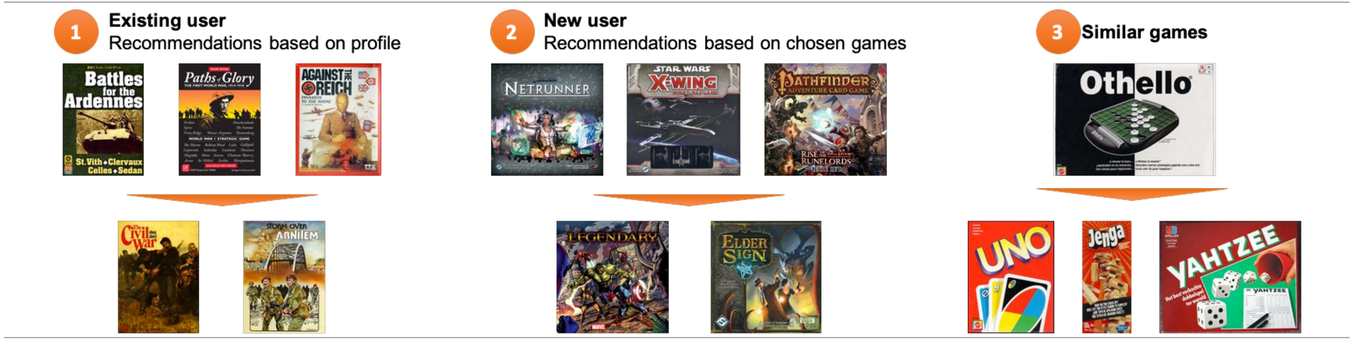
GetBoard: Recommendation Engine for Board Games

Escudero, Marco E
Georgia Institute of Technology
mescudero3@gatech.edu

Jimenez, Francisco J
Georgia Institute of Technology
fjimenez7@gatech.edu

Mai, Anna T
Georgia Institute of Technology
amai30@gatech.edu

Kumar, Ashok
Georgia Institute of Technology
akumar627@gatech.edu


1 Existing user
Recommendations based on profile
2 New user
Recommendations based on chosen games
3 Similar games

ABSTRACT

When confronted with the decision of choosing a table-top board game these days, a player is met with an endless amount of choices. With each game more different than the last. A board game could come in the form of: using cards, involving dice rolling, using game pieces, and even incorporating a phone application. This paper will explore how effective using a recommendation engine based on data known about board games is to help narrow the search for a new board game.

KEYWORDS

table-top board-games, recommendation engine, collaborative filtering, item similarity

1 INTRODUCTION

In recent years, table-top board games are becoming increasingly popular for all generations of gamers [9]. This has led to an explosion of choices, one such example is the website `boardgamegeek`, which lists close to 100k games. Therefore, it's becoming more difficult and time consuming for people to choose the next best board-game that they would want to play. Unlike digital media, the cost of a sub-optimal choice is much higher in the case of board-games. Our proposal is to build a recommendation engine using existing user play, collections, and ratings data to recommend a list of personalized board-games that match user interests with game attributes closely.

2 PROBLEM DEFINITION

Recommendation systems are a ubiquitous nature of our interactions on the internet. Advent of recommendations system has led to an increased diversity of products [12] and has helped the sales of relatively lesser known products [15]. While recommendation systems exist for various types of products and media, there is currently no recommendation engine specifically designed for board-games. As a result, a gamer is likely to spend significant amount of time in researching the next best board game that aligns with their interests. A personalized recommendation engine will eliminate this issue, and could boost demand and overall satisfaction.

`boardgamegeek` provides user-ratings, reviews, and other information for table-top board-games. Users can also record their game plays through `boardgamegeek` profiles. This data can be accessed via website's API.

We will leverage this data to build a recommendation engine.

3 SURVEY

Building a recommendation engine is a multi-step process that includes information collection, a learning phase, a recommendation phase, and a feedback loop. Different types of filtering techniques could be used for the learning phase such as: content filtering, collaborative filtering, and a hybrid filtering technique [8]. Content based filtering methods require user and item attribute vectors which may need to be built by hand. Collaborative filtering methods are able to use existing user and item interactions to learn a recommendation system [3]. Hybrid systems combine the two approaches and could produce an improved performance. Choosing the right method between collaborative filtering, regression models, or content based models is important to yield the best performance [10]. This guidance will benefit our application as we need to understand which model gives the best performance. A shortcoming is this article is that it does not compare other models which may best benefit our system. We have also reviewed an approach to leveraging collaborative filtering and clustering methods for a recommendation system for movies [13]. Following this approach can be a useful example during the design & development of our recommendation system.

An approach to label tweets as “good” and “bad” without relying on manually labeling tweets was presented in [6]. In this article the use of emojis and popular hash tags were used for training. An SVO weighting function is then used to recommend friends to the user based on their expressed attitudes (sentiment), the number of tweets written about a particular topic (volume), and the number of tweets that do not contain sent opinions (objectivity). While we cannot directly use the same training data for labeling, we can try to make use of sentimental words and board-game terminology in reviews to label our data and help save the time of manually labeling reviews. An approach of a classifier ensemble in analyzing sentiments of restaurant reviews [5]. We can utilize a similar classifier in our recommendation engines sentiment analysis. A shortcoming is that this study’s sentiment analysis may not apply directly to work with our data.

More recently deep learning methods have been applied to the purpose of build recommendation engines [14]. Progress has also been made in applying reinforcement learning to board games [2]. Although this is an interesting article, we will not be creating a reinforcement learning agent as it may be too complex to implement within the time frame, and classical methods seem to give a desirable performance. Crab [1] is a recommendation engine framework that utilizes Python’s SciPy, Numpy and Matplotlib. We can use it for the machine learning engine in our application.

We also surveyed few articles on the use of cutting edge methodologies into making successful interactive graphical user interfaces [11]. In order to make sure our application user interface is intuitive and promotes ease of use, we need to understand what the key components are to deliver a successful interactive user interface. Although we can’t tailor an application to work easily for all people, we will attempt to tailor ours to the gamer audience of users. Different perspectives on improving visual interface designs [7]. Learning these perspectives will guide us during our applications visual design. A shortcoming can be that the perspectives are not up to date with current web development styles. [4] describes the human psyche around board game interactivity. In order to design an application centered around how our users choose board games, we’ll need to understand human psychology related to board game playing. Although since human psychology can reveal an enormous amount of detail per individual, we can only tailor our app to a generalized representation of our gamers and not satisfy every human being.

We also studied how reviews are impactful to the sales of lesser known products [15]. This article gives us insights into product reviews and how they impact consumer purchases and board games. A shortcoming could be that we won’t get to use these insights in the development of our application. Impact of recommendation systems on consumer markets [12]. This will be useful in deciding how we will measure our success. A shortcoming is the article suggests recommendation engines recommend the most popular titles usually to the consumer which we want to avoid any bias in our application.

4 PROPOSED METHOD

4.1 Intuition

Currently, users rely on game suggestion engines on popular retail websites such as: Amazon, Target, GameStop, etc. These websites provide game suggestions based on multiple criteria: number of purchases, number of views, average customer ratings. However, these suggestions are not tailored to board games specifically. Further, unless the user has purchased all of their games from the same website, they would not have a full view of user's interests. Our solution will provide more relevant suggestions as we will be able to use much richer data from `boardgamegeek`.

4.2 Algorithms

There are various methods for building recommendation engines [8], the two broad classes being content-based filtering and collaborative filtering methods. A content-based filtering algorithm matches user interests with game attributes. To do this, a user preference vector (f_{u_i}) is built for each user, u_i, and a game attributes vector (f_{g_j}) is built for each game, g_j. These vectors will include information regarding genres, playtime, recommended ages, etc. These vectors will need to align with each other, e.g., an element for user vector could capture whether the user likes playing card-based games and the corresponding element for game vector will be if the game is card-based. A similarity score will then be calculated using a metric like Jaccard (shown in 1) for each user and game combination. Games with highest similarity score, excluding the games that user has already played, will be recommended to the user.

$$\text{Sim}(u_i, g_j) = \frac{f_{u_i} \cdot f_{g_j}}{\|f_{u_i}\| \|f_{g_j}\|} \tag{1}$$

Collaborative filtering, on the other hand, will use user-game interactions (ratings, collection or gameplay) data to build a model. There are three main approaches in collaborative filtering: item-item similarity, user-user similarity and matrix-factorization. Let's say there are n users and m games. $R \in \mathcal{R}^{n \times m}$ is the user-game ratings matrix. The user-user similarity method starts by constructing an user-similarity matrix, $U \in \mathcal{R}^{n \times n}$ that depicts the similarity between each combination of users. Cosine similarity and Jaccard measure are popular choices for the similarity metric. The recommendation score for user a for game b is given by $S(u_a, g_b) = \sum_i U_{a,i} R_{i,b} / \sum_i U_{a,i}$. Here, $U_{a,i}$ is the similarity score between users a and i, $R_{i,b}$ is the rating for game b by user i. The item-item similarity operates in a similar way but starts with constructing an item-similarity matrix, $G \in \mathcal{R}^{m \times m}$.

Another approach within collaborating filtering framework is called matrix-factorization. The user-game ratings matrix (R) has many blank values. Matrix factorization method tries to approximate the user-game ratings matrix using lower rank matrices consisting of latent features for users and games. Let's say the user latent features matrix is $L_U \in \mathcal{R}^{n \times k}$ and the game latent features matrix is $L_G \in \mathcal{R}^{m \times k}$. Then a minimization problem (2) could be solved to to estimate the latent feature matrices. Regularization terms could also be added to the optimization problem. As the optimization problem is non-convex, the solution cannot be reached in polynomial time but there are multiple numerical methods to solve it like singular-value decomposition (SVD), non-negative matrix factorization (NMF) and alternating least squares method (ALS).

$$\min_{L_U, L_G} J(L_U, L_G) = \|R - L_U L_G^T\|_2^2 \tag{2}$$

For building the recommendation engine, we explored multiple approaches. We concluded that while it is possible for us to build a content-based recommendation engine, it is not deemed as accurate as a collaborative filtering-based approach. We plan to use a train and test split approach to compared different modeling approaches in the collaborative filtering framework through evaluation metrics like precision, recall and RMSE. We also plan to experiment with hybrid approaches in order to improve the performance.

4.3 User Interface

We are planning to build a web-based interface for prospective users to interact with our application. Our user interface will have two designated workflows depending on if the user has existing information on `boardgamegeek` or not.

The first flow will be designated for existing users of `boardgamegeek`. They will start by entering their `boardgamegeek` username. They will be presented with their existing game play and collections data using the API. Recommendations will then be generated based on the pre-trained machine learning engine. The second flow for new users will provide them with a list of games to rate. After they rate these games, they will have recommendations generated based on the pre-trained machine learning engine.

In both these workflows, users can add additional constraints like game-cateogry, game-mechanic, play-time, number of players, etc. Users can also search and add more games to their lists to refine their recommendations. Users can also select a particular game and request for games similar to the selected game.

Screenshots of the user-interface are available in the README file, and can also be accessed here.

5 PLAN OF ACTIVITIES

At the beginning of the project, our action plan was divided into four main initiatives: data collection, building recommendation engine, building user-interface and integration. During the duration of the project, we met multiple times a week for brainstorming and working sessions. In the last few weeks, we were able to collaborate well to integrate various parts. We used `github` for hosting our code. All team members have contributed roughly the same amount of effort. We divided ourselves into two teams to work on these tasks in parallel. Our full plan is depicted in figure 1.

6 EXPERIMENTS & EVALUATION

6.1 Data

The website `boardgamegeek` provides an API access to various types of user and game data. For each user, the following info can be accessed: games that they own, their ratings and comments for games and the number of times they have played a game. For each game, we can access its release year, genres, game-family, game play time, recommended age, and other info. Our goal was to get user-interaction data that could be an input to our model. Our first approach was to build a list of games and then collect the user data for each game. But this approach would have meant making hundreds of API calls per game. So instead, we decided to build a list of users. But there is no API for pulling data by user. Fortunately, there is a webpage where one can search for users located in a zip-code. There are more than 33k zip-codes in the US. We picked the top 10k based on population (accounting for 85% of total US population) and then scrapped all users within 25 miles of each zip-code. This yielded us about 36k unique users. The API allows querying different types of data associated with each user. It includes their board-games collection, game-play information, ratings and comments for board-games. Extracting this data took about 4 days as we limited ourselves to about 1 call per 2 seconds. This yielded us more than 7.7 million user-game combinations with over 60% of having user-ratings. A user on average had rated more than 150 games. Further, we extracted about 1.2 MM user comments. The data has more than 62k unique games. The distribution of ratings is illustrated in figure 2

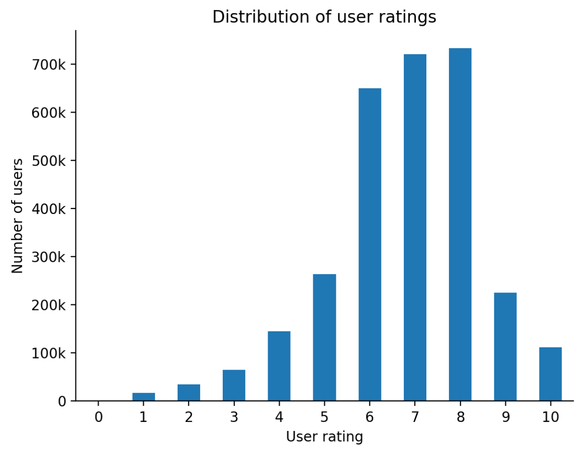


Figure 2: Distribution of ratings

6.2 Model Building

For building a collaborative filtering model, we explored multiple Python libraries and also explored building a model from scratch. Our first option was to use `Crab` but it is not actively maintained anymore. `surprise` is also a popular library but it does not take into account content based information like game genre etc. Finally, we chose `turicreate`. It is the open source version of `GraphLab` that was acquired by Apple Inc. in 2016.

`turicreate` has implementations for a number of collaborative filtering methods including item-item similarity, matrix-factorization and also a popularity based recommendation engine. To compare the performance of different models, we set up a process to compare the predictions from different models. To set this up, we first split our data-set into train and test by users. Once

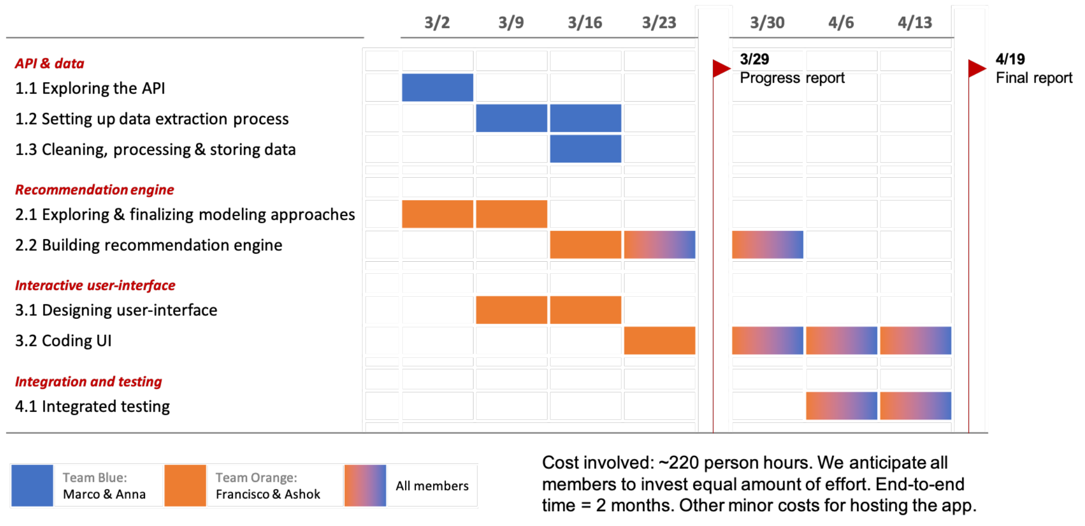


Figure 1: Project plan and timeline

the model is trained on train data-set, it can be used to make recommendations on the test data-set. The recommendations can be compared with actual ratings with metrics like precision, recall and RMSE.

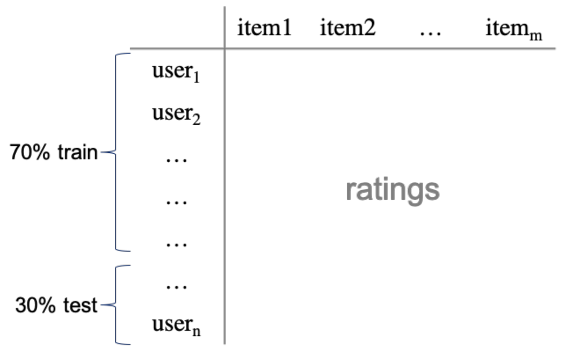


Figure 3: Train test split of data-set

Precision is defined ratio of relevant, retrieved recommendations and retrieved recommendations. Recall is defined as ratio of relevant, retrieved recommendations and relevant recommendations. If the set of actual items liked by a user are a and the set of top k recommendations is r_k, then precision and recall at cutoff k are defined by equation 3. The RMSE is defined by equation 4 where a_{ij} and r_{ij} are the actual and recommended ratings for user i and item j combination, respectively.

$$\text{Precision}_k = \frac{|a \cap r_k|}{|r_k|}, \text{Recall}_k = \frac{|a \cap r_k|}{k} \tag{3}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \left(r_{ij} - a_{ij}\right)^2}{mn}} \tag{4}$$

6.3 Experiments & Results

The library also provides methods for evaluation and comparison of different approach. We compared the models on precision, recall and RMSE measures. Our conclusion is that the matrix-factorization method yields the best performance. Comparison based on RMSE is shown in figure 4. Results are consistent when using other metrics like precision and recall. However, the item-similarity method has the distinct advantage that it can find games similar to a given game. Below we present some experimental results.

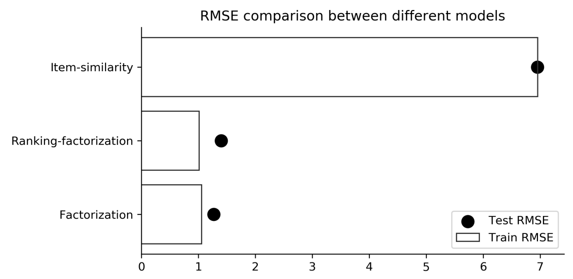


Figure 4: Comparison of algorithms

Example 1: Existing user
We take the example of xatsmann who is an existing user with over 100 games in their profile.

- Top current games: Istanbul, Against the Reich, Battle for Germany
- Top recommended games: Rise and Decline of the Third Reich, A House Divided: War Between the States 1861-65, The Civil War

The recommendations make strong intuitive sense. Both existing and recommended games are related to war-based strategy genres.

Example 2: New user who chooses only one game

- Chosen game: Scrabble
- Top recommended games: Catan, Carcassonne, Lost Cities

When a new user has chosen only one game, then the recommendations are biased towards popular games.

Example 3: New user who chooses multiple games

- Chosen games: Scrabble, Uno, Monopoly
- Top recommended games: Chess, Apples to Apples, Risk

As the amount of information fed to the recommendation engine is increases from 1 to 3 games, the recommendations make more intuitive sense. Both chosen and recommended games are in the family-friendly genre.

7 INNOVATION

One of the core sources of novelty in our application is that it applies the concept of recommendation engines to the domain of board-games for the first time. Furthermore, we are using a user's entire data for building the engine that includes collections, game-play, ratings and comments data. Our application will be useful for both users with existing data and users with no data. Both types of users will be able to increase the efficacy of results by adding more information. We believe that this innovation will be of immense help to gamers to discover new games.

8 CONCLUSION & DISCUSSION

Our goal was to build a recommendation engine to recommend new board games based on game play data that we knew from the user, whether they were a user from boardgamegeek or a completely new user. Our User Interface is able to handle recommending new games to existing users of boardgamegeek or users who do not have a account with boardgamegeek. In addition users are allowed to refine their recommendations by various filter criteria. We found the best model to use to build a recommendation engine was using item-item similarity. The RMSE comparison to other models we tried like ranking factorization and factorization align with our conclusion.

With the limited time, while we were able to implement a minimum viable product but we see a number of areas that could benefit from further work. From the perspective of improving the efficacy of the recommendation, we could do further research on state-of-the-art algorithms, especially in the field of deep learning. It may also be worthwhile to explore building hybrid models that can take the best of different approaches. We have rich user comments data which can be used to further fine tune the results. From the perspective of improving the user experience, we could add improvements like elastic-search, more information about games such a description popup windows or links to youtube tutorials to make our website a more wholesome experience, etc.

Overall, in this current form, this project should help board gamers in discovering new games that will help in fostering their love for gaming.

REFERENCES

[1] Marcel Caraciolo, Bruno Melo, and Ricardo Caspirro. 2011. Crab: A Recommendation Engine Framework for Python. Citeseer.

[2] Imran Ghory. 2004. Reinforcement learning in board games. *Department of Computer Science, University of Bristol, Tech. Rep* 105 (2004).

[3] Rafael Glauber and Angelo Loula. 2019. Collaborative Filtering vs. Content-Based Filtering: differences and similarities. arXiv:cs.IR/1912.08932

[4] Fernand Gobet, Jean Retschitzki, and Alex de Voogt. 2004. *Moves in mind: The psychology of board games.* Psychology Press.

[5] M Govindarajan. 2014. Sentiment analysis of restaurant reviews using hybrid classification method. *International Journal of Soft Computing and Artificial Intelligence* 2, 1 (2014), 17–23.

[6] Davide Feltoni Gurini, Fabio Gasparetti, Alessandro Micarelli, and Giuseppe Sansonetti. 2013. A Sentiment-Based Approach to Twitter User Recommendation. *RSWeb@ RecSys* 1066 (2013).

[7] Chun-Cheng Hsu. 2011. Factors affecting webpage's visual interface design and style. *Procedia Computer Science* 3 (2011), 1315–1320.

[8] FO Isinkaye, YO Folajimi, and BA Ojokoh. 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal* 16, 3 (2015), 261–273.

[9] Dan Jolin. 2016. *The rise and rise of tabletop gaming.* https://www.theguardian.com/technology/2016/sep/25/board-games-back-tabletop-gaming-boom-pandemic-flash-point

[10] Andreas Mild and Martin Natter. 2002. Collaborative filtering or regression models for Internet recommendation systems? *Journal of Targeting, Measurement and Analysis for marketing* 10, 4 (2002), 304–313.

[11] Brad A Myers and William Buxton. 1986. Creating highly-interactive and graphical user interfaces by demonstration. *ACM SIGGRAPH Computer Graphics* 20, 4 (1986), 249–258.

[12] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. 2014. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web.* 677–686.

[13] Phongsavanh Phorasim and Lasheng Yu. 2017. Movies recommendation system using collaborative filtering and k-means. *International Journal of Advanced Computer Research* 7, 29 (2017), 52.

[14] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[15] Feng Zhu and Xiaoquan Zhang. 2010. Impact of online consumer reviews on sales: The moderating role of product and consumer characteristics. *Journal of marketing* 74, 2 (2010), 133–148.