Champion Recommender System For League of Legends

Young Seok Kim
University of Washington
`tonykim7@uw.edu`

Pradeep Prabhakar
University of Washington
`prdp1992@uw.edu`

Aniruddha Dutta
University of Washington
`adutta29@uw.edu`

Abstract

Drafting in *Multiplayer Online Battle Arena* (MOBA) has inherent complexity and difficulties. In this project we propose a recommendation system for selecting champions in a popular MOBA game called *League of Legends*. Specifically, we propose an efficient algorithm to compute unbiased *synergy* and *counter* relationships that are normalized by popularity to capture the dynamic relationships between various champions. We then build a recommendation system by fully utilizing these *synergy* and *counter* relationships. Finally, in order to evaluate our recommended champions, we propose novel metrics called *Composite Win Rate* (CWR) and *Upper-hand* which is consistent with the recommendation objective while capturing small differences between each team's *composite win rate*. Our recommendation system achieved 80.79% average *Upper-hand* using the *sample threshold* of 20 matches.

1 Introduction

Multiplayer Online Battle Arena (MOBA) games refer to a sub-genre of real-time strategy games wherein two teams compete against each other, with an objective to destroy the opponent's base structure. A typical game is played between two teams of 5 players, where every player selects one champion/hero from a pool of over 100+ characters. Every champion has different abilities, strengths and roles. Two major aspects of the game that dominate the outcome of the match are the drafting phase (where players choose a champion) and the in-game mechanics. The widespread popularity and competitive nature of two games in particular - *DOTA 2* and *League of Legends*, have attracted a variety of research, with hero recommendation systems being one of them. For this project, we propose an algorithm to efficiently mine synergy and counter rules and build a subsequent recommender system, to recommend champions for a team in *League of Legends* during the drafting phase that maximizes the *composite win rate*.

2 Related Work

Previous work on hero-recommender systems were mostly based on two approaches - association rule mining based on selection frequencies, and historical win rates.

Hanke and Chaimowicz (2017) proposed a recommender system for Hero line-ups by mining association rules from historical line-ups and match results from DOTA2. They used Apriori algorithm to extract the rules that satisfy some minimum *support* and minimum *confidence* thresholds. The rules here are the frequent hero subsets that appear together either in the winning team (as allies) or in opposite teams (as counters) thereby corresponding to two sets of recommendations for each pick during the draft. Since their approach is based on frequency of champions co-occurring often either as an ally or counter (popularity), the method may not necessarily take into consideration the association that would maximize the win rates of the recommended picks.

Chen et. al proposed a Monte Carlo Tree Search (MCTS) based model for estimating the optimal values of hero combinations, where they modeled the drafting between two teams as a combinatorial game. The authors approximated the evaluation by using another model that predicts the win-rate of specific 5vs5 matchup. The authors noted that they did not consider hero bans nor the ordering of the picks.

3 Problem description

Ultimately, our recommendation system aims to provide champion recommendations during the drafting phase in 5 vs 5 solo rank games in *League of Legends* that maximizes the expected win probability, in which we approximated through *composite win rate*. (Although the same method can be used in other Multiplayer Online Battle Arena games such as *DOTA*).

To describe this problem in detail, we first define the champions set, C, which is composed of the champions that players can play in *League of Legends*. Every month or two, *League of Legends* releases a new patch with some enhancements to the champion abilities. For our analysis, we focus on the patch 10.9 which has 148 champions.

$$\begin{aligned} C \ &= \{Ahri, Ashe, Akali, Alistar, ..., Zyra\} \\ |C| &= 148 \end{aligned}$$

Each of the *champion composition*, h, is represented as a tuple of two sets of five champions. The left side of the tuple is Blue team, and right side of the tuple is Red team. Formally,

$$h = (\{c_1, c_2, c_3, c_4, c_5\}, \{c_6, c_7, c_8, c_9, c_{10}\}) \in H \ where \ c_i \in C for \ i = \{1, ...10\}$$

We find that there exists $|H| = \binom{148}{5} \times \binom{143}{5} \approx 2.566 \times 10^{17}$ *champion compositions*. For each of the *champion composition*, we assume that it is associated with a win probability. Formally, we have a function $P : H \rightarrow (0, 1)$ that measures a win probability (of Blue team) for each *champion composition*.

Our recommendation system will recommend one champion to the next drafting team, given a ***valid*** *incomplete champion composition*. (Here, ***valid*** means that the incomplete composition that can happen during drafting.) Note that in the drafting process, two teams alternate to pick heroes in a "1-2-2-2-2-1" order, during which heroes already selected are visible to both teams.

For example, assume that we have a *valid incomplete champion composition*, $\bar{h}$.

$$\bar{h} = (\{Ahri\}, \{Pantheon, Zyra\})$$

Then, we recommend a champion on Blue team (left side) that would make a good synergy with champion, *Ahri*, while counters *Pantheon* and/or *Zrya*. These *synergies* and *counters* are defined and described in more detail in Section 6.1, while maximizing the joint objective as a recommendation system is described further in Section 6.2.

3.1 System components

Our recommendation system is composed of three different technical components.

1. Mining synergy and counter relationships
2. Champion recommendation system that utilizes synergy and counter relationships
3. Evaluation scheme to measure how well our recommendations behave on the test set

First, we propose an alternative to association rules for mining synergy and counter relationships and an efficient algorithm to compute them. Second, our champion recommendation system recommends champions that maximizes the *composite win rate*. Lastly, we propose intuitive metrics to measure how our recommended teams behave on the test set.

4 Technical contribution

- To the best of our knowledge, this is the first champion recommendation system for *League of Legends*.

- For mining synergy and counter rules, our approach corrects the bias for popularity through normalization.
- We propose an efficient algorithm to calculate synergy and counter rules.
- Our recommendation system fully utilizes the metrics mined from synergy and counter rules.
- We propose a novel and efficient evaluation scheme that is consistent with the recommendation objective to measure how the recommended teams behave on the test set.

5 Data collection

Riot games, who developed the game *League of Legends*, provides an official API to developers and researchers to collect data from their servers.

5.1 Riot API

We used the following APIs.

- `/lol/match/v4/matches/{matchId}`
- `/lol/match/v4/matchlists/by-account/{encryptedAccountId}`
- `/lol/summoner/v4/summoners/by-account/{encryptedAccountId}`

The first API fetches information about the specific match that are uniquely identified by the `matchId`. The second API fetches a list of matches for a specific account, identified by the `encryptedAccountId`. The third API fetches a tier information for a specific player. We were particularly interested in 5 vs 5 solo rank queue type since it is the most famous one in league of legends and it is the queue type that we are considering in our problem formulation.

5.2 Pre-processing

Each of the API responds with a JSON serialized HTTPS response. We implemented multiple pre-processing scripts to only collect a subset of the information in the matches data which had picks, bans, spells, role, lane, accountID for all 10 players, and additional metadata information about the match. Finally, we filtered for the most recent patch (10.9) and filtered for the 5 vs 5 solo rank queues.

6 Methodology

In this section, we describe specific approaches and algorithms to the three components of our system.

Definition of matches

Before we dive into methodology, we describe a *match*, m, which we have collected through the Riot official API described in Section 5 as follows.

$$m = (\{c_1, c_2, c_3, c_4, c_5\}, \{c_6, c_7, c_8, c_9, c_{10}\}, b) \in M$$

where b is a boolean value, either *True* or *False* that describes whether Blue team (left side) won or not. Note that although this is very similar to champion composition described in Section 3, it is different because it has extra win/lose information. Also, note that m is not symmetric in terms of which team won because Blue team and Red team is not symmetric in *League of Legends* and we wanted to preserve the asymmetry. Formally,

$$(\{c_1, c_2, c_3, c_4, c_5\}, \{c_6, c_7, c_8, c_9, c_{10}\}, b) \neq (\{c_6, c_7, c_8, c_9, c_{10}\}, \{c_1, c_2, c_3, c_4, c_5\}, \neg b)$$

6.1 Synergy rules and Counter rules

6.1.1 Naive association rule method

We use association rules using the **FP-Growth** algorithm to recommend the champions for a particular line-up in *League of Legends*, similar to *Hanke and Chaimowicz (2017)*.

The algorithm aims to find the rules which satisfy both a minimum support threshold and a minimum confidence threshold. The "support" of the rule $S(I \Rightarrow j)$ is the fraction of observations in the union of items I and j in the complete item set K from which they were derived. The "confidence" of the rule is its support divided by the support of the antecedent I.

$$Confidence(I \Rightarrow j) = \frac{S(I \Rightarrow j)}{S(I)}$$

Since we want to recommend champions based on both allies and opponents composition, we extract two sets of association rules: one for the associations between champions that won together(ally team) and one for the champion that won against opposing champions(opponent team). As we want to recommend champions for every pick turn in the drafting process, we require association rules of champions given the number of champions already selected which ranges from 2 to a maximum of 5 in one team. The first champion is usually selected based on the bans and current patch meta.

For the allies set, we provide only the information of winning teams to the algorithm and extract any association with size ranging from 2 to 5, with a minimum support of $S_{min} = 0.00005$ (at least 2 games).

For the opponents set, we provide the information of both teams to the algorithm and extract associations between champions and their opponents. We extract 2-sized and 3-sized association rules with a minimum support of $S_{min} = 0.00005$. It is highly unlikely for a champion to counter all the opponent picks, so even if we are able to recommend a champion that counters 1-2 opponent picks it will improve the team's chances of wining.

For evaluation, we use these generated association rules on games from the test set to recommend champions and calculate the 'measure of upper-hand' for the recommended team.

6.1.2 Proposed new synergy and counter relationships

Naive association rules using the support threshold would bias the result towards popularity. For example, consider a champion *Miss Fortune*, who is very popular and has a low win rate of 48%, while another champion, *Ahri*,who is less popular, but has a higher win rate of 51%. Then, the naive association rules with a support would bias the result towards the popularity and suggest *Miss Fortune* as a better recommendation. However, since we are more concerned about the win probability, it would be better to suggest *Ahri*.

$$popularity(c_k) := P(c_k \in m[0] \vee c_k \in m[1])$$

With this limitation in mind, we develop new measures $synergy(c_i, c_j)$ and $counter(c_i \rightarrow c_j)$ that are normalized by the popularity, which are also interpretable as conditional win probabilities.

$$\begin{aligned} synergy(c_i, c_j) &:= P((c_i, c_j \in m[0] \wedge m[2] = True) \vee (c_i, c_j \in m[1] \wedge m[2] = False)) \\ &= \frac{|\{m|(c_i, c_j \in m[0] \wedge m[2] = True) \vee (c_i, c_j \in m[1] \wedge m[2] = False)\}|}{|\{m|c_i, c_j \in m[0] \vee c_i, c_j \in m[1]\}|} \\ &= \frac{|\{m|c_i, c_j \in m[0] \wedge m[2] = True\}| + |\{m|c_i, c_j \in m[1] \wedge m[2] = False\}|}{|\{m|(c_i, c_j \in m[0])\}| + |\{m|c_i, c_j \in m[1]\}|} \end{aligned}$$

As we can see from the definition, $synergy(c_i, c_j)$ is a conditional win probability given champion c_i and champion c_j are on the same team. We can also exploit the property that the champions are mutually exclusive to readily calculate this metric in our algorithm. (Two players cannot pick the same champion in *League of Legends* solo ranked games)

$$\begin{aligned} counter(c_i \rightarrow c_j) &:= P((c_i \in m[0] \wedge c_j \in m[1] \wedge m[2] = True) \vee (c_i \in m[1] \wedge c_j \in m[0] \wedge m[2] = False) \\ &= \frac{|\{m|(c_i \in m[0] \wedge c_j \in m[1] \wedge m[2] = True) \vee (c_i \in m[1] \wedge c_j \in m[0] \wedge m[2] = False)\}|}{|\{m|(c_i \in m[0] \wedge c_j \in m[1]) \vee (c_i \in m[1] \wedge c_j \in m[0])\}|} \\ &= \frac{|\{m|(c_i \in m[0] \wedge c_j \in m[1] \wedge m[2] = True)\}| + |\{m|(c_i \in m[1] \wedge c_j \in m[0] \wedge m[2] = False)\}|}{|\{m|(c_i \in m[0] \wedge c_j \in m[1])\}| + |\{m|(c_i \in m[1] \wedge c_j \in m[0])\}|} \end{aligned}$$

Similarly, we define $counter(c_i \rightarrow c_j)$ as a conditional win probability given champion c_i on the reference team and champion c_j on the opponent team.

Efficient algorithm to calculate synergy and counter

In order to efficiently calculate *synergy* and *counter* for all the champions, we propose an algorithm to calculate the conditional probabilities. We exploit that the size of our itemset, or the number of champions, $|C|$, is not quite large, compared to the number of matches, n, and assume that $O(|C|^2)$ fits in memory.

Algorithm 1: Algorithm to calculate synergy and counter

```
Result: Write here the result
initialize $\mathbf{S}, \mathbf{T_s}, \mathbf{C}, \mathbf{T_c} \in \mathbf{0}^{|C|\times|C|}$ ;
foreach $m \in M$ do
    foreach $c_a, c_b \in C^m_{winner}$ do
        if $a == b$ then
            continue
        end
        S[a][b] += 1
    end
end
foreach $C_{team} \in \{c^m_{winner}|m \in M\} \cup \{c^m_{loser}|m \in M\}$ do
    foreach $c_a, c_b \in C_{team}$ do
        if $a == b$ then
            continue
        end
        $\mathbf{T}_s$[a][b] += 1
    end
end
foreach $m \in M$ do
    foreach $(c_a, c_b) \in C^m_{winner} \times C^m_{loser}$ do
        C[a][b] += 1
        $\mathbf{T}_c$[a][b] += 1
    end
end
foreach $m \in M$ do
    foreach $(c_a, c_b) \in C^m_{loser} \times C^m_{winner}$ do
        $\mathbf{T}_c$[a][b] += 1
    end
end
return $\mathbf{S}/\mathbf{T_s}$, $\mathbf{C}/\mathbf{T_c}$
```

This algorithm essentially runs in $O(n)$ time where n is the number of matches. This is because each of the inner *for* loop can be considered as a constant because $\binom{5}{2} = 10$ and $5 \times 5 = 25$. This inner *for* loop only scales with the number of players within a team. However, this is fixed as 5 in *League of Legends*. The space (memory) complexity of this algorithm is $O(|C|^2)$ where C is the total champions set.

6.2 Champion recommendation system

Approach 1 - Equally weighted sum

Using the *synergy* rules and *counter* rules described and computed in Section 6.1, we try to optimize the following objective in order to recommend a champion, given a *valid incomplete champion composition*, $h = \{Allies, Opponents\}$

$$\underset{\substack{c \in C \\ c \notin Allies \\ c \notin Opponents \\ c \notin Bans}}{argmax} \left(\sum_{a \in Allies} synergy(c, a) + \sum_{o \in Opponents} counter(c, o) \right) \tag{1}$$

This distinguishes from the previous work by *Hanke et al.* where the authors chose a random champion from a set of champion pools suggested by the association rules while our recommendation system utilizes the actual rule metrics - *synergy* and *counter*. This objective function is also interpretable as choosing a champion that has the highest average conditional win rate if we include the normalization term. (The normalization term is excluded in the equation 1 above for simplicity. However, the argmax result is equivalent). Here is the equivalent formulation that is interpretable.

Approach 2 - Weighted average by sample size

We observed that the *synergies* or *counters* sometimes have outliers (extremely low win probability or extremely high win probability) due to small sample size. We wanted our objective to account for this sample size, so we have formulated the following weighted average approach, which is similar to Approach 1, but weighted by the sample size instead of equally taking the mean.

$$\underset{\substack{c \in C \\ c \notin Allies \\ c \notin Opponents \\ c \notin Bans}}{argmax} \left(\frac{\sum_{a \in Allies} synergy(c,a) \times \mathbf{T}_S(c,a) + \sum_{o \in Opponents} counter(c,o) \times \mathbf{T}_C(c,o)}{\sum_{a \in Allies} \mathbf{T}_S(c,a) + \sum_{o \in Opponents} \mathbf{T}_C(c,o)} \right)$$

6.3 Evaluation

6.3.1 Supervised learning for match result prediction

One way to evaluate our proposed recommendation for the champion lineup (blue team) is to predict whether the lineup has a higher chance of winning the match compared to the red team. In order to build this prediction model, we perform supervised learning wherein we make use of a function approximator to train the model, an approach similar to *Hanke and Chaimowicz* (2017).

Given we have the champion lineups of the two teams(a pregame feature) and the match result, we train multilayer perceptron (MLP) and gradient boosted tree models to learn the relationship between the champion composition and the match outcome.

For each match, the input value, $I \in \mathcal{R}^{|C|}$ is given by the following equation.

$$I_i := \begin{cases} 1 & \text{if } c_i \in \text{blue} \\ -1 & \text{if } c_i \in \text{red} \\ 0 & \text{otherwise} \end{cases}$$

Similarly, for each match, the output is represented as follows:

$$O := \begin{cases} 1 & \text{if blue won} \\ 0 & \text{otherwise} \end{cases}$$

With this approach, we trained multiple models with different hyper-parameters, but no model achieved a test accuracy of 60% or greater. We observed the same results even after considering games with highly ranked players (high ELO games). This experiment is included in Appendix A.

Based on our experiments, we speculate that this is due to the balanced nature of abilities between the champions in *League of Legends*, as no champion is significantly strengthened or weakened in any given patch. In order to overcome this limitation, we have come up with a set of measures called *upper-hand* and *composite win rate* that help us to capture even the slightest advantage in conditional win rates between the teams. We describe this further in the next subsection. (Section 6.3.2)

6.3.2 Measure of upper-hand based on synergies and counters

composite win rate (CWR)

We propose an alternative approach wherein we first calculate a metric called *composite win rate* (CWR) for both the recommended lineup and the opponent lineup, defined as the average of synergy win rates and counter win rates, formulated as follows:

$$\text{CWR}(A, O) = \frac{\sum\limits_{a1 \in A} \sum\limits_{\substack{a2 \in A \\ a1 \neq a2}} synergy(a1, a2) + \sum\limits_{a \in A} \sum\limits_{o \in O} counters(o, a)}{\binom{|A|}{2} + |A| \times |O|}$$

The *composite win rate* measure captures how well the the champions in a given lineup play with each other and how well they play against the champions in the opponent team. A and O here refers to Allies and Opponents, respectively.

An interesting note on *composite win rate* is that we can view our objective for recommendation in equation 1 as a greedy objective to achieve higher overall CWR. In other words, our algorithm described in Section 6.2 is greedily optimizing the CWR of the allies team. We can derive the term in 1 by subtracting the CWR of the champion composition before the recommendation from the CWR of the champion composition with the recommended champion.

upper-hand

Given we have the *composite win rate* for each team, we define a metric called *upper-hand*, an indicator variable that indicates whether the recommended team's *composite win rate* is higher than the opponent team's *composite win rate*, defined as follows:

$$\text{upper-hand}(A, O) = \begin{cases} 1, & \text{if } CWR(A, O) > CWR(O, A) \\ 0, & \text{if } CWR(A, O) \leq CWR(O, A) \end{cases}$$

While the recommendation is based on the *synergy* and *counter* mined through the training data, the *composite win rate* is calculated based on the *synergy* and *counter* mined through the test data, and so would be a good indicator of the performance of the recommended lineup. The process flow is as follows:

1. Split the data into train and test.
2. Calculate the synergy matrix and counter matrix for both the train and the test data
3. Pick a random champion for the opponent team. We pick a champion for the allies team based on the method described in Section 6.2.
4. Repeat (3) for each champion in the lineups
5. For evaluation, calculate the *CWR* for each team **based on the test data** and assign 1 as upper-hand to the allies if it has a higher composite win rate and 0 otherwise
6. Repeat steps (3) through (5) to get the average upper-hand for the allies team

7 Experimental Results

7.1 Synergy and Counter rules

7.1.1 Using association rules with *FP-Growth* algorithm

Using the naive approach described in Section 6.1.1, we are able to generate association rules for ally synergy and counters using FP-Growth algorithm. We sort them by lift in descending order.

antecedent	consequent	confidence	lift
[Leona, Nautilus]	[Karthus]	0.6666	59.9599
[Hecarim, Rakan]	[Xayah]	0.75	26.9301
[Taric, Zyra]	[MasterYi]	1.0	25.1952
[Syndra, Leona, Ashe]	[Olaf]	0.8	22.9892
[Rakan, Cassiopeia]	[Xayah]	0.5555	19.9482

Table 1: Synergy association rules mined with *FP-growth* algorithm

antecedent	consequent	confidence	lift
[Nautilus, Graves, MissFortune]	[Chogath]	0.2	50.9047
[Sivir, Shyvana]	[Zac]	0.2105	24.0147
[Illaoi, Aphelios]	[Ashe]	0.3333	23.7103
[Mordekaiser, Nami, Ezreal]	[Jax]	0.2	19.4363

Table 2: Counter association rules mined with *FP-growth* algorithm

Now considering that in a draft, we want to recommend champions for the 3rd pick turn, given we have information on 2 allies and 2 enemy picks. Assuming the ally picks = [Taric, Zyra] and enemy picks = [Illaoi, Aphelios], we can recommend the following champions: [MasterYi, Ashe]

To evaluate the recommendation system, we recommend champions for team blue using the team red picks from the test data-set and calculate the average *upper-hand* for recommended team. We observed that the recommended team had *upper-hand* in **53%** of the matches in the test dataset.

7.1.2 Using proposed new synergy-counter mining algorithm

Using a proposed method and algorithm described in Section 6.1.2, we find synergy rules and counter rules for all 148×148 scenarios. Table 3 shows the top 5 synergy rules in terms of the win rate.

Champion 1	**Champion 2**	**win rate (%)**	**Win count**	**Total Count**
Fizz	Ivern	86.36	19	22
Janna	Kennen	81.48	22	27
Ornn	Xin Zhao	79.17	19	24
Elise	Graves	78.26	18	23
Amumu	Zilean	76.92	20	26

Table 3: Top 5 Synergy rules mined with the proposed algorithm using sample threshold 20

This means that based on the matches in our train set, when champion *Fizz* and champion *Ivern* is on the same team, that team won 86.36% of the time. Here, we report the top 5 synergies for champions that played a minimum of 20 matches together. However, this *sample threshold* can be adjusted depending on which method we use for the recommendation system. We also report the result based on varying this threshold in future section.

Ally Champion	**Opponent Champion**	**win rate (%)**	**Win count**	**Total Count**
Qiyana	Tryndamere	81.82	18	22
Azir	Malzahar	80.95	17	21
Qiyana	Shen	80.00	16	20
Kalista	Warwick	79.31	23	29
Nunu and Willump	Cho'Gath	79.17	19	24

Table 4: Top 5 Couter rules mined with the proposed algorithm using sample threshold 20

Here, when champion *Qiyana* is on one team and champion *Tryndamere* is on the another team, the team with champion *Qiyana* won 81.82% of the time among the matches in the training data. Again, we report the top 5 counters for champions that played a minimum of 20 matches. A minor note is that "Nunu and Willump" is a name of a single champion in *League of Legends*.

7.2 Recommendation System Evaluation

Following the method and metrics described in Section 6.3.2, we report the *upper-hand* metric, by varying the *sample threshold* (minimum number of matches) for each of the two approaches described in Section 6.2 and 6.2. The baseline average upper-hand shown by the orange line is for two randomly-assigned teams, whereas the average upper-hand for the recommended team using the proposed approach is in blue line, which is measured by the procedure described in Section 6.3.2. The graph is read as follows: when the sample threshold applied is 15, the composite win rate for the recommended team is greater than the composite win rate of the opponent team for 70% of the matches, as seen in Figure 1.

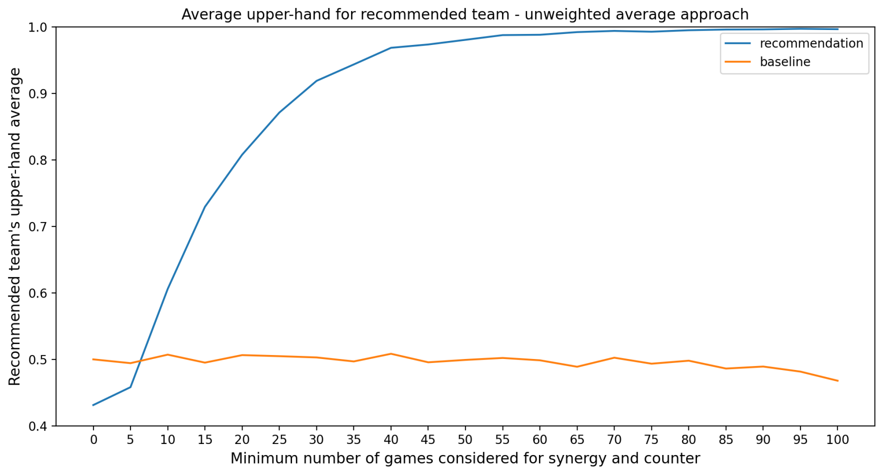

Figure 1: Average Upper-hand using un-weighted average

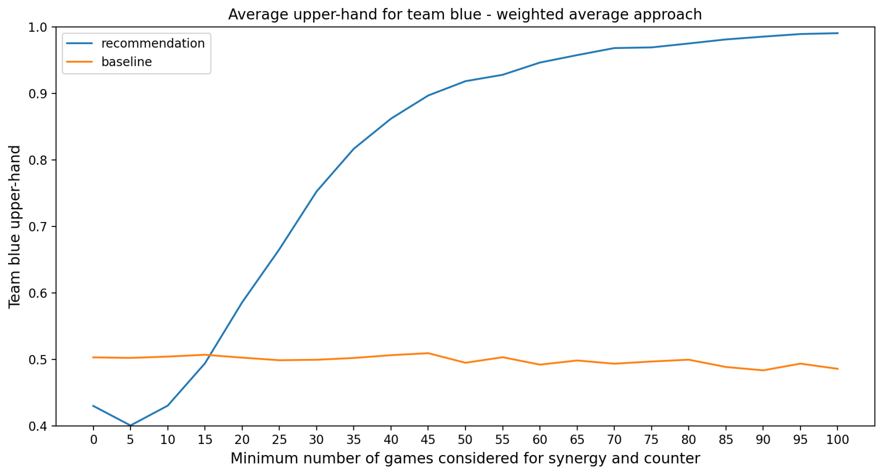

Figure 2: Average Upper-hand using weighted average by sample size

For both approaches, we see that the more restrictive we are in calculating the synergies and counters by increasing the minimum requirement, the higher are the *upper-hand* averages. This could be attributed to the fact that the synergies and counters on both the training and test sets are likely to exhibit similar behaviors with more matches.

However, for higher thresholds, although the *upper-hand* average approaches towards 1, in practice, we would use a threshold that is not too large as that would eliminate a lot of the champions in the training samples in the system and therefore, the system's recommendations would neither be diverse nor flexible.

Since we cannot just use the configuration with the highest *upper-hand* value, it can be easily seen that *upper-hand* metric is not the perfect metric. It does not fully capture the champion relationships (synergies and counters) if we use a higher threshold. However, it does capture the small difference in the CWR metric, which is what we want for our recommendation system.

8 Challenges

Data collection process

Data collection process was an exhausting task due to the rate limit imposed by *Riot Games*. Specifically, *Riot games* allows only 100 requests every 2 minutes and we had to use three different APIs simultaneously in order to collect more matches dataset with the tier information for each of the users. In addition, the API key expires every 24 hours. All these together significantly slowed our data collection process. However, we managed to collect approximately 130,000 games which, after filtering for patch 10.9 came down to 65,847 matches.

Evaluation models

While champion line ups are important in any MOBA game, game prediction based on just the champion lineups as input features, in general, has been a challenging problem, as it is difficult for the models to learn. Despite trying multiple models on the full dataset, we achieved 52% $\sim$ 54% test accuracy. We hypothesised that the accuracies might improve if we only consider games played by high tier players. However, the models still gave similar results. This compelled us to come up with an alternative approach for evaluation, which is also inline with the objective of our recommendation system.

9 Conclusion

For mining association rules, we used *FP-growth* as a naive algorithm and observed that the recommended team had *upper-hand* in 53% of the matches in test set. This is only marginally higher than the baseline upper-hand of 50% and can be attributed to the fact that the recommendations are susceptible to be biased by champion popularity ignoring champion *synergies* and *counters*. The algorithm also restricts the pool of recommended champions to only the most popular ones which are most likely to be banned in high skill-tier matches.

We formulated a new algorithm to compute *synergy* and *counter* matrices, which were utilized by the recommendation system. For evaluation,we formulated new metrics called *composite win rate* and *Upper-hand* as an alternative to win-prediction using supervised learning approach. This approach is consistent with our recommendation system objective while still capturing the subtle *composite win rate* differences between each team. The approach achieved an upper-hand of 80.72% for the recommended team,wherein the synergy and counters were calculated only for the set of champions who played a minimum of 20 matches either as allies or opponents. While the *composite win rate* is one of the ways to evaluate the strength of a team composition, further improvements can be made to the metric in future work. For example, we can modify our CWR metric to have good properties ($metric(A, O) + metric(O, A) = 1$) by adding opponent's synergy term.

$$\frac{\sum_{a1 \in A} \sum_{\substack{a2 \in A \\ a1 \neq a2}} synergy(a1, a2) + \sum_{a \in A} \sum_{o \in O} counters(o, a) + \sum_{o1 \in O} \sum_{\substack{o2 \in O \\ o1 \neq o2}} (1 - synergy(o1, o2))}{\binom{|A|}{2} + |A| \times |O| + \binom{|O|}{2}}$$

Also, it has to be noted that for the experiments, the opponent picks are completely randomized and variations such as restricting the opponent picks within top N counters/synergies for every recommendation can be explored in future work.

Distribution of work

Each of us did similar amount of work on fetching the data and writing the report, and spent significant time on the following activities.

- Young Seok: Formulating the algorithms, evaluation metrics, early prototype implementation of the recommendation system, implementing data fetching script.
- Pradeep: Supervised model implementation, running and coding experiments for evaluation metrics
- Aniruddha: Implementing and formulating approach for FP-growth algorithms, experiments for FP-growth algorithm.

A Appendices

A.1 Supervised models for evaluation

For evaluation, we used neural network and Gradient Boosting method to train the model. For neural network, the results are based on the following configuration.

- Hidden layer: 1
- Number of nodes : 75
- Dropout: 50% for the hidden layer

The below table shows the best results based on test accuracy for the above configuration

Skill level	# Games	Train accuracy %	Test accuracy %
All	65,847	70.79	51.78
Gold +	34,624	77.25	51.52
Platinum +	17,987	83.6	52.36
Diamond +	9,570	90.54	50.84

Table 5: Train-test accuracies of MLP model on various skill levels

For XGBoost classifier, the results are based on the following configuration.

- Number of estimators(trees) : 500
- lambda (l2 regularization parameter) : 0.1
- max tree depth : 6

The below table shows the best results based on test accuracy for the above configuration.

Skill level	# Games	Train accuracy %	Test accuracy %
All	65,847	55.98	53.13
Gold +	34,624	56.91	53.37
Platinum +	17,987	59.3	53.31
Diamond +	9,570	66.9	52.25

Table 6: Train-test accuracies of XGBoost model on various skill levels

As observed above, both MLP and XGboost models failed to generalize well on the test data across the skill levels and our hypothesis that the accuracies might improve for high ELO games did not hold, as the models were not able to learn the outcomes just based on the line ups of the two teams.

References

[1] Hanke, L., & Chaimowicz, L. (2017). A Recommender System for Hero Line-Ups in MOBA Games. AIIDE.

[2] Chen, Z., Nguyen, T.D., Xu, Y., Amato, C., Cooper, S., Sun, Y., & El-Nasr, M.S. (2018). The art of drafting: a team-oriented hero recommendation system for multiplayer online battle arena games. Proceedings of the 12th ACM Conference on Recommender Systems.

[3] Araujo, V., Rios, F., & Parra, D. (2019). Data mining for item recommendation in MOBA games. RecSys ’19.

[4] Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. SIGMOD ’00.