Final Project-Milestone

Michalis Baltsiotis 2578, Nikolaos-Gerasimos Zazatis 2723, team D

November 2022

1 Introduction

This document serves as Milestone to the final project, to make sure we are on track. In project proposal we gave a brief explanation of the Recommendation System and its applications on real world problems, its implementation using different ml models and what we aim to achieve. In the current state of the project, we dived deeper in the problem, using the book recommendation dataset, as mentioned in project proposal, performed eda, cleaned the data, implemented item based and latent factors collaborative filtering.

2 Dataset

The dataset consists of 3 csv files, one containing book information, the other containing user information and the third containing the ratings between users and books. The rating can be explicit, in the range of [1,10], or implicit, with the value of 0 meaning that the user has bought/read but did not rate the book. Because the value 0 will be interpreted as the lack of interaction between user and book, we replaced every 0 with the value 5.5, which is the mean of 1 and 10. The dataset can be found at
https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset.

3 EDA and Data Cleaning

Before joining the 3 files to one dataset, we need to perform EDA in order to recognize trends for different ages, authors etc. and invalid values/ outliers.

We also need to perform data cleaning before we can pass the dataset to a ml model. In the current dataset, we imputed the missing values to features Age and Year-Of-Publication with the mode, which is more robust that mean in presence of outliers, while adding to those values a random number drawn from a normal distribution with mean 0 and standard deviation 4 to make the distributions more gaussian like. Also, we filtered out the rows with Age bigger than 100 and Age less than 1, which can be considered an outlier, as their total number is too small compared to the whole dataset. Furthermore, we observed that the popularity of some books changes for different age groups. Finally, because of the size of the dataset, we choosed a subset where there are:

- Books who have at least 60 reviews
- Users who have reviewed at least 10 books

and split it to train and test with fraction 9/1. To make the Rating matrix we used the train dataset

4 Baseline

Before we can implement and evaluate different ml models, we need a baseline where every other model must perform at least a little better than the former. For our baseline we choose to predict a rating between a user and a book at random. By evaluating the baseline on test dataset we get:

- Mean Absolute Error (MAE) = 1.6068
- Root Mean Squared Error (RMSE) = 2.4452

5 Item Based Collaborative Filtering

For the first implementation of our recommender, we used item based collaborative filtering, which aims to find similar items/books that a user have reviewed to a book that the user has not reviewed. Then selects k-closest and averaging their rating to find the prediction for the book. To be more precise, every rating is weighted with the similarity, so that more similar books will have bigger impact on the rating than less similar books. The formula is as follows:

$$r_{ij} = \frac{\sum_{n=1}^{k} s_{nj} * r_{in}}{\sum_{n=1}^{k} s_{nj}}$$

where r_{ij} is the rating of book j from user i and s_{nj} is the similarity of book n with book j. To find the nearest items/books we used k Nearest Neighbors algorithm, with metric cosine. This returns the distances d_{ij} between books i,j in the range [0,1]. To find the similarity we just need to use the formula: $s_{ij} = 1 - d_{ij}$. We then return the top n recommendations for that user. To evaluate its performance, we only use the test dataset, as the kNN algorithm does not have weights or bias to learn. By doing so we get:

- MAE: 0.5555
- RMSE: 1.2303

which is much better than the random rating.

6 Latent Factors Collaborative Filtering

For the second implementation of our recommender, we used latent factors collaborative filtering, which factorizes the original rating matrix. In particular, we used SVD which factorizes the n*m R matrix to:

- U matrix, with dimensions n*k
- S diagonal matrix, with dimensions k*k
- V matrix, with dimensions m*k

, where k are the latent factors, a hyperparameter that needs to be tuned. A good rule of thumb is $k = \sqrt{min(n, m)}$. To revieve the new R' matrix we use the formula: $R' = USV^T$. The SVD can be calculated with direct methods or iterative methods. The most common approach on large scale problems is iterative methods, where it aims to: $R' \approx R$. Then, the m non-rated items with the highest value on R' are recommended. We evaluate this approach on both test and train dataset. On test dataset, every value is close to 0, because the original R matrix had 0 in those positions, so we need to add the value 5.5 to every prediction to get a more accurate rating. By doing so we get:

- Train evaluation: MAE: 4.6080 RMSE: 5.0258
- Test evaluation: MAE: 1.2370 RMSE: 1.7445

It seems to performing better than baseline but worse than item based collaborative filtering. But its positives are:

- Much faster to fill the R matrix than item based collaborative-filtering
- Can be stored with less memory (n*k + k*k + m*k is generally less than n*m)

7 Next Steps

After trying those simpler ML algorithms, we now aim to use deep learning to make a recommender. In particular, we aim to use more information about the dataset, such as age, year of publication and author. We will start with simple DL architectures such as Feed Forward NN, CNN and RNN and then we will use the functional API to combine some models together. Lastly, we will try to implement a hybrid model wich uses both collaborative methods and DL methods.