DSCI 351: Recommender Systems Final Project

Title: Amazon Books Recommender: Merging Ratings and Review Semantics

Authors: Anastasiia Poverenova, Claudia Adam

Dataset Description

For our dataset, we used Kaggle's *Amazon Books Review* data, which includes features from user reviews and Google Books metadata. The dataset contains two files with a total of over 420,000 rows, covering about 300,000 unique books and 500,000 users. The first file is on book ratings and includes features like book ID, title, user ID, rating (0.5-5), review summary, and several others. The second file has the book's metadata, with features like book descriptions, authors, categories, publication date, and average rating. After preprocessing, we merged the two files by book titles. Then we farther cleaned the data by normalizing test fields and filtering for users who rated at least three books and books with at least five ratings. The working dataset contained 1.34 million interactions across 62,804 items and 168,659 users.

Type of Recommender System

From here, we built a hybrid recommender system that utilizes content-based filtering as well as item based collaborative filtering. Our content-based model extracts semantic similarity between books by using TD-IDF on multiple features, such as title, book description, authors, categories, review text, etc. We implemented a KNN search with cosine similarity to gather recommendations of similar books. For the collaborative filtering model, we first constructed a user-item rating matrix and applied adjusted cosine similarity across items. We used mean-centered ratings to account for any user bias. To combine these two models, we took their normalized similarity scores using a weighted average parameter, this balances user behavior with content semantics.

Models and Methods

To farther explain our analysis, these are the methods and models used in each step of our code. For preprocessing and feature engineering, we normalized book titles to be able to merge datasets. Then, we cleaned and combined book metadata with reviews to form unified content strings. Also, to reduce sparsity, we filtered out low-activity users and items.

Our content-based filtering approach used TF-IDF vectorization with unigrams and stop word removal. From there, we built a sparse feature matrix, that had a dimension of about 62,000 books by 59,000 terms. Finally, we used the KNN method to compute similarity between items. For the collaborative filtering model, we constructed a sparse user-item ratings matrix with more than 800,000 non-null entries. Then, we applied mean centering per user to reduce user bias and fit the item-based KNN on the centered matrix.

In our hybrid model, we selected the top-rated items for each user, then retrieved its nearest neighbors from both models mentioned above. We normalized their similarity scores between 0 and 1 and combined them using a weighted average parameter. To test for best performance, we tested the weighted parameter at the values [0, 0.25, 0.5, 0.75, 1]. We achieved the best results at 0.25 which favored content-based filtering slightly more than collaborative.

Interpretation of Results

For the results, our hybrid model provided the top-k ranked list of book suggestions for the user. If the weighted average parameter equals 0, the results are purely from our content-based recommender system which means they are semantically related books. If the weight average is set to 1, the results come purely from the collaborative filtering model which means suggestions align with user co-preference. At our best performing weighted average of 0.25, the results balance thematic relevance with collaborative popularity trends. For example, one user received top recommendations that combined their interest in Jewish cooking with related Jewish holiday cookbooks, reflecting both content similarity and collaborative filtering co-reading patterns.

Evaluating System Performance

To evaluate our models, we used both rating prediction metrics and top-k ranking metrics. The root mean squared error is 7.12, which means on average, the system's predicted ratings differ from the true rating by about 7. Our mean absolute error is 3.41, which means prediction is about 3.4 away from the real rating, on average. This means that our baseline predictor (meaning rating per item) isn't very informative. This metric highlights how hard rating prediction is on this sparse dataset.

For the top-k rating metrics, we used the top 10 nearest neighbors and set the average weighted parameter to 0.25. Our results put precision at 0.026, recall at 0.26, and F1 score at 0.047. These numbers are quite low, but this makes sense for our large-scale

recommender system with extreme sparsity. Also, we produced a hit rate of 0.26 and coverage was 1.3% of the catalog. This shows that the hybrid system outperforms just the content-based or collaborative filtering model on their own.

Limitations and Future Improvements

One large limitation with our project is the data sparsity, because most users and books have very few interactions which limits the reliability of the collaborative filtering. Also, TF-IDF with over 60,000 items is memory intensive, which makes scalability difficult. Finally, there's difficulty making recommendations to new users without history, so the model defaults to popularity-based recommendations.

Some future improvements we could make include integrating matrix factorization, like SVD, to capture latent embeddings. We could also use transformer-based models to embed review text for broader content representation. Another suggestion is incorporating implicit feedback, such as clicks or shares, which add another level to user reviews.