Board games group recomendation using metadata

Eduardo Salinas [1] **Alfonso Badilla** [1] **Nicolás Gutiérrez** [1]

Abstract

In this document we propose a group recommendation system for board games using data from the Board Game Geek website, which includes attributes such as the number of players, average playtime, complexity, and user ratings. The implementation employs Python and the scikit-learn library, offering an efficient and scalable solution for group-based game recommendations.

1. Motivation and state of the art

Board games have been a popular form of entertainment for centuries, and their popularity has only increased in recent years. Even as digital games have become more prevalent, board games have remained a favorite pastime for many people, and the market for board games has grown significantly giving rise to plataforms such as Board Game Geek[1] (BGG). This website is a popular platform for board game enthusiasts, where users can rate and review games, as well as access information about them.

1.1. State of the art and related work

This plataform has a vast amount of data that has been used in the past to create recommendation systems. For example, github user richengo (2021) proposed a recommendation system for board games based on user ratings, using a collaborative filtering approach. However, this approach does not take into account the characteristics of many of the games, that are supposed to be played in groups. As such, we propose a group recommendation system to make recommendations tailored to groups of players.

Another example of group recommendation systems is the work of (Peska et al., 2023), who proposed a group recommendation system for tourism and movies[2]. This system uses regular clustering algorithms to group users based on their ratings, such as user k-nearest, item k-nearest and support vector decomposition (SVD), and then recommends movies and touristic locations based on the preferences of the group.

1.2. Proposal

We propose a refined version of that last system, using metadata-centric clustering to group board games based on their characteristics, and then recommending games to groups of players based on their summed preferences used the functions detailed on the previous work. This system is designed to be more accuarate than the previous one, as it takes into account the characteristics of the games, and not just the ratings of the users.

2. Data analysis

The dataset that will be used for this project[3], obtained from the website kaggle, has 10 different files, each one tith different information about board games. The main file is `user_ratings.csv`, which contains 19 million rows. Due to the limited processing power of Google Collab and our own local machines, we will use a reduced set of items and users for this work. The other files are:

- `games.csv`: This file contains 22 attributes about the rated games and includes 22,000 different games. Among the attributes, we have aspects such as the game description, which could be evaluated using language models, the year of publication (which distributes as shown in figure 1), the average rating received, among others.
- `ratings_distribution.csv`: Contains the total ratings for each board game. In general, these ratings distribute as shown in figure 2.
- `temes.csv`: Contains the themes of each game as a binary flag. The 50 most common themes can be seen in the graph in figure 3.
- `mechanics.csv`: Contains the game mechanics as

[1]Department of Computation, Pontificia Universidad Católica de Chile, Santiago, Chile. Correspondence to: Eduardo Salinas <esalinasbarros@uc.cl>, Alfonso Badilla <alfonso.badilla@uc.cl>, Nicolás Gutiérrez <njgutierrez@uc.cl>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

[1](BoardGameGeek, 2024)

[2]This work is hosted on a github repo and is intended to be used as a basic tutorial.

[3](Wadkins, 2023)

Figure 1. Amount of games (y axis) by year of publication(x axis)

a binary flag. The 50 most common mechanics can be seen in the graph in figure 4.

- `subcategories.csv`: Contains the subcategories of each game in binary flag format.
- `artists_reduced.csv`: Contains information about which artist created a certain game in binary flag format. Only artists with more than 3 games created will be considered.
- `designer_reduced.csv`: Contains information about which designer designed a certain game in binary flag format. Only designers with more than 3 games created are considered.
- `publishers_reduced.csv`: Contains information about the companies that sell these games in binary flag format. Only companies that sell more than 3 games are considered.

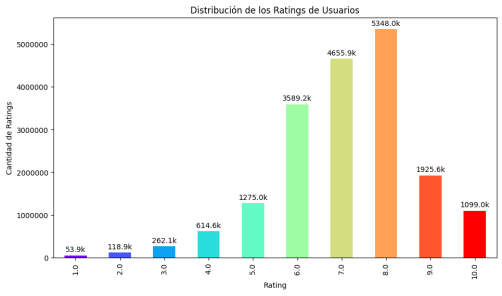

Figure 2. Games rating distribution (1-10)

3. Methodology

The methodology for this project is divided into three main stages: data preprocessing, group making, and recommendation.

For this project we will use the LightFM library[4] to make the recommendations, which is a Python implementation of a factorization machine for collaborative filtering.

[4](Kula, 2015)

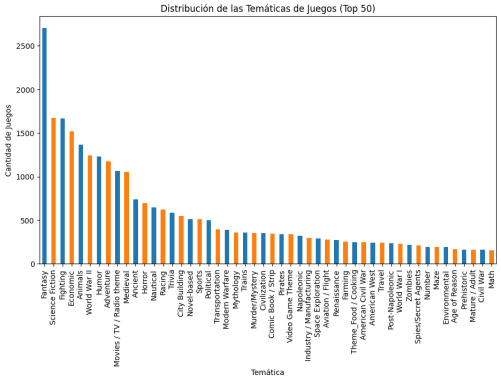

Figure 3. Most common game themes on the dataset

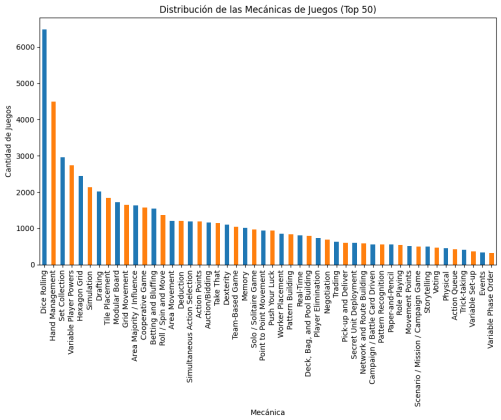

Figure 4. Most common game mechanics on the dataset

We will also use SVD models from the scikit-learn library[5] to compare the results of our model with a more traditional approach, and the most popular model as a baseline. SVD was selected above other models like k-nearest due to its usage as industry standard[6] and due to taking less time executing.

3.1. Data preprocessing

The first stage consists of reducing the dataset. This is important because the amount of data makes it take too long to calculate any meaningful metrics.

Part of the reduction is already done on the dataset by default, since there are no games with less than 20 ratings. Since we are going to also reduce the amount of users, we will delete games further up to 40 ratings, and then pick from 4 different game classes like the ones shown on figure

[5](Pedregosa et al., 2011)

[6](Roy & Dutta, 2022)

5.

With that in mind, we will reduce the amount of games to 1500, from the original >10k by taking 375 games from each class.

We will also reduce the amount of users to 15k, from the original >300k, first deleting all users with only 1 review. However, the classes will be more complicated to build, since we will be deleting a lot of games, so just making classes like in figure 6 will not be enough.

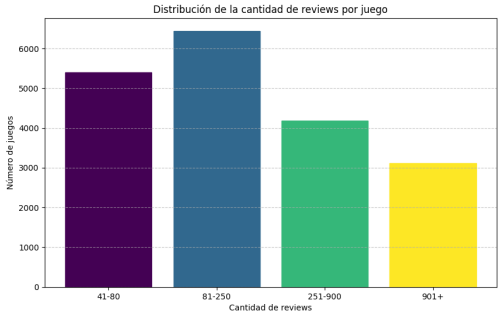

Figure 5. Amount of reviews per game

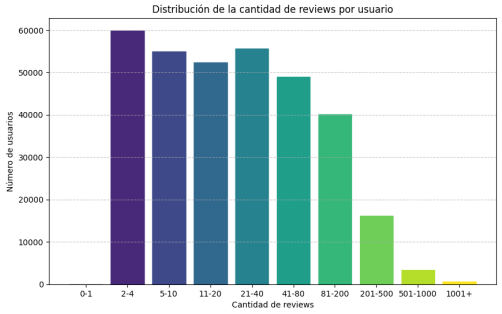

Figure 6. Amount of reviews per user (full dataset)

To do that, we will have to make the graph after selecting the games from each class. This graph can be seen in figure 7, and as seen on that figure, the classes have shifted due to less reviews overall. We will then make 3 classes as shown, and select 5k users from each class. Note that we will not select users with less than 3 reviews.

The final dataset will have 1500 games and 15k users, and will be used for the rest of the project. The dataset will be split into a training and a test set, with 75% of the data being used for training and 25% for testing.

The final amount of reviews per user and per game can be seen in figure 8 and figure 9 respectively.

With this, we expect to make a more efficient solution for group-based game recommendations, so that it can run with regular hardware.

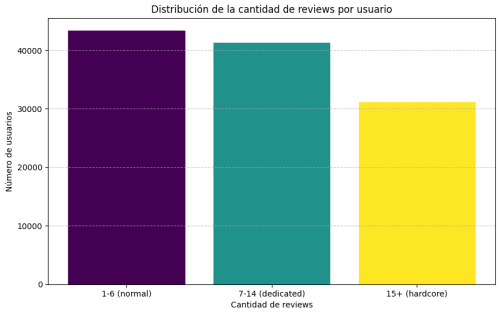

Figure 7. Amount of reviews per user (1500 items)

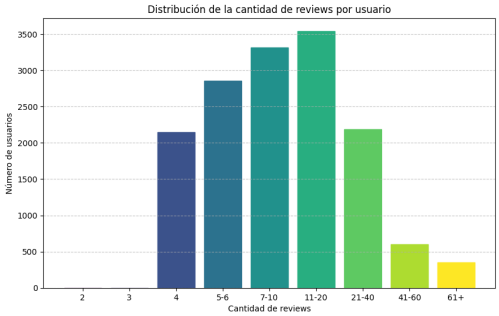

Figure 8. Amount of reviews per user (after full reduction)

3.2. Group making

The second stage consists of grouping the users based on their characteristics by using a correlation matrix. The function used is an adaptation of the one used in the "" work, which is based on the cosine similarity between the users. The function is as follows:

(1)

3.3. Recommendation

The third stage consists of recommending games to groups of players based on their preferences. Since the recommendations are supposed to be made to groups, the preferences of the group are calculated using a variety of methods, that can be described as follows:

1. Simple average: The preferences of the group are calculated as the simple average of the preferences of the individual players based on a top-100 recommendation list prior to the group formation. This can be done with the following formula:

$$\text{avg\_score\_item} = \frac{\sum_{i=1}^{n} \text{user\_rating}_i}{n} \tag{2}$$

with n being the number of users in the group, and u_i being the rating of the i-th user given an item.

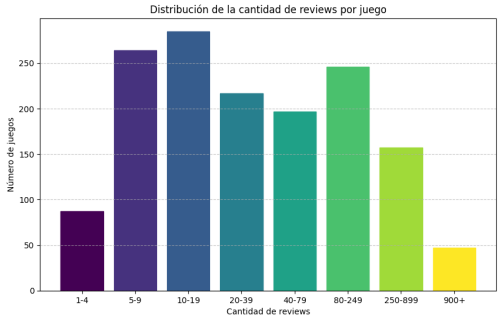

Figure 9. Amount of reviews per game (after full reduction)

Given the method explained above, we assign the score to each of the items in the top-100 recommedation list of each user on a group. Then we create a new list of items combining the top-100 recommendation lists of each user on a group, and we extract the top-n items with the highest average scores. Note that for this to work properly, if two users have a matching item in their top-100 recommendation list, the item will only be included as the sum of the average rankings by the users in the group.

4. Results

To evaluate the results of our model, we will compare it to most popular as a sanity check and to SVD applied to the same dataset and groups.

The metrics that will be used to evaluate the performance of each model are the following:

- Precision@k: This metric measures the proportion of recommended items that are relevant to the user, where relevance is defined as the items that the user has interacted with.
- Recall@k: This metric measures the proportion of relevant items that are recommended to the user.
- nDCG@k: This metric measures the normalized discounted cumulative gain at k, and is used to measure the quality of the recommendations.
- Relevance Score: This metric evaluates how relevant the recommendations are to the user, providing an indication of the recommendation quality.
- Diversity: This metric assesses the variety of the recommendations, ensuring a wide range of different items are suggested.
- Novelty: This metric measures how new or unexpected the recommendations are to the user, contributing to the overall recommendation quality.
- Fairness: This metric evaluates the fairness of the recommendations, ensuring that the recommendations are unbiased and equitable.
- Serendipity: This metric measures the likelihood of the recommendations to pleasantly surprise the user, enhancing the user experience.

For each metric, we used a cumulative avarage of the results of 300 different group formations, on which we applied the common formulas for each metric. Meaning that for each group, we calculated all of the metrics mentioned above, then we added them and divided by the number of groups.

As is shown on table 1, the LightFM model outperforms the other models in almost all metrics. This is to be expected, since its using not only user-item interactions but also metadata to make recommendations. The metadata recommendation allows for the model to recommend similiar items to two or more users that have similar preferences even if they have not interacted with any items in common. This is specially helpful when making groups recommendations, because it allows for greater diversity and novelty and creates a more personalized group experience.

One aspect of the recommendations that is worth mentioning is that the LightFM model is capable of superior results with more diversity. This means that all groups will get recommended different games, but they are tailored to the preferences of each user on a group. This in hand increases the novelty and the serendipity of the recommendations, creating more unique experiences for the users.

For each different group formation, the results are mostly the same. This is to be expected, since the group formation is based on the preferences of the users, and the recommendations are made based on the preferences of the group. Not on the preferences of the individual users. This is a good sign, beacuase it shows that the model does not overfit to any group formation and it is able to make recommendations based on the preferences of the group.

However, for new users, the results might not be as expected. The user needs to create user-item interactions in order to get proper recommendations. Untill then, the model should be able to recommend the most popular items, which are the most likely to be relevant to the user according to the very low information the model has.

Statistic	LightFM	SVD	Most Popular
Precision@5	0.3660	0.2020	0.3010
Recall@5	0.0942	0.0520	0.0766
nDCG@5	0.6232	0.4762	0.5903
Relevance Score	2.76	1.59	2.32
Diversity	0.03	0.01	0.01
Novelty	5.89	8.52	5.70
Fairness	0.60	0.72	0.66
Serendipity	4.51	2.80	3.63

Table 1. Random groups recomendation results

Statistic	LightFM	SVD	Most Popular
Precision@5	0.3350	0.2040	0.3240
Recall@5	0.0821	0.0498	0.0749
nDCG@5	0.5722	0.4736	0.5474
Relevance Score	2.57	1.62	2.48
Diversity	0.04	0.01	0.01
Novelty	5.99	8.60	5.70
Fairness	0.63	0.72	0.64
Serendipity	4.61	2.75	4.06

Table 2. Similar groups recommendation results

5. Sensitivity Analysis

6. Conclusions

7. Future work

Since this project was done with a reduced dataset, the obvious step would be to scale it up to the full dataset or at least a bigger portion of it with improved computation power. This would allow for more accurate results and a more robust set of models.

Another step would be to improve the group formation algorithm, as it is currently based on the preferences of the users, and not on the characteristics of the games they play (it does not use the available metadata). This could be done by using a clustering algorithm to group the games based on their characteristics, and then grouping the users based on the games they have interacted with. Future work could also use a hybrid approach, where part of how the groups are formed is random and another part is based on the similarity of the users.

Additionally, a notable drawback of this work is the lack of inclusion of the baseline "AGREE" recommended by the course instructors of IIC3633. Since the dataset selection process was finalized late in the development of this work, this implementation was not completed.

Finally, there is exploration to do regarding the use of the least misery and dictatorship aggregation functions, as well as the use of other aggregation functions that could be more suitable for group recommendations. This could be done by comparing the results of the different aggregation functions and selecting the one that provides the best recommendations. This barely didnt make it into the project, and could easily be implemented in the future.

References

BoardGameGeek. Boardgamegeek: The world's largest board game database, 2024. URL https://boardgamegeek.com. Accesed december 7th 2024.

Kula, M. Metadata embeddings for user and item cold-start recommendations. In Bogers, T. and Koolen, M. (eds.), *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, volume 1448 of *CEUR Workshop Proceedings*, pp. 14–21. CEUR-WS.org, 2015. URL http://ceur-ws.org/Vol-1448/paper4.pdf.

Ngo, R. Board-games-recommender, 2021. URL https://github.com/richengo/Board-Games-Recommender. Accesed december 7th 2024.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peska, L., Antonetti, L., et al. Group recommenders: Offline evaluation, 2023. URL https://github.com/barnap/group-recommenders-offline-evaluation. Accesed december 7th 2024.

Roy, D. and Dutta, M. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1):59, 2022. ISSN 2196-1115. doi: 10.1186/s40537-022-00592-5. URL https://doi.org/10.1186/s40537-022-00592-5.

Wadkins, J. Board games database from boardgamegeek, 2023. URL https://www.kaggle.com/datasets/threnjen/board-games-database-from-boardgamegeek. Accesed december 7th 2024.