CP421 Project Paper
Book Recommender System

Riley Adams 190416070
Torin Borton-McCallum 190824620
Chandler Mayberry 190688910
Rida Mohammed 190173550
Jessie Newman 200166070
Jasleen Pabla 190404270

December 6, 2023

Abstract

This project aims to create an innovative book recommendation system by integrating Knowledge-Based and Reinforcement Learning methodologies. The system's core focus is on tailoring book suggestions to user preferences, adapting to evolving behaviors through reinforcement learning, and considering contextual nuances. By integrating knowledge-based recommendations and reinforcement learning, the system aims to provide dynamic and personalized book recommendations. The primary goal is to recommend books aligning with user preferences while minimizing negative feedback and adapting to user interactions to refine future recommendations. The system utilizes predefined and user-defined profiles to initiate recommendations and adjust based on user feedback. Through the integration of these methodologies, the project endeavors to develop a versatile book recommender system that enhances user experience and recommendation accuracy.

1 Introduction

Recommender systems are a vital aspect of machine learning across multiple industries through its functionality to advertise relevant products to users of a service, ensuring that the recommendations are relevant to each user and that they are likely to both consume it and rate it highly. This is seen through streaming service suggestions, online shopping recommendations, and more. The system that we have implemented handles recommending relevant books to users through both Knowledge-Based and Reinforcement Learning recommendations and by considering users' ratings of different books.

2 Dataset

The dataset in our recommender system utilized data from three different files, User Data, Book Data, and Rating Data. The user data contains the user-id, location, and age of the readers. The book data provides information regarding each book such as the title, the year it was published and the author who wrote it. The rating data contains the user ID, the book ID, and the rating given to the book by the user.

The preprocessing required to clean the data to make it useable by the system happened in a few distinct steps. Firstly, the CSV files are read into the system and we drop any rows with null values. Next, we drop a few columns that are not required to give recommendations to users, the three image URL columns: "Image-URL-S", "Image-URL-M", and "Image-URL-L". Finally, we convert the "Year-Of-Publication" value to an integer and the rest of the columns to string values.

3 Model Implementations

For the recommender system we incorporated 2 different models. The first is Knowledge-Based Learning to provide personalized recommendations to our users. The second is reinforcement learning, in

particular Q-Learning. This allows the system to learn what our users' preferences are. Paired together with Collaborative Filtering, this allowed us to create a relatively robust recommender system.

3.1 Collaborative Filtering

Collaborative filtering works on the assumption that all users who agreed in the past will continue to agree in the future. Out of the two types of collaborative filtering we chose to employ user-based collaborative filtering. This allows us to recommend books based on the preferences of users deemed similar by the system to the target user.

3.2 Knowledge Based Learning

For the part of the system that uses Knowledge Based Learning, we create a coordinate system matrix that is based on the users' rating of different books. The row of the matrix represents the index, where the columns consist of the ISBN (International Standard Book Number) and ratings. For the implementation, we chose an existing user with book ratings so that the system did not have to start from nothing and try to recommend books to a user it has never seen before. We utilize the cosine similarity to calculate similar users to our target user, using the coordinates from the matrix. We recommend books to the target user by taking into account the top five most similar users and the books that they have rated.

3.3 Reinforcement Learning

The system applies Reinforcement Learning to adapt to user interaction and feedback with the application. Specifically, we utilize Q-Learning within our recommender system. Q-Learning is a method of machine learning that learns the value of taking an action at any given state by exploiting or taking the maximum reward value and sometimes exploring the state space to see what other actions could be taken. To be able to utilize Q-Learning we first have to model our problem as a Markov Decision Process. There are 4 key parts to a Markov Decision Process, the agent, the environment, the actions to take, and the rewards from taking said actions. In this case, we have a single agent being the recommender, the environment is the users, and the action we can take is recommending a book. The reward is given to the agent based on the rating the user gave the book that was recommended, also taking into account whether the user read it or not then normalizing and scaling appropriately. The epsilon in our epsilon greedy policy is 0.1, meaning that 10% of the time the agent explores instead of exploiting the already known values.

4 Hybrid Approach

We pair Knowledge-Based and Reinforcement Learning for a hybrid approach to help mitigate the downsides. One of which is the "Cold Start" problem. With existing data that allows the knowledge-based part of the system to make recommendations with limited user interaction data, instead of none at all. This pairs well with Reinforcement Learning because it will continue to improve the recommendations as the system gathers more data, which makes it still relatively effective in the opening and evolving stages.

5 Obstacles

5.1 Data Handling Challenges

One of the issues that occurred when trying to create the knowledge-based learning model was trying to fit the data inside a matrix called `book_to_user_rating_matrix` containing the User-ID as rows, ISBN as columns, and the values as the Book-Ratings. This needed to be done this way as the data was required to be in an array-typed format for sklearn cosine similarity to run against the dataset as the comparison metric.

Unfortunately doing this naively in a numpy array was not going to work as the combined dataset itself contains 753,295 rows of data and thus produces an even larger array in memory due to the organization of the data stated above. The first strategy to rectify this issue was to reduce the dataset in half as a test while continuing to do the naive approach. In terms of creating this `book_to_user_rating_matrix` this worked but due to the nature of the sklearn cosine similarity function, this approach failed as this method allocates a lot of memory to perform operations and output (319GiB of memory to be allocated in the last test).

After a few smaller attempts, it was decided to convert the data required into a sparse array instead of a regular array. This not only solved the problem by reducing the amount of memory required to store the matrix but also decreased the runtime of the code block significantly from well over 30s to less than 6ms in a quick test using the Python built-in time library. The method chosen to create the sparse matrix is called `coo_matrix` from the SciPy library, a familiar library used by some of the team in the past.

5.2 Hybrid System Debate

Another obstacle that was faced was determining whether the knowledge-based model implementation should include both types of collaborative filtering, being user-based and item-based. After discussions on the matter, it was determined that the user-based filtering was going to produce the best results due to the stipulations of the dataset. In the case of item-based filtering, this dataset gives no correlation or relation between books either than ratings, authors, and publishers. While it may be presumed that if a user enjoyed a book from a particular author or publisher they will enjoy more books from that same author and publisher. However, from simple reflection, it's clear that this is not always the case as a person may enjoy a single book from an author while having no interest in the rest of their novels. Making a broad assumption such as the one above creates an uncountable set of false positives and without data disproving the latter, results in a poor approach. The other useful way to perform item-based recommendations using this dataset would be to compare similarities in book ratings against all of the books. This once again, created bias and would result in nothing more than a similar-rated book selector, which could recommend any genre or author with no correlation between the books content. Therefore, in the end, user-based collaborative filtering was chosen as the assumption if a small subset of users enjoyed the same books the current user enjoyed, there is an increased likelihood that they will also enjoy books read by other similar users.

5.3 Model Environment

During development, the selection of Jupyter Notebook introduced a rather unforeseen issue of restricting user interaction and engagement with the recommender models. While this could be circumvented when using the knowledge-based recommender which functions more like back-end code, the reinforcement learning model struggled in the current use case. This is because the reinforcement model requires user engagement to function, learn, and create recommendations to the user. When a user is recommended a book from the model, the model requires the user to rate the book and thus receive a reward for future recommendations. Using Jupyter Notebook added the unanticipated issue of not allowing for an active running GUI between code blocks, thus limiting the user interaction to simple Python input() in the working environment. In particular, this made testing and tuning the model to be difficult as parameter adjustment could not be tweaked during active run time causing difficulties in final model accuracy. Although it was considered to be an obstacle it did not prevent the model from working properly as a proof of concept and could be fixed using a GUI front-end for future iterations.

6 Conclusion

In implementing both Knowledge-Based and Reinforcement Learning recommender systems, we have created two models for users to be recommended books with. These books are aimed to be as relevant to their personal tastes as possible. It is difficult, however, to realistically determine from this dataset alone whether or not a book recommended to the specified user is a good recommendation since such data is naturally subjective and thus cannot be adequately evaluated. Perhaps in future recommender systems, calculating popular books through high ratings and determining a way for them to be less

suggested to the reader could be implemented. We have learned a lot through implementing these systems, and understand more ways in which to expand on these concepts in the future.

7 References

"Book Recommendation Dataset." Kaggle, 7 May 2023, www.kaggle.com/datasets/arashnic/book-recommendation-dataset.