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 $\geqslant 7.0$ 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

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03.
https://doi.org/10.1145/3341105.3375780

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 proceed 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

Table 1: Evaluation results.

Recommender Type	***Collaborative Filtering***			***Autoencoder (AE)***			***Content***		***Hybrid***	***Base-line***
Rec. Name → Evaluation Type ↓	Memory (User)	Memory (Item)	Matrix Fact.	Classic	Den. AE	Var. AE	kNN	IDF	Hybrid	Popu-larity
Precision at 5	0.016	0.398	0.689	0.691	0.697	0.692	0.069	0.115	0.696	0.548
Precision at 10	0.014	0.379	0.628	0.640	0.651	0.648	0.068	0.106	0.639	0.544
Precision at 20	0.012	0.303	0.551	0.574	0.582	0.594	0.065	0.100	0.567	0.506
Precision at 100	0.011	0.130	0.364	0.404	0.408	0.418	0.040	0.072	0.403	0.361
Recall at 5	0.000	0.011	0.019	0.019	0.019	0.019	0.002	0.003	0.019	0.015
Recall at 10	0.001	0.020	0.034	0.035	0.036	0.036	0.003	0.005	0.035	0.030
Recall at 20	0.001	0.032	0.059	0.062	0.063	0.064	0.006	0.010	0.061	0.055
Recall at 100	0.006	0.066	0.184	0.206	0.209	0.214	0.019	0.034	0.206	0.188
NDCG at 5	0.002	0.129	0.339	0.339	0.339	0.317	0.024	0.046	0.339	0.244
NDCG at 10	0.002	0.138	0.337	0.340	0.340	0.316	0.024	0.046	0.339	0.259
NDCG at 20	0.002	0.133	0.335	0.337	0.332	0.319	0.026	0.049	0.335	0.264
NDCG at 100	0.003	0.108	0.331	0.344	0.343	0.337	0.028	0.057	0.344	0.284
MAP	0.023	0.017	0.017	0.022	0.022	0.073	0.000	0.005	0.021	0.006
Diversity	0.067	0.068	0.087	0.085	0.086	0.084	0.029	0.110	0.085	0.083
Novelty	0.076	0.075	0.084	0.083	0.084	0.082	0.063	0.109	0.083	0.081

the Hamming Distance by the total number of present attributes and then take the mean of all games. This is illustrated in Eq. 1. The essential difference between Diversity and Novelty is that the former represents the total of pairwise differences between items of the same set (Eq. 2), whereas the latter represents the pairwise differences in two distinct sets (Eq. 3). X and Y are binary attribute matrices. X is the list of recommendations. Y is the list of the user's liked items. The difference $X_i - Y_j$ denotes essentially the Hamming Distance between column j of matrix Y and column i of X.

$$Diversity/Novelty = \frac{\frac{\#DiffMechanics}{\#TotalMechanics} + \frac{\#DiffCategories}{\#TotalCategories}}{2} \quad (1)$$

$$Diff_{Diversity} = \sum_{i=1}^{k} \sum_{j=1}^{k} |X_i - X_j| \quad (2)$$

$$Diff_{Novelty} = \sum_{i=1}^{k} \sum_{j=1}^{l} |X_i - Y_j| \quad (3)$$

Train/Test Split. In order to produce training and test set, we randomly split the ratings of each user into 80% training data and 20% test data. We only consider users for training that have a minimum of 200 ratings, in order to draw useful conclusions.

Results. Table 1 summarizes the results. We make the following observations, first regarding ranking accuracy. Popularity performs rather well for such a simple approach. This is easily explainable: the most popular games are generally well rated, and lots of users have these games in their collection. Our baseline beats the content-based and neighborhood-based approaches in terms of Precision@k, Recall@k and NDCG@k. Regarding neighborhood-based collaborative filtering, User-User performs badly in all metrics, while Item-Item slightly better. Specifically for Item-Item, Precision@k vastly improves and is 0.398 for k=5 and still 0.303 for k=20. For k=100 it deteriorates to 0.130. Recall@k improves compared to the User-User approach and is the best for k=100 at 0.066. Compared to all our approaches, these results are still moderate.

Matrix Factorization based collaborative filtering yields an overall improvement across all metrics. Precision@k and Recall@k show results that are among the best for all our approaches, only beaten slightly by the autoencoders. The advantage compared to the previous user- or item-based approaches is, that it is always possible to predict ratings for all items in the dataset, even if only few users rated it (the user approach only takes ratings contained in a certain user neighbourhood; the item-based approach can only calculate correlation of existing ratings). This is due to the use of latent user and item vectors. The autoencoders have a similar approach, but their hidden layers seem to be an even better fit for modeling (non-linear) latent relationships in the ratings, explaining their slight evaluation improvements. In addition, NDCG is comparably better than in the previous approaches. MAP seems also to be mediocre indicating a not so good Recall/Precision tradeoff.

Overall, autoencoders exhibit the best ranking accuracy. They are comparable to those of matrix factorization, but a notch better. The basic autoencoder has the best values for almost all accuracy metrics. The slightly noisy input that is characteristic of the denoising autoencoder doesn't improve accuracy drastically. Precision@k and Recall@k results improve only by a very small margin. NDCG@k stays relatively the same as in the basic autoencoder. Similar observation hold for the variational autoencoder, which improves Precision@k and Recall@k and has the best MAE.

The content-based recommenders, given the relative short history they can learn from, fare worse than collaborative filtering. The IDF-based approach improves over the kNN in Precision@k, Recall@k and NDCG@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

[1] Pablo Castells, Saúl Vargas, and Jun Wang. 2011. Novelty and diversity metrics for recommender systems: choice, discovery and relevance. (2011).

[2] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. 2015. Semantics-Aware Content-Based Recommender Systems. In *Recommender Systems Handbook*. Springer, 119–159.

[3] Owen Duffy. 2014. Board games' golden age: sociable, brilliant and driven by the internet. *The Guardian* (2014). https://www.theguardian.com/technology/2014/nov/25/board-games-internet-playstation-xbox

[4] Milton Griepp. 2017. Hobby Games Market over $1.4 Billion in 2016. https://icv2.com/articles/news/view/38012/hobby-games-market-over-1-4-billion [Online; accessed 26-April-2018].

[5] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[6] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. *arXiv preprint arXiv:1802.05814* (2018).

[7] Ante Odic, Marko Tkalcic, Jurij F Tasic, and Andrej Košir. 2012. Relevant context in a movie recommender system: Users' opinion vs. statistical detection. *ACM RecSys* 12 (2012).

[8] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[9] Barry Schwartz. 2009. The paradox of choice.

[10] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

[11] Nick Wingfield. 2014. High-Tech Push Has Board Games Rolling Again. *The New York Times* (2014). https://www.nytimes.com/2014/05/06/technology/high-tech-push-has-board-games-rolling-again.html?_r=1

[12] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 153–162.