

When referring to users with variant gaming history, will novelty-optimized recommender systems recommend less popular items than baseline algorithms while maintaining relevance in the recommendations?

Konstantina Ellina

Abstract

Recommender systems for gaming platforms, such as Steam, frequently end up recommending more popular games, affecting the ability to find niche games for the users. In this project, it is examined if novelty-optimized algorithms reduce popularity bias, while preserving relevance in the recommendations, compared to some baseline algorithms, without forcing less popularity games to the algorithms. The two baseline algorithms used are Item-KNN and User-KNN. The three novelty-optimized algorithms are Item-KNN and User-KNN recommending only novel games and a hybrid model with a weight that tries to balance novelty and relevance. The metrics used for comparison and evaluation of these methods are novelty, recall, coverage, fairness, intra-list diversity, NDCG, and popularity, clearly showing trade-offs among the approaches studied, but in this study we focus more in novelty, popularity and relevance. The Item-KNN and User-KNN baseline algorithms indeed show increased popularity, while they do really well in recall and NDCG. However, the novelty-only methods and the hybrid algorithm show a slight improvement in novelty and less popularity, and they also do better even in relevance. These findings are examined in this paper, providing also insights about the different kinds of recommender systems tried.

1 Introduction

Nowadays, more and more recommender systems are used to provide solutions for giving to users personalized suggestions about items in their interests. Especially in the gaming industry, users are seeking item recommendations that fit with their desires and follow their preferences. However, many traditional recommender systems lead to popularity bias, where mostly the popular items appear in the final recommendations, frequently ignoring the potential of discovering unique and niche items. This problem can affect users with varied gaming histories that are willing to try new games.

Novelty can be a solution to this limitation. With the addition of novelty in the recommender systems and the popularity decreased, different

recommendations with games that the user has not played before can be provided. The users will then have more games they can explore on the platform and will be able to try something new to them. However, the trade-off between novelty and relevance is important to be taken into account. While novelty is increased, the relevance may be decreased, making the recommendations less aligned with the user's desires.

This project takes into account the concern about the popularity bias and how novelty could help with that, considering also the user's preferences, and focuses on answering the following research question: *“When referring to users with varied gaming histories, will novelty-optimized recommender systems recommend less popular items than baseline algorithms while maintaining relevance in the recommendations?”*

The scope of the project is to find the impact of novelty to recommender systems and if this addition to the well-known algorithms can produce less popular, but still relevant to the user’s preferences, games. The study tries to see how the balance to this trade-off can help improve user satisfaction and build more variant gaming platforms.

2 Literature

Novelty can be described in many ways, since it is a broad value and is used differently in many projects. In *Knowledge-Based Systems*(D. Kotkov et al., 2016), they mention four different meanings for novel items, but they use only one of them to continue with their research. Inspired by their paper, I am using the same meaning as them and referring to novel items as *“relevant or irrelevant items that a target user has never seen or heard about in his/her life”*.

In Kotkov et al. (2016), they mention the definition of serendipitous recommendations, which are novel and relevant items, but also unexpected for the user. Serendipity-based algorithms are frequently used to surprise the users and increase the user experience and engagement with the platforms. Also, in Ziarani and Ravanmehr (2021), they argue that serendipity is the one that can make recommendations meaningful and engaging to users, and not novelty. However, their study is focused on balancing accuracy and serendipity in algorithms. There, serendipity can be more useful than novelty, because relevance is important to be kept.

The difference between serendipity and novelty is discreet, but in this study we will refer only to novel items, meaning that the games will be unknown to the user, but they can also end up to be irrelevant. By using only novelty, we can clearly see how novelty, relevance and popularity perform in the different algorithms, without skewing the data to only relevant games, like serendipity would do.

3 Methodology

3.1 Collecting datasets and preprocessing

This section shows which datasets are used in this project and which methods of cleaning and filtering are applied on them.

3.1.1 Datasets

This study uses two datasets to build the recommender systems and analyze how they behave. These datasets are “train_interactions.csv” and “test_interactions_in.csv” and they represent the Steam platform. The datasets capture user interactions with the games available, showing which user has interacted with which games and how much they have played each game. The “train_interactions.csv” is used as the train dataset, and the “test_interactions_in.csv” as the test dataset.

3.1.2 Cleaning

After collecting the data, the preprocessing step begins. To make sure that the data have a good quality for their further usage, there are some cleaning steps applied for both datasets. The duplicate rows are removed and some checks about missing or invalid values are applied. Since the checks were clear and all the values are well-structured there is no further cleaning for the two datasets.

3.1.3 Sparse matrix

The two datasets are used to create sparse matrices for the interactions. The sparse matrix is a good solution for big datasets like these two, where the user-item interactions are not a lot, since most of the users have interacted only with a small part of the games. The use of Compressed Sparse Row(CSR) matrix is a memory-efficient way to hold sparse matrices, since they store only the non-zero values. The final matrices contain the ids of the users and the games as row and column, and the playtime of the users for the interacted games as its values.

3.1.4 Filtering

With the sparse matrix ready, a specific filtering is applied to the datasets. In this study, some users are filtered out to prioritize in them who have a significant interaction with the platform. Logically, users that have tried a lot of games already will want to try something new and they are the ones that will search for it in the platform. Users new in the platform or non-gamers will most usually play something that they have heard or seen somewhere, which makes this game more popular than the games we are looking for in this project. The users excluded are the ones with minimal gaming history, i.e. the ones that have played strictly less than 3 games and have less than 200 minutes in

their total playtime of games. They are considered outliers and keeping them could introduce noise in the dataset, because they don't provide the information that we want to take from the users. Focusing on active users, the data provided captures meaningful patterns and preferences to be used in the recommender systems later.

3.2 Dataset split

When the preprocessing is done, the test dataset is split to two smaller test datasets. The split is happening in half and we get one test dataset called *test.mat* and another one called *test.true*. This split is happening to evaluate the predicted recommendations from each algorithm with the true recommendations. In this case, the *test.mat* dataset is used in every algorithm for computing recommendations. The *test.true* is used for finding the ground truth for evaluation, including interactions for unseen data, and is going to be compared with the predicted recommendations that each algorithm found.

3.3 Recommender systems

The recommender systems used in this study include two baseline algorithms and three novelty-optimized algorithms, designed to see the trade-offs between novelty, popularity and relevance in the recommendations found. Using baseline methods can help with having a good reference to compare with, while the novelty-optimized methods can recommend items different than usual.

3.3.1 Baseline recommender systems

The baseline recommender systems used in this project are **Item-KNN** and **User-KNN**. These algorithms are simple, interpretable and well-known for recommendation tasks, being perfect for comparison with novelty-optimized algorithms. Item-KNN works by using the item-item similarities and finding similar items to those the user has liked in the past. Its recommendations come from items scored according to their similarity to the user's previous interactions with other items. By recommending the most similar ones, it ensures that the recommendations will respect the user's preferences. User-KNN works by using user-user similarities and finding items that similar users have interacted with before. This approach finds the "neighbors" of each user, i.e. those with same preferences, and recommends items that these "neighbors" have already tried. These two algorithms emphasize a lot in relevance, since the recommended items come

from similarity patterns, and in popularity, since the recommended items are similar to popular items in the dataset or are frequently interacted with by other users also.

3.3.2 Novelty-based recommender systems

The first two novelty-based algorithms used in this study are **novelty-only Item-KNN** and **novelty-only User-KNN**. Their implementation is really similar to Item-KNN and User-KNN baseline algorithms respectively with the only difference being that these two recommend only novel items to the users. These methods show how pure novelty changes the recommendations given in the corresponding baseline algorithms and how the trade-offs between novelty and relevance are. The algorithms prioritize games that are novel to the user, but still in their preferences or similar to their "neighbors".

The **Hybrid Novelty** algorithm is used as a balanced method to recommendations. Unlike pure novelty algorithms, this one balances novelty and relevance with the help of a weight variable. The main method is similar to Item-KNN as it uses the item-item similarity, but in this case it uses a weight. This weight has been tried to be 0.3, or 0.5, or 0.8. With weight=0.3, the algorithm introduces a bit of novelty in the recommendations, but relevance is still more important. With weight=0.5, we see a balance between relevance and novelty, since a bit more novelty is introduced while keeping relevance on mind. And with weight=0.8, the algorithm prioritizes novelty than relevance. This hybrid approach works as a "middle" algorithm from baseline Item-KNN and Novelty-only Item-KNN. If weight is 0, it results to having the same results as baseline Item-KNN, and if weight is 1, it results to having the same results as Novelty-only Item-KNN. It is used in the study to show how sensitive relevance is to novelty increment, and see how metrics behave when relevance is changed.

3.4 Sanity checks

When we get the results from all the algorithms, there are some sanity checks to see if the results provided are reproducible and not duplicated. First, reproducibility is checked by calling the algorithms three times to see if the results are going to be the same each time. And second, users are checked that they do not have duplicate recommendations. These checks are applied to all the algorithms and

Algorithm	NDCG	Recall	Coverage	Intra list	Gini Index	Publisher fairness	Novelty	Popularity
Item-KNN Baseline	0.1537	0.1681	0.5192	0.0851	0.8825	0.9047	0.9352	0.7927
User-KNN Baseline	0.1250	0.1757	0.2408	0.0678	0.8762	0.8931	0.9460	0.9497
Novelty-only Item-KNN	0.1680	0.1797	0.5256	0.0788	0.8785	0.9012	0.9370	0.7806
Novelty-only User KNN	0.1824	0.1989	0.2695	0.0574	0.8655	0.8869	0.9490	0.9395
Hybrid (More Relevance)	0.1627	0.1738	0.5210	0.0830	0.8811	0.9033	0.9357	0.7882
Hybrid (Balanced Relevance - Novelty)	0.1663	0.1776	0.5223	0.0804	0.8799	0.9025	0.9364	0.7847
Hybrid (More Novelty)	0.1679	0.1796	0.5227	0.0788	0.8785	0.9012	0.9369	0.7812

Table 1: Evaluation metrics for each algorithm tested locally

Algorithm	NDCG	Recall	Coverage	Intra list	Gini Index	Publisher Fairness	Novelty
Item-KNN Baseline	0.1145	0.1490	0.5192	0.2268	0.8825	0.9101	0.8285
User-KNN Baseline	0.0960	0.1640	0.2408	0.2750	0.8762	0.9052	0.8015
Novelty-only Item-KNN	0.1235	0.1571	0.5256	0.2097	0.8785	0.9068	0.8346
Novelty-only User KNN	0.1383	0.1815	0.2695	0.2411	0.8655	0.8992	0.8135
Hybrid (More Relevance)	0.1204	0.1530	0.5210	0.2205	0.8811	0.9091	0.8308
Hybrid (Balanced Relevance - Novelty)	0.1226	0.1558	0.5223	0.2140	0.8799	0.9082	0.8330
Hybrid (More Novelty)	0.1235	0.1571	0.5227	0.2099	0.8785	0.9070	0.8346

Table 2: Evaluation metrics for each algorithm tested in Codabench

since the checks are clean, we can be sure that the algorithms flow is correct.

4 Evaluation

For the evaluation of the algorithms above and for the analysis of the trade-offs considered, there are some metrics that are calculated:

- **NDCG and recall** for accuracy
- **Coverage and Intra List Similarity** for diversity
- **Gini Index** for item and publisher fairness
- **Item to History Distance** for novelty
- **Popularity measure** for popularity

From these metrics, the important ones for this study are **novelty**, which measures how well the system recommends new and unknown items to users, **recall and NDCG**, which measure the relevance of the items for users in the recommendations, and **popularity**, which measures how many recommended items are popular compared to the total recommendations. Popular items in this study are the items that their number of interactions is greater than or equal to the 90th percentile of all

items interaction counts in the dataset. These primary metrics with the combination of the other additional metrics can give a clear view for each algorithm’s performance and its trade-offs. The results of these metrics from local testing can be found in Table 1, and the results of Codabench in Table 2. The local results and the Codabench results have some differences, which could happen because of different split in test data or because locally I exclude some users as mentioned before in the Filtering section. However, both results conclude the same thing, so they can both be used for the study’s conclusion. This paper, when we are referring to results, we consider the Codabench results (Table 2), and add to them the popularity measure calculated in local testing. In Codabench, the last 7 files uploaded show the results that this paper shows. Their IDs are: 183612, 183611, 183610, 183579, 183549, 183480, 183462.

Item-KNN baseline

Item-KNN is a basic traditional algorithm used a lot for recommending items. Its high recall and NDCG can show that indeed the items recommended are relevant to the user’s preferences and the item-item similarities have played a crucial role to that. Also, the popularity is high enough showing the initiate consideration for baseline algorithms, which occur from similarities in items or users highlighting the most played games. However, a novelty score of

0.8285 indicates that it recommends some new unknown to the user games, while this is not its priority. The other metrics are normal for the specific algorithm with the intra-list similarity suggesting small diversity, coverage having a moderate score showing that a decent part of the items were discovered and suggested, while fairness and gini index are similar.

User-KNN baseline

User-KNN is another traditional algorithm for recommendations. Here, we see a novelty of 0.8015, and a high relevance as implied from the recall and NDCG. The popularity is really high, resulting in not being a good algorithm for providing niche games and most probably unknown to the users. The rest of the metrics are normal with coverage being moderate to low, fairness and gini index being high enough to say that the recommended items focus on a subset of similar users, and intra list suggesting small diversity in the items.

Novelty-only Item-KNN

This algorithm results in some improvement in the required metrics than baseline Item-KNN. It successfully prioritizes novel items with a small improvement in the score of the baseline algorithm Item-KNN. The recall and NDCG are higher showing that in this case the inclusion of novel games not only doesn't lose relevance, but it gains some relevance also. The popularity score is decreased, confirming that this change in the baseline algorithm helps with recommending less popular games. The rest of the metrics are really similar to the baseline Item-KNN. The comparison of the metrics between Novelty-only Item-KNN and its baseline is shown in Figure 1.

	Novelty	Popularity	Relevance
Item-KNN baseline	0.8285	0.7927	0.1490
Novelty-only Item-KNN	0.8346 	0.7806 	0.1571 

Figure 1: Performance of Item-KNN baseline and its extension

Novelty-only User-KNN

This novelty-based approach also achieves better results than its baseline algorithm, making it a better model for recommendations in this case. Even though the baseline has high novelty, this algorithm

increases it a bit more as expected from the implementation. The relevance is again higher than the User-KNN baseline, resulting to better performance of the model. And also, this approach recommends less popular games than the baseline, achieving the goal we are searching for. The other metrics are similar to the baseline User-KNN. The comparison between User-KNN baseline and its extension can be found in Figure 2.


	Novelty	Popularity	Relevance
User-KNN baseline	0.8015	0.9497	0.1640
Novelty-only User-KNN	0.8135 	0.9395 	0.1815 

Figure 2: Performance of User-KNN baseline and its extension

Hybrid novelty algorithm

Since this approach is supposed to work as an improvement to Item-KNN baseline, its comparison is only with Item-KNN baseline. Hybrid novelty algorithm's results differ a bit depending on the weight that balances novelty and relevance. In this case, the scores keep rising with small differences, until they reach at some point the Novelty-only Item-KNN results, if the weight is 1. This is normal since the Hybrid algorithm works as an intermediate method between Item-KNN baseline and Novelty-only Item-KNN to see how the metrics are changing. For weight=0.3, where relevance is prioritized, we have a bit higher relevance than Item-KNN and a slightly increased novelty score. Popularity of games is also decreased. As the weight is increased and novelty is more prioritized, both with weight=0.5 and weight=0.8, the metrics have the same behavior, with increased relevance and novelty and decreased popularity each time. The higher the weight, the better the results for novelty and popularity, and as not expected, the better the results for relevance too. The results of the metrics for all the weights tested in the model and their comparison with Item-KNN baseline can be found in Figure 3.

Overall comparison

Overall, while the novelty-only algorithms are really specific and a bit rough as recommender systems, in this dataset they perform really well and even though there are not extreme differences from the baselines, they complete the wanted results. Hybrid algorithm is tested to see how the

	Novelty	Popularity	Relevance
Item-KNN baseline	0.8285	0.7927	0.1490
Hybrid with weight=0.3	0.8308	0.7882	0.1530
Hybrid with weight=0.5	0.8330	0.7847	0.1558
Hybrid with weight=0.8	0.8346	0.7812	0.1571

Figure 3: Performance of Hybrid algorithm with different weights and compared to Item-KNN baseline’s performance

metrics change with the different weights used. It still gets better results than the baseline algorithm.

While the novelty is increased in the three novelty-based algorithms, as it was the main task for these algorithms, the recommendations contain less popular games and the relevance is increased than the baselines. The main concern of this study was to see whether with increasing novelty in algorithms, we would be able to decrease popularity and keep a decent amount of relevance, so the users would still get recommendations they would like. This is accomplished with these three novelty-optimized approaches, with each of them play a different role in the study. Novelty-only Item-KNN and User-KNN work as plain improvements of the baselines, and the hybrid method is used more for seeing how novelty, popularity and relevance behave when the weight is not only in relevance or in novelty. It could also be used as an improved method of the Item-KNN baseline with changes in the weight, depending on whether each platform wants to emphasize in recommending novel and relevant items or popular and relevant items.

5 Conclusion

In this study, five different recommender systems for Steam game platform were built to see if the novelty-optimized algorithms would perform better in novelty and popularity while keeping the same relevance in the recommendations than the baseline algorithms. The baseline methods, even though they had a strong performance in relevance, they promoted mostly popular games and not novel ones. In contrast, novelty-only algorithms prioritized their recommendations more in new and niche games, while not only maintaining relevance, but improving it, making them a good start for recommender systems that want to improve their basic recommendations and explore more metrics also. The hybrid algorithm is used to see how the metrics are changing and acts as a solution that could

be used for platforms that might not want to stand that much in novelty. They all perform better than the baseline algorithms if we want the recommendations to consider factors beyond simple relevance, providing a wider variety of information used for different purposes. They work as a good improvement of the baselines and can show that by improving novelty in the recommendations, we have lower popularity in them, without losing relevance, and in this case gaining some relevance also.

References

- [1] Kaminskas, Marius, and Derek Bridge. "Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems." *ACM Transactions on Interactive Intelligent Systems (TiiS)* 7.1 (2016): 1-42.
- [2] Kotkov, Denis, Shuaiqiang Wang, and Jari Veijalainen. "A survey of serendipity in recommender systems." *Knowledge-Based Systems* 111 (2016): 180-192.
- [3] Ziarani, Reza Jafari, and Reza Ravanmehr. "Serendipity in recommender systems: a systematic literature review." *Journal of Computer Science and Technology* 36 (2021): 375-396.