Game Data Analysis of League of Legends

Zhao Che
1155066626
simonzhao1024@gmail.com

ABSTRACT

In this team project, we crawl a large amount of crawled match data from an online battle game called League of Legends. Based on the data, we consider multiple analysis objectives such as game character performance ranking, character clustering and match prediction. Data analysis techniques in the area of machine learning and network analysis are applied. This individual report mainly focuses on our works on data crawling and character ranking.

Keywords

Online game analysis, data crawling, network analysis, network centralities

1. Introduction

The history of video games are quite short. A video game can be viewed as a system with many complicated rules. There are still many unsolved problems on how to well design a video game. An online video game usually allows multiple players to interact with each other. So this kind of game is even more complicated and difficult to design. For example, for a PVP (player v.s. player) type game, the balance of the game is quite important but really hard to handle. Analyzing historical data of players in the game to evaluate and improve game design is now becoming a hot trend. Also, in a PVP online game, analysis of the player behaviors provide insight to human interactions in the real world and the potential of virtual worlds for education, training, and scientific research.[1] [2]

This project chooses one of the most popular online PVP video game, League of Legend to experiment multiple analysis such as game character performance ranking, character clustering and match prediction. Data analysis techniques in the area of machine learning and network analysis are applied, which we think can help us understand deeply about the game and get insights on game design.

In **chapter 2** we will describe the significant features of the online video game we analyze. Then we introduce our crawling strategy and briefly summarize the dataset we obtained in **chapter 3**. **Chapter 4** is about explorations of champion ranking with different models and metrics.

2. Target Game

LoL (League of Legends) is a multiplayer online battle arena (MOBA) video game with 27 million daily players [3]. In LoL, a standard match consists of two opposing teams of five players. Every “summoner” (player) controls a "champion" (character) with unique abilities. The goal is to destroy the opposing team's "nexus", a structure which lies at the heart of a base protected by defensive structures called “turrets”. Each 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. [4]

During a match, a champion can be killed by other champions and then revive. The killer will get one killing score. If other champions deal damage to the victim right before a champion kills him, then will get one assisting score. This kind of killing scoring system is quite similar with real games like basketball. But the significant difference between LoL and basketball is that the champion character is unique, so while the result of a basketball match only depends on the player’s own skills, LoL also involves each champion’s unique ability setting.

Every champion is designed for different positions in the team. LoL has officially labeled each champion with these positions. The positions can be *Fighter* (high damage, close fighting), *Marksman* (high physical damage, distant fighting), *Mage* (high magical damage, distant fighting), *Tank* (high defence ability from damages), *Assassin* (high explosive damage and escaping ability), *Support* (support ability such as healing). Every champion has one or two position given by Riot. However, the real performance in matches of each champion can vary from the official definition. For example, a champion called Grey Foss is designed to be a Marksman, but his actual performance is more like a Fighter.

MOBA games like LoL have share many features with real sport games like basketball, such as position allocation and scoring rules. But there is a significant difference: while basketball only involves players, result of a LoL match can be effected by both skills of players and the ability of champions they choose. The balance of basketball is quite easy, all the rules and environments are obviously the same for players in both sides. But for LoL, there are more already 128 champions in LoL. In each match, ten champions chosen by players are different.

3. Data Crawling

3.1 APIs

Riot, the developer company of LoL has release a set of REST API (Riot API) [5]. We can use it to freely obtain a variety of data from players and their historical matches. Riot API methods return JSON format data from URL-like queries.

Cassiopeia is a open source library, which wraps the API in a Python environment [6]. With Cassiopeia, the original API methods are transformed to Python functions, the query sentences are automatically built from function parameters, and returned JSON data is transformed into Python objects. Our crawler is based on Cassiopeia.

3.2 Database Design

There is high hierarchical dependency between data objects we want to obtain from the API methods, as illustrated below:

- *Summoner*
  - *MatchList [Match]*
    - *Participants [Participant]*
      - *ParticipantStats*
      - *ParticipantTimeline*
    - *Timeline*
      - *Frames [Frame]*
        - *Events [Event]*
    - *Team*

We design a database to store the data without losing their relationships and also for the convenience of our analysis. The database contains seven tables for crawled data: *Summoner*, *Match*, *Team*, *TeamBan*, *Participant*, *ParticipantTimeline*, *FrameKillEvent. FrameKillEvent* contains all the champion killing events in every match we crawled. It contains the information of killer, victim and helper in every killing event. This table is highlighted because we use its data to do champion ranking experiments introduced in **4**.

3.3 Crawling Strategy

As an online game, LoL keeps updating. There are big differences between editions in each match season. In addition, LoL has different servers for worldwide players. We choose to focus on the North American area's data in

the Pre-Season 2016. Besides, because we can't crawl all the data in the season, which is too big to handle, we use a sampling strategy. In LoL, different players have different "tier", which quantifies their game skills. In order to make the sample more representative considering the average skill level of all players, we choose to crawl more data in tier "Silver", which has the largest percentage in the whole player group.

Our actual crawling process is as following:

(1) randomly choose 3 players (summoners) with tier "Silver", "Challenger" and "Diamond" separately as seeds, add them to an empty seed list;

(2) for each summoner in the seed list, have a processor crawl all of his historical matches' information in North American area during Pre-Season 2016;

(3) for each crawled much, obtain other summoners' ID. If the summoner has not been crawled, add its ID to the seed list;

(4) end the corresponding processor when we obtain about 100,000 matches from the Silver seed, 60,000 from the Challenger seeds and 60,000 from the diamond seeds;

(5) repeat step (2)

(6) merge the three crawled datasets into one;

This strategy is a kind of breadth first strategy, but it can still guarantee a low bias sampled dataset. This is because LoL has a Tier Matching system. It will randomly group players with similar levels to form a match. Besides, this system evaluates a player's level not only by his tier, but also his recent match records. This means that through three seed players, we can also obtain some (relatively less) data from players with other tiers like "Gold" and "Bronze".

3.4 Crawled Data

Here is a brief summary about the data we crawled:

SQLite Database size: 2.1 GB

Summoner: 487,484 records

Match: 222,652 records

Team: 445,304 records

TeamBan: 1,330,757 records

Participant: 2,226,520 records

ParticipantTimeLine: 8,906,080 records

FrameKillEvent: 21,692,852 records

4. Champion Ranking with Graph Model

Based on the historical data from 222,652 matches in the *FrameKillEvent*, we try to find some empirical methods to evaluate a champion's ability of killing and assisting and rank champions by the magnitude of abilities. There are mainly two ways for this objective:

One is to simply summarize the number of times a champion becomes a killer or helper in all the crawled matches and average by the times it is chosen (number of picks). The result is the average killing / assisting scores get per game of the champion, which can be used to rank champion's killing / assisting ability;

Another considers the contribution from the killing / assisting events between each pair of champions and transform the problem into a graph model, and then apply some network centrality metrics to evaluate the overall killing / assisting performance of a champion.

4.1 Champion Ability Ranking with Average Score

Here are the results of killing ability rank and assisting ability rank with the average score method mentioned above:

Table 1. Champion Killing Ability Rank with Average Score

	Champion	Score	Position		Champion	Score	Position
	Akali	9.031286	Assassin		Soraka	0.738921	Support, Mage
Top	Talon	8.934305	Assassin, Fighter	**Bottom**	Janna	0.967073	Support, Mage
10	Fizz	8.835916	Assassin	**10**	Braum	1.603554	Support, Tank
	Zed	8.432829	Assassin, Fighter		Nami	1.837762	Support, Mage
	Katarina	8.303200	Assassin, Mage		Alistar	1.851857	Tank, Support

Table 2. Champion Assisting Ability Rank with Average Score

	Champion	Score	Position		Champion	Score	Position
	Nami	15.268267	Support, Mage		Master Yi	5.003692	Assassin, Fighter
Top	Janna	14.727506	Support, Mage	**Bottom**	Tryndamere	5.052246	Fighter, Assassin
5	Braum	14.124870	Support, Tank	**5**	Riven	5.103746	Fighter, Assassin
	Soraka	14.058027	Support, Mage		Jax	5.268868	Fighter, Assassin
	Thresh	13.811561	Support, Fighter		Fiora	5.318172	Fighter, Assassin

4.2 Graph Model of Champion Relationship

We can think killing or Assisting as a kind of relationship between champions, and this can be modeled by a weighted directed graph: the nodes are champions, and the edges are the relationship, and the weights of the edges are the normalized killing / assisting scores (say, normalized by number of picks of A). For example, if champion A kills champion B for k times, then their would be an edge from A to B, weighted by the **k / picks(A)**. Written in matrix form, it is $\mathbf{K_{A,B} = k}$. It is easy to see this graph matrix is asymmetric (A kills B is not equal to B kills A), non-negative (the scores must be non-negative) and complete (the killing or assisting events could happen between each pair of champions).

4.3 Champion Ability Ranking with Graph Model

In this section, we will introduce champion ability ranking methods with graph model. Based on the graph model, we tried some network centrality metrics to rank champion's abilities. Here are the results of killing ability rank and assisting ability rank with eigenvector centrality and HITS centrality [7]. We also list each champion's official defined position for reference.

Table 3. Champion Killing Ability Rank with Eigenvector Centrality

	Champion	Score	Position		Champion	Score	Position
	Akali	0.137101	Assassin		Soraka	0.010750	Support, Mage
Top	Talon	0.136831	Assassin, Fighter	**Bottom**	Janna	0.014215	Support, Mage
5	Fizz	0.133501	Assassin, Fighter	**5**	Braum	0.023332	Support, Tank
	Zed	0.129720	Assassin, Fighter		Nami	0.026539	Support, Mage
	Katarina	0.123811	Assassin, Mage		Alistar	0.027421	Tank, Support

Table 4. Champion Killing Ability Rank with HITS Hub

	Champion	Score	Position		Champion	Score	Position
	Talon	0.012690	Assassin, Fighter		Soraka	0.001080	Support, Mage
Top	Akali	0.012320	Assassin	**Bottom**	Janna	0.001501	Support, Mage
5	**Twitch**	0.012129	Marksman, Assassin	**5**	Braum	0.002402	Support, Tank
	Fizz	0.011963	Assassin, Fighter		Alistar	0.002754	Tank, Support
	Katarina	0.011612	Assassin, Mage		Nami	0.002773	Support, Mage

Table 5. Champion Assisting Ability Rank with Eigenvector Centrality

	Champion	Score	Position		Champion	Score	Position
	Nami	0.150016	Support, Mage		Master Yi	0.051729	Assassin, Fighter
Top	Janna	0.144490	Support, Mage	**Bottom**	Tryndamere	0.052331	Fighter, Assassin
5	Braum	0.138364	Support, Tank	**5**	Riven	0.053255	Fighter, Assassin
	Soraka	0.137169	Support, Mage		Jax	0.054587	Fighter, Assassin
	Thresh	0.135735	Support, Fighter		Fiora	0.055673	Fighter, Assassin

Table 6. Champion Assisting Ability Rank with HITS Hub

	Champion	Score	Position		Champion	Score	Position
	Nami	0.016397	Support, Mage		**Vayne**	0.003561	Marksman, Assassin
Top	Janna	0.015918	Support, Mage	**Bottom**	**Tristana**	0.003760	Marksman, Assassin
5	Braum	0.015368	Support, Tank	**5**	**Draven**	0.003793	Marksman
	Thresh	0.014688	Support, Fighter		**Kalista**	0.003917	Marksman
	Soraka	0.014248	Support, Mage		**Lucian**	0.004176	Marksman

4.4 Result Analysis

There are quite a few interesting points from the results in **4.2** and **4.3**.

Firstly we compare champion killing ability rank results from different methods (**table 1, 3, 4**). We can see there is no significant difference of the top 5 and bottom 5 results among the methods. This may leads to a conclusion that if we want to rank champion's killing ability, average score method is good enough, as the other two methods are computationally more expensive. Besides, we can find the top 5 champions all have an *Assassin* position label, while the bottom 5 all have a *Support* label. This result fits the definition of the positions. *Assassin* champions are designed to have high killing ability, while *Support* champions' task is meant for supporting other teammates, which does not require high killing ability.

As for the assisting ranking (**table 2, 5, 6**), the top 5 rank results from three methods are also nearly the same, while the bottom 5 results are completely different from the other two methods. The bottom 5 given by average score and eigenvector centrality are exactly the same, of which all champions have both *Assassin* and *Fighter* labels, but the bottom 5 from Hub score of HITS are all champions with the label *Marksman*. In my opinion, these two results are both reasonable. *Assassin*, *Fighter* and *Marksman* champions usually have high damage ability, so the probability of killing is high, which reduces the probability of assisting. However, there is still big difference among these positions. For now we could not find a reason to explain this interesting phenomenon.

5. Future Works

There are still a lot of analysis works we expect to do. For the objectives we have tried, we think there are still large space for explorations. For example, for the data crawling part, we want to find a more elegant database structure to store the data because we often feels difficult and time-consuming to do some pre-processing works. And for the champion ability ranking, we can try to combine the match data with the champion's static data like basic damage value in every champion level. We can also extend this part to a deeper analysis of champion design.

REFERENCES

[1] W. S. Bainbridge, "The scientific research potential of virtual worlds," Science, vol. 317, pp. 472–476, 2012.

[2] M. D. Dickey, "Three dimensional virtual worlds and distance learning: Two case studies of active worlds as a medium for distance education," British Journal of Educational Technology, vol. 36, pp. 439–451, 2005.

[3] P. Tassi, "Riot's 'League of Legends' reveals astonishing 27 million daily players, 67 million monthly," Forbes, 2014

[4] "New Player Guide". League of Legends. Riot Games. Retrieved July 21, 2014.

[5] Riot Games, Inc , Riot Games API, https://developer.riotgames.com/api/methods/

[6] Meraki Analytics, Cassiopeia, https://github.com/meraki-analytics/cassiopeia/

[7] Newman M. Networks: an introduction[M]. OUP Oxford, 2010.