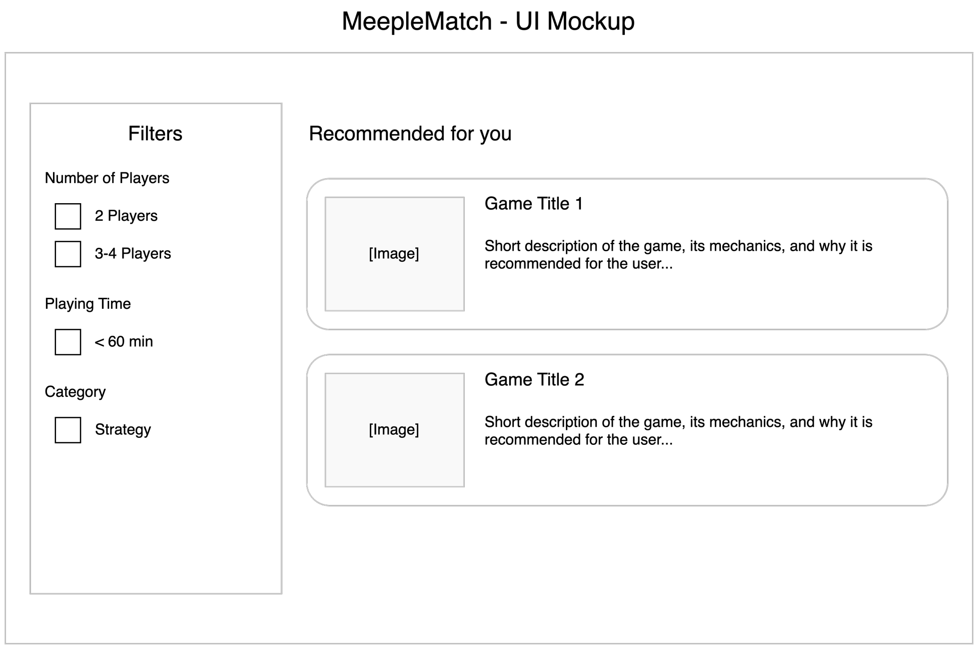
MeepleMatch - A Hybrid Recommendation Engine for Board Games

Abstract

This project aims to design and plan a personalised board-game recommender system named "MeepleMatch". The system will apply and evaluate several recommendation algorithms, combining them in a hybrid model that deals with cold-start issues for new users and new games in the hobby. The design document below outlines the system's scope, expected data sources, overall methodology, a clear evaluation framework, and a high-level timeline for implementation.

1 System Scope and Design

MeepleMatch is designed to give tailored game suggestions to all kinds of board-game fans, from newcomers hunting for their first title, to experienced players looking for new challenges. Recommendations will show up on a simple web page that arranges 5 to 10 titles in ranked order, making it easy for users to scan and click. The main interaction includes a broad filter and a live feed of cards, as shown in Figure 1.



MeepleMatch - UI Mockup
Filters
Number of Players
2 Players
3-4 Players
Playing Time
< 60 min
Category
Strategy
Recommended for you
[Image]
Game Title 1
Short description of the game, its mechanics, and why it is recommended for the user...
[Image]
Game Title 2
Short description of the game, its mechanics, and why it is recommended for the user...

Figure 1: User Interface Mockup for the MeepleMatch Recommendation Page

User feedback shapes the engine's ongoing learning cycle and system updates. After every suggestion, users can rate the recommendation on a 1-5 scale, save games into their wishlist, dislike the content, or briefly explain why a title missed the mark. Ratings and comments feed back into the model, helping future queries while protecting user privacy.

To tackle the cold-start problem in a fast-moving environment, the team proposes a set of targeted solutions. When a new player signs up, a short taste-profile quiz helps map their likes and dislikes. Whenever a new game is added to the catalog, its text and tag metadata fuel a content-based algorithm that spins out first-look recommendations. The overall model then undergoes an offline refresh each night, keeping its suggestions timely and on-point. Revenue will come through affiliate marketing, letting players jump from a winning tip straight to purchase via smooth e-commerce links.

2 Dataset Analysis

For this work, the team will draw on data from the BoardGameGeek (BGG) community, scraped fresh via the tool found at `https://gitlab.com/recommend.games/board-game-scraper`. Three core files make up the dataset: one holds rich metadata for each board game, a second tracks user ratings over time, and a third lists user details like IDs and activity dates. Together, these pieces form a sturdy backbone for the recommendation engine, with key fields and their roles laid out in Table 1.

Table 1: Key Fields from the BoardGameGeek Dataset and Their Applications

Filename	Key Fields	Description	Application in Model
GameItem.csv	boardgame-category, boardgame-mechanic	Category and mechanism tags	Feature vector construction for content-based filtering
GameItem.csv	minplayers, maxplayers, playingtime	Player count, duration, etc.	Used as filters and knowledge-based rules
RatingItem.csv	bgg_user, bgg_id, rating	User ratings for games	Construction of the user-item matrix for collaborative filtering
UserItem.csv	bgg_user, country, state	User's location information	Can be used for demographic analysis or regional recommendations

By using a web scraper, the system sidesteps the limits of static datasets. It pulls in fresh games and user ratings on the fly, so the data reflects the latest buzz in the board-game world. This living dataset is key to keeping the recommender up-to-date and useful. Of course, working with real-time, messy data also means dealing with duplication, typos, and missing values; those headaches will be tackled during the pre-processing step.

3 Methodology

To build a sturdy recommendation engine, this project plans to try out a range of algorithms and then weigh their strengths, with the end goal of packaging the best parts into a single, powerful hybrid system.

The first step uses content-based filtering as a baseline. It builds a detailed feature vector for each game by tallying category and mechanism tags, borrowing the TF-IDF trick to score each term. To make a recommendation, the system checks cosine similarity between a user's top-rated games and the pool of candidates. This approach is easy to explain and softens the sting of new items lacking past ratings.

Next, the system will add user-based collaborative filtering. In this step, a group of similar users, or k-nearest neighbors, is found by measuring how closely their ratings line up using the Pearson correlation. Recommendations for the target user then come from the liked items of these neighbors. This approach often surfaces surprising suggestions that a simple item-to-item match might miss.

The system will combine the two signals into a weighted hybrid model. Scores from the content-filter and the collaborative-filter are blended through the equation:

$$\text{Final\_Score} = \alpha \cdot \text{Score\_Content} + (1 - \alpha) \cdot \text{Score\_CF}$$

Here, Score_Content and Score_CF stand for the raw output from each engine, while α controls their balance as a tunable slider. By adjusting α, the design can favour one method over the other; for example, newcomers with few ratings may receive more help from the content line until their profile builds up.

4 Evaluation Framework

To gain a well-rounded picture of how the recommender system performs, we will blend offline tests with a short online user study.

In the offline phase, we will set aside 20 percent of the data while training the model on the remaining 80 percent, then apply a standard battery of evaluation measures. The primary focus will be on Top-N metrics-Precision@k, Recall@k, and Mean Average Precision (MAP)-which gauge not only whether relevant items appear but also their order on the list (Ricci, Rokach, & Shapira, 2011). To keep an eye on rating prediction, we will also report Root Mean Squared Error (RMSE), although ranking quality is the more urgent concern. Finally, results from all measures will be used to compare algorithms along the axes of accuracy, novelty, and diversity.

Moving to the online part, we plan a simple, low-impact user study with 5 to 8 volunteers who play board games at different levels. After logging into a clickable prototype, each user enters a few preferences and receives a tailored set of suggestions. Participants then complete a brief survey that asks about overall satisfaction, freshness of items, and how relevant the selections feel. Responses from this study will give us direct insight into user-centered qualities the numerical scores alone cannot capture.

5 Project Timeline and Milestones

To keep our work on track, we are suggesting the following draft timeline. Once the team is in place, we'll refine these dates further.

- Weeks 6-7: Data Processing and Baseline Model. Clean, pre-process, and explore the BGG dataset, then build and test a simple content-based filtering model.

- Week 8: Collaborative Filtering and Hybrid Model. Code the collaborative filtering algorithm and start linking it to the content model; run early tests with different weighting schemes.
- Week 9: Model Tuning and Evaluation. Carry out a set of offline experiments to compare models, find the best one, and fix its parameters; at the same time, lock in the user-study design.
- Week 10: Final Evaluation and Reporting. Run the user study, gather and analyze feedback, then pull together all results and draft the final report and presentation.

References

1. BoardGameGeek, “BoardGameGeek - Gaming Unplugged Since 2000,” 2023. [Online]. Available: `https://boardgamegeek.com/`. [Accessed: Dec. 2023].
2. F. Ricci, L. Rokach, and B. Shapira, Introduction to Recommender Systems Handbook. Boston, MA: Springer, 2011.