DraftRec: Personalized Draft Recommendation for Winning in Multi-Player Online Battle Arena Games

Hojoon Lee†
joonleesky@kaist.ac.kr
KAIST

Dongyoon Hwang†
godnpeter@kaist.ac.kr
KAIST

Hyunseung Kim
mymsng@kaist.ac.kr
KAIST

Byungkun Lee
byungkun@kaist.ac.kr
KAIST

Jaegul Choo‡
jchoo@kaist.ac.kr
KAIST

ABSTRACT

This paper presents a personalized character recommendation system for Multiplayer Online Battle Arena (MOBA) games which are considered as one of the most popular online video game genres around the world. When playing MOBA games, players go through a draft stage, where they alternately select a virtual character to play. When drafting, players select characters by not only considering their champion preferences, but also the synergy and competence of their team’s character combination. However, the complexity of drafting induces difficulties for beginners to choose the appropriate characters based on the characters of their team while considering their own champion preferences. To alleviate this problem, we propose DraftRec, a novel hierarchical model which recommends characters by considering each player’s champion preferences and the interaction between the players. DraftRec consists of two networks: the player network and the match network. The player network captures the individual player’s champion preference, and the match network integrates the complex relationship between the players and their respective champions. We train and evaluate our model from a manually collected 280,000 matches of Dota2. Empirically, our method achieved state-of-the-art performance in character recommendation and match outcome prediction task. Furthermore, a comprehensive user survey confirms that DraftRec provides convincing and satisfying recommendations. Our code and dataset are available at https://github.com/dojeon-ai/DraftRec.

KEYWORDS

MOBA, League of Legends, Dota2, Draft Recommendation

ACM Reference Format:

Hojoon Lee, Dongyoon Hwang, Hyunseung Kim, Byungkun Lee, and Jaegul Choo. 2022. DraftRec: Personalized Draft Recommendation for Winning in Multi-Player Online Battle Arena Games. In Proceedings of the ACM Web Conference 2022 (WWW ’22), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3485447.3512278

1 INTRODUCTION

Multi-player Online Battle Arena (MOBA) games such as League of Legends and Dota2 are one of the most popular online video games. For example, the annual world championship of League of Legends was reported to have nearly 50 million viewers in 2020 [28].

When playing MOBA games, multiple players participate in a single session, which we refer to as a match. Each match is divided into two stages: draft stage and play stage. In the draft stage, players are split into two teams and alternately select a virtual character (i.e., champion). In the following play stage, players control their selected champions and the first team that destroys the opponent’s main tower wins the game. The draft stage is a crucial component in MOBA games since the strategy of the subsequent play stage largely depends on the champions selected in the draft stage.

Champions in MOBA games vary in terms of their abilities and required player skills. Thus, it is vital to understand how different champions complement each other (i.e., synergy) and how they counter the abilities of the opponent team’s champions (i.e., competence) [9]. However, it is challenging to fully understand the synergy and competence since the number of champion combinations are exponential to the total number of champions. For example, in League of Legends, there currently exist 156 champions, leading to approximately $4.42 \times 10^{17}$ (i.e., $\binom{156}{5} \times \binom{151}{5}$) possible champion combinations for a single match where there are two teams of five players. Therefore, players commonly rely on game analytic web services to access information about the relationship between the champions. For instance, the game analytic web service "op.gg" [1] is accessed by 55 million League of Legends players per month.

To alleviate such difficulties, previous work focused on recommending champions with a high probability of winning by considering the synergy and competence of the champions [7, 49]. However, while matches in MOBA games are composed of various players with different champion preferences, none of these methods take the player’s personal champion preference into consideration. Therefore, in order to build an effective champion recommendation system, it is essential to consider each player’s champion preference and the complex interaction between the players.

To this end, we present DraftRec, a recommender system that suggests champions with a high probability of winning while understanding the champion preference of each player within the match. To reflect each player’s champion preference and then integrate that information, we construct a novel hierarchical architecture composed of two networks: the player network, and the match network. First, the player network captures the individual players’ champion preferences. Then, the match network integrates the complex relationship between the players and their selected champions by integrating the outputs from the player network.

The main contributions of this paper can be summarized as follows: (i) we formalize the personalized draft recommendation
Recommendation systems for MOBA games [7, 43] have also been studied, where [7] recommends champions by leveraging Monte-Carlo Tree Search and [43] recommends play stage strategies to increase the probability of victory for the player. However, our work differs with previous work in that they do not take players’ personal preferences into account.

3 PRELIMINARIES

This section provides background information and a formal problem formulation about the drafting process in MOBA games.

3.1 Draft Process in MOBA Games

Our work focuses on draft processes in competitive modes in MOBA games (e.g., the rank mode for League of Legends and the captain’s mode for Dota2). Since draft processes are similar across different MOBA games, with minor differences, we explain the draft process of MOBA games with an example from League of Legends.

As illustrated in Fig. 1(b), a total of 10 players participate in a single match, where they are divided into two teams: Blue and Purple. Each player is randomly assigned a particular turn (i.e., the order of selecting a champion) and a role (e.g., Top, Jungle, Middle, AD Carry, and Support) as shown in Fig. 1(b) and (d). After assignment, the ban process follows, where each player chooses champions to be illegal to select for the current match (Fig. 1(a)). Players then sequentially select champions (Fig. 1(c)) by considering one’s preferences as well as the synergy and competence of their team’s champion combination. The champion selection process is always conducted in a team-wise alternating order of “1-2-2-2-2-1” (i.e., 1-(2-3)-(4-5)-(6-7)-(8-9)-10 in Fig. 1(b)). In selection, previously selected champions (Fig. 1(c)) are visible to both teams and can no longer be selected by subsequent players within a single match.

Note that the information available to a particular player in the draft stage is partial. That is, as shown in Fig. 1(d) and (e), the teammates’ roles and player IDs are available while the opponents’ are not. “Summoner 1” shown in Fig. 1(e) is an anonymous nickname given to masked player IDs. Using IDs, players commonly search
3.2 Problem Formulation

The entire drafting process illustrated in Section 3.1 can be formalized as follows. Let us denote a single match as $m^i$ (for $1 \leq i \leq M$, where $M$ is the total number of matches in our training data) which is composed of 10 champion selections, one per each player. Players $\{p_1^i, \ldots, p_{10}^i\}$ (where $p_t^i$ represents the player at turn $t$ in match $m^i$) are evenly divided into two teams, Blue $\{p_1^i, p_2^i, \ldots, p_5^i\}$ and Purple $\{p_6^i, p_7^i, \ldots, p_{10}^i\}$ as shown in Fig. 1(b). Here, the ground-truth champion, role, match outcome (e.g., win or lose) and team of $p_t^i$ are respectively defined as $c_t^i, r_t^i, o_t^i$ and $team_t^i$.

The draft stage of MOBA games can be formulated as a variant of the sequential recommendation problem. The typical sequential recommendation problem aimed to predict the player’s most preferred champion (i.e., item) based on their champion interaction history [22, 40]. However, in MOBA games, we have to recommend champions based on not only a single player’s champion selection history but also on the teammates’ champion selection history. Therefore, we aim to encode each player’s champion preference information based on their past champion selection logs as well as their teammates’ selection logs.

Let $H_{p,L}^t$ denote the match history list of player $p_t$’s most recent $L$ matches before match $m^i$. Then, we denote $c_{p_t,L}^i, r_{p_t,L}^i$ and $ftr_{p_t,L}^i$ as the champion, role, and list of in-game features (e.g., number of kills, total earned gold, see supplement for feature design) of the $t$-th match in $H_{p,L}^t$, respectively. Thus, $h_{p,L}^t$ and $H_{p,L}^t$ are defined as

$$h_{p,L}^t = [c_{p_t,L}^i, r_{p_t,L}^i, ftr_{p_t,L}^i]$$

and

$$H_{p,L}^t = [h_{p,L}^1, h_{p,L}^2, \ldots, h_{p,L}^{t-1}, h_{p,L}^t]$$

(1)

where $h_{p,L}^t$ indicates the most recent match history and $h_{p,L}^t$ indicates the oldest match history. In addition, we denote the lists $H_{p,L}^t$ and $R_{team}^t$ of the players in the same team as $H_{team_t}^t$, $R_{team_t}$:

$$H_{team_t,L}^t = [H_{p,L}^t | team_t^k = team_t^j \text{ for } k = 1, \ldots, 10]$$

$$R_{team_t}^t = [r_{p_t}^k | team_t^k = team_t^j \text{ for } k = 1, \ldots, 10]$$

(2)

Then, as described in Section 3.1, for each champion selection, the players can access currently selected champions, their teammates’ match histories, and their respective roles. We denote the set of all observable information for the player $p_t$ at turn $t$ as state $s_t^i$.

$$s_t^i = \{c_{p_t,L}^i, H_{team_t,L}^t, R_{team_t}^t\}$$

(3)

Given $s_t^i$, our recommendation model $f_0$ outputs two different values, $\hat{p}_t$ and $\hat{o}_t$.

$$\hat{p}_t, \hat{o}_t = f_0(s_t^i)$$

(4)

For each output $\hat{p}_t$ and $\hat{o}_t$, the goal of the recommendation model is to accurately predict the ground-truth champion $c_t^i$ and the match output $o_t^i$ respectively.

4 PROPOSED METHOD

This section presents a detailed design of DraftRec. DraftRec exploits a hierarchical architecture with the two Transformer-based networks: the player network and the match network. While the player network focuses on capturing the individual players’ champion preferences, the followed match network integrates the outputs of the player network which will be explained in Sections 4.1-4.2. Then, in Sections 4.3-4.4, we describe the training procedure and the recommendation strategy of DraftRec.

Throughout this section, we simplify the notation by omitting the index of the match. To denote the player, champion, role, team, in-game features, match histories, and set of all partially observable information for a player at turn $t$ for match $m^i$, we denote them as $p_t, c_t, r_t, o_t, team_t, ftr_t, H_{p,L}^o$, and $s_t$, respectively. In addition, notations with a superscript will denote a vector embedding instead.

4.1 Modeling Player’s Dynamic Behavior

This section presents the details of the player network in Fig. 2(a).

**Embedding Layer.** First, we initialize a learnable embedding for the champion $E_C \in \mathbb{R}^{|C| \times d}$, role $E_R \in \mathbb{R}^{|R| \times d}$ and in-game features $E_{FTR} \in \mathbb{R}^{\sum FTR \times d}$ where $d$ indicates the number of hidden units for the embedding. We also initialize a pre-defined sinusoidal positional embedding $E_{pos} \in \mathbb{R}^{|L| \times 1}$ to represent history order.

Given the $l$-th match history $H_{p,L}^o$ of player $p$, the input embedding $H_{p,L}^0$ is constructed by summing up the corresponding champion, role, in-game feature, and position embedding, i.e.,

$$H_{p,L}^0 = E_{pos} + E_{c}^{p} + E_{r}^{p} + E_{ftr}^{p} + E_{pos}$$

(5)

where $E_{c}^{p} \in E_C, E_{r}^{p} \in E_R,$ and $E_{ftr}^{p} \in E_{FTR}$ are $d$-dimensional embeddings for the champion $c_{p,L}$, role $r_{p,L}$, and feature $ftr_{p,L}$ while $E_{pos} \in E_{pos}$ represents the positional embedding for position $l$.

Then, the list of $L$ match history $H_{p,L}$ is embedded as

$$H_{p,L} = [H_{p,L}^0, H_{p,L}^1, \ldots, H_{p,L}^0]$$

(6)

**Player History Encoder.** On top of the embedding layer, we place $N$ Transformer blocks $Trm$, each coupled with a multi-head self-attention network followed by a position-wise feed-forward network [42]. To further ease the difficulty of training, we employed a residual connection, layer normalization, and dropout, following [22, 40]. Through passing the Transformer blocks, the list of the embedded match history $H_{p,L}^0$ is transformed as

$$H_{p,L}^n = Trm(H_{p,L}^{n-1}), \forall n \in [1, \ldots, N]$$

(7)

After $N$ Transformer blocks, the final output $H_{p,L}^N$ encodes the information across match histories. Then, the last position of the output $H_{p,L}^N$ is defined as the player-level representation of a player given his/her match history list.

4.2 Modeling Match Information

This section explains the match network that integrates the outcomes of the player network as illustrated in Fig. 2(b).

**Embedding Layer.** Every match $m^i$ is composed of 10 players $\{p_1^i, \ldots, p_{10}^i\}$, their respective roles $\{r_1, \ldots, r_{10}\}$ and champions $\{c_1, \ldots, c_{10}\}$, corresponding teams $\{team_1, \ldots, team_{10}\}$, and the players’ past teammates’ match histories using game analytic applications to understand teammates’ champion preferences.
match history lists \([H_{p_t}, \ldots, H_{p_{T-1}}]\). At turn \(t\), the match information is partially observable for player \(p_t\), where only the champions selected before current turn \([c_k]\) and the teammates’ match histories \(H_{Team_{k-1}}\) and roles \(role_{k-1}\) are known. Therefore, we replaced the unavailable instances with the "\(\text{[unk]}\)" token. Then, we replaced the champion \(c_t\) with the "\(\text{[mask]}\)" token to indicate the query position. We add an extra learnable embedding \(E_{Team} \in \mathbb{R}^{d \times d}\) to model the players’ team information, and initialize the sinusoidal positional embedding \(E_{Turn} \in \mathbb{R}^{1 \times d}\) to represent the order of the turn.

Afterward, each player’s embedding \(p_k^{(n)}\) at turn \(k\) is constructed by summing up the embedding of the corresponding champion, role, turn and the output of the player network:

\[
p_k^{(n)} = c_k^{(n)} + r_k^{(n)} + \text{team}_k^{(n)} + \text{turn}_k^{(n)} + h_{p_{k,n}}^{N},
\]

where \(c_k^{(n)} \in E_C, r_k^{(n)} \in E_R, \text{team}_k^{(n)} \in E_{Team}\), and \(\text{turn}_k^{(n)} \in E_{Turn}\) are \(d\)-dimensional embeddings for the champion \(c_k\), role \(r_k\), team \(\text{team}_k\), and positional embedding for turn \(k\).

Thus, we represent the list of players in match \(m\) as:

\[
P^m = \{p_1^{(0)}, \ldots, p_0^{(N)}\}, \quad \forall n \in [1, \ldots, N]
\]

After \(N\) Transformer blocks, the final \(d\)-dimensional output \(P^N\) serves as the match representation where the personal histories of the players and available information for each turn of the match are aggregated.

Champion Prediction Head. We now predict the ground-truth champion \(c_t\) based on the hierarchically integrated player representation \(p_k^{N}\) at turn \(t\). In particular, two fully-connected layers with a GELU nonlinear activation is utilized where a softmax layer is followed to produce an output probability distribution over target champions as \(\hat{p}_t \in \mathbb{R}^{|C|}\):

\[
\hat{p}_t = \text{softmax}(\text{GELU}(p_t^j W^P + b^P) E_C^t + b^C) \tag{11}
\]

where \(W^P \in \mathbb{R}^{d \times d}\) is the learnable projection matrix and \(b^P, b^C\) are bias terms. To prevent the banned champions from being recommended, predicted scores of banned champions are masked out.

Match-Outcome Prediction Head. We jointly perform the match outcome prediction by comparing the representations of the two teams. Similar to [10, 13, 14], we obtain a \(d\)-dimensional representation vector of each team \(T_1, T_2\) by applying the average pooling operation for the player representations within each team:

\[
T_1 = \text{pool}(\{p_1^N, \ldots, p_0^N\}), \quad T_2 = \text{pool}(\{p_0^N, \ldots, p_0^N\}) \tag{12}
\]

Then, we subtract the two team representations and apply two fully-connected layers followed by a sigmoid function to obtain the match outcome prediction value \(\hat{v}_t\) such as:

\[
\hat{v}_t = \text{sigmoid}((T_1 - T_2) W^O + b^O) W^V + b^V \tag{13}
\]

where \(W^O \in \mathbb{R}^{d \times d}\) and \(W^V \in \mathbb{R}^{d \times 1}\) are a learnable projection matrix and \(b^O, b^V\) are bias terms. Note that since the match outcome prediction value \(\hat{v}_t\) is squashed to \([0,1]\), it indicates the predicted probability of winning.

4.3 Training DraftRec

This section explains the supervised training procedure of DraftRec. For each training match data \(m\) at turn \(t\), we first get state \(s_t\), which represents all partially observable information in respect to player \(p_t\). Then, as illustrated in Section 4.2, we obtain the predicted champion selection probability and predicted match outcome by forwarding the state \(s_t\) into the DraftRec model \(f_\theta\):

\[
\hat{p}_t, \hat{v}_t = f_\theta(s_t) \tag{14}
\]
Then, the network parameters $\theta$ are trained to maximize the predicted champion selection probability $\hat{p}_t$ of the ground-truth champion $c_t$ while minimizing the error between the predicted match outcome $\hat{q}_t$ and the ground-truth match outcome $q_t$.

Here, we denote $L_p$ as the champion prediction loss and $L_o$ as the match outcome prediction loss. We use binary cross-entropy for both $L_p$ and $L_o$ where L2 weight regularisation is utilized to avoid overfitting. In summary, the loss $L$ for each match is written as

$$L = \frac{1}{T} \sum_{t=1}^{T} \lambda L_p(\hat{p}_t, c_t^*) + (1 - \lambda) L_o(\hat{q}_t, q_t^*) + c||\theta||^2$$

(15)

where $\lambda$ controls the strength between the champion prediction loss and match outcome prediction loss, and $c$ controls the level of the L2 weight regularisation.

### 4.4 Recommendation Strategy

As described in Section 4.3, our model can predict the champion selection probability $\hat{p}_t$ and match outcome $\hat{q}_t$ for a given state $s_t$. Here, we denote the champion recommended by our model as $\hat{c}_t$ and the probability of selecting an arbitrary champion $c$ as $\hat{p}_{t,c}$. To estimate the match outcome of playing the champion $c$ at a given state $s_t$, we first modify the state $s_t$ and fill the "[mask]" token in position $c_t$ with champion $c$. Then, we replace the champion $c_{t+1}$ with a "[mask]" token. Next, we pass the modified state to the model and obtain the predicted match outcome which we denote as $\hat{q}_{t,c}$.

Since we can predict the expected winning probability $\hat{q}_{t,c}$ for all champions $c$, the straightforward recommendation strategy is to recommend the champion with the highest winning probability which already takes the player’s champion preferences into account.

$$\hat{c}_t = \text{argmax}_c \hat{q}_{t,c}$$

(16)

However, a recommendation system which solely rely on the match outcome prediction can exhibit unreliable behaviors. Fig. 3 demonstrates an illustrative example where the blue and red curve represents the distribution of $\hat{p}_{t,c}$ and $\hat{q}_{t,c}$ respectively. Since the states that are unseen within the training data might have arbitrarily inaccurate prediction values [17, 27], the highest match outcome value can be inappropriately assigned to champions which players do not prefer, such as positions indicated at Fig. 3(a).

Therefore, to properly integrate the player’s champion preferences and expected winning probability, our model recommends the champion with the highest match outcome where its selection probability exceeds a threshold value $\tau$ as in Fig 3.(b).

$$\hat{c}_t = \text{argmax}_c \hat{q}_{t,c} \mid \hat{p}_{t,c} > \tau$$

(17)

Recently, this idea of restricting the decision space within the training distribution has been actively studied to deploy safe decision-making systems [12, 25, 46]. In various real-world applications (e.g., recommender systems and robotics), these restrictions have shown reliable and robust results which are also corroborated by our experimental findings in Section 5.5.

### 5 EXPERIMENTS

This section presents the experimental setup, experimental results, and a detailed description of our user study.

#### 5.1 Dataset

To verify the performance of DraftRec, we conduct experiments on two MOBA game datasets: League of Legends (LOL) and Dota2. For each benchmark dataset, we sort the matches by time-stamps and take the first 85% matches as training set, next 5% matches as validation set, and last 10% matches as test set. The statistics for each dataset are given in Table 1.

| Dataset  | Matches | Champions | Players | Avg. Match History |
|----------|---------|-----------|---------|--------------------|
| LOL      | 279,893 | 156       | 62,466  | 66.38              |
| Dota2    | 50,000  | 111       | 140,931 | 2.12               |

#### League of Legends

We manually collected match data for League of Legends utilizing the publicly accessible API endpoint provided by Riot Games [5] and constructed a MOBA game match dataset with rich individual player history. To ensure the quality of each match outcome, matches of the top 0.1% ranked players from June 1, 2021, to September 9, 2021 were collected. Since the purpose of building a draft recommender system is to provide strategically advantageous suggestions, it is natural to train the model with matches from top rank players since they better understand the characteristics of champions compared to low rank players.

#### Dota2

Since previous work on champion recommendation for MOBA games [7] utilized a Dota2 dataset to validate the performance of their recommendation system, we also used a publicly available Dota2 dataset from Kaggle [2], which was collected from November 5, 2015 to November 18, 2015. The dataset contains matches from various ranks and the average match history length of each player is short, as shown in Table 1.

### 5.2 Experimental Setup

**Baselines.** To verify the recommendation performance, we compare our model with the personalized recommendation baselines:

- POP : It is the simplest baseline which ranks item based on the frequency within the given player history.
Table 2: Performance comparison of DraftRec and baselines on champion recommendation. Bold scores indicate the best model and underlined scores indicate the second best. The results are averaged over 10 random seeds.

| Models        | LOL          | Dota2        |
|---------------|--------------|--------------|
|               | HR@1 | HR@5 | NG@5 | HR@10 | NG@10 | HR@1 | HR@5 | NG@5 | HR@10 | NG@10 |
| POP           | 0.3212 | 0.5646 | 0.4497 | 0.6553 | 0.4792 | 0.0508 | 0.1131 | 0.0824 | 0.1647 | 0.0989 |
| NCF [16]      | 0.1376 | 0.4219 | 0.2823 | 0.5900 | 0.3367 | 0.0403 | 0.1384 | 0.0899 | 0.2407 | 0.1226 |
| DMF [48]      | 0.3243 | 0.5758 | 0.4567 | 0.6742 | 0.4887 | 0.0652 | 0.1488 | 0.1030 | 0.2302 | 0.1290 |
| S-POP [19]    | 0.4135 | 0.7404 | 0.5865 | 0.8469 | 0.6213 | 0.0554 | 0.1253 | 0.0901 | 0.1773 | 0.1004 |
| SASRec [22]   | 0.4497 | 0.7430 | 0.6071 | 0.8547 | 0.6368 | 0.0449 | 0.1477 | 0.0951 | 0.2542 | 0.1296 |
| DraftRec      | 0.1456 | 0.3950 | 0.2711 | 0.5675 | 0.3268 | 0.0403 | 0.1390 | 0.0905 | 0.2438 | 0.1239 |
| DraftRec-no-history | **0.5042** | **0.8025** | **0.6646** | **0.8836** | **0.6618** | 0.0434 | **0.1492** | **0.0962** | **0.2547** | **0.1496** |

Notes: "*" indicates the statistical significance (i.e., $p < 0.01$).

- NCF [16]: It captures the nonlinear interactions between players and items through a MLP with implicit feedback.
- DMF [48]: It optimizes the Latent Factor Model based on the explicit item selection ratio of each user.
- S-POP [19]: It is a variant of POP that ranks items based on the player’s most recent n history.
- SASRec [22]: It utilizes the uni-directional Transformer structure for modeling the player’s preference over time.

For the match outcome prediction, we compare our model with:
- Majority Class (MC) [7]: It is the simplest baseline which outputs the majority class (i.e., Blue for LOL and Radiant for Dota2).
- Logistic Regression (LR) [30]: It is a linear classifier with L2 regularization. We use the identical input format as [38].
- Neural Network (NN) [7]: It is the neural network model which follows the architecture and input format as [7].
- HOI [26]: It is based on factorization machines [35] and considers pair-wise interactions between players.
- OptMatch [13]: It exploits graph neural networks to obtain hero embeddings which are used to model players’ champion preferences and proficiency. It further utilizes multi-head self-attention to obtain team representations from the player representations.
- NeuralAC [14]: It is a self-attentive method which explicitly models the synergy and competence of different champions. However, it doesn’t utilize any in-game features or players’ match histories.

**Evaluation.** To evaluate the performance of the champion recommendation and match outcome prediction tasks, we measure the metric of each state $s_t$ at turn $t$ for all matches in the test dataset. Then, we report the average value of each metric.

We employ standard recommendation metrics, Hit Ratio (HR), and Normalized Cumulative Gain (NG), to evaluate the quality of champion recommendations. We report HR and NG with varying the rank $k$ from $\{1, 5, 10\}$. Since NG@1 is identical to HR@1, we omit NG@1. For match outcome prediction task, we consider Accuracy (ACC) and Mean Absolute Error (MAE) as our metrics. For all metrics except MAE, higher value indicates better performance.

5.3 Experimental Results

In this section, we study the performance of all methods on the champion recommendation and match outcome prediction task.

**Champion Recommendation.** Table 2 summarizes the performance of all models on the champion recommendation task. From our experimental results, we can observe the followings: All sequential methods (i.e., S-POP, SASRec, and DraftRec) outperform non-sequential methods (POP, NCF, and DMF) on all metrics except for HR@1 and NG@5 for the Dota2 dataset. This result shows that dynamically modelling players’ preferences improves champion recommendation performance.

Among all models, DraftRec achieved the best recommendation performance for all metrics and datasets except for HR@1 and NG@5 in Dota2. We speculate that this is due to the sparse player history record in the Dota2 dataset, making it difficult to learn the champion preference of each individual player. Compared to SASRec, DraftRec shows an improvement of 12.1% in HR@1, 8% in HR@5, 9.5% in NG@5, 3.4% in HR@10 and 3.9% in NG@10 for League of Legends. For Dota2, DraftRec shows an improvement of 15.4% in NG@10 against SASRec. In addition, when we trained DraftRec without any player match history information, the recommendation performance degrades by a significant margin. This confirms that integrating both player- and match-level representations is crucial in providing personalized champion recommendations.

**Match Outcome Prediction.** Table 3 summarizes the performance of all match outcome prediction baselines. We find that for all datasets, match outcome prediction methods which utilize player
After training DraftRec, we analyzed the model’s attention weights.

### Table 3: Performance comparison of DraftRec and baselines on the match outcome prediction task.

| Models          | LOL ACC | LOL MAE | Dota2 ACC | Dota2 MAE |
|-----------------|---------|---------|-----------|-----------|
| MC [7]          | 0.5040  | 0.4960  | 0.5180    | 0.4820    |
| LR [30]         | 0.5255  | 0.4973  | 0.5750    | 0.4819    |
| NN [7]          | 0.5263  | 0.4975  | 0.5748    | 0.4822    |
| HOI [26]        | 0.5264  | 0.4987  | 0.5716    | 0.4901    |
| OptMatch [13]   | 0.5411  | 0.4944  | 0.5751    | 0.4842    |
| NeuralAC [14]   | 0.5266  | 0.4977  | 0.5739    | 0.4841    |
| DraftRec-no-history | 0.5284  | 0.4942  | 0.5745    | 0.4757*   |
| DraftRec        | 0.5535* | 0.4842* | 0.5755    | 0.4782    |

Notes: "*" indicates the statistical significance (i.e., \( p < 0.01 \)).

match history information (i.e., OptMatch, DraftRec) show superior performance compared to methods which do not (i.e., LR, NN, HOI, NeuralAC, DraftRec-no-history). This demonstrates the importance of integrating the player’s match histories in order to understand the dynamics behind the match outcome between the players.

In addition, our proposed DraftRec outperforms all compared baselines for all metrics and datasets. For League of Legends, DraftRec shows an improvement of 2.3% in ACC and 2.1% in MAE. For Dota2, DraftRec shows an improvement of 1.3% in MAE. Identical to the findings from the champion recommendation task, we observe that utilizing both player-level and match-level representations is beneficial for the match outcome prediction task. For further experimental results of predicting the match outcome only at the post-draft stage (i.e., after the draft is completed), see Appendix B.1.

### 5.4 Interpreting the Attention Weights

After training DraftRec, we analyzed the model’s attention weights on League of Legends dataset to understand how the model learned the synergy and competence between the different roles of players.

Fig. 4 displays the average attention weights for the final self-attention layer of the match network over the test dataset. Blue (Orange) boxes show high attention weights within the same (different) team. From Fig. 4, we observe that champions with the Top and Middle roles have high attention scores between champions with the Jungle role. In addition, champions with the AD Carry and Support roles show high attention scores with each other. Interestingly, this interaction reflects the actual role interaction of League of Legends, where (Top, Jungle), (Middle, Jungle), and (AD Carry, Support) mainly interact with each other in a match. We conclude from the above analysis that DraftRec learns meaningful relationships between the players according to their role.

### 5.5 Evaluation for Recommendation Strategy

Here, we study the effectiveness of our recommendation strategy which recommends the champion with the highest winning probability among the champions preferred by the player. For comparison, we consider three different recommendation strategies.

- DraftRec\(_{p}\): Recommendation strategy which ranks the champions based on the champion selection probability \( \hat{p}_{t,c} \).
- DraftRec\(_{v}\): Recommendation strategy which ranks the champions based on the match outcome probability \( \hat{v}_{t,c} \).
- DraftRec\(_{p+v}\): Our proposed recommendation strategy described in Equation 17. We set the selection probability threshold \( r \) as 0.02 (i.e., \( \frac{1}{50} \)) to allow the model to recommend champions that were selected within the most recent 50 previous matches.

To compare the different strategies, a straightforward evaluation method is online A/B testing. However, this could be prohibitively expensive and may hurt players’ in-game experiences. Therefore, we utilize an offline evaluation method which directly estimates the results with a separate evaluation model [15, 44, 47, 51].

For each turn in all test match data, DraftRec recommends a champion according to its corresponding recommendation strategy and a separate match outcome evaluation model predicts the match outcome of the draft assuming that the recommended champion is selected. If we use an identical model for both of our recommendation and match outcome evaluation, the correlation of inaccurate predictions can be problematic [45]. Therefore, we use OptMatch [13] as our separate match outcome evaluation model. We report HR@10, NG@10, and the average predicted win rate (Win) with varying rank \( k \) from \( \{3, 10\} \) to evaluate each strategy.

### Table 4: Performance comparison of different recommendation strategies evaluated on the League of Legends dataset.

| Model            | HR@10  | NG@10  | Win@3  | Win@10 |
|------------------|--------|--------|--------|--------|
| DraftRec\(_{p}\) | 0.8836 | 0.6618 | 52.72  | 52.46  |
| DraftRec\(_{v}\) | 0.2051 | 0.0938 | 51.34  | 51.55  |
| DraftRec\(_{p+v}\)| 0.8495 | 0.5657 | 54.12  | 52.59  |

Table 4 summarizes the performance of different recommendation strategies on the League of Legends dataset. Our proposed recommendation strategy, DraftRec\(_{p+v}\), has achieved the highest predicted match outcome value with a win rate of 54.12% when one of the top-3 recommended champions from the model is selected. DraftRec\(_{v}\) shows the worst performance in all metrics where...
HR@10 and NG@10 drop significantly compared to DraftRec. We suspect that the poor performance of DraftRec is incurred from assigning inaccurate values to the champions outside of the training data distribution as illustrated in Section 4.4.

5.6 User Study

To further evaluate our model’s recommendations, we conducted a user study with a visualization of DraftRec’s recommendations.

Table 5: Results of the user survey (N = 84). Bold indicates the best rank for each question and underlined indicates the second.

| Rank of the Players | All (N=84) | Diamond† (N=14) | Platinum (N=23) | Gold (N=28) | Silver† (N=19) |
|---------------------|------------|-----------------|-----------------|------------|--------------|
| Q1. Are you familiar with the model’s recommended champions? | 86.90% | 100% | 91.3% | 85.71% | 73.68% |
| Q2. Do you think recommended champions will lead to victory? | 79.76% | 92.86% | 86.96% | 71.43% | 73.68% |
| Q3. Are the explanations of the recommendations reasonable? | 80.95% | 85.71% | 73.91% | 85.71% | 78.95% |
| Q4. Do you wish to use this recommender system afterward? | 86.90% | 92.86% | 86.96% | 89.29% | 78.95% |

Figure 5: A user study screen which recommends top 3 champions to the 9-th user. ©Riot Games.

The user study procedure. A total of 84 user study participants were gathered from various popular online League of Legends communities [3, 4]. When gathering players to participate in our user study, we explained that we are currently developing a recommendation system trained on top 0.1% ranked user data and that we aim to recommend personalized champions with a high winning probability. To conduct a personalized survey, we first received consent and then collected the account IDs of the participants. Then we gathered the 50 most recent match data of each player and reproduced the actual draft situation they experienced. For each match, the participants were given a display of 3 champion recommendations and a description made based on the attention weights, as shown in Fig. 5. The synergy (counter) champion indicates the champion with the highest attention value within the player’s opponent’s team. User study participants were given personalized surveys made based on their match history data.

User study results. The survey took an average of 30 minutes for each player. The survey results are presented in Table 5. In terms of Q1, we observe that participants are familiar with the recommended champions (All = 86.9%). With respect to Q2, we observe most participants think that our recommendation will lead to victory (All = 79.76%), but this perception is more positive to high ranked users (Diamond = 92.86%). With regard to Q3, we can see that participants found the explanations for the recommendations reasonable (All = 80.95%). Finally, in Q4, we notice that a great majority of the players wish to use our recommender system in the future (All = 86.9%). Interestingly, we were able to see a tendency that our method was more positive to high ranked players compared to low ranked players. We speculate that this is due to the difference in the understanding of MOBA games players have according to their rank. Another possible explanation is that high ranked players related more with the recommendations since our model was trained with the top 0.1% rank match data.

6 ETHICAL CONSIDERATION

We took careful steps to preserve the ethics of research. Specifically, the collected League of Legends dataset utilized the publicly accessible API endpoint provided by Riot Games [5] which includes an automatic account ID encryption stage, preventing any abuse of personal information. For the Dota2 dataset, we utilized a public dataset where all personal information were removed. In addition, the user study in Section 5.6 was approved by the Research Ethical Committee of KAIST.

7 CONCLUSION AND FUTURE WORK

In this paper, we present DraftRec, a novel recommendation system which understands each player’s champion preference and the complex interaction between different players within a match. DraftRec utilizes a distinctive strategy to recommend the champions with a high win rate among the preferred ones of each player. Through extensive experiments on two MOBA-game datasets, we empirically demonstrate the superiority of DraftRec over various baselines and through a comprehensive user study, find that DraftRec provides satisfactory recommendations to real-world players.

A limitation of our work is that while in-game features are dependent on the performance of other players in a match, we only consider them from a single player’s perspective. Expanding our method to further integrate other players’ performance when constructing the players’ match history is left for future work. We hope that our research will contribute to enhanced user experience for various MOBA-game-related web services and applications.

ACKNOWLEDGMENTS

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2019-0-00075, Artificial Intelligence Graduate School Program(KAIST), and No. 2020-0-00368, A NeuralSymbolic Model for Knowledge Acquisition and Inference Techniques).
[46] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. arXiv preprint arXiv:1911.11361 (2019).

[47] Teng Xiao and Donglin Wang. 2021. A general offline reinforcement learning framework for interactive recommendation. In Proc. the AAAI Conference on Artificial Intelligence (AAAI).

[48] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In Proc. the International Joint Conference on Artificial Intelligence (IJCAI). 3203–3209.

[49] Deheng Ye, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, Zhao Liu, Fuhao Qiu, Hongsheng Yu, Yuyutang Yin, Bei Shi, Liang Wang, Tengfei Shi, Quan Fu, Wei Yang, Lanxiao Huang, and Wei Liu. 2020. Towards Playing Full MOBA Games with Deep Reinforcement Learning. In Proc. the Advances in Neural Information Processing Systems (NeurIPS).

[50] Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. 2020. Mastering complex control in moba games with deep reinforcement learning. In Proc. the AAAI Conference on Artificial Intelligence (AAAI), Vol. 34. 6672–6679.

[51] Lixin Zou, Long Xia, Pan Du, Zhao Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. 2020. Pseudo Dyna-Q: A reinforcement learning framework for interactive recommendation. In Proc. of the Web Search and Data Mining (WSDM). 816–824.
A IMPLEMENTATION DETAILS

Here, we describe the implementation details of DraftRec.

A.1 Feature Selection

Table 6 and Table 7 lists the selected features for the League of Legends dataset and Dota2 dataset respectively. We selected features that express the characteristic and champion preference of each player through a careful discussion with 5 top 1% ranked players of each game.

Table 6: Selected features for League of Legends.

| Feature                        | Description                                                                 |
|--------------------------------|-----------------------------------------------------------------------------|
| KDA                            | \((#\text{kills} + #\text{assist}) / #\text{death}\)                        |
| Largest_killing_spree          | Largest killing spree                                                       |
| Largest_multi_kill             | Largest number of multi kills                                               |
| Killing_sprees                 | Number of consecutive kills                                                 |
| Longest_time_spent_living      | Longest time spent alive                                                    |
| Double_kills                   | Number of double kills                                                      |
| Triple_kills                   | Number of triple kills                                                      |
| Quadra_kills                   | Number of quadra kills                                                      |
| Penta_kills                    | Number of penta kills                                                       |
| Unreal_kills                   | Number of unreal kills                                                      |
| Total_damage_dealt             | Total damage deal                                                           |
| Magic_damage_dealt             | Magic damage deal                                                           |
| Physical_damage_dealt          | Physical damage deal                                                         |
| True_damage_dealt              | True damage deal                                                            |
| Largest_critical_strike        | Largest critical strike damage                                              |
| Total_damage_to_champions      | Total damage to champions                                                   |
| Magic_damage_to_champions       | Magic damage to champions                                                   |
| Physical_damage_to_champions   | Physical damage to champions                                                |
| True_damage_to_champions       | True damage to champions                                                    |
| Total_heal                     | Total heal amount                                                           |
| Total_units_healed             | Total number of units healed                                                |
| Damage_self_mitigated          | Damage self mitigated                                                       |
| Damage_dealt_to_objectives     | Damage deal to objectives                                                   |
| Damage_dealt_to_turrets        | Damage deal to turrets                                                      |
| Vision_score                   | Vision score                                                                |
| Time_ccing_others              | Time crowd controlling others                                               |
| Total_damage_taken             | Total damage taken                                                          |
| Magical_damage_taken           | Magical damage taken                                                        |
| Physical_damage_taken          | Physical damage taken has                                                  |
| True_damage_taken              | True damage taken                                                           |
| Gold_earned                    | Total gold earned                                                           |
| Gold_spent                     | Total gold spent                                                            |
| Turret_kills                   | Number of turret kills                                                      |
| Inhibitor_kills                | Number of inhibitor kills                                                   |
| Total_minion_killed            | Total number of minions killed                                              |
| Minions_killed                 | Minions killed by player                                                    |
| Minions_killed_team_jungle     | Team jungles killed by player                                               |
| Minions_killed_enemy_jungle    | Enemy jungles killed by player                                              |
| Total_time_crowd_control       | Time being crowd controlled                                                 |
| Vision_wards_bought            | Number of vision wards bought                                               |
| Sight_wards_bought             | Number of sight wards bought                                                |
| Wards_placed                   | Number of wards placed                                                      |
| Wards_killed                   | Number of wards killed                                                      |
| Wards_killed                   | Number of wards killed                                                      |

A.2 Hyperparameters

Table 8 describes the optimal hyper parameter configurations of DraftRec. All DraftRec models are trained using A100 GPU with 64 cores. For League of Legends, training takes 5 hours to complete and for Dota2, training takes 1 hour to complete.

B FURTHER EXPERIMENTS

B.1 Match Outcome Prediction

In our draft recommendation setting, the model needs to predict the match outcome at every turn for each match. Therefore, we trained and evaluated the match outcome prediction models with the partially observable information set at each turn. However, previous match outcome prediction research [7, 13, 14, 26, 30] focused on predicting the match outcome when a draft was finished (i.e., prediction was made after every player has selected a champion). Therefore, in Table 9, we report the performance of all models when they are trained and evaluated by following the experimental protocols of previous work.

Similar to the findings of our match outcome prediction experiments in Section 5.3, DraftRec performs the best among all methods for the League of Legends dataset. Compared to OptMatch [13], which is a state-of-the-art match outcome prediction model, DraftRec
Table 8: Hyperparameters of DraftRec on *League of Legends* and *Dota2* dataset.

| Hyperparameter     | LOL    | Dota2   |
|--------------------|--------|---------|
| Epoch              | 10     | 20      |
| Optimizer          | Adam   | Adam    |
| Adam $\epsilon$    | 1e-8   | 1e-8    |
| Adam $(\beta_1, \beta_2)$ | (0.9, 0.999) | (0.9, 0.999) |
| LR Scheduler       | Cosine | Cosine  |
| Initial LR         | 1e-3   | 1e-3    |
| Final LR           | 0      | 0       |
| Weight decay       | 1e-5   | 1e-4    |
| Batch size         | 512    | 512     |
| Hidden size        | 128    | 64      |
| Clip gradients     | 5.0    | 5.0     |
| Hidden dropout     | 0.1    | 0.2     |
| Attention dropout  | 0.1    | 0.2     |
| Num heads          | 2      | 1       |
| Num blocks $N$     | 2      | 1       |
| Max sequence length $L$ | 50    | 20      |
| Loss control $\lambda$ | 0.1  | 0.1     |
| Selection threshold $\tau$ | 0.02 | -      |

Table 9: Performance comparison of DraftRec and baselines on the match outcome prediction task when predictions are made after every player completed selecting a champion.

| Models              | LOL ACC | LOL MAE | Dota2 ACC | Dota2 MAE |
|---------------------|---------|---------|-----------|-----------|
| MC [7]              | 0.5040  | 0.4960  | 0.5180    | 0.4820    |
| LR [30]             | 0.5323  | 0.4975  | 0.6126    | 0.4682    |
| NN [7]              | 0.5335  | 0.4958  | 0.6108    | 0.4692    |
| HOI [26]            | 0.5339  | 0.4978  | 0.6076    | 0.4819    |
| OptMatch [13]       | 0.5449  | 0.4928  | 0.6109    | 0.4724    |
| NeuralAC [14]       | 0.5347  | 0.4962  | 0.6108    | 0.4712    |
| DraftRec-no-history | 0.5432  | 0.4893  | 0.6085    | 0.4592*   |
| DraftRec            | 0.5618* | 0.4826* | 0.6110    | 0.4642    |

Notes: "*" indicates the statistical significance (i.e., $p < 0.01$).

shows an improvement of 3.1% in ACC and 2.1% in MAE. However, for the *Dota2* dataset, we observed that a simple logistic regression model performed the best.

### B.2 Ablation on Maximum Sequence Length

In order to further analyze the importance of utilizing personal match histories, we conduct an ablation studies referring to the length of the player match history the models utilizes. Since *Dota2* dataset contains a scarce amount of players’ match histories, we only consider *League of Legends* dataset throughout this experiment. For all the experiments, we vary the length of player match histories $L$ from {1, 2, 5, 10, 20, 50, 100} and fix the remaining hyperparameters to each model’s optimal configurations.

Champion Recommendation. Fig. 6 displays the recommendation performances of sequential recommendation models which includes DraftRec, S-POP, and SASRec. We observe that player match history length $L$ is a crucial factor for the recommendation performances where both HR@10 and NG@10 increase when the player match history length $L$ increases. However, we also discovered that both HR@10 and NG@10 converges around $L = 50$ for all models. This implies that the player’s champion selections are more affected by their recent matches rather than their old ones.

Match Outcome Prediction. Fig. 7 shows the performance of the match outcome prediction models that utilizes player match history information which include DraftRec and OptMatch. We observed that both DraftRec and OptMatch benefits with longer player match history length $L$ and start to converge at $L = 50$. When the player match history length $L$ gets longer, the models may not benefit from a longer history length since extra noise can be injected along with the extra history information.