Book Recommender System

Layan Salem
Birzeit University
Layan.salem444@gmail.com

Raghad Jamhour
Birzeit University
raghadjamhour@gmail.com

Tasneem Shelleh
Birzeit University
Tasneemshelleh@gmail.com

Abstract - This project aims to develop a book recommender system based on collaborative filtering techniques to deliver personalized book recommendations. By analyzing user preferences and matching patterns in book ratings, the system provides customized suggestions. Three machine learning algorithms, Neural Networks, k-Nearest Neighbors (k-NN) and Clustering, are employed for their capability to identify similarities and patterns critical for accurate recommendations. The project evaluates the accuracy and performance of these algorithms to ensure the delivery of accurate and user-specific book suggestions.

I. Introduction

Recommendation systems are essential tools that personalize suggestions based on user preferences. In this project, we developed a book recommendation system using collaborative filtering, a technique that predicts a user's preferences by analyzing the behaviors and preferences of similar users. By leveraging patterns in user interactions, this method allows the system to make accurate recommendations, helping users discover books that align with their tastes.

To achieve this, we employed three algorithms: K-Nearest Neighbors (KNN), Artificial Neural Networks (ANN), and clustering. A major challenge was dealing with imbalanced data, where a significant portion of ratings were unimportant (e.g., zeros) where the user didn't rate the book. By carefully preprocessing the data and dropping these unimportant entries, we created a more meaningful dataset to train and evaluate our models. Our work not only highlights the strengths of each algorithm in handling recommendation tasks but also highlights the impact of data preprocessing in improving model performance. Most full papers are around six pages in length.

II. Dataset

The dataset utilized in this study comprises three core components: books, users, and ratings. The Books Dataset contains detailed metadata for each book, including ISBN, Title, Author, Year of Publication, and Publisher. The Users Dataset provides demographic information such as User-ID, Location, and Age. The Ratings Dataset records the ratings provided by users to books on a scale, capturing their preferences.

Preprocessing was applied to ensure data quality, including data cleaning and filtering. Ratings with a value of zero (or no value) were removed due to their large number, as they could skew the analysis. Additionally, users with fewer than 200 ratings and books with fewer than 50 ratings were filtered out to retain meaningful interactions. A user-item interaction matrix was constructed from this data, representing user-book relationships with ratings as values. Sparse matrix representations were employed to efficiently handle the large-scale dataset, and dimensionality reduction techniques, such as Truncated SVD, were used to retain the most informative features while reducing computational complexity. The maximum length is eight pages, which includes references and all figures and/or tables.

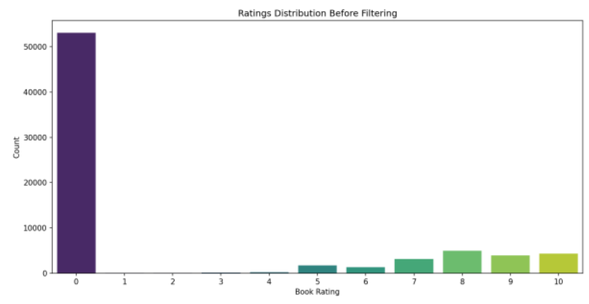

FIGURE 1 RATING DISTRIBUTION BEFORE FILTERING

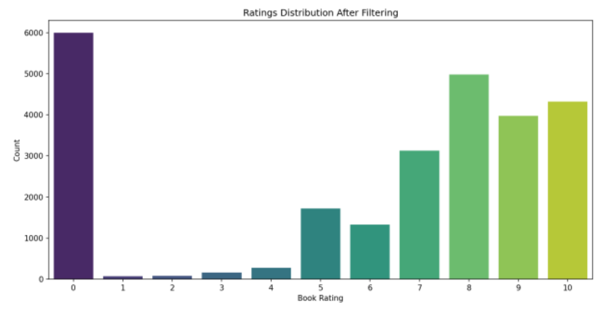

FIGURE 2 RATING DISTRIBUTION AFTER FILTERING

III. Neural Collaborative Filtering

In this implementation, we developed a recommendation model that combines matrix factorization (MF) and multilayer perceptron (MLP) architectures. The approach leverages embeddings for users and items, which are learned during training. The MF component captures latent interactions between users and items using Generalized Matrix Factorization layer (GMF) through an element-wise dot product, while the MLP component models' non-linear relationships by concatenating the user and item embeddings and passing them through several dense layers. We utilized a deep learning framework, TensorFlow, to construct and train the model, applying techniques like early stopping to prevent overfitting. The model uses binary cross-entropy loss and metrics like precision, recall, and F1-score to evaluate performance. To address class imbalance, dynamic sample weights are applied, ensuring that the model gives more importance to underrepresented classes during training along with dropping irrelevant zeros. This approach is inspired by Neural Collaborative Filtering paper. [1]

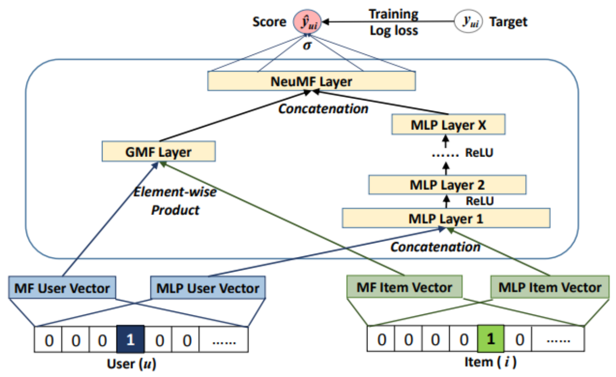

FIGURE 3. NEURAL COLLABORATIVE FILTERING [1]

IV. K-Nearest Neighbors (KNN)

In this implementation, the K-Nearest Neighbors (KNN) algorithm was employed to build a collaborative filtering recommendation system by identifying users with similar rating patterns to predict preferences for unrated books. The process began with data preprocessing, where users with fewer than 100 ratings and books with fewer than 50 ratings were filtered out. To address imbalanced data, a subset of zero-rated entries was sampled while retaining all non-zero ratings, and a user-item matrix was created with missing ratings filled as zeros. Truncated SVD helps transform the large and sparse user-item matrix into a compact, manageable representation that retains the most relevant information, enabling the KNN model to run efficiently.

The model utilized cosine similarity to compute user similarity, and the Nearest Neighbors class from scikit-learn was employed to train the model. Recommendations were generated by aggregating ratings from the top 5 nearest neighbors, with fallback popularity-based recommendations for new users. The system was evaluated using precision, recall, F1-score, and accuracy, and a confusion matrix provided insights into model predictions, showcasing the effectiveness of collaborative filtering and dimensionality reduction

V.Clustering

K-means clustering is a machine learning technique that helps group data into K clusters based on how similar the points are to each other. It begins by picking K random points as starting centers (called centroids). Then, each data point is grouped with the nearest centroid based on distance, can be measured by cosine similarity distance. The centroids are then recalculated as the average position of all the points in their group. This process repeats until the centroids settle into stable positions, creating clusters where the points in each group are as similar as possible.
In this implementation, we built a book recommendation system using K-means clustering to group users based on their rating patterns.

The system starts by processing the data to create a user-item matrix, where each user's ratings for different books are captured. Users are then grouped into clusters that reflect similar preferences. Recommendations are made by identifying other users in the same cluster, by finding books highly rated by similar users. To ensure the system works well, we evaluate it using metrics like precision, recall, F1-score, and accuracy. We also handle missing data by filling in default ratings and focus on active users and popular books to keep the recommendations relevant. This approach combines collaborative filtering and clustering to generate personalized and diverse book recommendations, leveraging the scikit-learn framework.

VI. Evaluation

The models were evaluated using a range of metrics, including precision, recall, F1-score, and accuracy, to ensure comprehensive performance assessment. Confusion matrices were analyzed to understand the balance between true and false predictions. For clustering-based and collaborative filtering models, cosine similarity metrics were leveraged to evaluate recommendation quality, while classification models were tested for their ability to correctly predict user preferences. The results demonstrated the robustness of the models in accurately recommending books based on user interactions and preferences. Additionally, stratified train-test splits ensured that data distribution was preserved across sets, providing a reliable testing environment for model evaluation. These metrics, combined with detailed analyses, confirmed the effectiveness of the implemented approaches in handling sparse data and generating personalized recommendations.

VII. Model metrics examples

Neural Collaborative Filtering			
Accuracy	Precision	Recall	F1 Score
0.87	0.9	0.96	0.93

K-Nearest Neighbors			
Accuracy	Precision	Recall	F1 Score
0.74	0.92	0.69	0.79

Clustering			
Accuracy	Precision	Recall	F1 Score
0.27	0.89	0.2	0.33

VIII. Metric Variations Across Algorithms in Recommender Systems

When testing different algorithms, it's natural to observe variations in the performance metrics because each algorithm processes data and learns patterns in a unique way. For instance, K-Nearest Neighbors (KNN) makes predictions by finding the closest data points in the feature space, which means it relies heavily on the structure of the data and the distance metric used. On the other hand, Neural Collaborative Filtering uses deep learning to uncover hidden relationships between users and items, often capturing more complex interactions that simpler methods might miss. Clustering algorithms, by contrast, are designed to group similar data points together and may focus on entirely different aspects of the dataset, such as underlying distributions or natural divisions.

These differences can lead to variations in metrics like accuracy, precision, recall, and F1-score, as each algorithm performs better or worse depending on the dataset's characteristics such as sparsity, size, or noise. For example, KNN might perform well with dense data but struggle with sparsity, while Neural Collaborative Filtering may excel at capturing latent features in sparse datasets. Additionally, factors like hyperparameter tuning, feature selection, and the algorithm's computational complexity also influence the results. Testing multiple algorithms not only highlights these differences but also helps identify the most effective one for the specific task. This process ensures that the chosen model aligns with the problem's requirements and maximizes the overall performance of the recommender system.

References

[1] L. L. H. Z. Xiangnan He, "Neural Collaborative Filtering," 2017.