

Board Game Recommender

Joep Kapma

joep.kapma@student.auc.nl

Brook Sorensen

brook.sorensen19@gmail.com

Iason Matiatos

iason.matiatos@student.auc.nl

Abstract

We created the “Board Game Recommender” (BGR) software that uses written text by users highlighting their preferences over boardgames to return three appropriate suggestions. Through BERT-fuelled keyword extraction, the BGR recognises the most important and relevant words out of the user’s inputted text. Then, through cosine similarity, it maps these words to the categories existing in our database that are semantically closest. Finally, it returns the best rated, most rated (which is associated with most played) and one randomly chosen board game that fulfils the user’s preferences.

Background

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). We will use the term recommender system (RS) broadly in this paper, as any system that helps users find products from a given set (Felfernig et al., 2014). Today, Streaming services (Netflix or Spotify), social media (Facebook), online shopping websites (Amazon), dating services (Tinder) all use RS’s and the list goes on and on (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). By analysing data on the behaviour of many users, these platforms can predict the type of content that will attract new users who fit into specific interest categories.

The way that most all recommender systems are designed today is by using 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 in an algorithm created by Netflix engineer Toby Mao (Alden, 2019). The closest attempt of non-insiders that we could locate uses data of 200 users that was individually provided through personal connections (Zalewski et al., 2019). Overall, several attempts have been made to do exactly what we tried, although they use advanced machine learning and statistical techniques unfamiliar to us (Woodward & Woodward, 2019; Ngo, 2021; Ion, 2018).

Our project's original contribution & relation to Text Mining

Out of necessity, we used content-based filtering. We decided to ask users for direct input of 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. All popular applications of recommender systems readily available ask users to provide one or more board games they enjoy, and make suggestions accordingly. Our recommender, however, allows users to input written text, which can allow for more 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 the existing solutions.

Furthermore, this is exactly the relation of our project to text mining techniques. 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 category 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.

Author Contributions

Since the beginning we divided tasks efficiently among the three of us. In broad terms, Joep was mostly preoccupied with the graphical user interface and extracting keywords from the users' answers. Brook made the code that maps these keywords to our own database categories. Iason

gathered and pre-processed the data, and made code that returns the correct recommendations based on a user's preferences (already mapped onto our categories).

Joep

Made a modern, clean-looking, user friendly interface from scratch using CustomTkinter library. It displays the questions, allows the user to type in their answers and move through the questions (back and forth). Upon handing in the answers, a popup warns the user the questionnaire is about to be submitted and can't be changed. After submitting, a loading screen appears while in the background the recommendation is calculated. If no games are found a popup appears informing the user there are no matching games and requests the user to fill the questionnaire in again. If there are matches found a new interface appears. This one shows 3 games: the highest rated game, the most rated game (which is associated with the most played game) and a random game. All of these fit the user's preferences. Using a button the user can load a different random game. All the game's descriptions are also presented.

Furthermore, I was responsible for how we could interpret the users' answers, namely how we extract keywords from the answers to later match with data from the database. This proved to be quite difficult to extract keywords we are interested in. However, using a specialised BERT model which also filters on a given list of words, I succeeded in retrieving the keywords. In the sentences where the numbers contain the most important information like for example max playtime, I used another function. This function uses regular expressions to retrieve any written out numbers (like four), and a `isdigit()` function to check for any other number (like 4). If one number is found, it is assumed this is the max, so either max playtime or max number of players. If more than 1 number is found, it is assumed the lowest number is the minimal and the highest is the maximum number.

Brook

Created `user_to_feature` which uses the built in Word2Vec Cosine Similarity Function, with the Gensim GloVe 100 Dimensional Word Vector pretrained model. The `user_to_feature` function takes in the extracted keyword, a tokenized list of features, and a non tokenized list of features that matches the format found in the `final_data.csv`. The highest similarity between the keyword and the feature list is found by averaging the similarity between all the words within each feature, as some have multiple words such as "city building" and returning only the feature with the highest similarity in the non tokenized format.

Also wrote the code for the analysis, using the Google Form survey. The survey has the user enter answers in full sentence format and then they themselves pick out a single keyword. The analysis then goes through and finds the similarity between the extracted keyword and the user keyword. It also finds the difference in similarity found between the extracted keyword and the closest feature and the user keyword and the closest feature. It also finds the accuracy when it comes to both the extracted keyword and the user keyword returning the same feature.

Iason

Filtered through the 286,000 initial data entries to create a useful database of 10,000 games. These are all without missing values, with at least 20 user ratings, and a rating score above 5/10. The code used for this processing has been uploaded to the GitHub repository as 'TM_data_making.ipynb'. This data was further edited, because we noticed some characters in the game descriptions that should not be there when printed to the user, such as “
”. These characters were edited out using regex.

In addition, wrote code that takes in user preferences on five parameters (playtime, number of players, mechanisms, theme and type) and filters through our data frame to find the subsection of games that match these preferences. Out of these, three are selected using a second set of functions. At the same time, was responsible for GitHub updates, and for the majority of this report including the literature review.

Method

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 ≥ 20 user ratings. This is to increase the chances that the rating is accurate.
5. Keep only games with a rating $\geq 5/10$. The rest would not be worth recommending to someone.
6. Remove unnecessary characters from the game descriptions, such as “
”, 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.

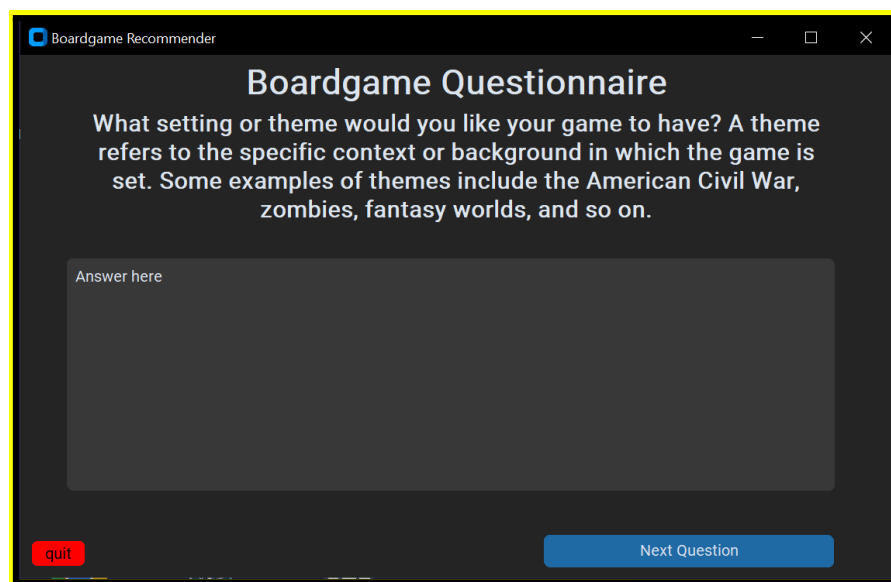
BGR Software

Our software can reasonably be divided into 5 parts.

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:

- a. 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..”
- b. 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.”
- c. 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 controlment, luck-based or dice rolling games.”
- d. Max time: “What is the desired maximum duration for the game? Please provide the answer in minutes.”
- e. Max players: “How many players would you like the game to accommodate?”



The screenshot shows a window titled "Boardgame Recommender" with a subtitle "Boardgame Questionnaire". The main text asks: "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." Below this is a large text input field with the placeholder "Answer here". At the bottom left is a red "quit" button, and at the bottom right is a blue "Next Question" button.

The screenshot shows a window titled "Boardgame Recommender" with a dark background. The main heading is "Boardgame Questionnaire". Below it, the question is "How many players would you like the game to accommodate?". There is a large, empty text input field. At the bottom, there are three buttons: a red "quit" button on the left, and two blue buttons labeled "Previous Question" and "Submit" on the right.

The screenshot shows a confirmation dialog box with a light gray background and a black border. The title bar says "SURE TO SUBMIT?". The main text asks, "Are you sure you want to submit your answers? You cannot change your answers afterwards." At the bottom, there are two buttons: "Cancel" and "Submit".

2. Keyword extraction

Using a specialised BERT-fuelled software, we created a function that selects and extracts the most important and relevant parts of the user's written answers. For instance, in the question about the game's Theme, from an answer such as "I tend to like wargames but I'm not that strict about it" the <function> would isolate the word "wargames". Similarly for Type and Mechanisms. In the other two questions regarding the number of players and playing time, another <function> spots and isolates numbers, whether digits or strings. The user's answers are finally stored as dictionary values, corresponding to the respective key of the question.

3. Category mapping

Using cosine similarity, the <function> compares the keywords of the user to each of the categories within our database to find the closest match. For instance, given the word "wargames" from above, the nearest word in terms of semantics can be "fighting," which is a game category in our database and can thus be used to select all and only those games that

fall within it. 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.

4. Data selection

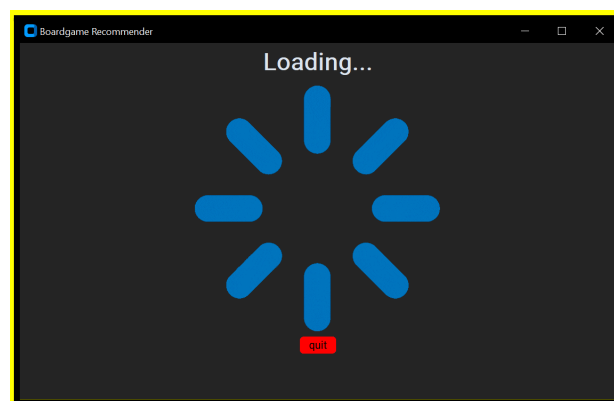
We created several mini functions, one for each type of preference, which take in the appropriate element from the above-created list and use data frame indexing to create a sub-data frame containing all and only those entries that are in line with the user's preferences. After this subselection, a second 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 database as a pandas data frame and returns a list of three board game names.

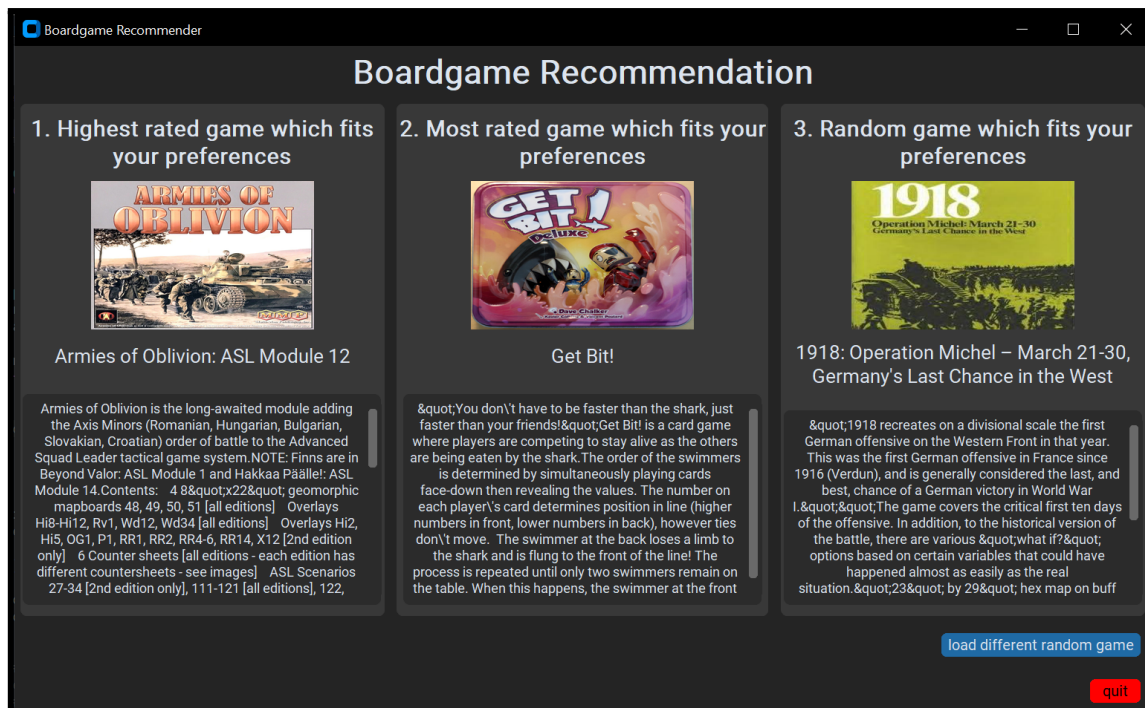
5. User output

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 in a friendly way.

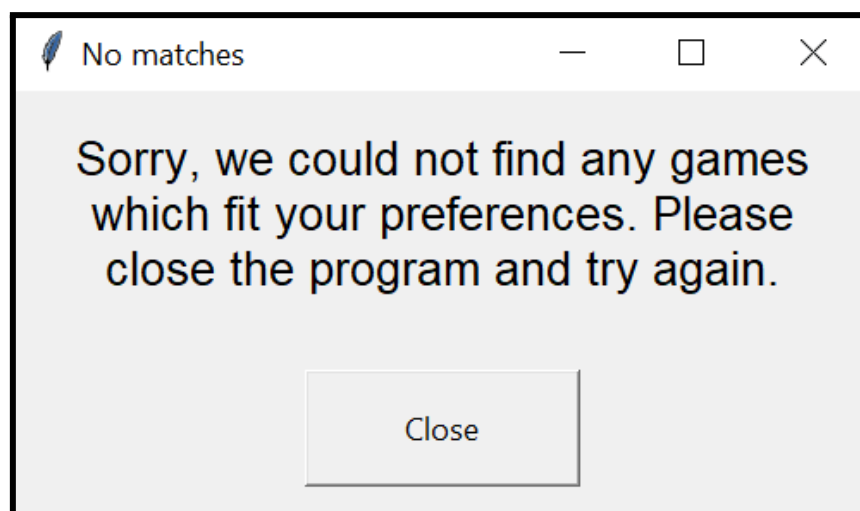
Results

The code works. It runs without errors and does 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 the user's experience even more, we included a loading screen that is presented after all preferences have been written and before the results show up, to let the user know that the program did not freeze, but instead is processing the responses. If no board games fulfil the user's preferences, the user is notified.





Or

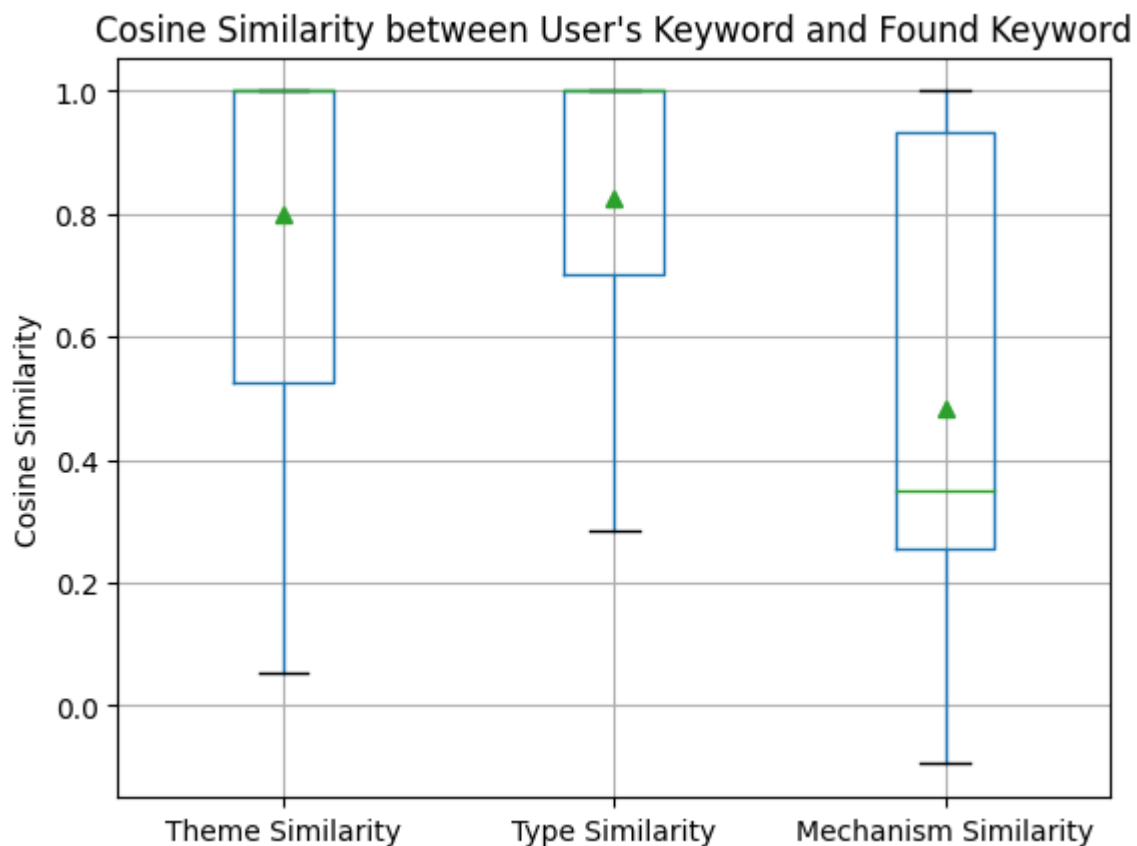


Evaluation

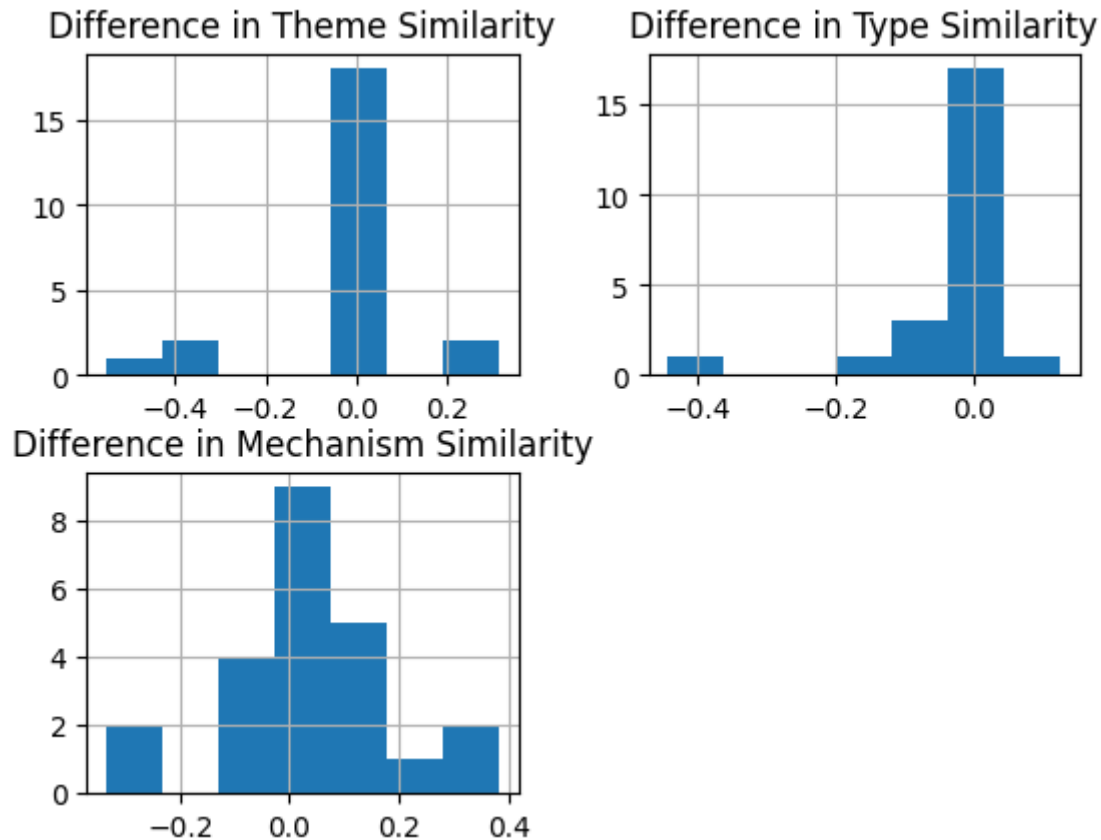
Areas where our code could plausibly be improved are the keyword extraction from user answers and the mapping of those keywords onto the appropriate dataset categories. The optimal strategy for evaluation would be to compare the board games returned by our algorithm against those returned by other similar algorithms for the same sets of user preferences, to have a baseline of performance. Unfortunately, these datasets do not exist.

Thus, we decided to create a google form where the user can manually select one word from each of their answers that should be given the most attention. This way, 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 as well as how close they are to features.



The results of our evaluation showed that when it came to finding a keyword for the theme and type of the game, the extraction was fairly accurate. While there was a bit more variability when it came to the similarity between the extracted theme keyword and user keyword, the mean tended towards having the same keyword for each. When it came to mechanism keywords, there was a significant dip in accuracy. While there were still keywords that were the same there were also quite a few that were extremely far away in it came to similarity.



When it came to the difference in similarity between the user's keyword and the extracted keyword with the highest feature, the majority of the time the most similar feature was the same for both keywords. However, when they were not the same for theme and type the users keyword was found to be closer to a feature than the extracted keyword. This is shown through a negative difference as the similarity for the user's keyword was higher than that of the extracted keyword. Mechanism had a fairly even distribution when it came to which keyword had the highest similarity with the extracted keyword tending to have a slightly higher similarity.

For both theme and type, there was a fairly high accuracy percentage. The accuracy in this case being defined as having the same feature found from both the extracted keyword and the user's keyword. Theme had an accuracy of 78.26% while type had an accuracy of 73.91% out of a total of 24 responses. Mechanism had a much lower accuracy of 34.78%.

Discussion/ conclusion

There are two main things that it would make sense to try and improve when it comes to the keyword extraction process. The first is that the keyword extraction might be more effective if it took into account the feature categories when it was determining the most important

word. Despite the fact that there was a fairly high accuracy rate, the times when it was not accurate the user's keyword tended to have a high similarity with a specific feature. Trying to make sure that the extracted keyword has a high similarity with one of the features may help it to be more accurate or simply work better with the function. Another way that we could try and compare it is instead of taking a singular keyword from the user's input, we could average multiple keywords to see if that might improve the similarity with a feature.

There seems to be an opposite issue when it comes to the keyword extracted for mechanisms. While there is not a very high similarity when it comes to the extracted keyword and the user's keyword, the extracted keyword tends to have a higher similarity with the features. This means that it was finding something closer in relation than it would have if it was only taking the user's keyword. However, the accuracy of the mechanisms was not great. A way that we could try and improve that in the future is to make it more clear what mechanisms there are to choose from if the similarity is under a specific threshold. Another way that we could work on this is giving a game that is well known and listing the mechanisms within that so there is something for the user to compare their answers to.

References

Alden, S. (2019). *Announcing New Feature - Game Recommendation System* | BGG.

BoardGameGeek.

<https://boardgamegeek.com/thread/2140686/announcing-new-feature-game-recommendation-system>

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>