Project Report: Book Recommendation System

By: Ann Okafor
Springboard – Data Science

Abstract:

The book recommendation system is a valuable tool in the digital age, addressing the challenge of finding relevant books in a vast sea of options. By leveraging machine learning and data mining techniques, this system analyzes user behavior and book data to generate personalized recommendations. In this project, we explored three recommendation models: content-based filtering, KNN (collaborative filtering), and random forest.

Content-based filtering focuses on book data, extracting features and identifying similarities to recommend books with similar content to those the user has enjoyed. Collaborative filtering examines user behavior, finding similarities between users and suggesting books enjoyed by similar users. The random forest model combines multiple features, including book titles, authors, genres, and ratings, to generate accurate recommendations.

Throughout the project, we delved into data preprocessing, feature engineering, and model selection techniques to develop robust recommendation systems. Evaluating the models with a test set of user preferences revealed the random forest model as the most accurate, showcasing its ability to provide reliable and personalized book recommendations. By understanding the strengths and weaknesses of each approach, readers can select the most suitable recommendation strategy for their own book recommendation systems.

Introduction:

The book recommendation project aims to develop an intelligent system that suggests personalized book recommendations to users. The project starts by collecting a dataset of book data, including attributes such as titles, authors, genres, ratings, and descriptions. This data serves as the foundation for building various recommendation models.

The project employs three main approaches for generating book recommendations: content-based filtering, collaborative filtering using KNN (k-Nearest Neighbors), and a random forest model. Each approach utilizes different techniques and algorithms to analyze book attributes and user preferences, ultimately providing recommendations based on similarity and user behavior.

Data preprocessing plays a crucial role in preparing the dataset for analysis. Missing values are handled, text data is cleaned and normalized, and categorical variables are transformed into numeric representations. Exploratory data analysis techniques are applied to gain insights into the dataset and understand the distribution of book attributes.

The content-based filtering approach focuses on the textual content of books. It employs TF-IDF vectorization to analyze book descriptions and calculate similarity scores between books. The model recommends books with similar content to the one provided by the user.

Collaborative filtering using KNN considers the preferences of similar users to make recommendations. It computes cosine similarity between users' preferences and recommends books that neighbors with similar preferences have interacted with.

The random forest model combines various book attributes and uses a random forest regressor to generate recommendations. It considers features such as titles, authors, genres, ratings, and numerical attributes to calculate similarity scores and suggest similar books.

Data Collection and Preprocessing:

The data collection process involves gathering book data from various sources. Attributes such as book titles, authors, genres, ratings, and descriptions are collected and compiled into a dataset. This dataset serves as the foundation for building the book recommendation system.

Once the dataset is collected, it undergoes preprocessing steps to handle missing values, clean text data, extract relevant information, and transform categorical variables into numeric representations. Missing values in columns such as description and language are filled with appropriate values like "No description available" and "Unknown," respectively. Numeric columns like pages are processed to convert them to the appropriate data type and handle missing or erroneous values.

Text cleaning techniques are applied to columns such as titles, authors, and descriptions to remove non-essential characters, normalize text, and prepare it for further analysis. For example, the 'unidecode' library is used to convert special characters to their ASCII equivalent. The 'extract_year' function is employed to extract the year from the 'publishDate' column. The 'modify_year' function adjusts the year to a valid range.

To improve the analysis and reduce the number of unique genres, a mapping dictionary is created to map specific genres to broader categories. This mapping allows for a more general understanding of book genres and simplifies subsequent analysis. The 'map_genres' function is applied to assign books to their corresponding genre categories. Additionally, empty brackets in the 'genres' column are replaced with empty strings, and the 'genre' column is filled with 'Unknown' for missing values.

Duplicate rows are removed from the dataset to ensure that each book is represented only once. Irrelevant columns, such as 'isbn', 'series', 'characters', and 'publisher', are dropped as they are not needed for the analysis. The remaining columns are reordered to facilitate readability and maintain consistency.

Overall, the data collection and preprocessing steps aim to create a clean and structured dataset that can be used for further analysis and model building.

Content-Based Filtering:

The content-based filtering model focuses on analyzing the content of books to generate recommendations. It leverages the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique, which converts textual data into numerical representations. In this case, book descriptions are vectorized using the TfidfVectorizer from the scikit-learn library.

The TF-IDF vectorization process involves transforming the book descriptions into a matrix of TF-IDF features. This matrix represents the importance of each word or term in a book description relative to the entire corpus of book descriptions. The resulting matrix captures the essence of the book content by assigning higher weights to words that are more specific to a particular book.

To generate recommendations, the content-based filtering model first identifies the index of a given book title in the dataset. It then computes the cosine similarity scores between this book and all other books based on their TF-IDF feature vectors. The cosine similarity measures the similarity in the content of two books, with values ranging from 0 (no similarity) to 1 (identical content). The model sorts the books based on their similarity scores and recommends the top N most similar books.

To evaluate the content-based filtering model, a test set of user preferences is used. For each user, the model predicts book recommendations based on their preferences. The accuracy of the model is calculated by comparing the predicted recommendations with the user's actual preferences. This evaluation allows us to assess the effectiveness of the content-based filtering approach in generating relevant recommendations based on book content.

KNN (Collaborative Filtering):

The KNN (k-Nearest Neighbors) model for collaborative filtering takes into account the preferences of similar users to make book recommendations. It operates on a user-item matrix, where each row represents a user, and each column represents an item (book).

The model calculates the cosine similarity between users' preferences, using the ratings they have given to different books. The cosine similarity measures the similarity between two users' preference vectors, with values ranging from -1 (dissimilar) to 1 (identical preferences). The higher the cosine similarity between two users, the more similar their preferences are.

Given a user ID, the KNN model first identifies the index of the user in the user-item matrix. It then finds the k nearest neighbors of the user, based on their cosine similarity scores. The neighbors are users who have similar preferences to the target user. Finally, the model recommends books that the neighbors have interacted with but the target user has not.

To evaluate the KNN model, a test set of user preferences is used. For each user, the model predicts book recommendations based on their preferences. The accuracy of the model is calculated by comparing the predicted recommendations with the user's actual preferences. This evaluation allows us to assess the effectiveness of collaborative filtering in generating relevant recommendations based on similar users' preferences.

Random Forest:

The random forest model combines various book attributes, including titles, authors, genres, ratings, and numerical features, to generate book recommendations. It employs a random forest regressor, a machine learning algorithm that combines multiple decision trees to make predictions.

In the random forest model, categorical features such as genres and authors are encoded using techniques like label encoding, which assigns a numeric label to each category. Numerical features such as ratings and pages are used as-is.

The feature matrix is created by combining the encoded categorical features and numerical features. The target variable is the 'likedPercent' column, representing the percentage of users who liked the book. The random forest regressor is trained on this feature matrix and target variable.

To get recommendations based on a given book title, the random forest model calculates the cosine similarity between the features of the given book and the rest of the dataset. It retrieves the indices of the most similar books based on their cosine similarity scores. Finally, the model recommends the top N most similar books.

The random forest model is evaluated using a test set of user preferences. For each user, the model predicts book recommendations based on their preferences. The accuracy of the model is calculated by comparing the predicted recommendations with the user's actual preferences. This evaluation allows us to assess the effectiveness of the random forest model in generating relevant book recommendations based on various book attributes and user preferences.

Conclusion:

In conclusion, the book recommendation project demonstrates the application of different recommendation techniques to provide personalized book recommendations to users. The project begins with the collection and preprocessing of book data, ensuring a clean and structured dataset.

The content-based filtering approach leverages TF-IDF vectorization to analyze the textual content of books and recommend similar ones. Collaborative filtering using KNN considers user preferences and identifies similar users to generate recommendations based on their interactions. The random forest model combines various book attributes to calculate similarity scores and generate recommendations.

The project evaluates each recommendation model using a test set of user preferences and compares the predicted recommendations with the actual user preferences to measure accuracy and effectiveness. The evaluation helps assess the performance of each model and identify areas for improvement.

Overall, the book recommendation project showcases the potential of leveraging data analysis, machine learning techniques, and user preferences to generate personalized book recommendations. The combination of content-based filtering, collaborative filtering, and the random forest model provides diverse approaches to cater to different user preferences and enhance the user experience in discovering new books. This project can serve as a foundation for developing more advanced recommendation systems and can be extended to other domains beyond books.