Designing a Recommender System for Board Games

Michael Ion  
TU Wien, Austria

Dimitris Sacharidis  
TU Wien, Austria

Hannes Werthner  
TU Wien, Austria

ABSTRACT
Interest in board games has grown dramatically in the recent years, and so has the number of releases per year. Consumers can find it hard for themselves to choose the next board game to delve into, and often rely on curated lists and expert recommendations. At the same time, the broad availability of qualitative and quantitative data about board games makes this domain ripe for the application of automated recommendations. In this paper, we employ existing and novel techniques for recommending board games.

CCS CONCEPTS
• Information systems → Recommender systems.

1 INTRODUCTION
Sales and releases of new board games have been increasing in the last years by about 25% each year [3, 11]. One possible explanation of this boom is that people are looking for a pastime that doesn’t involve “staring at a screen”, as digital technology increasingly pervades everyday life and modern working environments. The total market is estimated at around $1.5 billion [4].

The vast number of releases increases diversity and quality of available board games. However, too many options can make customers experience “decision paralysis” when choosing what game to purchase next [9]. Therefore, it is interesting to investigate how recommender systems can be used in the domain of Board Games. The aim of this work is to design a system that provides board game suggestions. Its functionality is simple: the user inputs a collection of games that they enjoy, and the system then recommends several board games that are likely to interest the user. There are various signals from which the system can learn from, including the historical ratings, and the description of games (categories, mechanics, etc.). We consider several existing ideas as well as a hybrid approach, and use data publicly available to evaluate the ranking accuracy and degree of novelty and diversity of the recommendations. Our findings show that modern collaborative filtering (CF) techniques are highly effective but suffer in terms of novelty and diversity. In contrast, simple content-based (CB) approaches offer more diverse and novel recommendations. A hybrid CF-CB method is found to offer the best compromise in terms of the examined metrics.

2 DATA
An extensive source of board game data is the foundation of our recommenders. We have accumulated this data from the biggest online community for board games, BoardGameGeek.com. It has 4 million unique monthly visitors and 1.5 million registered visitors. Users contribute in forums, share industry news and upload images or write game reviews. Games are rated on a scale from 0 (worst) to 10 (best). Besides ratings, it also includes a wealth of descriptive attributes for each board game, such as the category and complexity of the game, involved game mechanics, or required playing time.

Over its publicly available API, we have extracted information on 80,474 games. This includes more than 13 million total ratings from about 249,186 different users. About 85% of users rated more than 50 games. This gives a healthy data set for recommendation purposes. Ratings per user are not evenly distributed. About 50% of ratings per user are given in the range of 6–8 in the 10-point-scale. Ratings are normalized per user. Another peculiarity is that the average rating of games increases over the years, implying released games get better with time. This might be linked to growing competition and better understanding of customer needs.

3 METHODS
The recommendation techniques used can broadly be classified as collaborative filtering, content-based, and hybrid approaches.

3.1 Collaborative Filtering
The following approaches are based solely on user ratings.

User - User. The basis of this approach is the computation of a similarity matrix of all users, calculated using mean-adjusted cosine similarity over the ratings of users [7]. We experimented with neighborhood sizes of 1, 2, 5 and 10 and found that a neighborhood of 5 users gave the best result.

Item - Item. Similar to the User - User approach, we consider all ratings ≥ 7.0 and calculate a similarity matrix of all games to each other by using mean-adjusted cosine similarity [8].

Matrix Factorization. This approach embeds users and items in a low-dimensionality latent space [5]. Training the model involves setting various hyperparameter values. The dimensionality of the latent space was set to 5. Number of epochs is 20, regularization strength is 1e-4, while the learning rate of stochastic gradient descent is set to 1e-4.

Autoencoder. Autoencoders generalize matrix factorization in that they learn non-linear embeddings (for users, items or both) using neural networks [10]. We implemented an autoencoder with a single hidden layer that contains 1,024 nodes. Number of epochs is 50, regularization strength is 1e-2, and batch size is 128.

After tuning, we found that best results are given by using 1024 hidden nodes, 50 training epochs processing batch sizes of 128. The Regularization term is 0.01. The Relu activation function and the adadelta optimizer give the best results.

Denoising Autoencoder. A denoising autoencoder adds distortion to its input in order to learn more robust embeddings [12]; the
best denoising factor is 0.1. The other hyperparameters are best set as in the basic autoencoder, except the number of epochs set to 100.

**Variational Autoencoder.** The variational autoencoder learns a latent statistical model from the input data and learns its parameters during training [6]. Binary cross entropy loss, rather than mean squared error, measures error. We find that the variational autoencoder needs less hidden nodes (256) than the other autoencoders. We use 50 epochs, batches of 64, and regularization of 1e-05.

### 3.2 Content-Based Recommenders

The following approaches are based on the attributes describing board games [2].

**k-Nearest-Neighbor.** For kNN, we use Categories, Mechanics, Playing Time, Weight, Minimum Players and Minimum Age. Categories depict the different genres of the games on a broader scale, e.g., Economic, Science Fiction or Negotiation. Mechanics are a more fine-grained characteristic and describe the gameplay elements of the board game, e.g., Dice Rolling, Modular Board or Set collection. There exist 84 different Categories and 51 different Mechanics. Both of these are turned into a binary attribute vector for distance calculation. Playing Time can have any value, but is most commonly around 30–180 (minutes). Weight is a measure of the complexity of the game’s rules. It is based on a vote by the BoardGameGeek.com community and ranges from 1 (least complex) to 5 (most complex). Minimum Players are often 2 or 3, Minimum Age is most commonly in the range of 8–14. All these attributes are equally weighted in the distance calculation. After the distance is measured, the results are sorted and the games with the k smallest distances returned. We do not include the attribute Family, as it has 2,761 different values and they are not very consistent — they can be very specific or very vague. We found best results using Euclidean distance.

**IDF-based.** The inverse document frequency (IDF) based recommender calculates the importance of each attribute, which is inversely related to its occurrence frequency (fitted to a logarithmic scale): rare categories are usually more descriptive of a game. For calculating IDF, we only use the Category and Mechanic attribute vectors of each game because of their categorical nature. We cannot calculate IDF based on continuous measures like Complexity or Playing Time or non-distinct values like Minimum Age.

IDF is very high for rare attributes and low for frequent attributes. This IDF value is assigned to each existing category value. This means, that all 84 distinct category values and each of the 51 mechanics values have an IDF value attached to them. Therefore, we have an IDF vector for categories and one for mechanics. The recommendation process is as follows. Let’s assume we want to have recommendations based on one input game. First, we get all the existing mechanics and categories of this game as a binary vector. We then process to create a bitwise AND operation with the binary mechanics and categories vectors of all other games. Now, we have all the mechanics and categories all games have in common with the input game. Then, we replace these with the IDF values for mechanics and categories. When we sum these together, we have a score for each game based on one input game. The higher the score, the higher the ranking of the recommended game. The games with the highest scores are output as recommendations.

### 3.3 Hybrid Recommender

We choose to combine a CF with a CB method. Specifically, our hybrid recommender can be seen as an extension of the previously presented autoencoder approaches. We combine ratings data together with data descriptive of the content of our items. In the training phase, the parameters of the hidden layers are still trained on a per user basis. The target is to most closely resemble the output layer, which still consists of the user’s real ratings of their chosen items. However, there is an extra layer now that abstracts additional information on the input side: This is categorical information describing the overall type of items the user prefers.

Categorical information has three types: Categories, Mechanics and Other Attributes. Other Attributes consists of Playing Time, Weight, Minimum Players and Minimum Age. For Minimum Players and Minimum Age we chose the minimum values as these are more restrictive (and more expressive) than the upper bound.

To make the recommender work, we had to compute values for this categorical information for each user. For the first two attributes, Categories and Mechanics, we used the concept of TF-IDF. For each user, we took the games the user rated above 7.0 as a basis for the calculation and computed a TF-IDF value of each possible category and mechanic. For the Other Attributes we simply calculated the average values of all games. For the recommendation phase, this means that those values need to be calculated for each set of inputs as well. We combined these three sets of categorical data with the ratings of the users. After tuning, we found that best results are given by using 1,024 hidden nodes, 50 training epochs processing batch sizes of 128. The Regularization term is 0.01.

### 4 Evaluation

**Methods.** We compare the various algorithms presented in Section 3 with a simple baseline, the popularity-based recommender that returns the most frequently played games.

**Evaluation Metrics.** We used the following metrics to measure the performance of our recommender systems.

- **Precision@k** presents the fraction of the relevant games in the recommended set to the total number of games in the recommended set. k is the number of retrieved items considered in this calculation.
- **Recall@k** is the fraction of relevant games in the recommended set compared to the total count of relevant games. k determines the size of the recommended set.
- **Normalized Discounted Cumulative Gain (nDCG)** uses a graded relevance scale to attribute the usefulness of recommended results based on their rank. This gain is based on the position in the result list - results on the top have a higher usefulness.
- **Average Precision (AP)** for a specific user is calculated as the average precision score from every recall value from 0 to 1. The mean of all APs is called MAP.
- **Novelty and Diversity** measure how similar are recommendations to the historically liked items of a user, and among each other, respectively [1]. Thus, they both compare two lists of items. We compute the pairwise distances of the games’ Mechanics and Categories to each other. These attributes are stored as boolean arrays. We use Hamming Distance to count the number of positions at which vectors differ; the higher its value, the more dissimilar the items are. In the computation of Diversity and Novelty, we divide
The essential difference between Diversity and Novelty is that the user neighbourhood; the item-based approach can only calculate to predict ratings for all items in the dataset, even if only few users results that are among the best for all our approaches, only beaten all improvement across all metrics. Precision@k and Recall@k show to all our approaches, these results are still moderate. k=100 it deteriorates to 0.130. Recall@k improves compared to the Item-Item slightly better. Specifically for Item-Item, Precision@k Recall@k and NDCG@k. Regarding neighborhood-based collaboration-based and neighborhood-based approaches in terms of Precision@k, have these games in their collection. Our baseline beats the content-based and neighborhood-based approaches in terms of Precision@k, NDCG@k and Recall@k. But still has overall subpar performance. The hybrid approach cannot improve over the non-content-aware autoencoders. This shows that ratings are a more expressive indicator of good recommendations (based on our metrics). Just including similar attributes does not necessarily bring added value when ratings are already involved.

Regarding the diversity and novelty metrics, we observe that the content-based IDF approach appears to be performing much better than all other approaches.

**REFERENCES**


