A recommendation system approach for World of Warcraft Classic Hardcore in-game information distribution

Enzo Morata
Pontificia Universidad Católica de Chile
enzo.morata@uc.cl

Fabrizio Garcia
Pontificia Universidad Católica de Chile
fsgarcia1@uc.cl

Jose Caraball
Pontificia Universidad Católica de Chile
jtcaraball@uc.cl

Abstract

In this work we propose the use of an in-game implementation of a recommender system as an alternative of the current solutions for risk advices for the game World of Warcraft Classic Hardcore. To achieve the best performance we compare different models between Collaborative Filtering, Implicit Feedback and Machine Learning using ranking metrics and exploring different approaches to incorporate relevant contextual data to the system.

Keywords:

Recommender Systems. Contextual Recommendation. Ranking Metrics. World of Warcraft.

1. Introduction

World of Warcraft Classic Hardcore is an alternative version of the popular massive multiplayer online role-playing game (MMORPG) released on August 24, 2023, that adds the Hardcore rule set to the original game. As with many MMORPGs World of Warcraft Classic is a game in which players invest considerable hours to building and personalizing their characters, obtaining powerful items, and defeating increasingly harder enemies. Taking advantage of this investment the Hardcore rule set attempts to add a new layer of challenge to the game by making it so that, if a player ever dies, their character is locked from the world, their progress is lost and are thus forced to create a new one to start over from the beginning.

In this context it is imperative that players undertaking this challenge have the necessary tools to minimize the risk of losing their characters and when it comes to MMORPGs few things are more important than knowledge. This is particularly the case for new players who venture into the world without knowing which enemies pose the most dangers to them given their character's class and level. In fact, Blizzard (2023), the game developer and publisher, has reported that in the first three months since the game mode released almost three million characters have died with all top ten sources of death corresponding to enemies or environmental hazards present in the early zones that players visit during their adventures.

Currently there are some efforts that try to help with providing players with information about the dangers of the world of Azeroth. All these solutions fall in one of two groups: guides and data driven analysis. The first group consists mainly of free and paid guides where knowledgeable players lay out guides to traverse the world while minimizing risk based on their experiences. The second one, mainly spear headed by the open-source project Deathlog by aaronma37 (2023), try to aggregate data of characters death given manually by players to derive useful information from basic statistical analysis.

We believe that these approaches fall short of solving the problem in two mains aspects:

1. They amount to non-personalized recommendation systems that fail to consider the class and level of the player's character or the similarities between them. For example, Warlock and Hunters share a similar playstyle but the latter is far more popular so when advising Warlock players, one may lack observations for the class but could obtain valuable information from looking at Hunters interactions.

2. They are accessed outside of the game, are locked behind pay walls or are hidden in cumbersome menus that the players must traverse to get important information.

In this work we explore different recommendation methods and compare their performance on the dataset compiled by the Deathlog project using standard recommendation metrics. The goal is to find a system that allows us to recommend to players, given the class they are playing, the level of their character and the zone in the world they are currently in, which adversaries are

the most dangerous and thus arm them with knowledge so they may better asses how they should traverse in their adventure.

From the candidates we find the best technique that exceeds the results of the underline model of the existing solutions and show an in-game implementation that allows players to access the recommendation system inside of the game with minimal effort.

2. Dataset

The dataset used in this work was obtained from the data compiled by aaronma37 (2023) in the open-source project Deathlog. This database is being actively updated by the project maintainers, but the version used here is comprised of 337623 death recordings that, after being cleaned and grouped, correspond to 48184 unique character-source interactions each one accompanied by the number of times it is observed. In figures 1 and 2 we can see the distribution of the sum of interactions by character class and level.

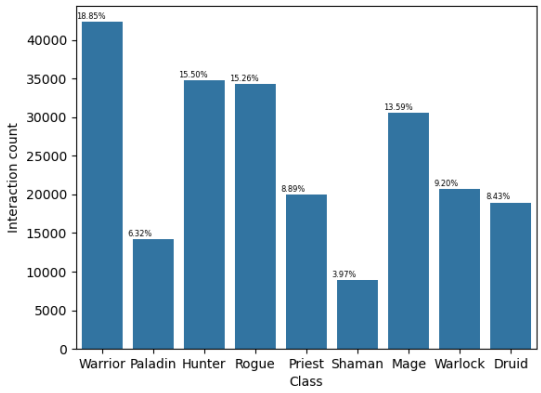

Figure 1: Interaction count distribution by class.

We will be referring to 'sources' as the entities, be it an enemy or an environmental hazard (like drowning) that causes a player death and to 'characters' as the avatar that the players control. In the following section we explore the challenges and considerations that come into play when trying to define how to represent the concept of a character in a recommendation system.

3. Methodology

3.1. Contextual Recommendation

The first consideration that raises when trying to recommend items that are contextually exclusive is how to incorporate this context into the recommendation

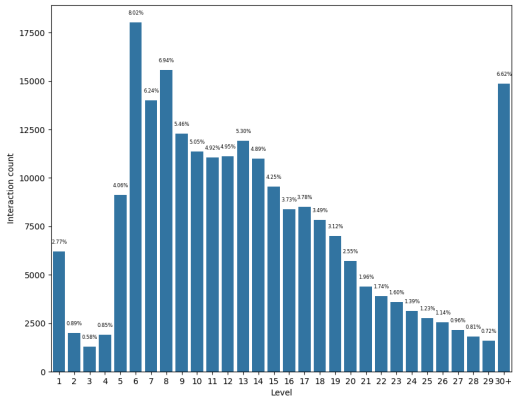

Figure 2: Interaction count distribution by level.

system. In the case of this work, the sources of danger may appear only in a specific area and thus a user should never see a recommendation that is not present in the current zone they are in. Adomavicius et al., 2011 explores many possibles ways in which one can incorporate contextual information and we make use of two of them to attempt to solve this problem. These are Contextual Modeling (CM) in which the contextual information is integrated as a part of the recommendation method and Contextual Post Filtering (CPF) that, as the name suggests, makes use of contextual data to filter recommendations. In our case the CM approach is applied by defining characters as triples (class, level, zone) and passing recommendations as given by the system while CPF is implemented using only pairs (class, level) as a way of defining characters and filtering out results that do not correspond to the target zone.

3.2. Metrics

As mentioned earlier one the main goals for this work is to deliver information to the players in a precise and convenient fashion. To do this we believe its necessary to be able to capture the most important sources of danger and recommend the smallest set of dangerous possible. Based on experience we have settled on only recommending the top five results. The small number of recommendations imposes a particularly high level of demand on the top results and that is why we proposed the use of strict Ranking Metrics to evaluate the models. The selected metrics were: Mean Average Precision (MAP@5) and Mean Normalized Discounted Cumulative Gain (nDCG@5).

3.3. Models

We evaluate several different models to find the one with the best performance and portability for the recommendation in-game. These models cover different types of recommendation strategies, being these: *Implicit Feedback*, *Collaborative Filtering* and *Machine Learning*. Also we define as underline models to compare the performance, the *Most Popular* and *Random* recommendations.

3.3.1. Most popular

Most popular is a non-personalized recommendation method which output corresponds to the k items with the greatest number of interactions recorded. This method is particularly relevant as it is the underline system that all existing solutions employ to advise players. Both guides and Deathlog, which main recommendation interface is a list of the sources with most recorded kills for each class by zone, rely on sources' historic predominance to derive its relevancy in all situations.

3.3.2. Random

The Random recommendation method, included mainly for the sake of completeness, creates a recommendation list at random assuming that interactions with items follow a given probability distribution. The specific method used is RandomNorm as implemented by Hug, 2015 which assumes a normal distribution.

3.3.3. Implicit Feedback

Within the implicit feedback model framework, the emphasis lies in capitalizing on player interactions and in-game actions, eliminating the necessity for explicit feedback (Koren et al., 2009). In this specific context, the Alternate Least Squares (ALS) matrix factorization model, renowned for its robustness and efficacy, will be trained as part of the proposed approaches within our comprehensive methodology.

The rationale behind opting for the ALS model stems from its adeptness at handling extensive datasets and swiftly converging towards optimal solutions. This attribute proves particularly valuable in the dynamic gaming environment, where player actions furnish invaluable signals about preferences and experiences.

To enhance the effectiveness of the ALS model, two distinct data segmentation approaches were investigated. One approach involves grouping data that share the same class, level, and zone (contextual modeling), while the other focuses on data with the same class and level (contextual post filtering). The objective is to derive a model that more accurately mirrors reality in each recommendation, considering the specific context of a character's class, level, and gaming zone.

Prior to the training phase, a weight-balancing procedure was implemented to counteract imbalances in group popularity. This adjustment proved pivotal in achieving a more equitable distribution, aligning the data with a normal distribution, and instilling confidence in the data quality used for the model.

In the experimentation phase, matrix factorization was conducted using the Implicit Library. An exhaustive exploration considered various combinations of crucial hyperparameters influencing the performance of the ALS model. These hyperparameters, including the number of iterations, latent factors, regularization factor, and confidence factor (α), each played a pivotal role in optimizing the model's effectiveness in delivering impactful recommendations. (Koren et al., 2008)

Conclusively, after conducting multiple iterations among the four aforementioned hyperparameters, we have arrived at the optimal combination for the ALS model based on implicit feedback:

- The number of latent factors is equal to 50; this choice signifies an optimal balance between precision and efficiency, allowing for a detailed representation of relationships between users and items.
- The number of iterations is equal to 5; this indicates that the model has reached an appropriate convergence level after these iterations, reflecting stability and optimal fitting based on the training dataset.
- The regularization factor is 0.01; this value denotes a suitable balance to prevent overfitting, avoiding excessive model weight. Thus, effective generalization is achieved without significantly compromising performance.
- The confidence factor (α) is equal to 1; this implies assigning moderate confidence to observed interactions, deemed appropriate for influencing the importance attributed to different interactions in recommendation generation.

3.3.4. Collaborative Filtering

Collaborative Filtering (CF) is a simple approach used in recommender systems recommendations are generating using notion of similarity between the users and items represented as the ratings they give or receive

respectably. The selected methods of Collaborative Filtering were CF Based on Users, CF Based on Items (Schafer et al., 2007) and Slope One (Lemire & Maclachlan, n.d.). To implement these methods we used the Surprise library (Hug, 2015). The particular implementations details are as follows:

- Collaborative Filtering Based on Users; For this model the algorithm KNNWithMeans was employed using the cosine similarity. The values of K used for Contextual Modeling (CM) and Contextual Post Filtering (CPF) where $K = 12$ and $K = 6$ respectably.
- Collaborative Filtering Based on Items; The same way as with CF Based on Users the method used was KNNWithMeans with cosine similiraty. Th values of K used for Contextual Modeling (CM) and Contextual Post Filtering (CPF) where $K = 6$ and $K = 12$ respectably.
- Slope One; the default implementation of the algorithm was used.

The hyperparameters values used for CF Based on Items and Users were selected by trial and error until the best result was found. All results can be seen in the tables 1a and 1b.

K	Contextual Post Filtering	Contextual Modeling
6	0.01082	0.00392
8	0.01263	0.00378
10	0.01421	0.00368
12	0.01610	0.00364

(a) Hyperparameter search for *CF Based on Items*

K	Contextual Post Filtering	Contextual Modeling
6	0.03417	0.00527
8	0.03325	0.00583
10	0.03364	0.00583
12	0.03313	0.00607

(b) Hyperparameter search for *CF Based On Users*

Table 1: Hyperparameter search for KNNWithMeans implementation using nDCG@5 metric.

3.3.5. Neural Networks

The neural network used in this work takes as input, for a given pair character-source, the vector resulting of the concatenation of the character's number of observed interactions with every source and the number of observed interactions of the source with every character, and outputs a single value corresponding to their expected number of interactions. Of course, hiding these value on both the instances it appears in the input. It is comprised of two fully connected layers with 512 neurons each. As the input is incredibly sparse the objective of this method was to try to take advantage of the power of deep learning to see if it was possible to obtain meaningful inference.

Because using inputs that vary depending on zones would demand a variable input vector a model using the CM approach was not possible and thus only a CPF variant was implemented.

The model was implemented using Pytorch (Paszke et al., 2019) and it used $Mean\ Square\ Error\ (MSE)$ as the loss function, Adam (Kingma and Ba, 2017) as the optimizer, Sigmoid as the activation function, a dropout value of 0.1 and a learning rate of 0.001. These parameters where primarily chosen by trial and error until we found the best result. Though, it is important to remark that this is but one out of a vast universe of possible architectures and thus not necessarily indicative of the capabilities of neural networks in tasks of this nature.

4. Results

In this section we will describe the results obtained by the different models. Almost all models were tested using both contextual recommendation methods: Contextual Modeling (CM) and Contextual Post Filtering (CPF), using the metrics nDCG@5 and MAP@5. The results can be observable in the table 2.

For the underline models, Most Popular and Random recommendations. In first place, Most Popular by its nature obtained the same results for Contextual Modeling (CM) and Contextual Post Filtering (CPF) methods. In second place, Random (CM) has the worst performance of all models, but Random (CPF) can achieve a better recommendation than Most Popular.

For the Neural Networks model, as mentioned before only the Contextual Post Filtering approach can be implemented. Its metrics results were $\mathrm{nDCG@5} = 0.01309$ and $\mathrm{MAP@5} = 0.01116$. Between the proposed models with the CPF approach it has the worst performance, being behind of the results from CF Based on Items (CPF).

For the Collaborative Filtering (CF) models; Slope One, Based on Items and Based on Users. In general, the results were better for the Contextual Post Filtering method than Contextual Modeling method. Also, all the results were better than the current solutions' underline models. The models ordered by their results are: CF

Based on Items, Slope One and CF Based on Users. Being CF Based on Users (CPF) the best model in Collaborative Filtering, with $\mathrm{nDCG@5} = 0.03417$ and $\mathrm{MAP@5} = 0.03262$.

For the Implicit Feedback model, Matrix Factorization ALS. It has the best performance between all the models, with the CM approach, $\mathrm{nDCG@5} = 0.04060$ and $\mathrm{MAP@5} = 0.03926$. Also, the CPF approach has a very good performance with $\mathrm{nDCG@5} = 0.02775$ and $\mathrm{MAP@5} = 0.02423$, being just behind from the metrics obtained by the CF Based on Users (CPF).

Model	nDCG@5	MAP@5
Most Popular (CM)	0.00141	0.00138
Most Popular (CPF)	0.00141	0.00138
Random (CM)	0.00000	0.00000
Random (CPF)	0.00320	0.00280
M.F. ALS (CM)	**0.04060**	**0.03926**
M.F. ALS (CPF)	0.02775	0.02423
Slope One (CM)	0.00723	0.00636
Slope One (CPF)	0.02307	0.02049
CF Based on Items (CM)	0.00392	0.00376
CF Based on Items (CPF)	0.01610	0.01506
CF Based on Users (CM)	0.00607	0.00552
CF Based on Users (CPF)	0.03417	0.03262
Neural Networks (CM)	-	-
Neural Networks (CPF)	0.01309	0.01116

Table 2: Models metrics on Deathlog dataset.

5. Implementation

The in-game implementation, written in the game's native language LUA, required several challenges to be overcome, mainly in the form of computational and storage limitations imposed by the players and the game itself. For the recommendation system to be usable, as it shares resources with the game, it needed to be computationally efficient and lightweight. To achieve this, we took advantage of the relatively small universe of characters and sources and implemented the system as a look up table where the key corresponds to the character triple (class_id, level, zone_id) and the value to a pre-computed list of the top 5 recommendations created by the trained Matrix Factorization ALS model. The resulting module operates in constant time and when loaded weighs 8.38 kB which is well within the acceptable range.

The mode of use of the system is straightforward. When a player enters a zone an in-game event is triggered, their characters information is queried to the game and the recommendations are looked up and printed out to the chat interface. An example of a level 20 rogue entering the Redridge Mountains zone can be seen in figure 3.

Figure 3: In game implementation of recommendation system.

The implementation was privately tested with a small group of friends and family. The feedback received was limited but generally positive, with the recommendations being in line with the expectation of the veteran players in the group. The main concern raised involved characters that were under or over the appropriate level for a zone, in this scenario, as fewer deaths are recorded, the quality of the recommendations tend to decline. As more instances are included we expect that situations like this should become rarer.

6. Conclusions

All models trained in this study surpassed the underline model. Particularly noteworthy is the model that yielded the most favorable results, utilizing implicit feedback through matrix factorization (ALS) with Contextual Modeling (CM). This approach proved highly effective, outperforming alternative methods in terms of performance. The success can be attributed to its consideration of the character's class, level, and zone, highlighting its capability to offer personalized and precise recommendations.

Quantitative evaluation metrics, such as MAP@5 and nDCG@5, robustly support the effectiveness of the proposed approach, affirming the reliability and relevance of the implemented recommendation system.

The introduction of a Neural Networks model underscores the potential of artificial intelligence and deep learning within the domain of game recommendation systems. While refinements are necessary, this exploration implies promising prospects for future innovations.

The decision to integrate the recommendation

system directly into the game reflects a commitment to continually enhancing the user experience. The simplification of external menus underscores a strategic emphasis on user convenience and accessibility for players in WoW Classic Hardcore.

The personalized recommendations provided valuable guidance to players in WoW Classic Hardcore, enhancing their ability to make informed decisions in the game. This not only contributed to a more gratifying experience but also mitigated the risk of significant setbacks.

References

aaronma37. (2023). *Deathlog*. https://github.com/aaronma37/Deathlog

Adomavicius, G., Mobasher, B., Ricci, F., & Tuzhilin, A. (2011). Context-aware recommender systems. *AI Magazine*, *32*(3), 67–80. https://doi.org/10.1609/aimag.v32i3.2364

Blizzard. (2023). *2,961,039 permadeaths in classic hardcore!?!* https://www.youtube.com/watch?v=Jv0HYpGLXgA

Hug, N. (2015). *Surprise*. https://surprise.readthedocs.io/en/stable/index.html

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization.

Koren, Y., Bell, R., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. *IEEE*. https://doi.org/10.1109/ICDM.2008.22

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37. https://doi.org/10.1109/MC.2009.263

Lemire, D., & Maclachlan, A. (n.d.). Slope one predictors for online rating-based collaborative filtering. In *Proceedings of the 2005 siam international conference on data mining (sdm)* (pp. 471–475). https://doi.org/10.1137/1.9781611972757.43

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: Methods and strategies of web personalization* (pp. 291–324). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-72079-9_9