Book Recommendation System

Sandesh Gaikwad
Data Science Trainee,
AlmaBetter, Bangalore

Abstract:

Recommender systems are machine learning systems that help users discover new products and services.

A recommendation system helps an organization to create loyal customers and build trust by their desired products and services for which they came on your site. A book recommendation system is a type of recommendation system where we have to recommend similar books to the reader based on his/her interest. The books recommendation system is used by online websites which provide ebooks like google play books, open library, Goodreads, etc

Keywords: machine learning, regression, predictive modeling, trip time prediction

Problem Statement:

Use the given dataset to build a recommender system. In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy, or anything else depending on industries).

Data Overview

The Book-Crossing dataset comprises 3 files.

- **Users**

Contains the users. Demographic data is provided (Location, Age) if available. Otherwise, these fields contain NULLvalues. The Data set Contains 271354 Rows

- **Books**

Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given (Book-Title, Book-Author, Year of publication, Publisher), obtained from Amazon Web Services. Note that in the case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavors (Image-URL-S, Image-URL-M, Image-URL-L), i.e., small, medium, and large. These URLs point to the Amazon website. The Data set Contains 278858 Rows

- **Ratings**

Contains the book rating information. Ratings (Book-Rating) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0. The Data set Contains 1149780 Rows

Data Cleaning and Preprocessing:

BOOKS:

All the columns in the book table are just to give more information about the books. Missing value from this table won't impact the recommender algorithm.

Ratings: there are no null values in the rating column but a lot of ratings are just zero we remove that and only consider where explicit 0 to 10 ratings are available.

User: we have info on the demographic location and age of the user but we won't be writing any algorithm that uses this information.

Merge all datasets :

We will merges users and ratings datasets on user_id and books dataset using ISBN

Feature engineering: ‘

Weighted Rating: We will create new weighted rating using below formula

$$wr = [vR/(v + m] + [mC/(v + m)]$$

Where , v = number of votes
M = minimum votes required to be listed on chart,
R = average rating of book

Recommender system :

Popularity based:

We will only consider top 5 % books. Using the weighted rating we recommend top rated books. To all the new users.

Collaborative filtering:

This method makes automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on a set of items, A is more likely to have B's opinion for a given item than that of a randomly chosen person

Collaborative Filtering (CF) has two main implementation strategies:

Memory-based

- This approach uses the memory of previous users interactions to compute users similarities based on items they've interacted (user-based approach) or compute items similarities based on the users that have interacted with them (item-based approach).

- A typical example of this approach is User Neighbourhood-based CF, in which the top-N similar users (usually computed using Pearson correlation) for a user are selected and used to recommend items those similar users liked, but the current user have not interacted yet. This approach is very simple to implement, but usually do not scale well for many users.

Model-based

- In this approach, models are developed using different machine learning algorithms to recommend items to users. There are many model-based CF algorithms, like Neural Networks, Bayesian Networks, Clustering Techniques, and Latent Factor Models such as Singular Value Decomposition (SVD) and Probabilistic Latent Semantic Analysis.

Matrix Factorization

- Latent factor models compress the user-item matrix into a low-dimensional representation in terms of latent factors. One advantage of using this approach is that instead of having a high-dimensional matrix containing an abundant number of missing values we will be dealing with a much smaller matrix in lower-dimensional space.

- A reduced presentation could be utilized for either user-based or item-based neighborhood algorithms. There are several advantages to this paradigm. It handles the sparsity of the original matrix better than memory based ones. Also comparing similarity on the resulting matrix is much more scalable, especially in dealing with large sparse datasets.

- An important decision is choosing the number of factors to factor in the user-item matrix. The higher the number of factors, the more precise the factorization in the original matrix reconstructions. Therefore, if the model is allowed to memorize too many details of the original matrix, it may not generalize well for data it was not trained on. Reducing the number of factors increases the model generalization.c

Evaluation

In Recommender Systems, there are a set of metrics commonly used for evaluation. We choose to work with **Top-N accuracy metrics**, which evaluate the accuracy of the top recommendations provided to a user, compared to the items the user has interacted in the test set.

This evaluation method works as follows:

- For each user For each item, the user has interacted in the test set
- Sample 100 other items the user has never interacted with.
- Ask the recommender model to produce a ranked list of recommended items, from a set composed of one interacted item and the 100 non-interacted items
- Compute the Top-N accuracy metrics for this user and interacted item from the recommendations ranked list
- Aggregate the global Top-N accuracy metrics

Conclusion :

- We get hits in the top 5 for 26 % of the times
- We get hits in the top 10 for 36 % of the times
- The high volume of data is limiting the model complexity
- We need to build a more complex model for the recommender system to improve the accuracy
- The model should account for age, location and we can ask the user for interest in the genre to further narrow down the results

References:

- Almabetter Notes
- Kaggle
- Medium articles