Making a Book Recommender from Goodreads Data

Catherine Johnston and Yevgeniya “Jonah” Tarasova
With Contributions from Katie Johnston and Ava Bamforth
GitHub Repository

Introduction:

Finding your next book can be difficult, given the sheer volume of choices. Readers often rely on personal recommendations, but as the number of available books grows, a more systematic approach can be beneficial. In this project, we used data scraped from Goodreads to develop a book recommendation system designed to help readers discover new books that match their preferences.

Stakeholders and KPIs:

Our stakeholders include readers, publishers, booksellers, libraries, and community book providers. To evaluate our model's performance, we used "hits" as our key performance indicator. A "hit" represents a recommended item that a user rated positively and on which the model had not been trained.

Methods and Modeling Approach:

We created our dataset of books by scraping data from Goodreads https://www.goodreads.com/, particularly books from Goodreads' yearly Readers' Favorite Books lists (2011-2024) and books that had at least one review in a random sampling of user reviews that we scraped. Relevant information scraped for each book includes title, author, book description, genres, average user rating, number of user ratings, and Goodreads URL. After cleaning and preprocessing our data, the dataset consists of around 71,000 books.

Our book recommendation system uses a hybrid approach that combines content-based and collaborative filtering. In the content-based filtering model, books are represented as feature vectors (genres and keywords), and similarities are measured using cosine similarity. A user’s preferences are computed through a weighted average of these similarities. The collaborative filtering model factorizes the user ratings matrix into a book “feature” matrix and user preference matrices, and incorporates bias vectors. These matrices and vectors are optimized via gradient descent. For new users, Ridge regression learns individual preferences. Final recommendations are generated by multiplying the scores from both methods.

Generating the keyword vectors involved a two-step process: First, the KeyBERT tool extracted keywords directly from the book descriptions. Then, to manage similar terms (like “crime” and “criminal”), the keywords were clustered into semantic groups using Sentence-BERT embeddings.

An alternative recommendation method is to create clusters of similar books using K-modes and then to recommend popular books within the clusters that contain books specified by the user. K-modes is an unsupervised machine-learning algorithm that groups books into clusters based

on their feature vectors. The algorithm randomly assigns a predetermined number of books as cluster centroids and then moves books between clusters to minimize the cosine dissimilarities between the centroids and the rest of the books.

Results and Implications:

Before coding our models, the book data was divided into training and testing sets. User ratings were split by user into training, cross-validation, and testing sets. The cross-validation set was used for model selection, while the final model was evaluated on the testing set. In both phases, each user's ratings were split in half: one half to simulate a "new user," the other as unseen data for evaluation.

During cross-validation, we optimized the parameters for our recommender system components. This involved determining the ideal number of features (600 is ideal) and regularization strength (5 is ideal) for collaborative filtering, and the optimal weighting of genre vectors versus keyword vectors for content-based filtering (80% weight for genres is ideal). We compared pure collaborative filtering, pure content-based filtering, and a hybrid model using score multiplication. After selecting these parameters, a random baseline was added for the final evaluation.

Our final tests confirmed that the hybrid model, combining scores by multiplication, overall delivered the best performance, with a higher average number of hits and fewer instances of no-hit recommendations. The content-based method also performed strongly, and outperformed the hybrid model when we had more limited information on a user, while collaborative filtering alone performed worse than the other two. All methods showed improved performance with an increasing number of relevant books, but the hybrid score multiplication method particularly excelled for users with extensive relevant book histories. The random baseline, as expected, consistently yielded much poorer results, highlighting the effectiveness of the other, more sophisticated recommendation strategies.

Future Work:

In addition to finer parameter tuning and expanding the size of the dataset, we propose a few extensions to the project:

Addressing duplicate books: The current dataset sometimes contains multiple different versions of the same book, such as non-English translations or collector’s editions. The presence of non-English translations could be addressed by detecting the language used in the book’s description, allowing users to input their preferred language, and then making recommendations accordingly.

Optimizing K-modes initialization: Instead of randomly assigning the initial K-modes cluster centroids, more robust centroid initialization methods could be used. For example, the method of Huang [1997, 1998] ensures that initial centroids are maximally dissimilar from each other, whereas the method of Cao et al. [2009] prevents choosing outliers as centroids.

Including additional features: Feature vectors could be expanded to include features other than genres and keywords, such as popularity (number of ratings), level of engagement (reviews-to-ratings ratio), publication year, number of pages, or author name.

Finalizing the app: Our models should eventually be trained on the full datasets, and the app itself should be optimized to be more user-friendly. Moreover, allowing users to make an account and store/save their data would allow us to further fine-tune the app.