Board Game Recommender

Joep Kapma JOEP.KAPMA@STUDENT.AUC.NL
Amsterdam University College

Brook Sorensen BROOK.SORENSEN19@GMAIL.COM
Amsterdam University College

Iason Matiatos IASON.MATIATOS@STUDENT.AUC.NL
Amsterdam University College

Abstract

We created the "Board Game Recommender" (BGR) software that uses written text by users highlighting their preferences over board games to return three appropriate suggestions. Through BERT-fuelled keyword extraction, the BGR isolates a keyword out of the user's inputted text. Then, using cosine similarity, it maps these to the our dataset's categories. Finally, it returns the best rated, most rated, and one randomly chosen board game that fulfils the user's preferences.

Keywords: Recommender Systems, Board Games, User Preferences

1. Background

1.1. Literature Review

Recommender systems have been used at an increasing pace since the 90's, mirroring the expansion of information and content available online (Konstan & Riedl, 2012). By recommender system (RS) we will refer to any system that helps users find products from a given set (Felfernig et al., 2014). RS applications today include streaming services (Netflix or Spotify), social media (Facebook), online shopping websites (Amazon), dating services (Tinder), inter alia (Rocca, 2019). Behind the widespread use of these systems lie the same root causes: information size and preference customization. In platforms containing an overwhelming amount of choices, there needs to be an automated sorting system (Isinkaye et al., 2015).

Most recommender systems at present rely on machine learning methods. Broadly, these are either content-based filtering - selecting products of similar content to the ones a user likes - or collaborative filtering - presenting content that was liked by other users with similar preferences - or a hybrid of both (Isinkaye et al., 2015). In our case, there exists no publicly available database of user data for board games, which makes it impossible to use collaborative or hybrid filtering. However, the website that holds all the board game and user data, boardgamegeek.com, does employ this type of filtering, as do small projects we found online, which rely on individual requests for a small sample of user data (Alden, 2019; Zalewski et al., 2019). Overall, existing attempts of board game recommenders request current board games as input and use advanced machine learning methods to find similar games (Woodward & Woodward, 2019; Ngo, 2021; Ion, 2018).

1.2. Our project's original contribution & relation to Text Mining

Out of necessity, we used content-based filtering. We asked users directly for their preferences which were used as filters. This is not unusual. Existing websites employ similar filters (Woodward & Woodward, 2019), while requesting direct user input is often used in recommender systems, albeit for feedback and evaluation purposes rather than initial selection (Isinkaye et al., 2015).

The novelty of our project lies in the way user preferences are collected. Our recommender, allows users to input written text, which allows for much flexibility in suggesting new games tailored to their current interests. Anyone seeking to explore new kinds of games based on new preferences, and who has no expansive knowledge of hundreds of available games to input an existing game as an indicator of their preferences, would benefit greatly by our software's architecture compared to existing recommenders.

Furthermore, this is exactly the relation of our project to text mining. It is BERT-fuelled keyword extraction methods that allowed us to spot the useful parts of the users' written inputs, and word vector cosine similarity techniques that helped us map these arbitrary preferences to our own feature space. Overall, we used text mining techniques to provide a helpful solution to the well known problem of changing preferences that often underlies recommender systems.

1.3. Author contributions

Joep Made the graphical user interface (GUI) from scratch using the CustomTkinter library, uncluding the questionnaire, all popups and the user output window. Furthermore, I was responsible for keyword extraction, using specialized BERT models.

Brook Made the code for category mapping, using state of the art models for optimization. Also wrote the code for the evaluation and its visualizations, using the Google Form survey. Moreover, created the GitHub repo and contributed to its updates.

Iason Pre-processed the data to create our working dataset. Also made code that matches mapped user preferences to board games. Finally, was responsible for GitHub updates and the majority of this report including the literature review.

2. Method

2.1. Data

The data used came from a database created by a Kaggle user who scraped the website boardgamesgeek.com and created four dataframes, each of which contains some type of characteristic for each of the website's board games. These original data frames can be found under this link: https://www.kaggle.com/datasets/caesuric/bgggamesdata

Our pre-processing aimed to

1. Connect all data frames into one.
2. Delete characteristics that are of no use to our analysis, such as "Bayes rating".
3. Delete board games with missing values on any of the useful categories.
4. Keep only games with over 20 user ratings. This is to increase the chances that the rating is accurate.
5. Keep only games with a rating over 5/10. The rest would not be worth recommending to someone.
6. Remove unnecessary characters from the game descriptions, such as "br/", which served a purpose in the presentation of the descriptions within the original website, but obscured our own output.
7. Finally, the resulting data frame consisted of 10,000 board games, which were exported as a .csv file.

2.2. BGR software

Our software can reasonably be divided into 5 parts.

2.2.1. Questionnaire

Using the CutomTKinter library, we were able to create a pop-up questionnaire for the user in the form of a Graphical User Interface (GUI), which inquires about his/her preferences. The questions asked are:
1. Theme: "What setting or theme would you like your game to have? A theme refers to the specific context or background in which the game is set. Some examples of themes include the American Civil War, zombies, fantasy worlds, and so on.."
2. Type: "What specific genre or type of game would you prefer? Examples of game types include card games, word games, memory-based games, and so on."
3. Mechanisms: "What kind of gameplay mechanics would you like your game to feature? In simpler terms, what type of actions or interactions should the game revolve around? Examples of gameplay mechanisms include voting, grid movement, zone control, luck-based or dice rolling games."
4. Max time: "What is the desired maximum duration for the game? Please provide the answer in minutes."
5. Max players: "How many players would you like the game to accommodate?"

2.2.2. Keyword extraction

Using keyBERT, a specialised BERT-fuelled software, we created a function that extracts the most important keyword from each of the user's written answers. For instance, from an answer "I tend to like wargames but I'm not that strict about it" the function KeyBert().extract_keywords(user_input, stopwords) isolates the word "wargames". This applies to the questions of Theme, Type and Mechanism. In the other two questions, another function - find_all_numbers(user_intput) - extracts numbers, whether digits or strings, using python's .isdigit() and

regular expressions. If one number is found, the function assumes it is the maximum value and automatically sets the minimum to zero. If more then one value is found, the lowest is assumed to be the min. and the largest is assumed to be the max. Keywords from the user's answers are finally stored as dictionary values, corresponding to the respective key of the question.

2.2.3. Category Mapping

Using the built in Word2Vec Cosine Similarity Function, the Scos (x, y) master function compares the keywords of the user to each of the features within our dataset to find the closest match. For the word matrix we use the Gensim GloVe Wiki Gigaword 100 model. This has been shown to outperform Word2Vec in similarity tasks (Cothenet, 2020).

The goal is to find the closest feature given a user input. For instance, given the word "wargames" from above, the nearest word semantically is "fighting," which is a game category in our dataset and can thus be used to select all and only those games that fall within it. Because some feature names consist of more than one word, the similarity comparison takes their average. This applies to Theme, Type, and Mechanisms, since the numeric answers of the other two questions suffice as they are. The results are stored as strings and floats in a list of mapped preferences and the most similar feature is returned in the format of a tokenized string that matches one name from the features list.

To do this, the feature list is tokenized using NLTK's built in tokenizer. Stopwords are also removed from the list. We also found that there were a fair amount of repeat words in features, the main one being "game." During tokenization, these words are removed.

2.2.4. Data selection

We created several mini functions, one for each type of preference, which take in the appropriate element from the above-created list and use data frame indexing to create a sub-data frame containing all and only those entries that are in line with the user's preferences. After this subselection, a second set of functions chooses the best rated, most rated, and one random board game. These functions are eventually aggregated inside a master function, which takes in the list of mapped preferences and our dataset as a pandas data frame and returns a list of three board game names.

2.2.5. User output

The list of names is used for data frame indexing to select the appropriate games as rows in our data frame. The name, description, and image of these games is then taken into the GUI and presented to the user.

3. Results

The code works. It runs without errors and can present to the user board games in line with the given preferences. It is relatively fast, reaching completion in ¡ 1 min, and looks very presentable and user friendly. To improve user experience, we included a loading screen while user responses are being processed. If no board games fulfil the user's preferences, the user is notified and kindly asked to start restart the program.

4. Evaluation

For evaluation, we created a Google form asking each user to manually select one word from each of their answers that should be given the most attention. This allows us to evaluate against human standards. Our form received 23 answers. This sample size is large enough to allow for some basic observations and descriptive statistics within the scope of this paper.

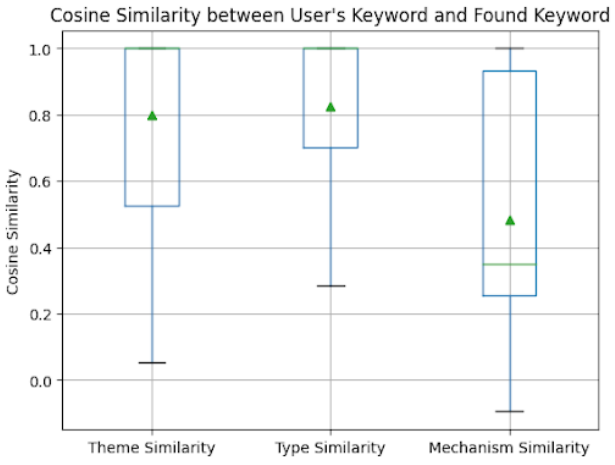

Figure 1: keyword extraction

Regarding keyword extraction, we can compare the word chosen by the BGR to the word picked by the user and see how semantically close they are using cosine similarity. For the evaluation of category mapping, we can use the same cosine similarity metric to examine how close each of the isolated words is to our dataset's existing categories.

The results of our evaluation showed that when it came to extracting a keyword for the Theme and Type of the game, the keyword extracted by the BGR and the word chosen by the user were semantically similar. Our software had a mean accuracy rate of around 80%, with Theme similarity exhibiting a higher variance compared to Type. This contrasted with the similarity of Mechanism keywords, where accuracy dipped significantly to approximately 50%, while variance increased substantially. This is clearly where the BGR had the most unpredictable results. While some keywords were the same between BGR and user, there were also quite a few with extremely low cosine similarity ratings.

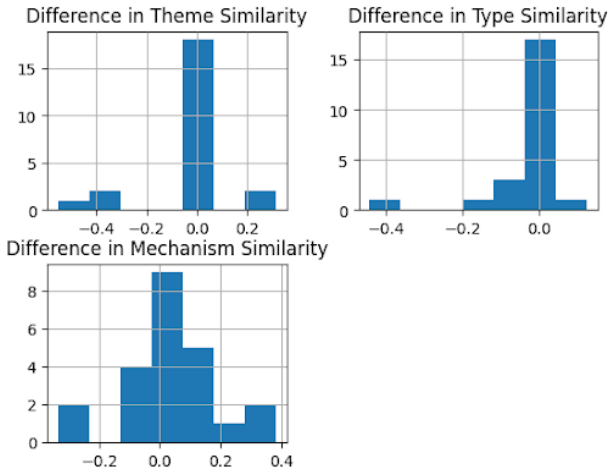

Figure 2: category mapping

As for category mapping, for each user we found the highest similarity rating of the BGR's extracted keyword to our database's features, then the highest similarity between the user's chosen word and our features. To compare, we subtracted the user's similarity rating from the BGR's rating. The difference between the two reveals which one was closer to our existing categories and by how much. It is assumed that if a word is closer to any of our categories to begin with, it is more likely to get mapped correctly during category mapping.

According to our results, both Theme and Type had a fairly high accuracy percentage. Accuracy here is defined as mapping to the same feature from both the user's and the BGR's keywords. Theme had an accuracy of 78.26% while type had an accuracy of 73.91%. Again, Mechanism had a much lower accuracy of 34.78%.

5. Discussion

There are two main areas for potential improvement; category mapping and keyword extraction. We propose two possible improvements for future applications.

One obvious limitation of our code is its low accuracy when selecting a Mechanism. This could possibly be because the list of mechanisms within our dataset is too limited for the variety of user preferences, or because the concept of "mechanisms" is itself too vague. These could be ameliorated by replacing the existing set of mechanisms with a new one that is more extensive and intuitive, perhaps created by users. Of course, this goes beyond the scope of our project, but it could be very useful for optimising filters in all kinds of board game recommenders.

The second suggestion is that the BGR could average multiple keywords from each user answer and use that averaged vector instead of simgular words, to improve its extraction accuracy.

In conclusion, despite all its shortcomings, we were successful in creating a board game recommender that uses written text to provide recommendations. In most cases it performs well, is fast and user-friendly. Future efforts could improve on keyword extraction and category mapping, to expand on our idea of a recommender using text mining rather than boardgame similarity.

References

Alden, S. (2019). Announcing New Feature - Game Recommendation System — BGG. BoardGameGeek. https://boardgamegeek.com/thread/2140686/announcing-new-feature-game-recommendation-system

Cothenet, C. (2020). Short technical information about word2vec, glove, and Fastext. https://towardsdatascience.com/short-technical-information-about-word2vec-glove-and-fasttext-d38e4f529ca8

Felfernig, A., Hotz, L., Juha Tiihonen, & Bagley, C. M. (2014). Configuration-Related Topics. 21–27. https://doi.org/10.1016/b978-0-12-415817-7.00003-7

Ion, M. A. (2018). Designing and evaluating a board game recommender system. Repositum.tuwien.at. https://repositum.tuwien.at/handle/20.500.12708/1444

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261–273. https://doi.org/10.1016/j.eij.2015.06.005

Konstan, J. A., & Riedl, J. (2012). Recommender systems: from algorithms to user experience. User Modeling and User-Adapted Interaction, 22(1-2), 101–123. https://doi.org/10.1007/s11257-011-9112-x

Ngo, R. (2021). Board Games Recommender. GitHub. https://github.com/richengo/Board-Games-Recommender

Rocca, B. (2019). Introduction to recommender systems. Medium; Towards Data Science. https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

Woodward, P., & Woodward, S. (2019). Mining the BoardGameGeek. Significance, 16(5), 24–29. https://doi.org/10.1111/j.1740-9713.2019.01317.x

Zalewski, J., Ganzha, M., & Paprzycki, M. (2019). Recommender system for board games. 2019 23rd International Conference on System Theory, Control and Computing (ICSTCC). https://doi.org/10.1109/icstcc.2019.8885455

Appendix A. Graphical User Interface

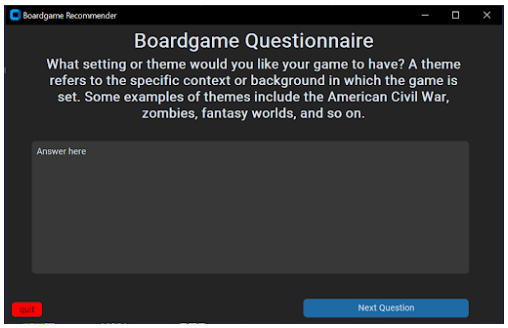

Figure 3: Theme

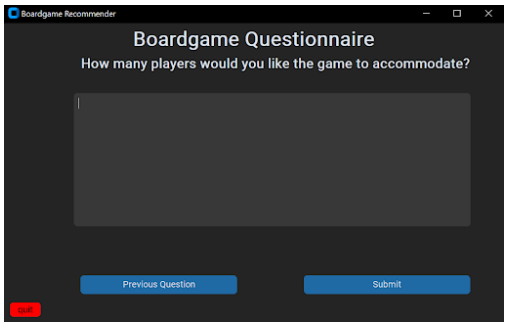

Figure 4: Number of players

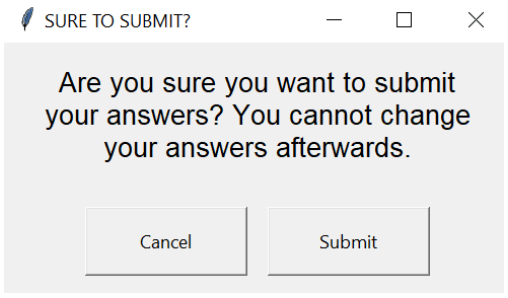

Figure 5: Sure to submit

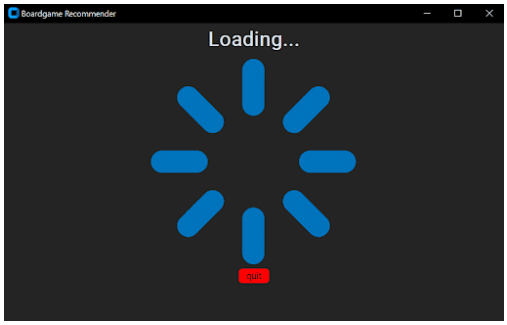

Figure 6: Loading screen

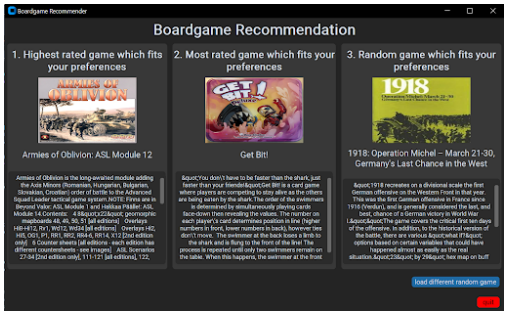

Figure 7: User output

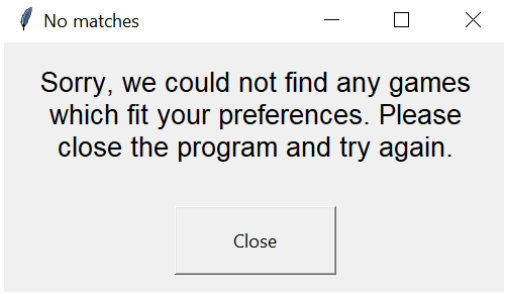

Figure 8: No games found

Appendix B. Link to Google form

https://forms.gle/Zx4S9hud5TtBubNYA