Personalized Sequential Recommendation for Adaptive Itemization in MOBA Games

Zachary Novack
znovack@ucsd.edu

INTRODUCTION

Multiplayer Online Battle Arena games (or MOBAs for short), are subgenre of video games that combine elements of both action games and strategy games. MOBAs comprise some of the most popular games within eSports, with League of Legends (LoL) representing one of the most popular video games worldwide. In LoL, two teams of five players (each playing as some "champion"), fight to destroy the other teams' base. Along the way, players can buy items with in-match gold, earned from completing objectives and killing enemy players, in order to improve their champions' stats and odds of winning.

However, the task of choosing which items to buy during a game is not a trivial problem, involving knowledge of in-game mechanics, personal playstyle, the current set of ally and enemy champions, AND each champion's current set of items *they* bought. Thus, the task at hand is whether **we can effectively design a machine learning system to *recommend* which items a user should buy sequentially given the game state.**

1 PART 1: DATASET

For this analysis, we turn to a dataset of 108,000 top-ranked League of Legends Games in Korea from Kaggle[1]. For every match in the dataset, we have access to which users played the match, which champions they were playing, who won, and most importantly for our purposes, a record of which items each player had by the end of the match. As this dataset is not specifically designed for recommendation-like tasks, extensive preprocessing was done to make the dataset suitable for our task.

We follow the setup from Araujo et al. [2022], wherein we do the following: for every match in the dataset, we extract a processed data point for each user on the *winning* team (as we make the assumption that the winning team chooses items better) that coincides to that user's data and all ally and enemy data, with the corresponding label representing the list of items bought during the game. This essentially quintuples the size of the dataset (with one data point per winning player per game). We also filter the dataset for only matches played in season 13 of LoL, as significant game-mechanic changes, including the exclusion or creation of entirely new items, may occur season to see. Additional preprocessing was used in order to extract a semi-ordered representation of the item list (as the raw dataset only included the items as represented by their placement in the user's inventory, labeled 1-6). Based on interviews with a small pool of domain experts, we used the following heuristic to induce an ordering to each user's item set in a given match:

(1) Consumable "potions" that give in-game health back to the player are generally bought right at the start of the game with the baseline gold.
(2) "Boots", a unique type of item that increases player movement speed, is one of the first items bought as the player accrues gold.

Thus, every user's item set in every match was reorder to include consumables, followed by boots, and finally with all other remaining items. We make an additional simplifying assumption in that all item purchases among players happen at the same time (i.e. when a player buys their 3rd item, we assume all players in the game have bought 2 items). We recognize this as a inherent limitation to our work, and hope that in future endeavors a dataset with temporally correct item logs can be used.

We then split the data 80-20 for training and validation, wherein the processed data is split temporally such that the validation set includes the most recent match, and notably no match appears in both sets (to

[1]https://www.kaggle.com/datasets/gyejr95/league-of-legendslol-ranked-games-2020-ver1

# Champions	148
# Items	244
# Players	99366
# Train/Test Split	331920 / 82980

Table 1: Processed Dataset details

avoid data leakage). The final processed data details are shown in Table 1.

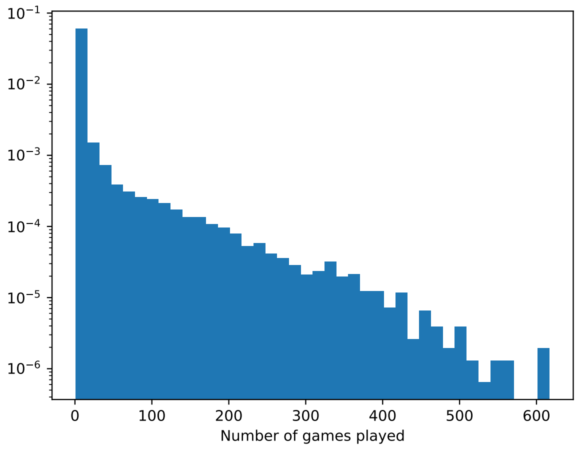

Figure 1: Distribution of Games Played per User

In particular interest for the project was incorporating *player* information into the recommendation pipeline, as such personalization has not been a part of recent works on MOBA-related recommendation tasks [Araujo et al. 2022; Chen et al. 2018; Looi et al. 2018; Villa et al. 2020]. In Figure 1, we show the distribution of number of games played across all unique users in the dataset. Notably, we see that roughly 66% of users only played one game. While the sparsity of this feature is somewhat to be expected (as ranked games are more competitive than casual unranked games and are such played less often for most users), this shows that there is at least some amount of historical user data in the other third of users that we may leverage in order to improve item recommendation.

2 PART 2: PROBLEM SETUP

Here, we formalize the task of personalized sequential recommendation for item choice in MOBAs, which is heavily inspired by Araujo et al. [2022]. For a given match, we have a set of users $\mathcal{U} = \{\mathbf{u}_k\}_{k=1}^{10}$, a set of champions $C = \{\mathbf{c}_k\}_{k=1}^{10}$ corresponding to each user, and a list of items for each user's champion $\mathcal{I}_{\mathbf{c}_k} = \{\mathbf{i}_j^{(\mathbf{c}_k)}\}_{j=1}^{|\mathcal{I}_{\mathbf{c}_k}|}$. As correct itemization in LoL not only depends upon a player's current item history, but also their playstyle, their own champion choice, and the champion choice and items bought by all allies and enemies, the task can be phrased as estimating the following conditional probability: at a given item slot j, user $\mathbf{u}_k$, champion $\mathbf{c}_k$, item history $\mathbf{i}_{\leq j}^{(\mathbf{c}_k)}$, allied champions $\mathbf{c}^a$ and items $\mathbf{i}_{\leq j}^{(\mathbf{c}^a)}$, and enemy champions $\mathbf{c}^e$ and items $\mathbf{i}_{\leq j}^{(\mathbf{c}^e)}$, we estimate

$$p(\mathbf{i}_{j+1}^{(\mathbf{c}_k)} \mid \mathbf{u}_k, \mathbf{c}_k, \mathbf{i}_{\leq j}^{(\mathbf{c}_k)}, \mathbf{c}^a, \mathbf{i}_{\leq j}^{(\mathbf{c}^a)}, \mathbf{c}^e, \mathbf{i}_{\leq j}^{(\mathbf{c}^e)}) = p(\mathbf{i}_{j+1}^{(\mathbf{c}_k)} \mid \mathbf{d}_j), \tag{1}$$

where $\mathbf{d}_j$ is the "description" of the game state when buying the $j + 1$th item. In words, we seek to use the current game context to predict the next item a player should buy. We can then assess the quality of our predictions using standard metrics such as Precision@k, Recall@k, and Mean Reciprocal Rank (MRR) @k.

Given this setup, there are many possible choices for how to structure extracting user, item, and champion level feature representations from the data. As one simple baseline comparison, we draw from Araujo et al. [2022] and propose **POP**. **POP** completely forgoes user information or any sequential modeling and simply implements a standard popularity-based heuristic, recommending the most popular set of items (in decreasing relevance, implicitly assuming that more popular items should be bought earlier) for each champion, where popularity is determined by frequency in the training set.

For our main model of interest, we base our model off the architecture HT4Rec from Araujo et al. [2022] (detailed more in Section 3). Notably, we do not use any of the contextual stat-based information for items or champions and instead model each item and champion through latent representations. However, modeling of the *user* representations is approached differently.

While we could take a latent factor-inspired approach to modeling users, such an approach may not scale in practice with the massive player base of LoL or update quickly to account for fast changes in user playstyle. Thus, we turn to a parametric feature extraction approach for the user representation. A key facet of user representations that we wish to capture is the idea of *user skill*: While certain item-champion configurations may be optimal for pro-players, such combinations may lend themselves to playstyles that are more difficult for beginning players or involve “active" items that require their own learning curve to use effectively. Firstly, we draw from standard league metrics[2] and calculate the *Dominance Factor* for each user k in each match m:

$$DF_{k,m} = 2K_{k,m} + A_{k,m} - 3D_{k,m} \tag{2}$$

Where $K_{k,m}, A_{k,m}, D_{k,m}$ are the number kills, assists, and deaths respectively (an assist in this context refers to when a player damages an enemy shortly before they are killed but do not land the killing blow). We then extend this to account for historical trends by calculating the moving average dominance factor aDF for a user over the past t matches (i.e. for a player’s nth match, $aDF_{k,n} = \frac{1}{t}\sum_{m=n-1}^{n-t} DF_{k,m}$), where t is a hyperparameter. In this sense, aDF captures (for a given game) the recent ability of the user. We then discretize the aDF into the following final user input, which we denote their “elo" $\mathbf{e}_{k,m}$:

$$\mathbf{e}_{k,m} = \{b : \mathbf{Q}_{b-1} \leq aDF_{k,m} \leq \mathbf{Q}_b\} \tag{3}$$

Where $\mathbf{Q}_b$ is the bth *quintile* of all $DF_{k,m}$ across all matches and users. In words, this quantity captures what bracket of all players is the current player’s ability well situated in. While an obvious issue with this method is that calculating each $\mathbf{Q}_b$ across the entire dataset inherently involves data leakage, we manually verify that the quintiles are remarkably stable over small (5% of the data) subsets of the original dataset (see Figure 2), and thus use the global quintiles given the stationarity. For users with no historical data, we default set this value to the centermost quintile. This discretization allows us to extract temporal player-ability information that is in the same format as champions and items (that is, discrete tokens over a fixed vocabulary).

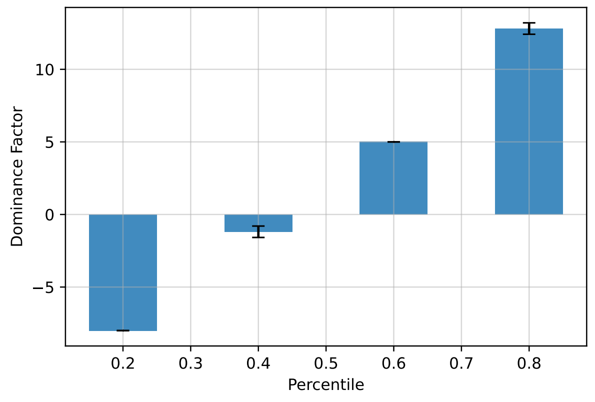

Figure 2: Average Quintile across disjoint subsets of dataset.

3 PART 3: MODEL

We base our main model architecture off of [Araujo et al. 2022]’s **HT4Rec**, which is a hierarchical transformer model that is the current state of the art for sequential item recommendation in MOBAs. However, HT4Rec noticeably did not include any *user* information in its structure, which is where we propose **HT4Rec4U** (**H**ierarchical **T**ransformer **f**or **R**ecommendation **f**or **Y**ou), a personalized form of HT4Rec. We now describe the model architecture, and omit match subscripts from our notation for clarity.

For a given user in a given match, we first extract a learned champion embedding $\mathbf{c}$, a learned team embedding $\mathbf{t}$ to differentiate between allies and enemies, and a learned *elo* embedding $\mathbf{e}$ extracted from the user’s recent game history as shown in the previous section. At a given recommendation step j, we compute item embeddings $\mathbf{i}_j^{(\mathbf{u})}$, with our final input embeddings $\mathbf{x}_j$ calculated as follows:

$$\mathbf{x}_j = (\mathbf{c} + \mathbf{t} + \mathbf{e}) \oplus \mathbf{i}_j^{(\mathbf{u})}, \quad \mathbf{x}_j \in \mathbb{R}^d \tag{4}$$

where $\oplus$ is the concatenation operation. We use $\mathbf{X}_j$ to denote the input embeddings for all players in the game.

After this step, our method proceeds identically to HT4Rec. Namely, we use a Contextual Transformer [Villa et al. 2020] module CT to calculate interactions among the different players in the game, which outputs a contextual embeddings $\mathbf{a}_j^{(\mathbf{u})}$ for a given user and timestep. This embedding is then added to a learned positional embedding $\mathbf{p}$, and the resultant embedding is fed into a Sequential Transformer [Kang and McAuley 2018] block ST to generate a latent embedding $\boldsymbol{\gamma}_j^{(\mathbf{u})}$. Finally, this latent embedding is taken with a linear recommendation layer $\mathbf{W}$ and a softmax activation σ,

[2]https://leagueoflegends.fandom.com/wiki/Kill_to_Death_Ratio

which gives us the final output probability:

$$p(\mathbf{i}_{j+1}^{(\mathbf{u})} \mid \mathbf{d}_j) = \sigma(\mathbf{W}\boldsymbol{\gamma}_j^{(\mathbf{u})}) \tag{5}$$

$$\boldsymbol{\gamma}_j^{(\mathbf{u})} = ST(\mathbf{a}_j^{(\mathbf{u})} + \mathbf{p}_j) \tag{6}$$

$$\mathbf{a}_j^{(\mathbf{u})} = CT(\mathbf{X}_j) \tag{7}$$

In comparison to HT4Rec, our HT4Rec4U architecture's inclusion of the elo embedding seeks to use the relative skill levels of the players in order to provide better recommendation. Like Araujo et al. [2022], we train our model to minimize the cross entropy loss with the correct item predictions using the Adam optimizer [Kingma and Ba 2014]. We note that one possible drawback of HT4Rec4U is the entanglement of the elo embedding with the champion/team embedding (which was done to not increase the dimensionality d of the embedding too much), which could negatively impact performance. Additionally, in order to maximize results given compute restrictions, we use the default model specifications from Araujo et al. [2022] for HT4Rec4U.

4 PART 4: RELATED LITERATURE

The space of recommendation for MOBA games has seen growing attention in the past few years [Araujo et al. 2022; Chen et al. 2018; Looi et al. 2018; Villa et al. 2020], and the most common datasets used are the Season 7 US-based LoL matches[3] and a dataset from another popular MOBA, DotA 2[4]. These datasets were not used for the present work, as they do not include any cross-match user information, thus making them not suitable for personalized recommendation. Within MOBA games, there are two main tasks with regards to recommendation: item recommendation (the focus of the present work), and champion recommendation.

Champion recommendation is concerned with helping a player choose which champion to play at the start of the match. During the champion selection phase, each player on each team is allowed to ban a champion from being played by *anyone* that match (which happens all at once), and is then proceeded by a draft selection phase, where teams alternate in picking champions to play. There is a breadth of work that has looked at learning optimal ways to play this minimax game, leveraging association rules [Hanke and Chaimowicz 2017], classical ML algorithms [Porokhnenko et al. 2019], deep-learning approaches [Chen et al. 2021; Gourdeau and Archambault 2020], and Monte-Carlo Tree Search [Chen et al. 2021, 2018] methods.

Within item recommendation, the majority of works are not concerned with the sequential and/or multi-agent aspect of the game itself. Namely, Looi et al. [2018] is primarily concerned with single item recommendation only given the player's character and currently bought. Araujo et al. [2019] and Villa et al. [2020] both focus on context-aware (i.e. concerning other players) recommendation of the full item set. We note Araujo et al. [2022] as our main inspiration, as they are the only other work (to our knowledge) that tackles both the sequential and context-aware aspects of item recommendation in MOBAs. Here, they use the aforementioned Dota 2 dataset, which contains a more accurate log of item purchases.

5 PART 5: RESULTS & CONCLUSION

We compare our three models (POP, HT4Rec, and HT4Rec4U) on their performance on the validation subset of the original data. For HT4Rec and HT4Rec4U (as no training is required for POP), we train for 50 epochs with a learning rate $\eta = 2 \times 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and weight decay = 1×10^{-4}. For the model architecture, we use a single layer for the contextual and sequential transformer encoders, a latent dimensionality $d = 512$, and a max sequence length of 6 (as only 6 items can be held at once).

Metric	POP	HT4Rec	HT4Rec4U
Precision@1	0.1037	0.1968	**0.3122**
Recall@1	0.1037	0.1968	**0.3122**
MRR@1	0.1037	0.1968	**0.3122**
Precision@3	0.0367	0.1312	**0.1841**
Recall@3	0.1100	0.3937	**0.5522**
MRR@3	0.1068	0.2817	**0.4167**
Precision@6	0.0194	0.0921	**0.1188**
Recall@6	0.1164	0.5528	**0.7130**
MRR@6	0.1078	0.3153	**0.4508**

Table 2: Quantitative Results on Validation Set

[3]https://www.kaggle.com/datasets/paololol/league-of-legends-ranked-matches
[4]https://www.kaggle.com/datasets/devinanzelmo/dota-2-matches

In Table 2, we can see that our model, HT4Rec4U, performs best across all metrics at different values of k. While the subpar performance of POP is not surprising (as it breaks the sequential nature of the problem), the boost over HT4Rec gives evidence that leveraging even simplified personalized information can drastically improve the quality of the recommendations. We do note that in general, the metric values are quite low across all tested models, which opens the door for further research on more well-formed datasets with perhaps more involved personalization modeling.

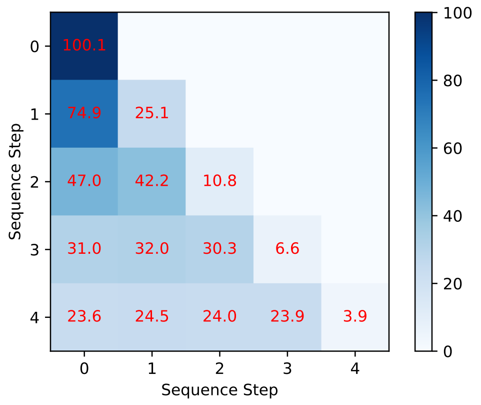

Figure 3: Average Attention weights for Sequential Encoder

In order to get a sense of *what* our model actually learned, we visualize the attention weights for the sequential transformer encoder (as we a priori know each sequence has at most 6 items and thus 5 predictions), averaged across all data in the validation set. In Figure 3, we can see that our model not only attends to all of the past time steps when predicting the next recommended item. This helps us build the intuition that our model has learnt reasonably specific temporal dependencies and uses the *entire* set of currently bought items to influence the next item.

As a whole, our present project represents the first work to recommend items in MOBA games through a sequential, context-aware, and *personalized* modeling approach. Given the success of incorporating user skill-based information into the recommendation, we hope that this work inspires further research into fine-grained, user-specific item recommendation in MOBA games, and more broadly bringing user skill information into the space of recommendation for games.

REFERENCES

Vladimir Araujo, Felipe Rios, and Denis Parra. 2019. Data mining for item recommendation in MOBA games. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 393–397.

Vladimir Araujo, Helem Salinas, Alvaro Labarca, Andres Villa, and Denis Parra. 2022. Hierarchical Transformers for Group-Aware Sequential Recommendation: Application in MOBA Games. In *Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization.*

Sheng Chen, Menghui Zhu, Deheng Ye, Weinan Zhang, Qiang Fu, and Wei Yang. 2021. Which heroes to pick? learning to draft in moba games with neural networks and tree search. *IEEE Transactions on Games* 13, 4 (2021), 410–421.

Zhengxing Chen, Truong-Huy D Nguyen, Yuyu Xu, Christopher Amato, Seth Cooper, Yizhou Sun, and Magy Seif El-Nasr. 2018. The art of drafting: a team-oriented hero recommendation system for multiplayer online battle arena games. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 200–208.

Daniel Gourdeau and Louis Archambault. 2020. Discriminative neural network for hero selection in professional Heroes of the Storm and DOTA 2. *IEEE Transactions on Games* (2020).

Lucas Hanke and Luiz Chaimowicz. 2017. A recommender system for hero line-ups in MOBA games. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Wenli Looi, Manmeet Dhaliwal, Reda Alhajj, and Jon Rokne. 2018. Recommender system for items in dota 2. *IEEE Transactions on Games* 11, 4 (2018), 396–404.

Iuliia Porokhnenko, Petr Polezhaev, and Alexander Shukhman. 2019. Machine learning approaches to choose heroes in dota 2. In *2019 24th Conference of Open Innovations Association (FRUCT)*. IEEE, 345–350.

Andrés Villa, Vladimir Araujo, Francisca Cattan, and Denis Parra. 2020. Interpretable Contextual Team-aware Item Recommendation: Application in Multiplayer Online Battle Arena Games. In *Fourteenth ACM Conference on Recommender Systems*. 503–508.