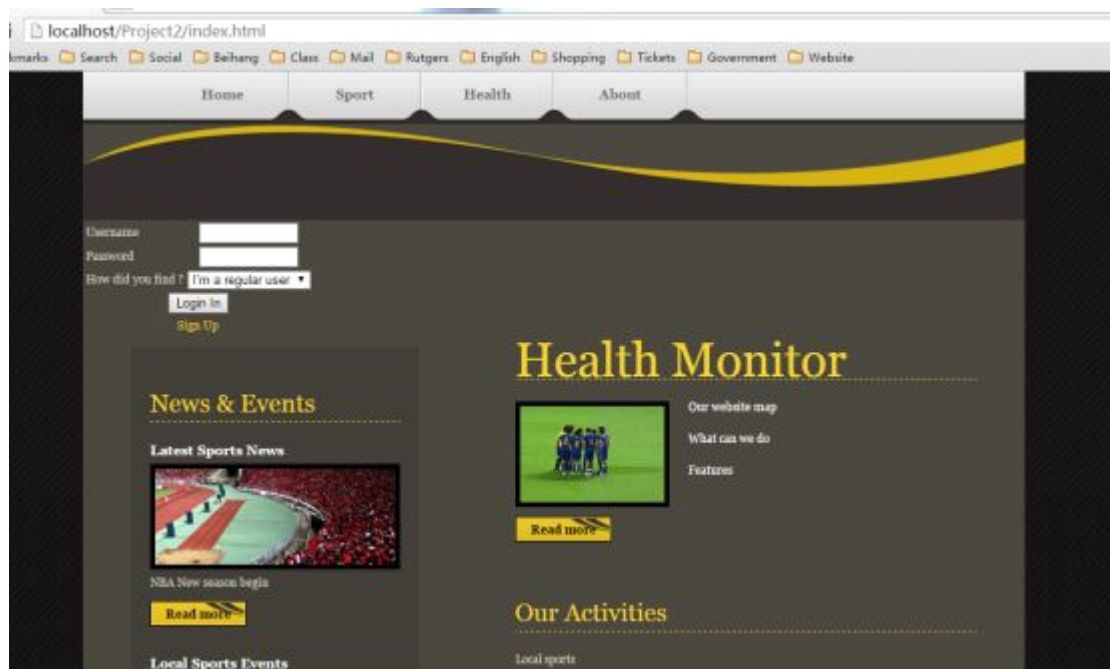


Figure 5-1

In local host, which is our projects root route, is our index html page. On the top, there are three sub-pages directing to three sub sections, which are sports, health and about. We will come to these sub-pages later. On the left side, there are log in and sign up bar to choose either choice. In general, we have two ways to extract data from the users, one is to get data from twitter with API and analysis with algorithms, this is the main way to conduct and get re results, the other source of data is to let user to sign up their own account and add their data to those data from twitter for analysis. So we can see that we do need a sign up page.

## 5.1.2 Login and Signup page

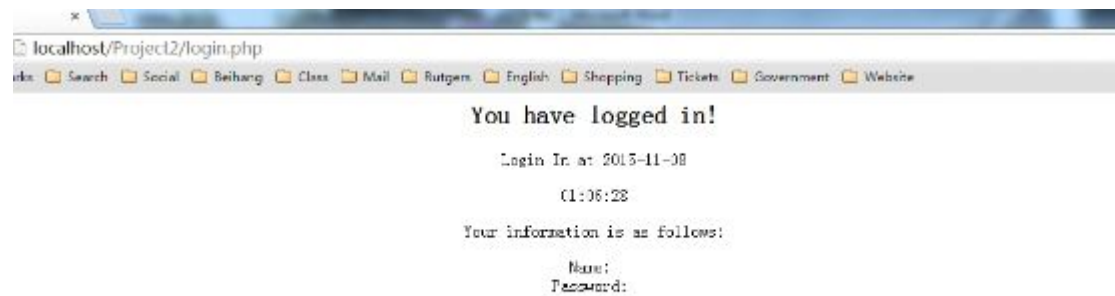


Figure 5-2

From this screenshot we can see that it is using php to achieve submitting the user's data, and there are indexes such as time or date. In next stage, we will enrich our database for a new and more efficient table to store the user's data such as username or password. To have this table, we first have to let user to input the data, which is sign up, after sign up, the user's data are stored in the database, when they log in, what is input are compared with the database, if match, then log in success.

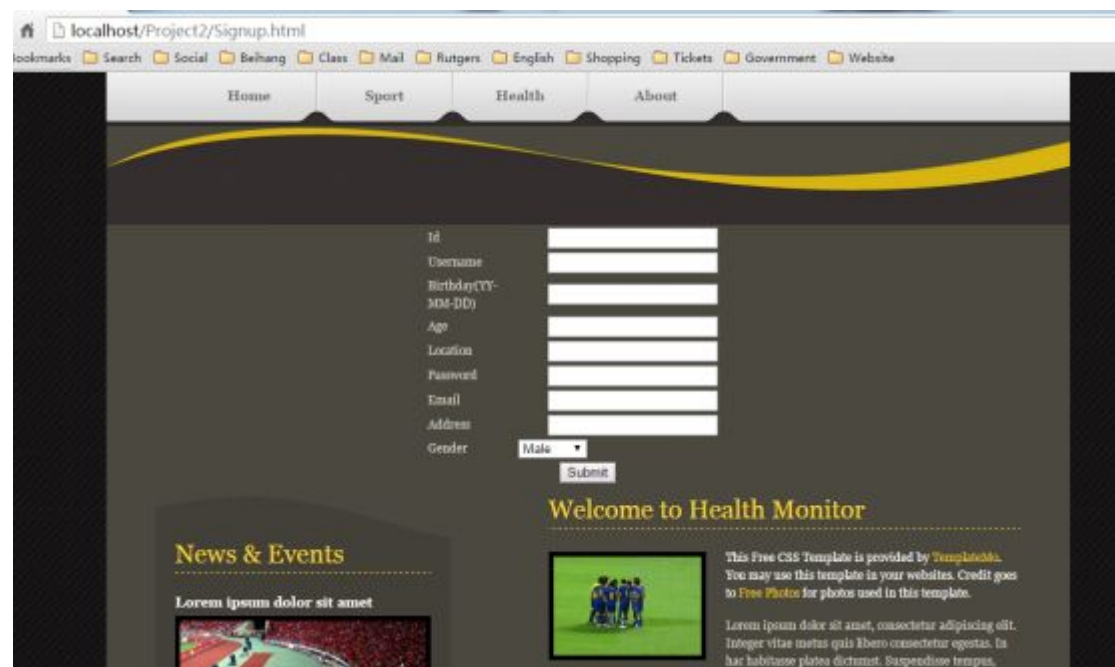


Figure 5-3

This is the sign up page. Users type in their basic information, and initialize a table in the table in database called by the user's id defined in the database.

### 5.1.3 Sports page

The sports page is mainly about the results of data analysis. The data analysis results, is mainly divided into two parts, which to our definition, is sports intension and sports involvement. Intension is people talk or tweet about specific kind of sports, take basketball for example, someone tweets about basketball and said watching it but not playing it, we can use our algorithm to analyze that and use figure to show this result. We can also see from the charts that if some kinds of sports are not “easily physical accessible”, such as swimming, there are not too much people talk about it but there are many people actually swimming, then we can say that people’s involvement of swimming is higher comparatively. These data, coming from the database, is linked from php selection, and then use javascript to display.



Figure 5-4

Above is the chart showing how the result is displayed.

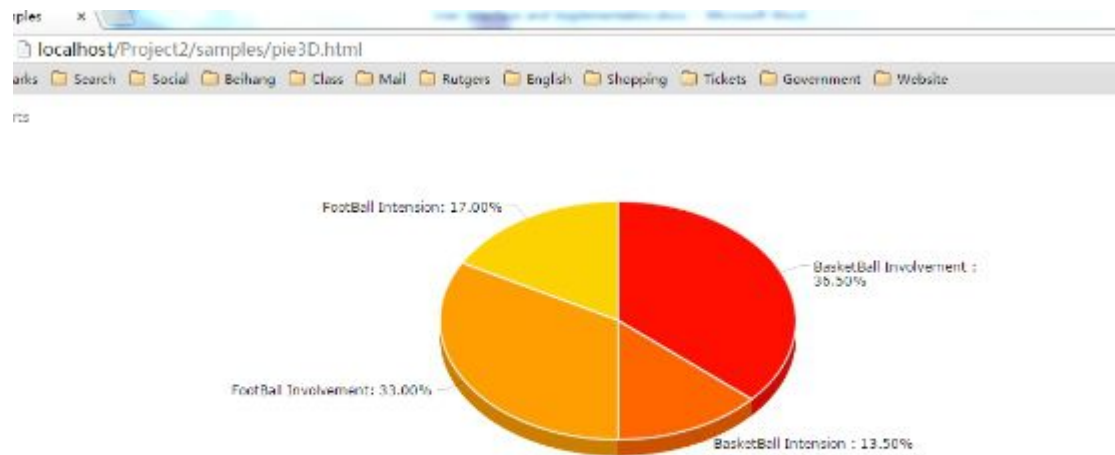


Figure 5-5

Above is the ratio of two kinds of sports' intension and involvement.



Figure 5-6

Our website can also display the results of data during some time, although it is not that accurate, we do select some data from each month and analysis them to display. For the next phase of work, we will figure out more straightforward description to show the results.

## 5.2 Android Application

Now it has a login function and a registration function. After logging in, a picture of Michael Jordan will show up. All the data is stored in local database now, has not connect to our main database yet, that is the part I will be working on.

This is what you will see after opening the application. If you already registered, you will be able to log in.

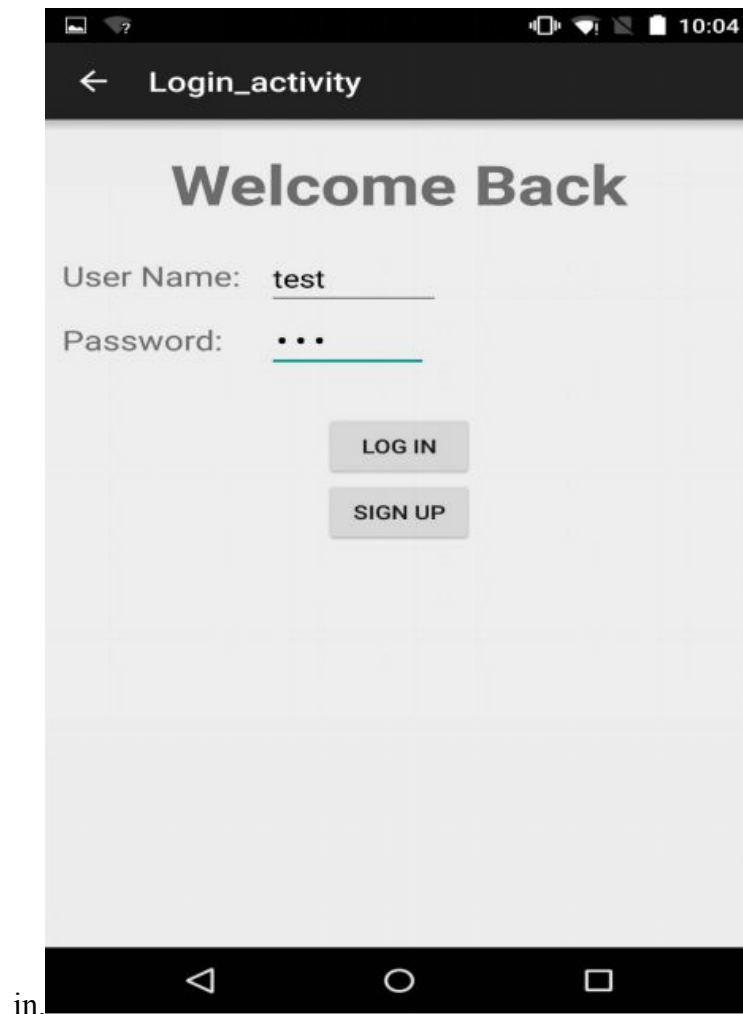


Figure 5-7

And this is what you will see after logging in.

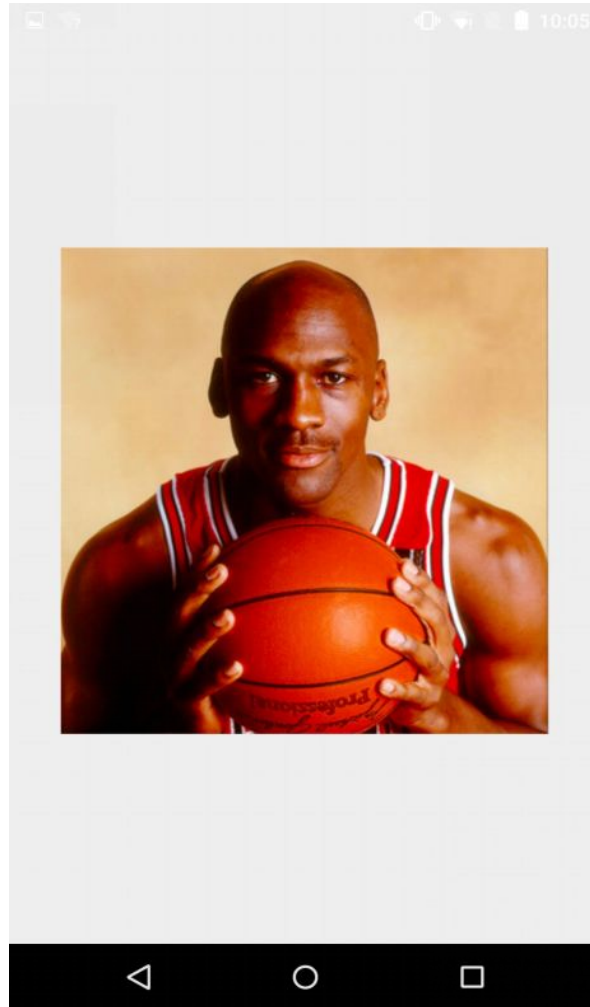
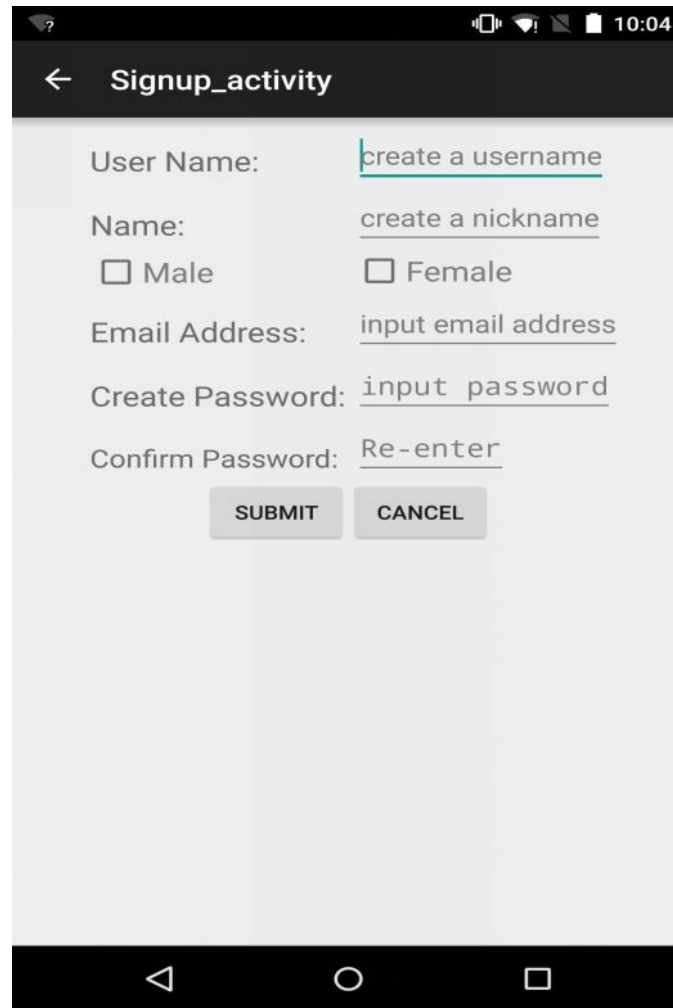


Figure 5-8

If you have not registered yet, just push the sign up button and the sign up page will show up.



← Signup\_activity

User Name: create a username

Name: create a nickname

☐ Male ☐ Female

Email Address: input email address

Create Password: input password

Confirm Password: Re-enter

SUBMIT CANCEL

Figure 5-9

For now, we only accomplished these simple functions because I just got started in learning Android application developing, I think I will be making progress in connecting this app to our database.



## 6. Design of Tests

### 6.1 Overall Description

The part of design of tests is to test the basic function of our system. The test will be divided into two sections: the function unit test and the Integration testing strategy. Because some function units use the same coding methodology, therefore we group them as a class and choose one unit to test. The test table is shown in table 6-1.

Function unit group	Unit within group	Test unit
Twitter Data Capture	Twitter retrieve	Twitter retrieve
Data base set up	Data base	Twitter retrieve
Tweets data filter	Useful tweets and advertising	related and unrelated tweets
Related data selection	Self-involve and “audience” tweets	self-involve and “audience” tweets
Histogram show	Heat map and Sports intension&involvement	Heat map and Sports intension&involvement
Exercise hit trend	Exercise history	exercise history

Table 6-1

### 6.2 Function unit test

#### 6.2.1 Test Unit: Twitter retrieve

Tweets are the basic data source for our system. They are captured by the streaming twitter API. We use two php profiles to capture data. OauthPhirehose.php and Phirehose.php achieve the function.

Input Requirement	Mysql database is running. Apache webserver is running. New database is created in Mysql database. Account and pin of twitter API is applied. Then, run the php profiles.
-------------------	---

Expected Output	Several tables are created in the database already created. Each table contain the required tweets data such as location, created time and etc.
Succeed/ Fail	Succeed if the right data is stored in corresponding database and format is right. Fail if there is no table created in database or no required data in tables.
Comment	This is to test whether the necessary could be collected successfully

Table 6-2

### 6.2.2 Test Unit: Data base

Input Requirement	JOSN File Extract From Twitter
Expected Output	Tweet Text, User Profile, other Information that could be useful for system requirement
Succeed/ Fail	Succeed if Data Base Set Up With Specific Data Fail if Data Base Set Up With Other Data
Comment	This test is to make sure that our database will contain the exact data extract from twitter

Table 6-3

### 6.2.3 Test Unit: Useful tweets and advertising

After tweets are stored, the system need to sort and filter the data from the database and create two new tables. They are used for further achievement of specific function of system. And they contain the necessary data for other functions.

Input Requirement	Mysql database is running. Apache webserver is running. Tweets are stored in the corresponding database. Then run datafilter php profile.
-------------------	---

Expected Output	New tables which contain useful tweets are created in the database for system. The format and content in each table are right. If there are more than one database for our system, every database would have two more tables.
Succeed/ Fail	Succeed if two new tables are created in corresponding database. Fail if the connection to database is timeout or tables cannot be created for every database.
Comment	This is to test whether the system could filter the data from original database and be preparing for further functions.

Table 6-4

### 6.2.4 Test Unit: Self-involve and “audience” tweets

The data filtered is ready for analyzing and processing in functional units. DataSelect is implemented to select request data from database for different functions. It has to confirm successful connection to database and select out right data. Then it will bring proper information or data to the specific function analyzing units.

Input Requirement	Mysql database is running. Apache webserver is running. Tweets are filtered in the corresponding database. Then run dataselect python programs.
Expected Output	Connect to database steadily with account and password. Select data out and store them into different sorted arrays. So the system will get two kinds of data which one is about self-involving exercise and the other one is about watching or discussing sports.
Succeed/ Fail	Succeed if each arrays are print out in the browser and the statistic appears is right. Fail if there is nothing in the browser or wrong is displayed in the browser.
Comment	This is to test whether the system could select the required data and ready to be called by functions.

Table 6-5

### 6.2.5 Test Unit: Heat map and Sports intension & involvement

Histogram is one key function of our system. User could see these tables or charts after choosing to click histogram button on screen.

Input Requirement	Mysql database is running. Apache webserver is running. Data is ready for draw real-time histograms. Internet connection to Google map api is running. Then run these section codes.
Expected Output	Web page for displaying histograms is opened. Histograms are set in defined space and basic map function such as enlarge and shrink is normal. Three kind of histograms show regularly. They are heat map for showing the popularity of sports, sports intension among area's residents and sports involvement degree.
Succeed/ Fail	Succeed if histograms display and all sub-functions run normally. Fail if the web page cannot be opened or displays of histograms are wrong or no specific markers on map.
Comment	This is to test whether the histograms could show and all functions could be switch randomly and run steadily.

Table 6-6

### 6.2.6 Test Unit: Exercise history

Collecting more data and categorize them by different slots (i.e. morning/ afternoon/ evening)

In this feature, we are able to analyze what is suitable for exercising dependent to specific time. By reaching this point, user is allowed to appoint where and when is most suitable for joining in others exercise event. For example, if someone is willing to play basketball at 8pm, but cannot gather enough teammates to play a game, then maybe he/she can find some information that show the location where other people played most of times before a few days at 8pm.

Input Requirement	Mysql database is running. Apache webserver is running. Data source is ready for Trend plotting. Then run the Trend profiles and click on Trend button.
Expected Output	Web page for displaying histogram is opened. Trend set in defined space. Different lines with different color appear in the same chart.
Succeed/ Fail	Succeed if the line chart appears. Values and description of axis are accurate. Fail if the web page cannot be opened or there is no trend chart display or the chart cannot show all required statistics.
Comment	This is to test whether the trend is plotting right and implement in the right space on webpage.

Table 6-7

### 6.2.7 Test Unit: Android application log in

States	States Description	Input Requirement	Output expected	Comments
Valid	The user will be able to log in our application after clicking log button and input valid account. If the user has not registered yet, the system will tell the user to register.	No starting with empty blank and caps lock matters. The input have to be a exact match with the previous input.	Successfully log in.	This test is to make sure that the log in function of the android program works fine.
Invalid	The user will be able to log on the system no matter if he registered or not	Input a account that has not been registered or input with different characters.	System tells the user to type in again and if fails again, tell users to register.	

Table 6-8

### 6.2.8 Test Unit: Android application registration

States	States Description	Input Requirement	Output expected	Comments
Valid	The user will be able to register for our program only if he type in the password the same twice.	Input the password the same twice.	Successful register	This test is to make sure that our user can successfully register for our system.
Invalid	Not able to register or type in different passwords but still able to register.	Type in password different for the two times. Start with empty blank.	Not able to register	

Table 6-9

### 6.2.9 Test Unit: Android application display page

States	States Description	Input Requirement	Output expected	Comments
Valid	The picture of Micheal Jordan shows up	No	The picture of Micheal Jordan shows up	Test if the display page works.
Invalid	Nothing shows up	No	Nothing	

Table 6-10

## 6.3 Test Coverage

Test coverage is a method and measure used to describe the degree to which the source code of a program is tested by a particular test suite. Our tests covered most part of our codes. We will test our system and revise our codes every certain time. Every testing cycle, measuring code coverage, writing tests, running tests and revising code will be repeated.

## 6.4 Integration testing strategy

To test the integration feature of our system, it is essential for us to focus on the assigned php file because this file takes all responsibilities to enable the communication between the database and the front-end. Once we make sure it works successfully, the integration test is finished.

Prerequisites:	<ol style="list-style-type: none"><li>1. MySQL is open</li><li>2. Sever is connecting to the internet</li><li>3. Local server is on service</li></ol>
Test by steps:	<ol style="list-style-type: none"><li>1. Data Collection Input the 'TWITTER_CONSUMER_KEY', 'TWITTER_CONSUMER_SECRET', 'OAUTH_TOKEN', 'OAUTH_SECRET', which you get from Twitter to use their 36 Streaming API, in the 140dev.php file. Change FilePath in get_tweets.php and change the database information which contains host IP Address, username, password and the database name which you want to store the collected data in db_config.php. Then run the get_tweets.php to retrieve data from Twitter, which will be stored as json file in database. Run parse_tweets.php that is a separate background script that takes that value from our json_cache table and parses it into the rest of the DB by removing it into tweets, tweet id, gps coordinates, et cetera.</li><li>2. DataFilter Change the database setting in filter.php first. Then run it to filter all history data and store the information of users who are in U.S and their tweets also in two tables.</li><li>3. Run our program in browser Make sure all the settings about the database are correct. Run index.html.</li></ol>
State:Success	<ol style="list-style-type: none"><li>1.The jsonsender can communicate with the website and the database</li><li>2.Database table with user's information</li><li>3.Data can be request.</li></ol>

State:Fail	1.The jsonsender could not communicate with the website and the database 2.Database table with user's information 3.Data could not be request
------------	---

Table 6-11



## 7 Plan of Work

Meeting date and location: Our team holds group meeting twice a week on Monday and Wednesday at Study Room 1 or 3, Library of Science and Medicine.

### 7.1 Merging the Contributions from Individual Team Members

This report was compiled by all the team members based on our own works.

There are several issues encountered when combining our works. The first and vital problem is that when we want to analysis the row data and kick the irrelevant tweets out of database, it is hard to determine which one is the irrelevant. If we kicked out the actually relevant tweets or stored irrelevant tweets, either of them will influence the accuracy of the final analysis. We spend lot of time and energy to solve this issue, and we finally get more accurate data since report 1. However, we still think that we can enhance the accuracy by modifying our algorithms.

Another problem we met is that when we try to use more than one computer to retrieve the tweets, it is hard to combine mass data from several computers into one. And since the data size grow bigger and bigger, it took too much time to analysis all the data we retrieved by our own computer.

### 7.2 Project Coordination and Progress Report

#### Schedule before report 2

Data Collecting (Server)	09/14/2015	10/16/2015
Investigate Twitter API	09/14/2015	09/18/2015
Investigate the database codes from the former groups	09/17/2015	09/23/2015
Re-establish the database from the former group 2 and refine the keywords	09/17/2015	09/23/2015
Discontinuously download the data by Twitter streaming API	09/17/2015	09/23/2015

Implement Twitter API	09/24/2015	09/26/2015
Investigate text analytics APIs for demography information	09/29/2015	10/10/2015
Figure out gender, age and type by text analytics API	10/11/2015	10/16/2015
Data Analyzing (Database)	09/16/2015	10/16/2015
Investigate the features in reports from former groups	09/16/2015	09/22/2015
Tweet heat: geographical distribution	09/23/2015	10/16/2015
Tweet heat: user ranking	09/23/2015	10/16/2015
Exercising duration	09/23/2015	10/20/2015
Personal Diagnosis	10/10/2015	10/18/2015
Data Displaying (Website)	09/20/2015	10/20/2015
Set up the communication between front-end and rear-end	09/20/2015	10/20/2015
Data Collecting (Server)	10/17/2015	10/27/2015
Download data for long period	10/17/2015	10/27/2015
First demo	10/26/2015	11/05/2015
Structure the website	10/17/2015	11/05/2015
Implement the website	10/17/2015	11/04/2015

Table 7-1 Previous Work

The table 7-1 shows the description of what we have done till this report.

## 7.3 Plan of Work

### After Report 2

Data Analyzing (Database)	10/20/2015	12/05/2015
Tweet sentiment	10/20/2015	11/17/2015
Word frequency	10/26/2015	11/17/2015
Personal diagnosis	11/05/2015	11/17/2015
Refine the implemented features	11/05/2015	12/04/2015
Discuss new features	11/09/2015	12/04/2015
Data Displaying (Website)	10/15/2015	12/05/2015
Refine the website	11/15/2015	12/01/2015
Refine the method for exercise frequency	11/20/2015	12/01/2015

Improve the accuracy of other features	11/20/2015	12/05/2015
Second demo	12/05/2015	12/05/2015

Table 7-2 Future work

## 7.4. Breakdown of Responsibilities

### 7.4.1. Project basic structure work

We have one or more team members to work on each project basic structure. Since every basic structure works are mostly fixed and low-layered, little changes need to be implemented when we develop the high-layered feature upon them. Therefore we suggest everyone to work on the basic structure first and integrate with other team member later. By doing so, we are able to speed up the developing process and diminish the overlap and conflict of different team members working on the relevant part. It is true that each team member may mainly working on a specific part of the project. Yet everyone needs to have a good understanding of the whole project. That's why during our group meeting, we will report out own work to the whole group. To push the whole group moving forward, we have specific deadline for each part of the work as presented in the following sections.

Assignments	Junjie Feng	Kai Kang	Ruotian Zhang	Tianzhe Wang	Weidi Zhang	Zhijie Zhang
Data Collecting		√		√		√
Data Management	√		√		√	
Website Display			√	√		√
Integration and Management	√	√			√	

Table 7-3 Basic structure

The assignments of basic structure are shown in Table 7-1. Data collecting is to get the public information and tweets from Twitter. Also using demography speculation API to get users' private information. The team members who are responsible for this part need to download data to our local database. And then they need to export and import the database structure and the data for other team member to use them. Data management means designing the tables in our database. The website display parts are responsible for design and implement the UI and display the results. The last part is

for combining everyone's work and integrates it to the whole project. Integration also involves communication between front-end and rear-end which is using JSON files.

## 7.4.2 Product ownership (Features)

Assignments	Junjie Feng	Kai Kang	Ruotian Zhang	Tianzhe Wang	Weidi Zhang	Zhijie Zhang
Tweet heat in geographical distribution					√	
Tweet heat in variation tendency		√				
Tweet heat in exercising classification			√			
Tweet heat in demography						√
User ranking				√		
Exercising duration	√					
Exercising frequency		√				
Word frequency				√		
Correlation topics						√
Tweet sentiment			√			
Personal diagnosis					√	

Table 7-4 Feature distribution

As shown in the Table 7-2, every team member is assigned with several feature works. Therefore every one will work on the data analyzing of the project. If any team member comes up a new feature, the whole group will discuss the new feature and decide whether that feature should be added into our project.

### 7.4.3 Breakdown of responsibilities (Report 2)

Assignments	Junjie Feng	Kai Kang	Ruotian Zhang	Tianzhe Wang	Weidi Zhang	Zhijie Zhang
Interaction Diagrams	√	√	√	√	√	√
Class Diagram	√					
Data Types and Operation Signatures		√	√			
Traceability Matrix						√
Architectural Styles				√		√
Identifying Subsystems				√		
Mapping Subsystems to Hardware				√		
Persistent Data Storage					√	
Network Protocol					√	
Global Control Flow & Hardware Requirements					√	
Algorithms		√			√	
Data Structures		√			√	
User Interface Design and Implementation				√		√
Design of Tests			√			
Project Management and Plan of Work	√					
References	√					

Table 7-3 Report writing

As shown in Table 7-3, the assignments of the report 1 are distributed to different team members. Although each team member only needs to write several parts of the report, the whole group revises the report 1 together.

## 8 References

- [1] Ivan Marsic. Software Engineering. September 10, 2012.
- [2] Nathan Fraenkel, Varun Gupta, Jason Lucibello, Boyang Zhang. Language Disambiguation for Disease Analysis. 2013-2014
- [3] Setup Mysql: <https://www.cnblogs.com/onlycxue/p/3291889.html>.
- [4] Setup PHP & Apache & Mysql: <http://www.jb51.net/article/53787.htm>.  
[http://www.cnblogs.com/good\\_hans/archive/2010/04/01/1702059.html](http://www.cnblogs.com/good_hans/archive/2010/04/01/1702059.html).
- [5] Steven Bird, Ewan Klein, Edward Loper. Nature language processing with python. June 2009.
- [6] Matthew A. Russell. Mining Social web. January 2011.
- [7] Setup Apache:  
<http://jingyan.baidu.com/article/c85b7a642df6f7003bac95d9.html>
- [8] Twitter API: <https://dev.twitter.com/overview/documentation>
- [9] Rest API: <https://github.com/abraham/twitteroauth>
- [10] Demographic API: <http://textalytics.com/core/userdemographics-info>
- [11] Google API: <https://developers.google.com/apis-explorer/#p/>
- [12] Setup PHP Mysql environment:  
[http://www.cnblogs.com/good\\_hans/archive/2010/04/01/1702059.html](http://www.cnblogs.com/good_hans/archive/2010/04/01/1702059.html)
- [13] A maximum entropy approach to natural language processing
- [14] Phirehouse: <https://github.com/fennb/phirehose>
- [15] S. Kumar, F. Morstatter, H. Liu. Twitter Data Analytics. Aug. 19, 2013.
- [16] JSON: <http://en.wikipedia.org/wiki/JSON>