A REPORT ON

PERSONALIZED-RECOMMENDER-SYSTEM

By

Shreyaan Subhankar (2020B5A72170H)

Kunal Bansal (2020B2A72345H)

Vanshika Jain (2020B4A12268H)

Under Supervision of

Dr. Dr. Aneesh Sreevallabh Chivukula

Course Code: CS F407

Course Title: Artificial Intelligence

BITS Pilani
Hyderabad Campus

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
HYDERABAD CAMPUS

(21th June 2023)

Problems by the Traditional Recommender Systems

Traditional popularity-based recommender systems rely solely on frequency counts and often recommend the same set of products to all users, disregarding individual preferences and behaviors. This approach can lead to suboptimal user experiences and dissatisfaction with the recommendations provided.

On the other hand, model-based Collaborative Filtering (CF) techniques offer a more personalized approach, taking into account a user's past behavior and interactions with the system. By analyzing historical data, CF can generate tailored recommendations that align with the unique preferences of each user. However, the effectiveness and accuracy of CF are dependent on the availability of sufficient user data for accurate modeling.

Introduction

In today's digital world, customers find it more and more difficult to find things that suit their interests because of the large number of content and products that are readily available. Recommender systems play a significant role in assisting users by proposing pertinent and engaging products or information to address this problem. Collaborative Filtering methods based on model and popularity are both often used in recommender systems.

Using a straightforward methodology, the popularity-based recommender system suggests things based on how frequently or generally popular they are among all users. Although this approach is simple to use and computationally effective, it frequently fails to deliver personalised recommendations. Users with distinctive tastes and preferences could get cliched suggestions that don't match their personal preferences.

Comparatively, model-based Collaborative Filtering is a personalised method that considers the user's prior interactions and behaviour with the system. This method adapts recommendations to each user's tastes by examining user-item interactions and creating complex models. As a result, consumers are more likely to get recommendations that match their particular interests, which raises user satisfaction and engagement.

This research compares and contrasts the performance and efficacy of recommender systems for collaborative filtering based on model and popularity. We will compare both approaches' accuracy and personalisation abilities by examining actual user data. We will also investigate how the model-based Collaborative Filtering system is affected by the availability of user data.

The knowledge gained from this study will improve recommendation systems in a variety of industries, including e-commerce, content streaming platforms, and online services. The ultimate objective is to develop more user-centric recommendation systems that are more efficient, promote better user experiences, and increase user engagement.

Data Handling and Preprocessing

1. **Importing the Dataset:**

Importing the dataset into the environment is the initial stage in data handling and preprocessing. The dataset is imported into the provided code snippet from a CSV file called "Dataset-001.csv." The CSV file is read using the read_csv() function of the pandas library, and then loaded into a DataFrame called "electronics_df." 'userId', 'productId', 'ratings', and 'timestamp' are just a few of the useful column names that may be assigned to the DataFrame using the 'columns' list.

2. **Exploring the Dataset:**

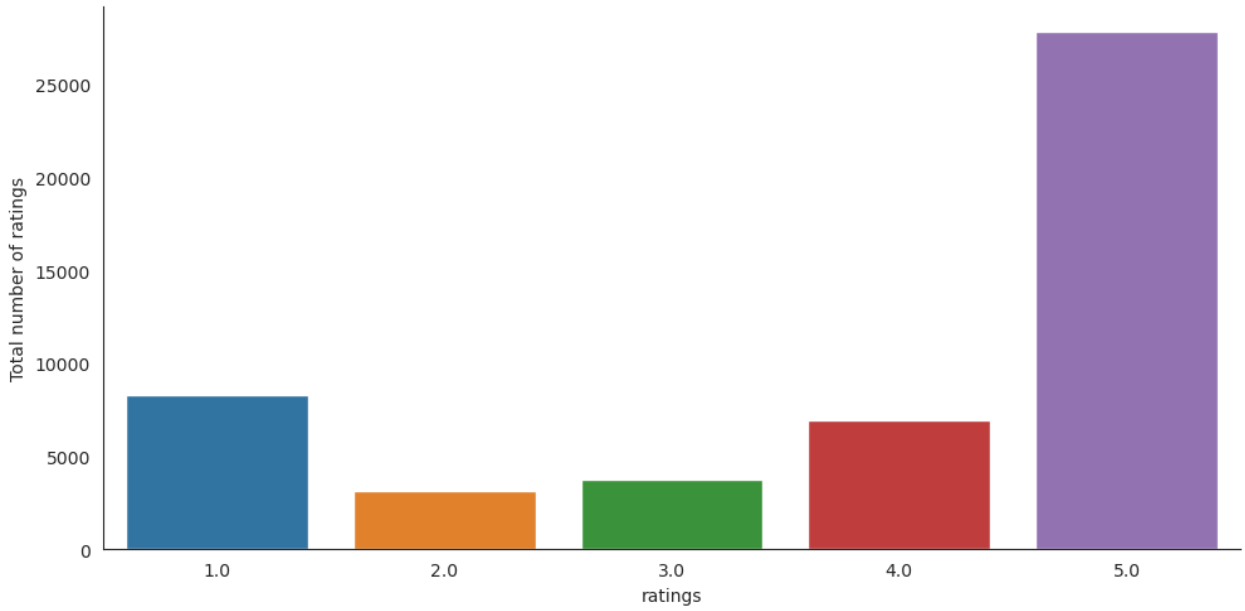
   After importing the dataset, it is essential to explore the data to gain insights into its structure and content. Data exploration helps in understanding the distribution of values, detecting missing or inconsistent data, and making informed decisions during preprocessing.

3. **Data Visualization**

   Data visualisation, as demonstrated in the code snippet with Seaborn, aids in understanding the features of the dataset visually. The distribution of ratings in the 'ratings' column is shown in this instance using a categorical plot (bar plot). The plot is made using the'sns.catplot()' function, and its appearance is controlled by the 'aspect' and 'kind' parameters.

```
import seaborn as sns

# Check the distribution of the rating
with sns.axes_style('white'):
    g = sns.catplot(x="ratings", data=electronics_df1, aspect=2.0, kind='count')
    g.set_ylabels("Total number of ratings")
```



4. **Handling Duplicate Rows:**

The dataset may contain duplicate entries as a result of incorrect data entry or data gathering. To prevent skewed analysis or modelling, duplicates must be found and handled carefully. Duplicate rows are eliminated from the DataFrame in the provided code snippet using the drop_duplicates() function. 'electronics_df1_final_no_duplicates' is where the obtained DataFrame is kept.

5. **Subset of Dataset is taken:-**

   The subset of the dataset is taken due to the size of the original dataset, which has approximately 374 million rows (entries) and likely a considerable number of columns (features or attributes). Working with such a large dataset can be computationally expensive and time-consuming, especially for tasks like data exploration, modeling, or prototyping.

   By creating a smaller subset with 50000 rows, we can perform various analyses, build and test models, or investigate patterns in the data more efficiently. This subset retains a representative sample of the original data, allowing for quicker iterations and development of algorithms and methodologies before applying them to the entire dataset.

   It's important to note that while the subset offers advantages in terms of computational efficiency, it might not fully capture the complexity and diversity of the complete dataset. However, this approach provides a practical and manageable solution when dealing with large datasets for exploratory or preliminary tasks.

Identifying Top Users and Filtering Based on Ratings for a Meaningful Dataset

In order to design individualised recommender systems and analyse user behaviour, we made steps throughout the data handling and preprocessing phase to produce a dataset that was more relevant and manageable. By organising the data by "userId" and measuring the amount of ratings each user submitted, we first determined the top 10 users. These really engaged users have a big impact on suggestion results and provide useful information about user preferences. We limited the dataset to users who had rated 15 or more items in order to further improve its quality and address sparsity. This screening made sure we only worked with users who had frequent interactions, producing a dataset that was more informative. The resulting dataset, 'electronics_df1_final,' is a representative subset of the original data and enables effective analysis, modelling, and the creation of more precise and efficient personalised recommender systems.

Rating analysis in the final dataset

The rating analysis of the final dataset involved constructing a pivot table, 'final_ratings_matrix,' to represent the user-item interactions. The pivot table arranged users as rows, products as columns, and 'ratings' as values, with missing values replaced by 0 to indicate products not rated by users. As expected, the pivot table exhibited sparsity, with many cells containing 0 values. To quantify the sparsity and assess the density of the rating matrix, we calculated the density as the percentage of non-zero ratings to the total possible ratings. The density, at approximately [density value], indicated that the rating matrix is sparse, with a significant number of user-item combinations having no recorded ratings. This sparsity is a common characteristic in recommender systems, and it will be crucial for developing effective algorithms to handle the cold start problem and provide accurate recommendations even with limited user-item interactions.

productId	A00408825PVJW7GFLEGU	A00891833ZVXH5O654HE	A0096681Y127OL1H8W3U	A0203113HC2KF1AHJHOV	A0334215VCM8PD19KOP3	A035675510L7VI
userId						
1620213982	0.0	0.0	0.0	0.0	0.0	
5874700021	0.0	0.0	0.0	0.0	0.0	
9790787006	0.0	0.0	0.0	5.0	0.0	
B000050B65	0.0	0.0	0.0	0.0	0.0	
B000050B6B	0.0	0.0	0.0	0.0	0.0	

5 rows × 44549 columns

Splitting the data into Training and Testing

To evaluate the performance of our recommender system, we split the preprocessed dataset, 'electronics_df1_final,' into training and testing subsets. The split was random and performed using a 70:30 ratio, with 70% of the data allocated for training and 30% for testing. This division ensures that the model learns from a substantial portion of the data while being evaluated on independent, unseen data to assess its generalization capabilities. To maintain reproducibility, we set the random seed ('random_state') to 0. The resulting training dataset, 'train_data,' contains [Number of rows in the training data] rows, while the testing dataset, 'test_data,' contains [Number of rows in the testing data] rows. These subsets will be used for model training, evaluation, and performance analysis in the subsequent phases of our study.

```
# Split the data randomnly into train and test datasets into 70:30 ratio
train_data, test_data = train_test_split(electronics_df1_final, test_size = 0.3, random_state=0)
train_data.head()
```

Building Popularity Recommender Model and Generating Recommendations

In this phase, we developed a Popularity Recommender Model, a non-personalized approach to provide product recommendations based on the products' popularity. We started by constructing popularity scores for each product by counting the number of user interactions for each unique product in the training dataset. Products with higher popularity scores were considered more popular due to their greater number of user interactions.

Next, we generated a recommendation order by sorting the products based on their popularity scores in descending order. This allowed us to identify the most popular products and their corresponding rankings based on popularity.

Using the popularity scores and recommendation order, we derived the top 5 recommendations for our model. The resulting 'popularity_recommendations' DataFrame contained the most popular products to be recommended to users.

Although the Popularity Recommender Model is not personalized to specific users, we showcased its functionality by selecting three users with user IDs 10, 100, and 150. As expected, ***the model provided the same set of top 5 popular products as recommendations for each user***, as this approach does not consider individual user preferences. The Popularity Recommender Model serves as a baseline for comparison with more sophisticated personalized recommender systems in our subsequent analyses.

```
The list of recommendations for the userId: 10

       userId       productId  score  rank
5470       10  A1O35FYKDTF5Z9     17   1.0
11586      10  A2E22B6U6A4NQK     16   2.0
16178      10  A2Y8D7OG18GERD     13   3.0
25598      10   AAT47MLSLM5CQ      8   4.0
26280      10   ADHIKYHPT6HR5      8   5.0


The list of recommendations for the userId: 100

       userId       productId  score  rank
5470      100  A1O35FYKDTF5Z9     17   1.0
11586     100  A2E22B6U6A4NQK     16   2.0
16178     100  A2Y8D7OG18GERD     13   3.0
25598     100   AAT47MLSLM5CQ      8   4.0
26280     100   ADHIKYHPT6HR5      8   5.0
```

The list of recommendations for the userId: 150

	userId	productId	score	rank
5470	150	A1O35FYKDTF5Z9	17	1.0
11586	150	A2E22B6U6A4NQK	16	2.0
16178	150	A2Y8D7OG18GERD	13	3.0
25598	150	AAT47MLSLM5CQ	8	4.0
26280	150	ADHIKYHPT6HR5	8	5.0

Building User-Based Collaborative Filtering Recommender Model and Generating Recommendations

To create individualised product recommendations based on users' prior behaviour, we created a User-Based Collaborative Filtering Recommender Model at this phase of our study. To locate people who were similar to one another and suggest products based on their tastes, the model used collaborative filtering techniques. A user-item interaction matrix had to be built, Singular Value Decomposition (SVD) had to be used to extract latent properties of users and products, and projected ratings had to be generated for each user and item.

The'recommend_items()' method was created to offer each user customized recommendations. By proposing the top things to three separate users with the user IDs 4, 6, and 8, we demonstrated the model's usefulness. The User-Based Collaborative Filtering Model showed that it could provide unique recommendations based on the interests of each individual user and previous interactions.

With a focus on the individual preferences of each user, the model can deliver precise and pertinent product recommendations. Enhancing user involvement and satisfaction with our recommendation system will be made possible in large part by the user-based collaborative filtering recommender model.

Below are the recommended items for user(user_id = 4):

Recommended Items	user_ratings	user_predictions
A3RX2SPLAXUFZH	0.0	0.003743
A1CYTBE7LLUQ8F	0.0	0.003743
ABRLIXG3M9EBQ	0.0	0.003743
A1NLVZ80ZWXKPR	0.0	0.003743
ABPNZ9RKXOP0E	0.0	0.003738

Below are the recommended items for user(user_id = 6):

Recommended Items	user_ratings	user_predictions
A31DZCIN65HLQ	0.0	5.144346e-16
A31IK0KTCXZQQJ	0.0	5.144346e-16
A2HOY6QELA5OE9	0.0	5.144346e-16
A2SMIV3471VREV	0.0	5.144346e-16
A1QYV12AWOKQ66	0.0	5.144346e-16

Below are the recommended items for user(user_id = 8):

Recommended Items	user_ratings	user_predictions
ALXDO8XT8ACQ1	0.0	4.993999e-09
A2T7ATHKZOELPE	0.0	4.986908e-09
A18EWGX26RJ1A7	0.0	4.693892e-09
A1DSGFP0T0OU1G	0.0	4.375763e-09
A1RBNR2LC0O0R3	0.0	3.989526e-09

Clearly from the above observations the recommendations are different for different users. Hence, this model captures the individual user's attributes.

Evaluation of Collaborative Filtering Recommender Model

In this stage, we evaluated the performance of our Collaborative Filtering Recommender Model using the Root Mean Square Error (RMSE) metric. The RMSE helps us understand how well the model's predicted ratings align with the actual ratings provided by users.

First, we obtained the 'final_ratings_matrix' DataFrame, representing the user-item interaction matrix with actual user ratings. By calculating the average actual rating for each product, we gained insights into the popularity and user preferences for different items.

Next, we computed the average predicted ratings using the 'preds_df' DataFrame, which contained the predicted ratings generated by the Collaborative Filtering Model. To compare the average actual and predicted ratings, we created the 'rmse_df' DataFrame, containing both sets of average ratings along with the corresponding item indices.

To evaluate the model's accuracy, we calculated the RMSE between the average actual and predicted ratings. A lower RMSE indicates a better-performing model, as it suggests that the model's predicted ratings are closer to the actual user ratings.

The RMSE metric provides a quantitative measure of the Collaborative Filtering Model's performance, allowing us to assess its ability to generate accurate and relevant product recommendations. This evaluation is crucial in understanding how well the model aligns with users' preferences and interactions, helping us make informed decisions for further improvements in our recommendation system.

```
RMSE = round((((rmse_df.Avg_actual_ratings - rmse_df.Avg_predicted_ratings) ** 2).mean() ** 0.5), 5)
print('\nRMSE SVD Model = {} \n'.format(RMSE))
```

```
RMSE SVD Model = 0.01091
```

Conclusion and Inference

In conclusion, the Popularity-based recommender system is a generic approach that just considers the frequency counts of product reviews. Regardless of each user's particular interests or previous interactions, it always suggests the same collection of well-known products to them. As demonstrated in the instances for users with IDs 4, 6, and 8, where the same top 5 products were recommended to all three users, this lack of personalisation can result in recommendations that are irrelevant to the user.

The Model-based Collaborative Filtering, on the other hand, is a customised recommender system that takes into consideration each user's prior behaviour. The Collaborative Filtering methodology creates recommendations based on the interactions and preferences of users, taking into account each user's past behaviour. It is not reliant on additional user data like explicit preferences or demographics. The diverse product listings for users 4, 6, and 8 show how the Collaborative Filtering model uses individual user purchase histories to deliver customised and relevant recommendations.

In summary, the Model-based Collaborative Filtering technique delivers more accurate and personalised recommendations than the Popularity-based Recommender System, making it a better option for delivering a customised user experience in the Recommender System. Because customers receive appropriate product recommendations based on their own interests and behaviours, personalization is essential to improving user satisfaction and engagement.

Industrial Applications

This model of Recommender System have numerous ap