**武汉大学计算机学院**

**课程论文**

**Game Data Analysis of**

**League of Legend**

**课程名称 商务智能**

**专业年级班级 2016级 卓越二班**

**姓名 刘瑞康**

**学号 2016302580242**

**学期 2018-2019 学年 第一 学期**

**成绩 任课教师签名**

Content

[1. Background 3](#_Toc532557557)

[1.1 League of Legends 3](#_Toc532557558)

[1.2 Data analysis on games 4](#_Toc532557559)

[2. Requirement Analysis 4](#_Toc532557560)

[3. Data Set 5](#_Toc532557561)

[4. Running Environment and Dependencies and Technolgy Used 5](#_Toc532557562)

[5. Complete Solution & Whole Process 5](#_Toc532557563)

[5.1 Data source for analysis 5](#_Toc532557564)

[5.2 Data storage 7](#_Toc532557565)

[5.3 Data retrieval 8](#_Toc532557566)

[5.4 Data cleaning 10](#_Toc532557567)

[5.5 Typical value computing and data normalization 11](#_Toc532557568)

[5.6 Hierarchical clustering analysis 12](#_Toc532557569)

[5.7 Principal component analysis 15](#_Toc532557570)

[6. Conclusions 16](#_Toc532557571)

[7. Details And Problems 19](#_Toc532557572)

[7.1 About implementation 19](#_Toc532557573)

[7.2 About results 20](#_Toc532557574)

[8. How to see results 21](#_Toc532557575)

# Background

## 1.1 League of Legends

League of Legends (abbreviated LOL) is a multiplayer online battle arena video game developed and published by Riot Games.



In League of Legends, players assume the role of an unseen "summoner" that controls a "champion" with unique abilities and battle against a team of other players. The goal is usually to destroy the opposing team's "nexus", a structure that lies at the heart of a base protected by defensive structures. Each League of Legends match is discrete, with all champions starting off fairly weak but increasing in strength by accumulating items and experience over the course of the game.

LOL was generally well received upon its release in 2009, and has since grown in popularity, with an active and expansive fan base. In September 2016, the company estimated that there are over 100 million active players each month. The 2017 World Championship had 60 million unique viewers and a total prize pool of over 4 million USD. The 2018 Mid-Season Invitational had an overall peak concurrent viewership of 19.8 million, while the finals had an average concurrent viewership of 11 million.

## 1.2 Data analysis on games

Data analysis are widely used in game data mining to collect information for game designers and game players. It can be used for game balance, champion positioning and champion recommendation and so on. In fact, the champion team iG takes advantage of the predictive analysis in its ban/pick phase.

# Requirement Analysis



In LOL, there are six official positions for champions: Fighter, Marksman, Mage, Tank, Assassin, Support. However, some of the positions are based on their characteristics like their attack mode, and some are just based on the designer's original intention in designing phase. For example, all the champions with attack mode shooting are regarded as Marksman, and there are many champions which are designed as Tank but sometimes play a role as Fighter or even Assassin. So I’m always curious about their typical roles in the real game instead of the official positions – which of them are more similar and which of them are quite different from the original settings and so on.

# Data Set

All intermediate data and final data that is used in this project are stored in the directory **data** with json files and a database file.

# Running Environment and Dependencies and Technolgy Used

The solution projecet is run on JetBrains **PyCharm** Community Edition, based on **python 36** with urllib, sqlite3, numpy, scipy, matplotlib and sklearn, using **hierarchical clustering analysis** and **principal component analysis**

# Complete Solution & Whole Process

## 5.1 Data source for analysis

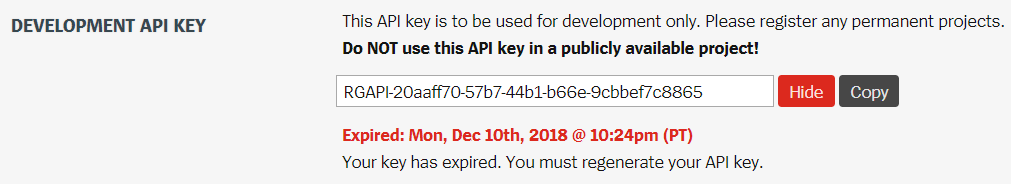
For analysis, an enormous amount of match data are needed. Since I’m a LOL player using chinese client and server, firstly I excepted to get the match data from its agent in China – Tencent. However, after browsing the official website carefully, I’m sure that it doesn’t provide any API for the match data.

Fortunately, the North America(NA) server provides its match data free of charge by calling API . So the data used in this project is all from **NA server**.

To use the API, a developer need to register in <https://developer.riotgames.com/> and has a game account in NA server. And all the APIs provided are listed in <https://developer.riotgames.com/api-methods/>.

For each API request, an API key must be included your API key using the api\_key parameter on each request, that’s why a developer need to register.

Developers can easily generate API key, but the API key string will be **expired in 24 hours** after applying each time, so it need updating mostly every time.



Registering permanent projects can provide a permanent API key, but my application is still pending when I’m writing this report.



## 5.2 Data storage

Since the project is based on python36 and the data volume needed is not much huge, sqlite3 is the best choice. All the operations related to database are defined and implemented in **utils/SQLiteUtils.py**.

The database called info.db has 5 data table – account, match, championMatchData, championData and championInfo.

account is used to store the accounts searched or to be searched for match data.

match is used to store the match id of the match data.

championMatchData stores the numerical details of a specific champion in a match.

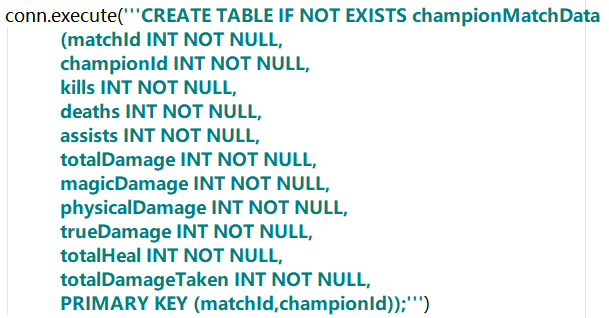
championData is used to store the typical value for each champion

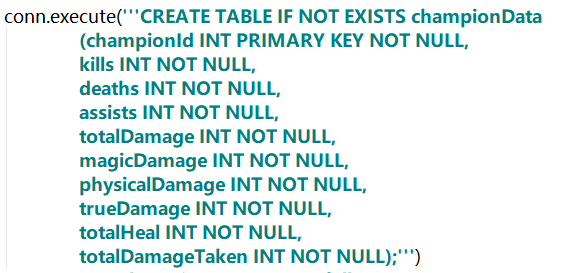
championInfo stores the detailed information for each champion id

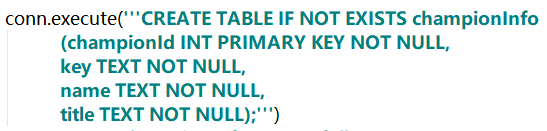
The data table structure are listed below.











## 5.3 Data retrieval

The main two API used in this project are:

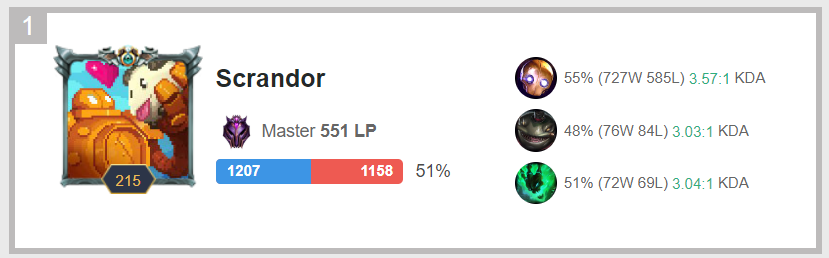
/lol/match/v3/matchlists/by-account/{accountId}

/lol/match/v3/matches/{matchId}

The first one is used to get the details of a specific match with match id matchId, the other is used to get match list of a specific player with account id accountId. These APIs return json files. The sample json file returned by the first API is **data/matchlist.json**, and the sample json file returned by the second one is **data/singlematch.json**.

Since there are no APIs to get a sufficient number of match data, it’s required to start from a seed account id, get the match list of the player, and then for each match, get all of the participants of the match, and repeat this process.

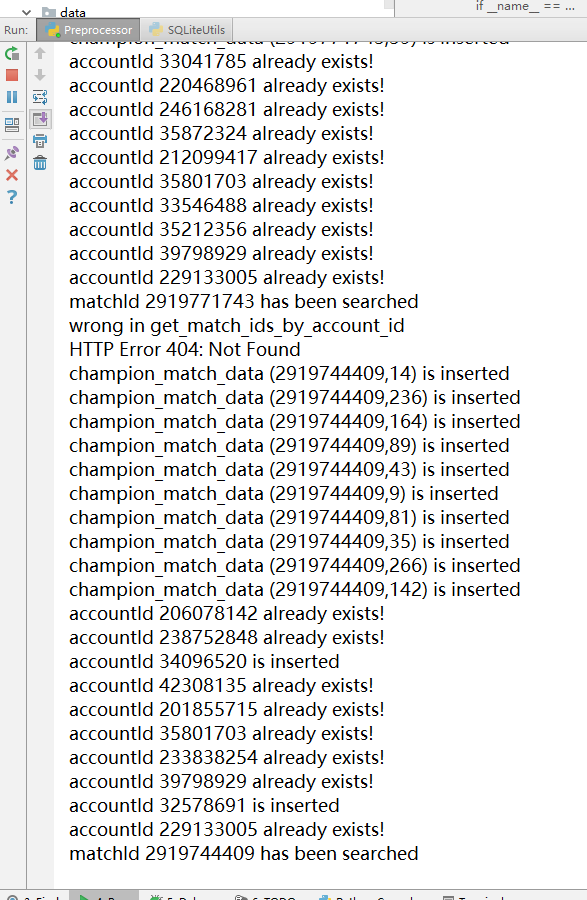
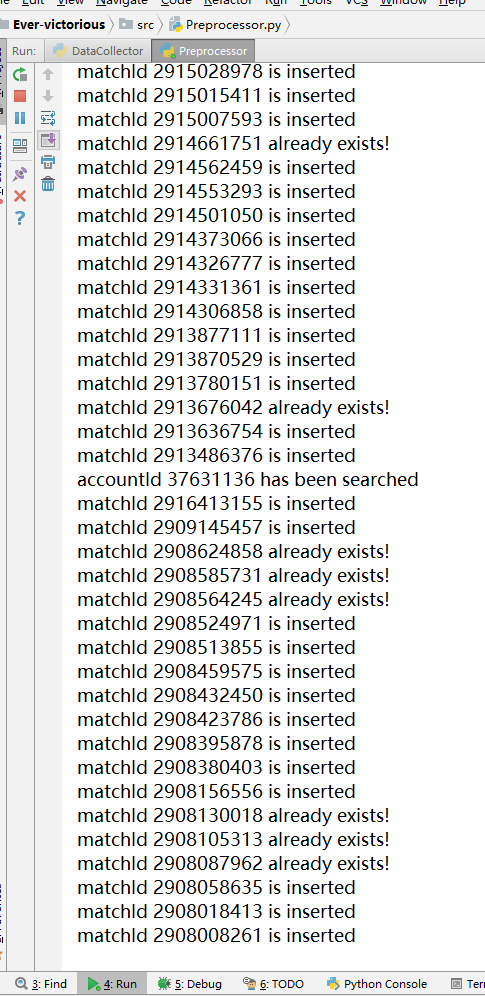
The seed account id chosen is the player with currently ranking #1 in NA server. (The rankings come from <http://www.op.gg/>). The account id obtained by API /lol/summoner/v3/summoners/by-name/{summonerName} is 37281196.



All the operations related to API are defined and implemented in **utils/DataCollector.py**.

In the file **src/Preprocessor.py**, **SQLiteUtils.py** and **DataCollector.py** are used to define methods to implement the loop of getting enough match data and store the data into corresponding data table.

Part of logs during data retrival are as follows:

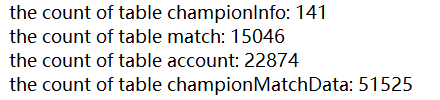
 

## 5.4 Data cleaning

After data retrieval, invalid data need to be removed. For now, with only the data in the json files, there is no way to judge whether the player is active during the game. So personally conclude that if all of kills, deaths and assists are less than 2, the player is regarded as inactive and the match data of the champion usd by the player is invalid.

The method for data cleaning is defined in **SQLiteUtils.py** and is called by **Preprocessor.py** after data retrieval.

After data cleaning, the data in the database is as follows:



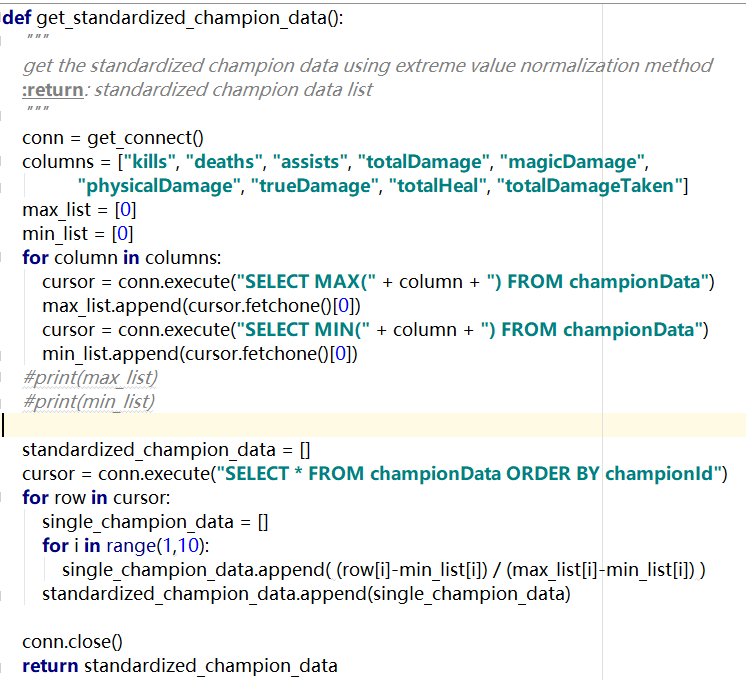
## 5.5 Typical value computing and data normalization

At this stage, simply regard the average value among the matches of a specific champion as its typical value for analysis.

Part of the logs for getting typical value are as follows:



Different data indicators often have different dimension and dimension units, which will affect the results of data analysis. In order to eliminate the dimension effect between indicators, data standardization is needed to solve the comparability between data indicators. To get the standardized champion data, extreme value normalization method is used to scale all the data into [0,1]



## 5.6 Hierarchical clustering analysis

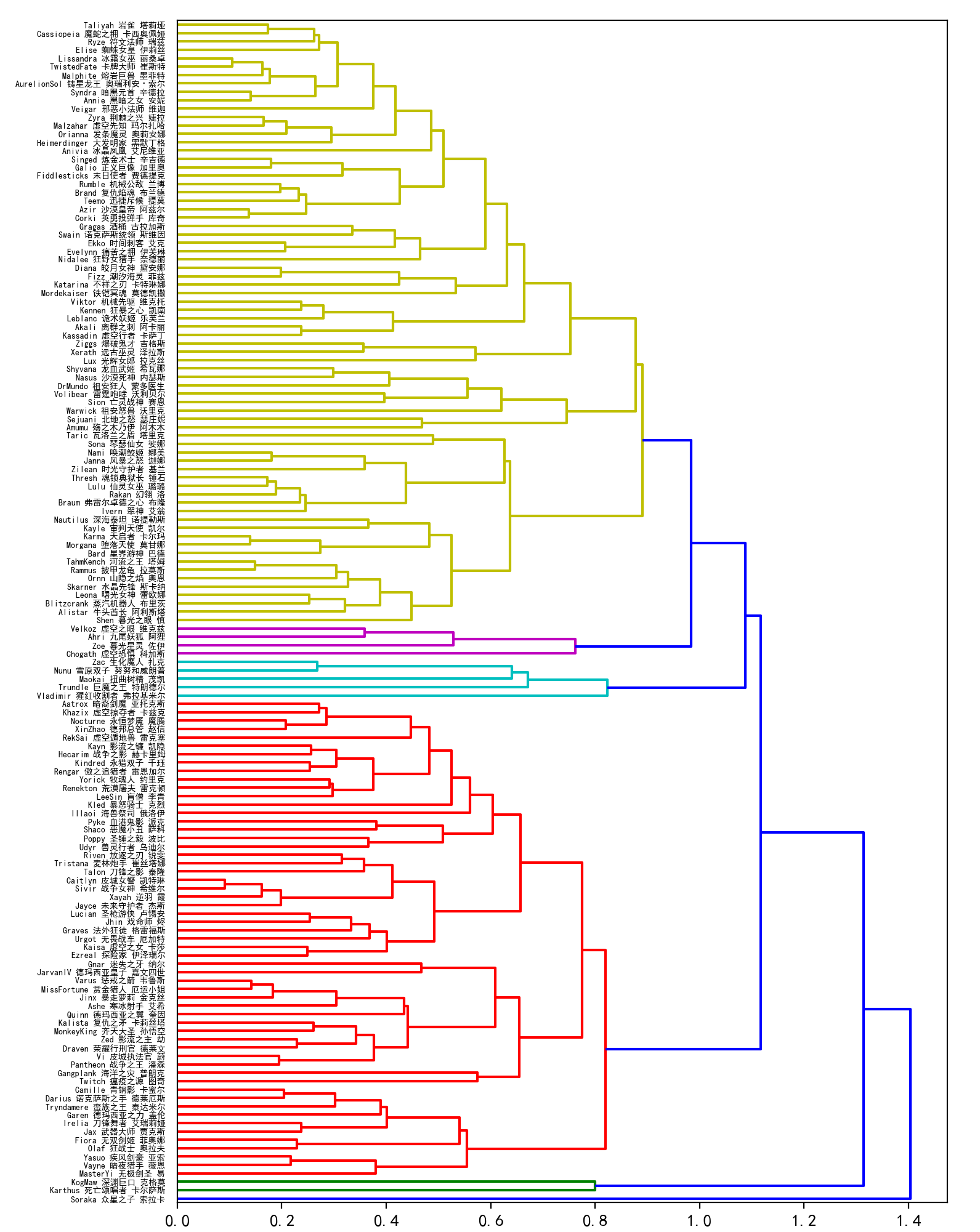
For analysing the similarity and mining, unsupervised learning approach is more suitable for this project.

At first, I tended to use k-means clustering method for analysis since we have learned it in the class. However, after actual implementation, the results expose many shortcomings of k-means clustering method:

* Clustering results are affected by the initial number of clustering centers
* Clustering results are influenced by initial center selection
* Clustering results are influenced by the order of sample input
* Non-convergence case will occur

For such reasons, hierarchical clustering method are chosen. Hierarchical clustering basically overcomes the shortcomings above. It can display the whole clustering process from discrete sample points to finally a single group with no conception of initial centers, center number, input order and non-convergence.

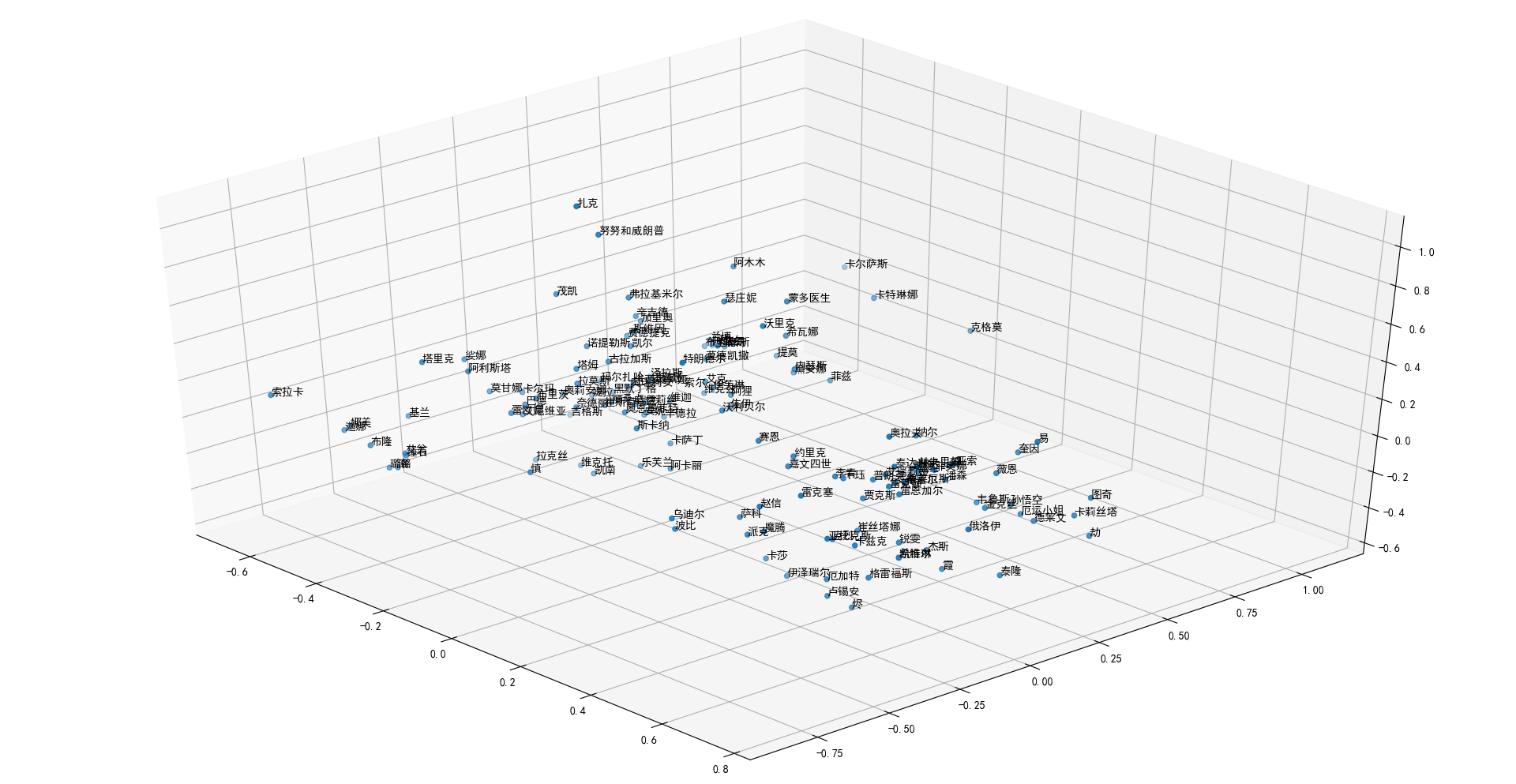
The hierarchical clustering result is as follows and saved in the same directory of this report named **clustering\_figure.png**:



## 5.7 Principal component analysis

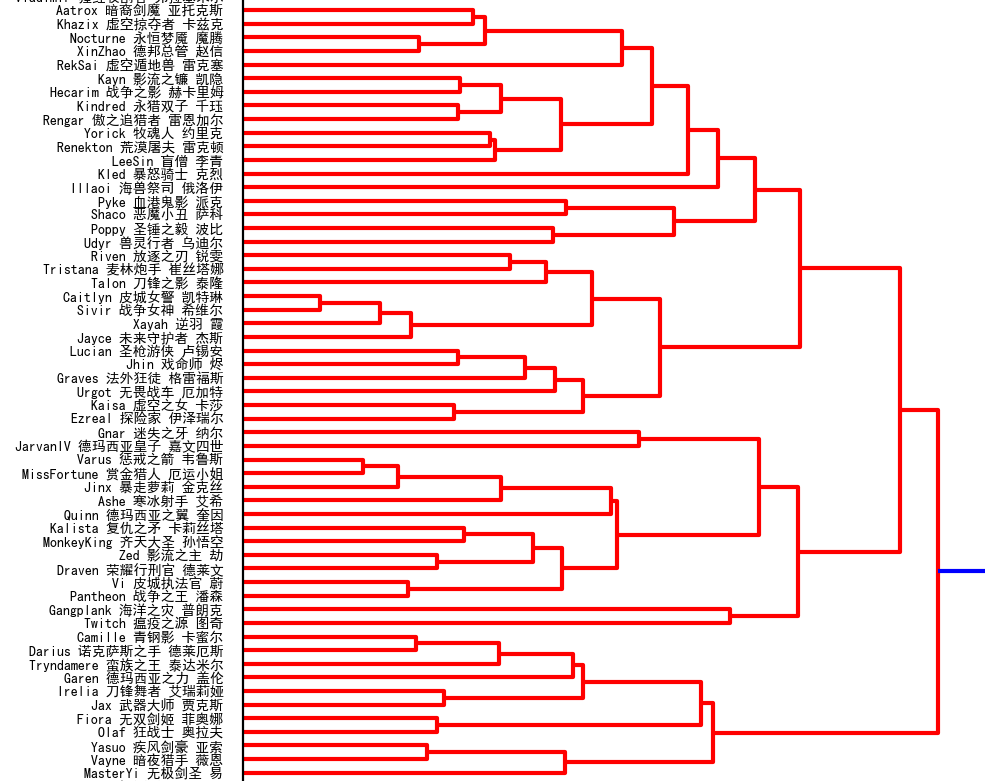
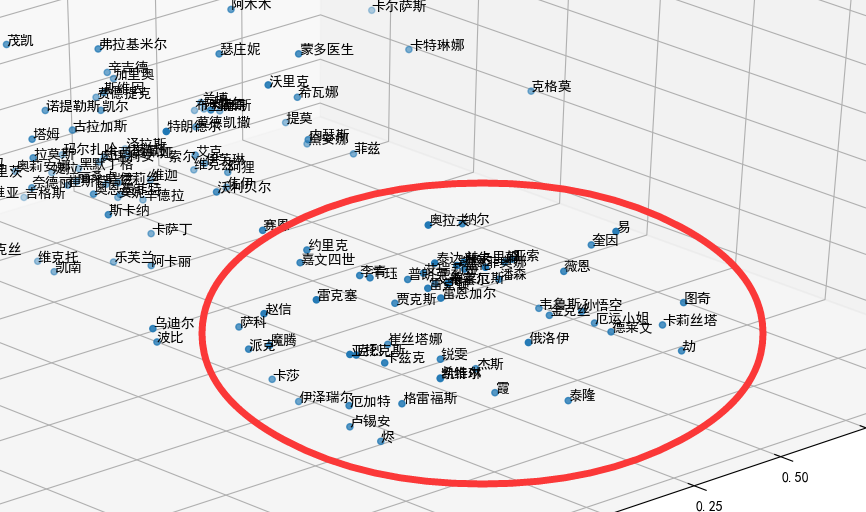
In order to show the similarity among champions in a more intuitive way, principal component analysis is applied to implement data dimension reduction to 3d.

The principal component analysis result is as follows and saved in the same directory of this report named **pca\_figure.png**:

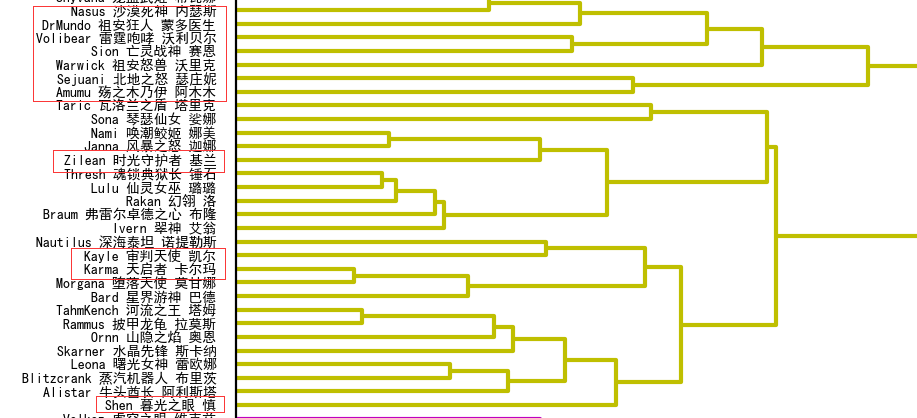


# Conclusions

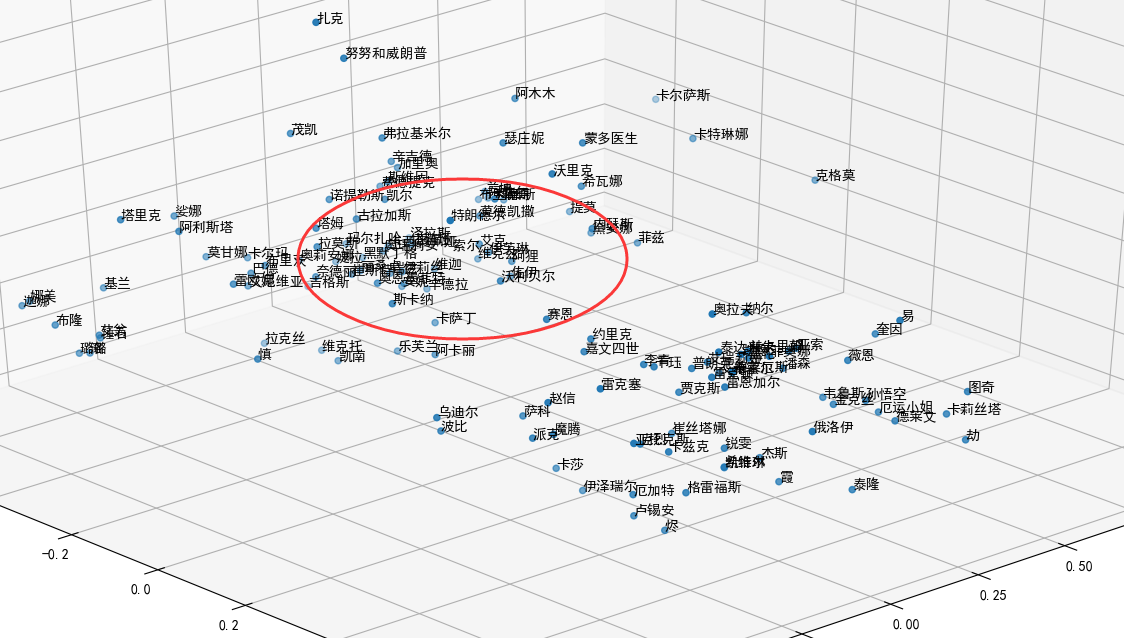
1. In both Hierarchical clustering analysis and Principal component analysis, most of the champions with position Marksman perform similarly with Fighters and Assassins. That means that they typically play the same role in a game, so it’s unadvisable to have too many such champions in one team.



1. Some funcitional Tanks and Mages perform similarly with Supports. The champions with position Support are designed for funcitional supports to other champions. But there are also champions with other positions can do the same things.



1. Some typical value of champions tend to be neutral. Most of those champions play flexible roles in games based on the items it equiped, which is determined by the game styles and preferences of the players.



**There are some rare extreme cases, too.**

1. Mage “Vladimir “ performs like Tank. This Mage can continuously suck blood from enemy, which make it the ability of taking a lot of damage like Tank





1. Marksman “KogMaw” performs neither like Marksman nor Mage, but in between. That’s because this champion’s skills make it the ability of causing both magic and physical demage.



1. Support “Pyke” performs like Assassin and Marksman. That’s because its unique skill can directly kill enemies with blood less than a certain amount.



**And there are many other interesting results.**

# Details And Problems

## 7.1 About implementation

In the process of implementation, firstly I tend to implement both the Hierarchical clustering analysis and Principal component analysis by myself. After the initial implementation and testing, it seems that the space efficiency and time efficiency of my implementation are not high enough. What’s more is that my implementation of Hierarchical clustering can only display the clustering process in text, which has poor readability.

After searching related information on the Internet, the python libraries **spicy.cluster** and **sklearn.decomposition** meet the needs of the project respectively for Hierarchical clustering analysis and Principal component analysis. That’s also the final version of the approaches adopted.

## 7.2 About results

In the process of Hierarchical clustering analysis and Principal component analysis, champions with positions Fighter, Assassin and Marksman have a mixing trend from the very beginning. But in real games, those champions are supposed to work together to win the game instead of having many champions of the same position. I suspect that there are two possible reasons for this situation.

The first reason is in my normalization process, each data indicators are scaled to [0, 1], which are used directly to compute the similarity distance. But in fact, these indicators should have different weights since the indicators represent different features of the champions. However, after trying many different weights combination, I still cannot get ideal clustering process that can divide those champions thoroughly, so in the final version of the project, the default weight 1 is remained.

The second reason is the lack of enough indicators. The match data returned by the API only contains values for the whole match. As a matter of experience, the Fighters and Assassins play an important role in the early stages of the game, while the Marksmen play a more crucial part in the later stages. If such information related to different stages can be obtained, at least the Marksman will separate from the mixing trend.

# How to see results

Each part of the process is separated in its corresponding .py files with condition “if \_\_name\_\_ == ‘\_\_main\_\_’”, so load the project into PyCharm and then it’s easy to run any part of the project by run the related files. Since the final data is stored in the database, it’s also convenient to directly run the **Analysis.py** to see the final analysis results.

For each API request, an API key must be included your API key using the api\_key parameter on each request, that’s why a developer need to register.

Developers can easily generate API key, but the API key string will be **expired in 24 hours** after applying each time, so it’s needed to update the **api\_key** in the file **utils/DataCollector.py** when wanting to call the methods of this file**.**

The steps for generating new api key:

1. Open <https://developer.riotgames.com/>
2. Login with your account, or my account with username RuikangLiu and password 123456Lrk
3. Open the DASHBOARD
4. Click the button “:REGENERATE API KEY” in the bottom of the page
5. Copy the new api key into the python file **utils/DataCollector.py** to replace the old string of **api\_key**