Book Recommender

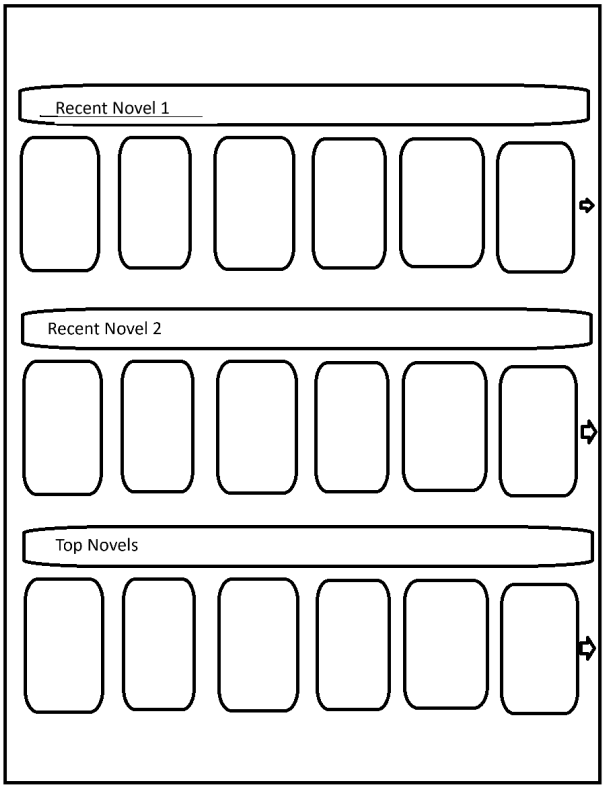
Author: Alex Zhao

Domain

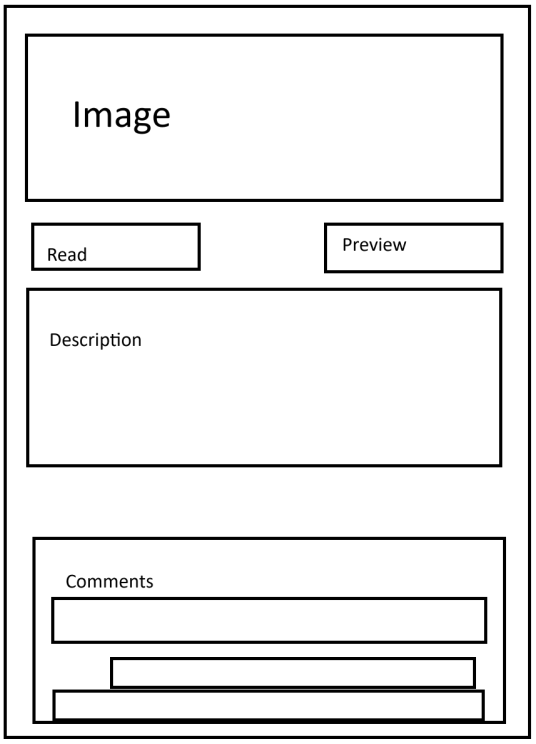
This recommender will be on an online book selling/library app aimed to recommend books based on the users taste. The design will consist of a list of recommended books (6 at a time) similar to each recent book they have read, up to 5 books. Upon clicking on a novel, there should be a short description, a list of reviews for that novel and a free sample for the user. There will also be a list of trending and all time top novels to handle the cold start problem and to introduce new genres to readers. The recommendation should be updated after each novel read.

An online book library generates money through the selling of books, hence the recommender can increase the revenue by recommending books that users are likely to click on and purchase - which can be done by providing similar books to recently books they liked.

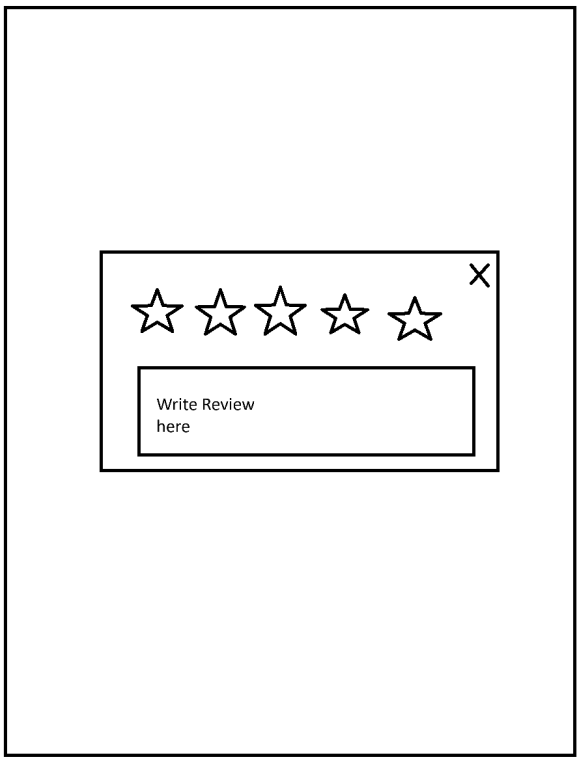
An example interface of the home page:



Interface of book page:



Users will also be prompted to rate the novel upon completion or exiting:



Dataset

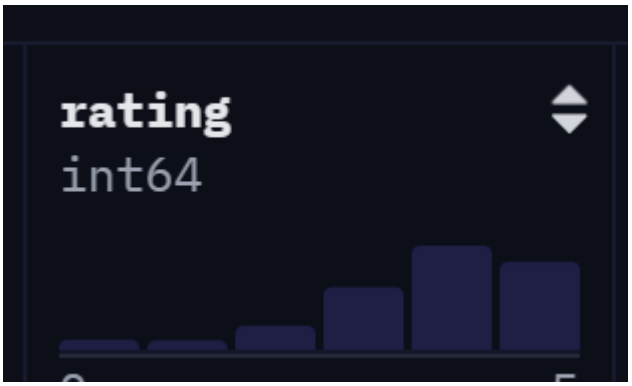
Dataset link: https://huggingface.co/datasets/qmaruf/goodreads-rating

The above dataset consist of a list of book reviews from GoodReads (total 900,000), with each review containing

- Review Text
- Rating
- Date added
- Date Updated
- Time when the novel was finished
- Time when the novel was first read
- Upvotes for the review
- Number of comments for that review

However, the dataset has a few limitations.

- First of all, the ratings are biased towards positive reviews.



- The dataset is also limited - with the dataset being 2 years old, this list of reviews will not contain the latest novels. There are also only 32 unique users.
- Most importantly, this only shows the list of people who reviewed - it is missing a lot of metadata such as the amount of pages read to help create accurate ratings of books.
- This dataset also does not show the prices of books, hence we will have to assume each book is the same price, where in reality we may want to recommend more expensive books.
- The dataset has already been sanitised, where each review has a comment - in reality there will be reviews that only have a star rating and no text.
- However, unlike many datasets, this does not appear to be part of a competition, so the dataset should more accurately reflect the real world scenario.

Method

Content Based recommendation - this type of recommendation given our dataset as it does not include the content of books, which although we can find, the size is very large and not feasible to use.

Collaborative filtering - This is the most recommended method, as we can choose either item-based or user-based filtering.Both will involve us creating a rating matrix for each row of users and each column of book. Then we can use a similarity measure (such as cosine similarity) on either the items or the users to estimate how a given item will do given on the n-most similar user/items. One method to potentially speed up the process and store less data would be to factorise the matrix first.

Knowledge-Based Recommendation - knowing what genres the user likes/searches for can greatly improve the likelihood the user buys the book. However, it is not viable as a solo methodology as there are too many books to completely narrow down, and hence it is best combined with other recommendation systems. However, we do not have the list of book content downloaded, so we can only use the recommendations to define loose categories for each book.

The basic approach will be a simple collaborative filtering method, either basing the similarity metric from the items or the users. However, a more advanced method will include knowledge based recommendation where we predict the genre of a book and allow users the option to pick or ban certain categories that will be listed - where we can add a weighting to increase or decrease the final predicted rating for each given book. This method should be more effective as readers are generally only interested in the same genre for a while - and hence why we only use the most recently liked books to recommend to the user.

Evaluation

Testing the system will require us to predict a users rating given they have read and reviewed a certain amount of books already. Hence our training/test must be split to ensure that reviews with the same user id are together in the same set, or else it will turn into a cold start problem. The cold start problem should be handled by a list of currently popular novels, instead of relying on our recommendation system.

A few metrics we can use include the f1 scores, precision, and recall for the first n items. However precision and f1 score is most important as we care a lot more about the precision, the ratio between true positives (liked novels) and all positives (recommendations). Since we have the true ratings and predicted ratings in our test set, we can also calculate the MSE between the two to determine how accurate our ratings are. We will test multiple recommendation methods and pick the one that gives the highest average precision and f1-score, and also average MSE for the top N recommendations.

One feasibility issue with collaborative filtering is scalability. Because we are factorising a large matrix, this can be very time consuming and inaccurate if the matrix becomes very sparse. However, books generally take a long time to read, so it is likely we do not need to update in real time and it is fine to have a slow computation.

A user study will consist of a user first picking up a few novels they have already read and rating them. Then the recommender will give a few novels for them to read and rate (with time constraints, they only have to read the summary and give a rating). We can determine how effective our system is by the number of books the user says they will be willing to give a try (since thats how the app earns money). Aside from the above, other user feedback can

include general thoughts on the system and also the number of books they completely would never give a chance to read.