1. Introduction

Recommender system, one of the primary applications of data, has been rather prevalent in our daily life due to the boom of big data technology. It seems that people can be recommended to do something as long as the records of their historical behaviors are obtained and learned by this smart platform. In the domain of game, especially electronic sports (e-sports), data becomes more and more important since it can be generated considerably within a game and it usually provides some potential information for strategy making.

League of Legends (LOL) is a multiplayer online battle area game (MOBA) which features two teams made of five players each competing against each other. Prior to a battle, each teammate selects a champion to fight against the opponent team. Champion is a virtual fighter in game controlled by player. There are 148 champions and none of them is perfect. That means they are strong or weak at various tasks and affect the win of game differently. In this project, details covering the process of battle would not be discussed, instead we are only interested in how to select champions better to win the game.

However, there are some specific rules about the selection of champions. One of the most crucial rules is that a champion cannot be repeatedly chosen in a game. Besides, the selection can be briefly thought as player 1 in team 1 > player 1 in team2 >… > player 5 in team 1 > player 5 in team 2, and each team can ban serval champions that have not been picked during this process. In other words, players should be flexible as their intended champions may be suddenly banned by the opponent as well as they can ban champions that may constrain ones they have picked.

In short, the crucial problems of recommender system in this case are that how to pick champions that intend to win under different phases of selection, what is the alternative if an intended champion is banned and what champions should be ban to avoid countering. In fact, these are also be concerned by professional LOL teams, so it seems be very necessary to conduct such a system.

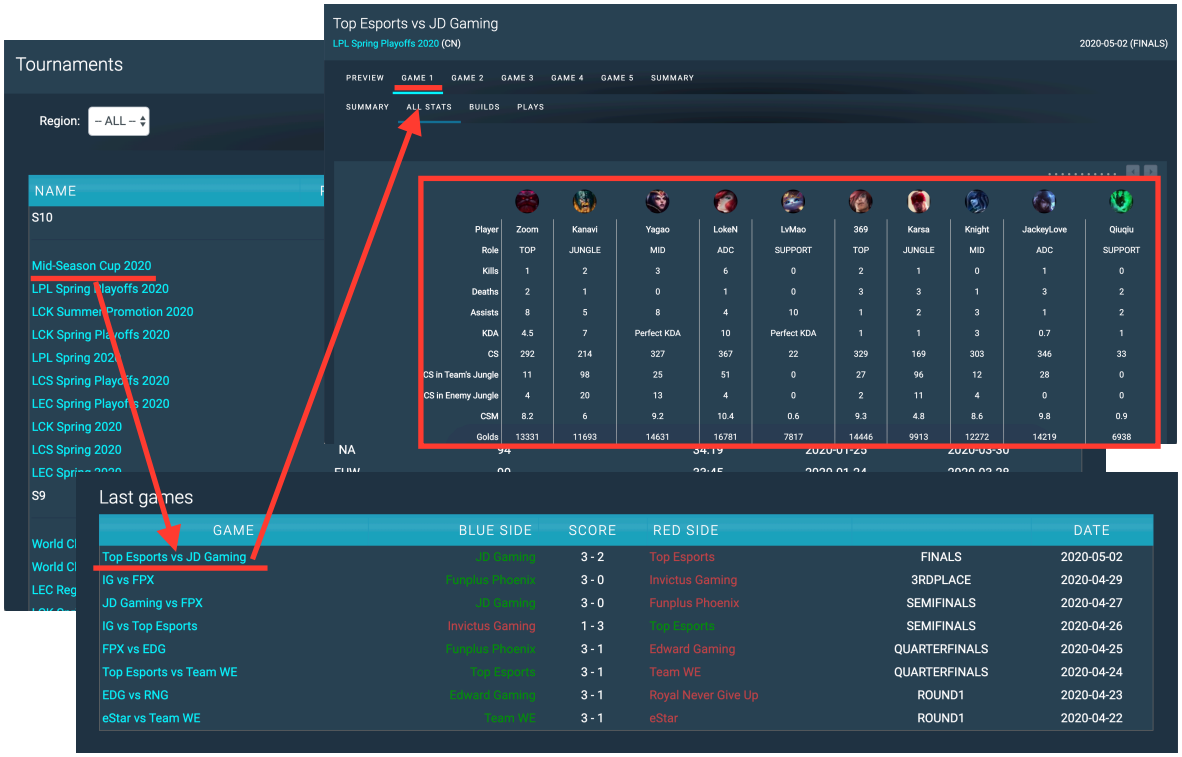


Figure 1. Web Scraping Logic

2. Data Acquisition

All data in this project was scrapped from Games of Legends (https://gol.gg/esports/home/), a website that collects the historical professional tournaments of LOL. Figure 1 indicates the process of web scrapping logic. In this project, tournaments held within the past two years and in top leagues were included. In details, each game contained result (win or lose), game time, 10 champions picked by two teams and some indicators that measured the performance regarding various tasks for each champion in game. Also, images of the champions were included for easier recognition. Finally, totally 22770 rows (equivalent to 2277 games since 10 rows for a game) and 13 columns (image was excluded) were obtained to form the original data set.

Figure 2 shows the instance of the original data set. There was two teams that fought against in a game. The interpretation for each attribute is in the following:

- **GameID**: a unique code for each game.
- **Game time**: game duration in minute.
- **Result**: win or loss.
- **Champions**: each champion has a unique name and photo.
- **Role**: a team is made up of Top, Jungle, Mid, ADC, Support. In a team, each champion plays a role taking some specific tasks.
- **Others**: indicators that measures the performance for some tasks in game in minute.

GameID	Game time	Result	Champion	Photo	Role	KillsPM	DeathsPM	AssistsPM	CSM	GPM	DPM	HPM	DTPM
23705	35.40	WIN	Ornn		TOP	0.028249	0.056497	0.225989	8.2	377	387	166.016949	682.627119
			Olaf		JUNGLE	0.056497	0.028249	0.141243	6.0	330	281	441.412429	808.813559
			LeBlanc		MID	0.084746	0.000000	0.225989	9.2	413	516	326.666667	567.288136
			Miss Fortune		ADC	0.169492	0.028249	0.112994	10.4	474	441	19.067797	253.672316
			Yuumi		SUPPORT	0.000000	0.000000	0.282486	0.6	221	215	335.423729	87.146893
		LOSS	Sett		TOP	0.056497	0.084746	0.028249	9.3	408	353	82.627119	823.418079
			Lee Sin		JUNGLE	0.028249	0.084746	0.056497	4.8	280	128	231.723164	684.604520
			Lissandra		MID	0.000000	0.028249	0.084746	8.6	347	440	151.045198	507.853107
			Aphelios		ADC	0.028249	0.084746	0.028249	9.8	402	597	62.598870	351.525424
			Thresh		SUPPORT	0.000000	0.056497	0.056497	0.9	196	75	35.988701	281.412429

Figure 2. Preview of the Original Data

Apart from the description, some basic data visualization were implemented to understand the inside information. In Figure 3, the spider plots shows the difference the external difference between two groups and the internal difference among all roles within a group within game 23705. It is easy to claim that not only teams performed differently but also champions did. Even of some indicators like CSM, GPM and DTPM, champions in same roles showed more similar performance while they were opponent. Thereby, it was intended to demonstrate that champions and roles exactly differ for tasks distribution in game. When comparing the similarity of champions, the above indicators could be referred.

Figure 2 shows the top 10 champions with most presence and wining rate, respectively. The bar chart of wining rate filtered those champion with less than 20 presence. Even though the winning rates among all champions did not differ so much, it could be contributed to team evaluation in the recommender system.

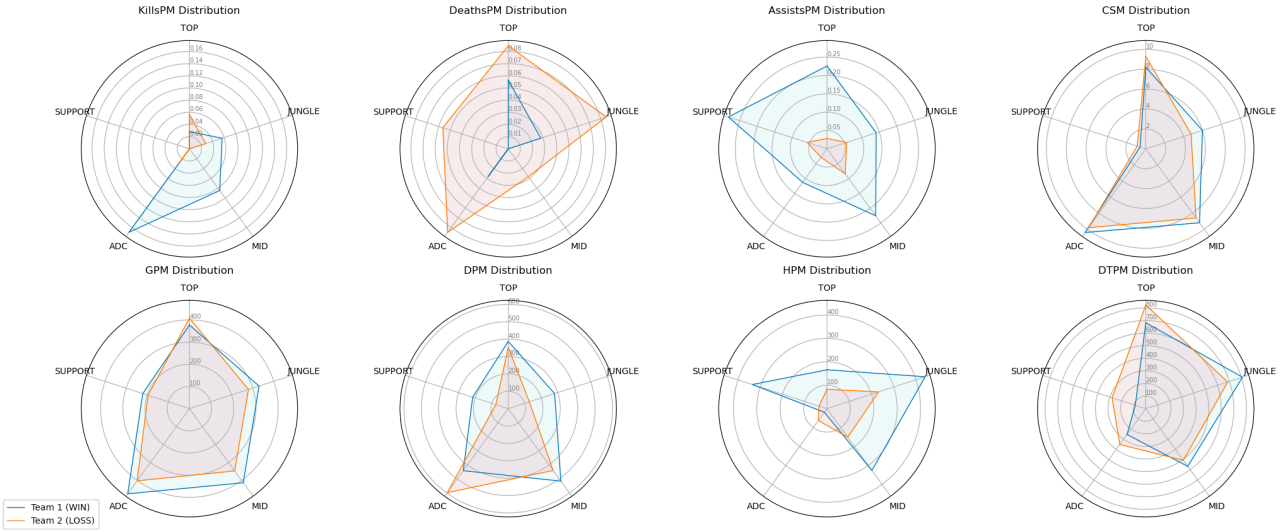


Figure 3. Spider Plots of Indicators in A Game

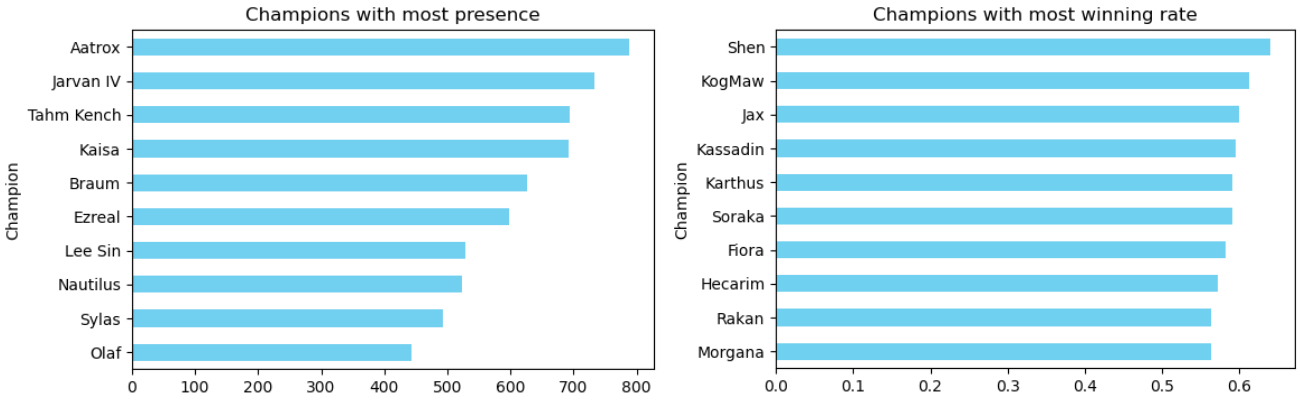


Figure 4. Bar Charts of champions with Most Presence and Wining Rate

3. Data Processing

Apart from the current indicators, it required more indictors to compute the similarity of champions more precisely. Based on the original data set, we could compute the probability of champions being placed on various roles and the wining rate of champions in different stages of games. The first one told what champions are more likely to be placed on same roles, hence, when the recommender was making selection for a role, it would be more possible to pick those champions that frequently appeared on it. The second one could provide an evaluation score for any combination of champions to form a team. Since the champions are so many and the count of duplicated teams in the original data set is few, it is not reasonable to give a score for a team based on the historical presence of it, meanwhile, this approach is not that smart. To overcome the problem, an approach being used was computing wining rate for each champions in a phase of game and combining them for scoring a team.

3.1 Average performance via indicators

Firstly, the average performance of each champion was computed. Average ‘Game time’ amounts to average game time. Average ‘Result’ amounts to average winning rate.

Champion	Game time	Result	KillsPM	DeathsPM	AssistsPM	CSM	GPM	DPM	HPM	DTPM
Aatrox	32.453249	0.500000	0.074642	0.076892	0.125133	7.952284	376.769036	360.276650	328.170685	756.168021
Ahri	31.317500	0.750000	0.032675	0.006482	0.238946	8.450000	368.250000	365.250000	127.629706	346.269217
Akali	33.310619	0.469027	0.108425	0.063172	0.114629	8.439823	388.713864	384.215339	232.132788	564.919651
Alistar	32.690299	0.524457	0.022661	0.094076	0.248639	1.607609	256.130435	116.934783	194.338432	450.023260
Anivia	30.760000	0.666667	0.035891	0.062490	0.078085	9.066667	366.666667	214.000000	83.425731	490.418586

Figure 5. Head of Average Data for Champions

3.2 Probability that being placed on roles

Secondly, the probability for each champion to be placed on each roles was computed. For each champion, the result was the presence of it on each role divided by all presence. According to Figure 6, it can be indicated that, for instance, ‘Aatrox’ never be placed on ADC and Support, while it was placed on Top most time and on Jungle and Mid seldom. Another champion such as ‘Ahri’ must be placed on Mid. If champions have similar distribution of the probability, they are considered as a category of champions.

Champion	ADC_prob	JUNGLE_prob	MID_prob	SUPPORT_prob	TOP_prob
Aatrox	0.0	0.098985	0.083756	0.0	0.817259
Ahri	0.0	0.000000	1.000000	0.0	0.000000
Akali	0.0	0.000000	0.560472	0.0	0.439528
Alistar	0.0	0.000000	0.000000	1.0	0.000000
Anivia	0.0	0.000000	1.000000	0.0	0.000000

Figure 6. Head of Probability Data for Champions

3.3 Winning rate in different stages of game

The game duration can be discretized into five phases including Early (less than 25 min), Early-to-middle (more than 25 min but less than 30), Middle (more than 30 min but less than 35), Middle-to-late (more than 35 min but less than 40) and Late (more than 40). Why wining rate at various phases to be concerned is that champions also differ on strength in different time. For example, some of them may be very strong at the early phase, while they tend to be weak at the late phase due to the balance of game.

For each champion, the result was the wining times at each stage divided by all presence at each stage. In Figure 7, it shows that 'Aatrox' seems to perform better after Middle as the wining rate grows from 0.47 at the earlier stage to more than 0.5 at the later stage.

This computation not only could be used to measure the similarity of champions, but also contributed to score a team later.

Champion	Early	Early-to-middle	Middle	Middle-to-late	Late
Aatrox	0.470588	0.495798	0.498141	0.515924	0.506849
Ahri	0.460788	1.000000	1.000000	0.000000	0.485682
Akali	0.529412	0.382979	0.514286	0.428571	0.586957
Alistar	0.600000	0.583333	0.478632	0.475610	0.555556
Anivia	0.460788	0.500000	0.480454	1.000000	0.485682

Figure 7. Head of Winning Rate Data for Champions

3.4 Merging and filtering

The above results were merged together and dropped those champions out that appeared less than 20 times. The final new data set for champions contained 100 champions and 20 indicators.

4. Similarity among champions

After computing all indicators, the similarity among champions could be implemented. Figure 8 indicates a cluster map using correlation among champions and indicators.

```
data = similar_champions('Sett')
HTML_show(data)
```

Champion	Photo	Similarity
Ornn		0.999863
Kled		0.999015
Poppy		0.998412
Sion		0.997352
Wukong		0.995487

```
data = similar_champions('Ashe')
HTML_show(data)
```

Champion	Photo	Similarity
Caitlyn		0.999721
Sivir		0.999255
Jinx		0.999173
Varus		0.999055
Orianna		0.998103

Figure 9. Top 5 Similar Champions for 'Sett' and 'Ashe'

In terms of the relationship among different indicators, the horizontal hierarchical tree describes that, for example, ‘AssistPM’ and ‘SUPPORT_prob’ are highly correlated which means Support was more likely to give assistance in game. Additionally, ‘Result' is highly correlated to ‘Early-to-Middle' and ‘Middle’ which means most games was wined prior to the middle stage of game.

In terms of the relationship among champions, instead of showing by cluster map, a function was defined to input a champion and output the top N champions that were similar to it. Figure 9 shows a demo regarding this function when input champions, ’Sett’ and ‘Ashe’.

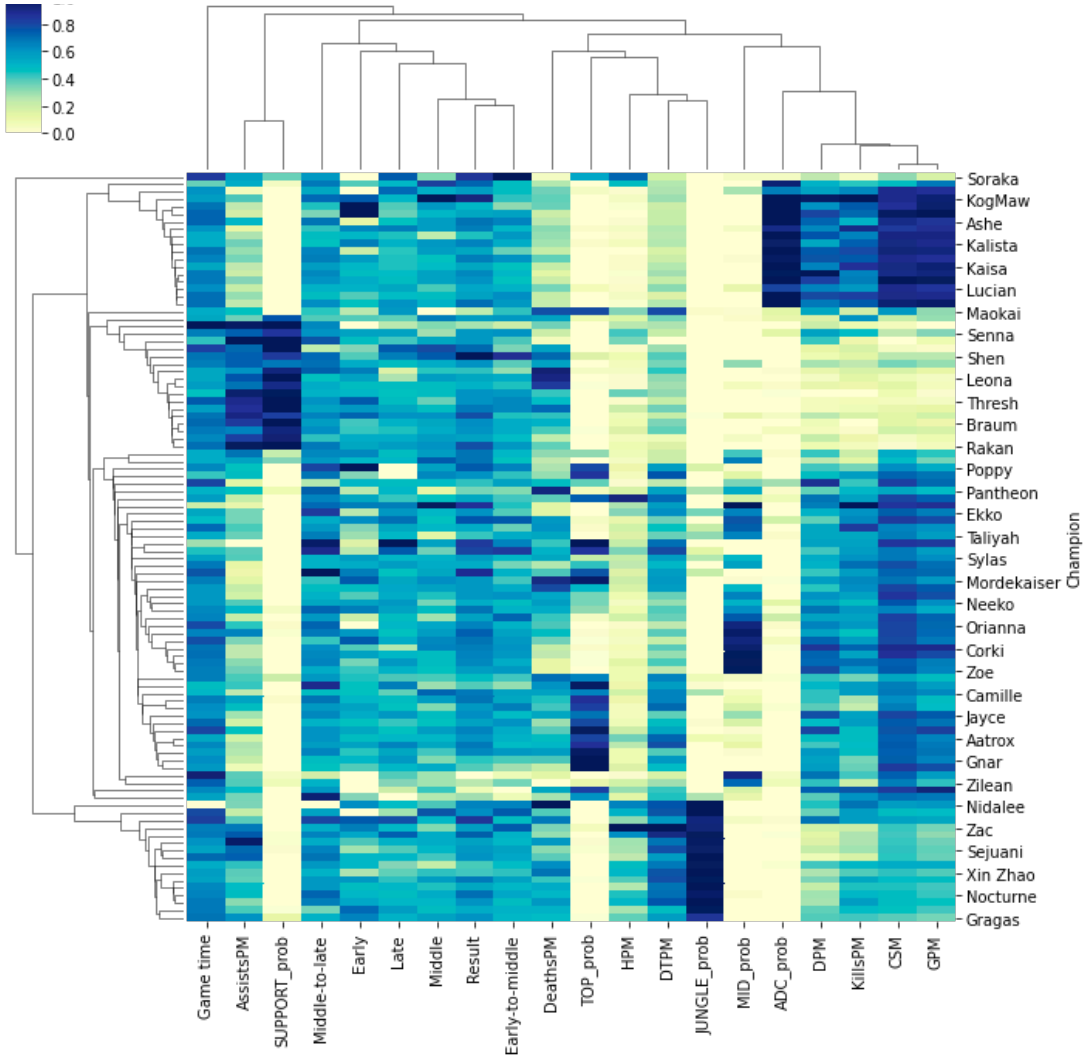
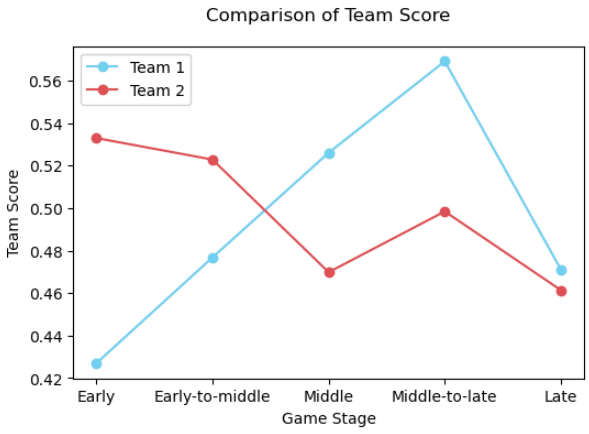


Figure 8. Cluster Map of Merged Champions Data

5. Team score

In the previous part, the winning rate in five phases of game for all champions has been computed. Thereby, given a team, a team score could always be deduced accordingly. Figure 9 shows an example. If inputting a team [‘Ornn’, ‘Olaf’, ‘LeBlanc’, ‘Miss Fortune’, ‘Yuumi’] and the other team [‘Sett’, ‘Lee Sin’, ‘Lissandra’, ‘Aphelios’, ‘Thresh’], a comparison between two teams was plotted in the left by averaging five champions’ real time wining rate in the same team. Besides, a table including all details of was shown in the right. As we can see, Team 2 was intended to be stronger than Team 1 before Early-to-Middle, whereas Team 1 reserved the situation when the game becomes longer.



Team	Champion	Photo	Early	Early-to-middle	Middle	Middle-to-late	Late
Team 1	Ornn		0.500000	0.523256	0.542056	0.578947	0.463415
	Olaf		0.600000	0.531250	0.539568	0.510000	0.439024
	LeBlanc		0.350000	0.506329	0.478632	0.555556	0.600000
	Miss Fortune		0.214286	0.397727	0.545455	0.550725	0.431818
	Yuumi		0.470588	0.425287	0.524390	0.650000	0.421053
Team 2	Sett		0.444444	0.433333	0.443038	0.500000	0.407407
	Lee Sin		0.461538	0.485915	0.463277	0.500000	0.493151
	Lissandra		0.608696	0.533333	0.544000	0.574713	0.457143
	Aphelios		0.400000	0.578125	0.479452	0.403846	0.484848
	Thresh		0.750000	0.583333	0.419355	0.513514	0.464286

Figure 9. Comparison of Team Score

6. Detection of counters

So far we have already known champions are different regarding the performance in game and the probability to win at various stages of game. Moreover, countering relationship among all champions is also explicit due to the design of game. Some champions were designed to counter another ones. Based on this data set, it could be computed that for a champion how likely that others beat it. In details, we could count the games in which a champion fought against each of others, then the probability to be beat by them could be obtained. A champion countered another one, if it had a very high probability to win when they fought against.

```
counters = find_counters('Sett')
HTML_show(counters)
```

Champion	Photo	Counter Rate
Trundle		0.800000
LeBlanc		0.760000
Taric		0.652174
Varus		0.644444
Jarvan IV		0.590909

```
counters = find_counters('Ashe')
HTML_show(counters)
```

Champion	Photo	Counter Rate
Sylas		0.761905
RekSai		0.714286
Kennen		0.666667
Varus		0.600000
Galio		0.560000

Figure 10. Top 5 Counters for ‘Sett’ and ‘Ashe’

Similar to Section 4, Figure 10 shows a demo after inputting champions, ‘Sett’ and ‘Ashe’. For instance, 80% of game in which ’Trundle’ fought against ‘Sett’ was won by ‘Trundle’ in the past, therefore, ‘Trundle’ was absolutely the best counter of ‘Sett’. By the way, this function also did not include it if a champion fought against someone less than 20 times, so some champions might have rare counters even no counters.

7. Recommender

Back to our main purpose, a recommender system has been possible to be conducted based on the previous computation. The recommender aimed to output a complete teams (five different champions corresponding to Top, Jungle, Mid, ADC, Support roles) after inputting a subset of them or none. Note that there was a confusion if we directly retrieved teams that satisfied the condition from the original data, it was possible to output nothing especially when 3-4 champions were input. There were totally 4554 teams in the original data set while 4445 of them were unique. Besides, there were 100 usual champions but the permutations of them were much larger than 4445.

The other confusion was that, even though the team score could be contributed to evaluate teams better, the recommended champions would constantly be the ones with the highest wining rate on their roles. Generally, it was not reasonable to merge the top champions on their roles to form a recommended team, because champions were not independent.

To guarantee more teams that could be recommended and retain the dependence among champions, with the assistance of similarity among champions, a recommender was built in the following algorithm. Figure 11 shows STEP 3 to STEP 6 in an example for better understanding, as Top is ‘Sett’ and ADC is ‘Ashe’ and the recommender should recommend the champions on else roles.

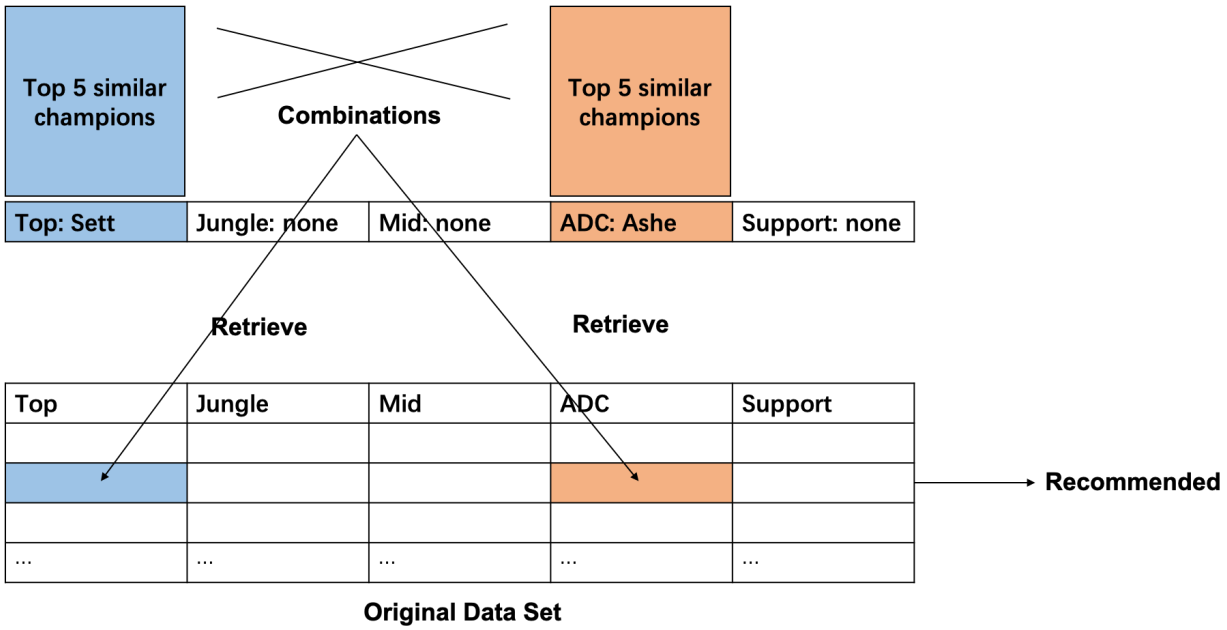


Figure 11. Graph Describing STEP 3 to STEP 6

Algorithm

STEP 1: Input none or a subset of champions with specified roles, and the evaluating method: Early score, Early-to-middle score, Middle score, Middle-to-late score or Late score.
STEP 2: If input is none, then output the best N teams that existed in the original data set based on the given evaluating method.
STEP 3: If input is a subset, then compute top 5 similar champions for each one in the subset.
STEP 4: Based on the given roles, find all possible combinations among them as well as the given champions.
STEP 5: For each combination, retrieve teams that contain it in the original data set. Iteratively, forms a list of recommended teams.
STEP 6: In the list, replace champions on the given roles with the given champions correspondingly.
STEP 7: Sort all teams in the list based on the given evaluating method.
STEP 8: Output the best N teams.

```
top=False
jun=False
mid=False
adc='Ashe'
sup='Yuumi'
recom_team = recommender(top,jun,mid,adc,sup,num=3)
HTML_show2(recom_team)
```

	TOP	JUNGLE	MID	ADC	SUPPORT	Middle Score	Photo
0	Ornn	Lee Sin	Kassadin	Ashe	Yuumi	0.567791	
1	Kled	Jarvan IV	Corki	Ashe	Yuumi	0.547773	
2	Mordekaiser	Jarvan IV	Syndra	Ashe	Yuumi	0.544987	

```
top=False
jun=False
mid=False
adc='Varus'
sup=False
recom_team = recommender(top,jun,mid,adc,sup,num=3, expect_stage='Late')
HTML_show2(recom_team)
```

	TOP	JUNGLE	MID	ADC	SUPPORT	Late Score	Photo
0	Fiora	Karthus	Jayce	Varus	Tahm Kench	0.655661	
1	Fiora	Karthus	Azir	Varus	Tahm Kench	0.635661	
2	Kennen	Karthus	Zed	Varus	Blitzcrank	0.634520	

Figure 12. Demo of Recommender

Figure 12 shows the demo of this recommender. The first example was given ‘Ashe’ on ADC’, ’Yuumi’ on Support and score in Middle as the evaluation method. The top 3 recommended teams were output in a table. The other example displayed the output given ‘Varus’ on ADC and score in Late was concerned.

So far a recommender has been completed but it could be improved more with something has been done before. The recommender was capable of telling users how to form a team better, but not considering about what else champions could replaced for each of the current champions and what else champions should be

```
merged_recommender(adc='Ashe', sup='Yuumi', num_team=3,num_counter=2, expect_stage='Middle')
```

Middle Score		Recommended team	Similar Champions	Counters & Counter Rate
	TOP	Ornn	[Sett, Poppy, Kled]	[(RekSai, 0.63), (Senna, 0.62)]
	JUNGLE	Lee Sin	[Sejuani, Jarvan IV, Skarner]	[(Karma, 0.7), (Ornn, 0.68)]
0.567791	MID	Kassadin	[Qiyana, Akali, Mordekaiser]	[]
	ADC	Ashe	[Caitlyn, Sivir, Jinx]	[(Sylas, 0.76), (RekSai, 0.71)]
	SUPPORT	Yuumi	[Senna, Sona, Kayle]	[(Tahm Kench, 0.68), (Sylas, 0.62)]
	TOP	Kled	[Sett, Sion, Ornn]	[]
	JUNGLE	Jarvan IV	[Lee Sin, Sejuani, Skarner]	[(Vladimir, 0.69), (Ashe, 0.68)]
0.547773	MID	Corki	[Jayce, Azir, Orianna]	[(Galio, 0.64), (Jayce, 0.62)]
	ADC	Ashe	[Caitlyn, Sivir, Jinx]	[(Sylas, 0.76), (RekSai, 0.71)]
	SUPPORT	Yuumi	[Senna, Sona, Kayle]	[(Tahm Kench, 0.68), (Sylas, 0.62)]
	TOP	Mordekaiser	[Sylas, Fiora, Kassadin]	[(Renekton, 0.59), (Ornn, 0.57)]
	JUNGLE	Jarvan IV	[Lee Sin, Sejuani, Skarner]	[(Vladimir, 0.69), (Ashe, 0.68)]
0.544987	MID	Syndra	[Caitlyn, VelKoz, KogMaw]	[(LeBlanc, 0.71), (Ryze, 0.63)]
	ADC	Ashe	[Caitlyn, Sivir, Jinx]	[(Sylas, 0.76), (RekSai, 0.71)]
	SUPPORT	Yuumi	[Senna, Sona, Kayle]	[(Tahm Kench, 0.68), (Sylas, 0.62)]

The first recommended team

The second recommended team

The third recommended team

banned due to the countering relationship.

Figure 13. Demo of Merged Recommender

A merged recommender system was implemented and shown a demo in Figure 13.

The input was kept constant as one of the previous example as ‘Ashe’ on ADC and ‘Yuumi’ on Support’ and score in Middle. Different to the output of the previous one, this output additionally consisted of the top 3 similar champions as well as the top 2 counters for each one in the recommended teams. Due to the limitation of banned champions in a game, it is more appropriate for players to ban the most threatening counters so the counter rate were also given. As the figure shows, for the first recommended team of ‘Ornn’, ‘Lee Sin’, ‘Kassadin’, ‘Ashe’ and ‘Yuumi', the most threatening champion can be considered as ‘Sylas’, with the probability of 0.76 to counter ‘Ashe’. Thereby, it was reasonable to ban it. By the way, those empty output indicated there was no obvious counters for the champion due to the lack of presence.

As we know, a threatening can be banned by a team so it is possible that the intended champions which a team plan to pick are banned by the opponent. At this time, those similar champions output by the merged

recommender can be considered as replacement. In this case, if ‘Orrn’ was banned by the opponent before it was picked, then ‘Sett’, ‘Poppy’ and ‘Kled’ could be alternative for ‘Orrn’ based on the highest similarity.

8. Conclusion

This project purposed a recommender system to help players make decision regarding picking and banning champions prior to battle. Even though the original data set was not very all-round to provide all kinds of information, the recommender was capable of making selection based on the average performance of all champions, the similarity among them, the winning rate of them in various stages of game and the countering relationship among them with some computation. It not only provided some suggestion to complete a team, but also pointed out the alternative plans and the threat that should be concerned.

9. Reference

- Games of Legend: https://gol.gg/esports/home/
- Building a League of Legends team recommender in Go: https://towardsdatascience.com/building-a-league-of-legends-champions-recommender-system-in-go-and-how-to-deploy-it-in-the-cloud-1ee7a4fb55ee