Dual-Component Book Recommendation System Project

TzuHung Cheng

Executive Summary:

Introduction and Project Goals

This project aims to develop a sophisticated book recommendation system leveraging the extensive Goodbooks-10k dataset. The primary goal is to enhance the reading experience by providing personalized book recommendations through a dual-component system. This system comprises two main modules: a user-based recommendation engine and an item-based recommendation engine. By employing advanced collaborative filtering techniques, the project seeks to significantly improve user engagement and satisfaction with tailored book suggestions.

Methods and Approach

The methodology adopted for this project is comprehensive and multifaceted, involving several critical stages: data preprocessing, exploratory data analysis (EDA), algorithm selection and optimization, and user interface (UI) development.

- Data Preprocessing: This stage focuses on preparing the dataset for analysis by addressing missing values and preprocessing data as necessary by merging all the datasets to be used in models. Key attributes such as user IDs, ISBNs, and ratings are identified for subsequent analysis.
- Exploratory Data Analysis (EDA): A thorough examination of age, ratings, users is done to understand the data in detail.
- Algorithm Selection and Tuning:
    - **Item-Based Recommender System:** The objective is to develop an item-based collaborative filtering book recommendation system using the K-Nearest Neighbor (KNN) model, incorporating Adjusted Cosine Similarity to address scale sensitivity in vectors.
    - **User-Based Recommender System:** The objective here is to develop a user-based collaborative filtering book recommendation system using the Nearest Neighbors model using the similarities of the users and the item (books in this case).

Conclusions and Evaluation

We successfully developed item-based and user-based recommender systems leveraging the K-Nearest Neighbor (KNN) algorithm. Through careful preprocessing, we refined our datasets to focus on users who have provided 50 or more ratings and books that have received significant attention, ensuring the recommendations' relevance and reliability. The core of our system lies in creating a sparse matrix from user ratings, enabling efficient computation of nearest neighbors. By fitting the KNN model into this matrix, we established a foundation for making personalized book recommendations.

Our system uniquely addresses the challenge of scalability and accuracy in collaborative filtering by selecting a subset of users and items that contribute most to the recommendation process. The use of Adjusted Cosine Similarity further refines the model by mitigating the impact of varying rating scales among different users, enhancing the quality of recommendations.

Item Based Collaborative Filtering

1. Objective

In this section, our goal is to build a book recommendation system based on item-based collaborative filtering using the K-Nearest Neighbor(KNN) model. We also address the problem of the sensitivity on the scale of vectors with Adjusted Cosine Similarity. Moreover, we apply Root Mean Square Error (RMSE) to evaluate the model and make observations on the impact of hyperparameters. In the end, we build a simple interface that can recommend books to the user.

2. Methodology

2.1. Data collection

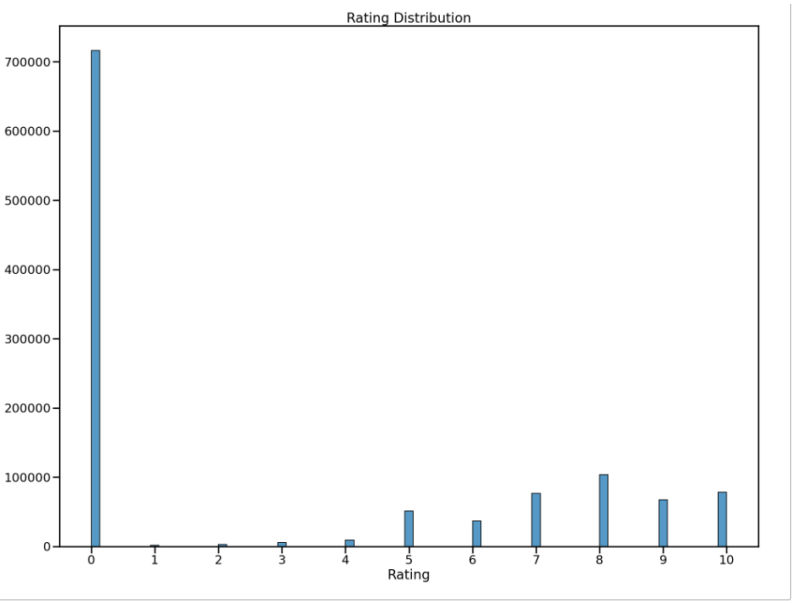
We chose Goodbooks – 10k as our dataset. It not only contains the necessary attributes we need but also provides enough data for training the model. Below are a few columns from our dataset.

	ISBN	BookTitle	BookAuthor	YearOfPublication	Publisher	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com.
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com.
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com.
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com.
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	http://images.amazon.com.
5	0399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	http://images.amazon.com.
6	0425176428	What If?: The World's Foremost Military Histor...	Robert Cowley	2000	Berkley Publishing Group	http://images.amazon.com.
7	0671870432	PLEADING GUILTY	Scott Turow	1993	Audioworks	http://images.amazon.com.
8	0679425608	Under the Black Flag: The Romance and the Real...	David Cordingly	1996	Random House	http://images.amazon.com.

	UserID	ISBN	BookRating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6
5	276733	2080674722	0
6	276736	3257224281	8
7	276737	0600570967	6
8	276744	038550120X	7
9	276745	342310538	10

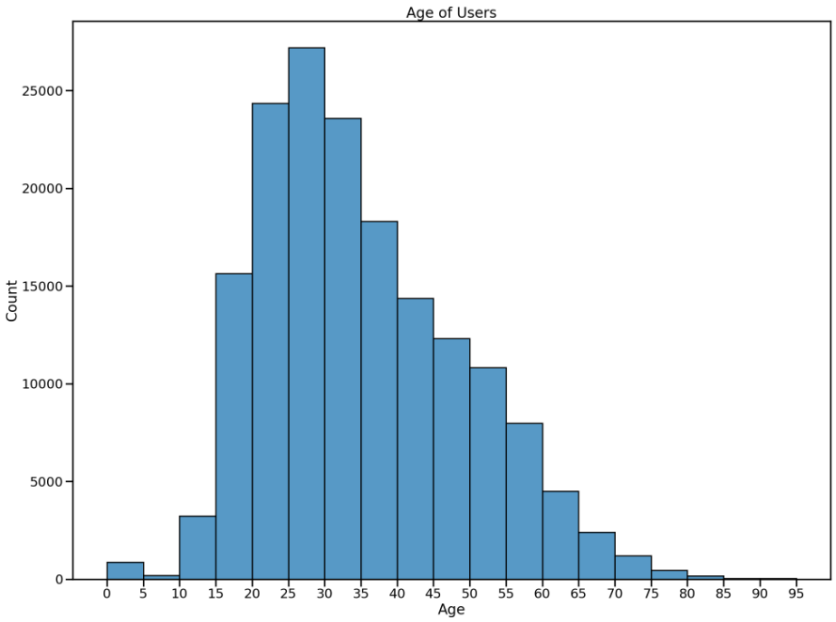
2.2. Exploratory Data Analysis (EDA)

2.2.1. Rating Distribution



We observed that rating 0s occupied a large proportion of our dataset. It is difficult to make assumptions that the interactions between the book and the user have not occurred or have not been recorded. These 0 values not only enhance noise but also enlarge the sparsity of our dataset.

2.2.2. Age Distribution



The age distribution of users is very similar to the Gaussian distribution. Most of the users are between 20 to 50 years old. This distribution might cause trouble in precisely recommending books to kids, teenagers and elders.

2.3. Data preprocessing

Before we started building the recommendation system, we checked if there were any missing data.

```
In [4]: (books.isnull() | books.empty | books.isna()).sum()
```

```
Out[4]: ISBN                 0
        BookTitle            0
        BookAuthor           2
        YearOfPublication    0
        Publisher            2
        ImageUrlS            0
        ImageUrlM            0
        ImageUrlL            3
        dtype: int64
```

```
In [8]: (ratings.isnull() | ratings.empty | ratings.isna()).sum()
```

```
Out[8]: UserID        0
        ISBN          0
        BookRating    0
        dtype: int64
```

We filtered out books and users with a total count of ratings greater than 50 from BX-books.csv and BX-Book-Ratings. Then, we merged the two data frames on 'ISBN' and dropped the other attributes, only keeping 'ISBN', 'UserID', and 'BookRatings'.

```
# Filter out the users with less than 50 reviews
ratings_count = ratings.groupby('UserID')['ISBN'].count()
active_users_id = ratings_count[ratings_count > 50].index.tolist()
active_users = ratings[ratings['UserID'].isin(active_users_id)]
active_users
```

```
#Filter out the books with less than 50 ratings
item_ratings_count=active_users.groupby('ISBN')['UserID'].count()
popular_items_id = item_ratings_count[item_ratings_count > 50].index.tolist()
popular_ratings = active_users[active_users['ISBN'].isin(popular_items_id)]
popular_ratings
```

	UserID	ISBN	BookRating
173	276847	0446364193	0
413	276925	002542730X	10
426	276925	0316666343	0
429	276925	0385504209	8
453	276925	0804106304	0
...	...	...	...
1149696	276688	0446604666	0
1149712	276688	0553569155	0
1149715	276688	0553575104	6
1149726	276688	0679751521	0
1149731	276688	068484267X	0

99223 rows × 3 columns

Next, we create an item-user pivot table. Each column represents the user's rating among different books and each row represents the book's rating among different users. Since it is almost impossible for a book to be read by every user. Most of the values in the data frame will be NaN. We need to fill these empty values with 0.

UserID	243	254	507	626	638	643	741	882	929	1025	...
ISBN											
002026478X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
002542730X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060008032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060085444	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060096195	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
006016848X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060173289	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060175400	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
006019491X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060199652	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060391626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060392452	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060502258	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060915544	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
0060916508	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

We can now split the data into training data and testing data. We randomly sample 20% of the data to test data and use the remaining data as training data.

```
split_ratio = 0.2
test_df = ratings_pivot.sample(round(len(ratings_pivot)*split_ratio), axis=1)
train_df = ratings_pivot.drop(test_df.columns, axis=1)

# Convert to Numpy array
test = test_df.values
train = train_df.values
print(test.shape,train.shape)
```

```
(1063, 213) (1063, 2942)
```

2.4. Model Approach

Metrics used:

- **Cosine Similarity:**
  - This is the raw cosine similarity between two items. It is calculated as the dot product of the item rating vectors divided by the product of their magnitudes.

- **Adjusted Cosine Similarity:**
    - This is a modification of cosine similarity that aims to address the issue of user rating bias. It subtracts the average rating of each user from their ratings for a given pair of items before calculating the cosine similarity.

3. **KNN Model Evaluation**

   The code evaluates the KNN model's performance using Root Mean Squared Error (RMSE) for different values of k (number of neighbors) and both Cosine Similarity and Adjusted Cosine Similarity metrics.

   - **KNN Model Training:** The KNN model is trained on the train_df. For each item in the test set, the model predicts ratings for unseen items based on the weighted ratings of its k most similar neighbors identified in the training set.
   - **RMSE Calculation:** The RMSE is calculated by comparing the predicted ratings from the KNN model with the actual ratings in the test_df. Lower RMSE indicates better model performance, meaning the predictions are closer to the actual user ratings.

4. **RMSE Evaluation**
   - **Cosine KNN Model**: RMSE was calculated for k values ranging from 1 to 20. The analysis took approximately 12 minutes and 47 seconds, indicating the computational intensity of calculating pairwise item similarities.

   - **Adjusted Cosine KNN Model**: Similarly, RMSE values were assessed for the same range of k, with the computation taking around 2 minutes and 59 seconds. This faster performance suggests an efficiency improvement with the Adjusted Cosine measure.

5. Evaluation

Next, we used test data to simulate how the trained model predicts new data. The prediction method is broadly similar to the above, with the difference being that we need to reserve a portion of the scores so that the model can use these ratings to calculate the similarity values for individual items. Therefore, based on the original ratings, we reserved 80% of the ratings to predict the remaining 20%. Moreover, we used the two different similarity methods mentioned above for prediction, enabling us to obtain the RMSE values for the test data to compare which method performs better. Consistent with theoretical expectations, we found that adjusted cosine similarity provides a better prediction of ratings.

6. Application

We designed an interface where users can obtain recommended books by typing the book title. The output will show the information about each book including the book’s name, author, year of publication and cover image, etc.

7. Conclusions

In this implementation, we not only learned the principles of recommendation systems but also discovered the strength of item-based collaborative filtering. Due to its exceedingly simple computation, it can make predictions quickly. However, overly simplistic methods often come with numerous limitations. After using this method to build a recommendation system, we identified several shortcomings of this model. For example, when using sparse data, this model cannot predict accurately and is easily influenced by noise. Additionally, if there are just a few ratings, it cannot predict accurately, especially when new items are introduced, often resulting in inaccurate predictions due to the cold start problem.

8. Future Work:

In future plans, we can attempt to combine it with other models, such as using a content-based model. Since it does not require interactions between users and items to make predictions, it can effectively solve the cold start problem. Furthermore, combining it with Singular Value Decomposition (SVD) can also bring significant improvements. SVD can decompose data into several matrices and reduce the dimensions of data, thus occupying less space. Not to mention its data extraction capability, as it can identify features of greater importance from the data. I believe that by combining multiple models, we can achieve a more accurate recommendation system.

User-Based Collaborative Filtering:

1. Objective:

This project report outlines the development and implementation of a user-based book recommendation system employing the K-Nearest Neighbors (KNN) algorithm. Utilizing datasets from the BX-Books project, including book details, user ratings, and user demographics, the system identifies similar users based on their rating patterns to recommend books. Through meticulous data preprocessing and analysis, the project demonstrates the application of collaborative filtering to enhance personalized reading experiences.

2. Methodology:

While most of the preprocessing and visualizations remain the same as the Item-Based Collaborative Filtering.

2.1 Preprocessing:

- **Books:** The code selects relevant columns (ISBN, title, author, year, publisher) and renames them for clarity.
- **Ratings:** The code renames columns (user ID, ISBN, rating) and keeps only users with at least 50 ratings to ensure a sufficient data base for recommendation.
- **Users:** The code renames columns (user ID, location, age) and explores the user data shape.

2.2 Filtering and Merging:

- The code filters ratings to include only users with at least 50 ratings.
- It then merges the filtered ratings with the preprocessed books data based on the ISBN to create a combined dataset containing book information and user ratings.
- It calculates the number of ratings for each book and merges this information with the previously combined dataset.
- Finally, it filters the data to include only books with at least 50 ratings and removes duplicates based on user ID and book title to ensure each user-book interaction is counted only once.

3. Creating the Rating Matrix

- The code creates a user-item rating matrix (ratings_pivot_table) as a pivot table from the filtered and merged data. This table has users as rows and books (titles) as columns, with each cell representing a user's rating for a particular book.
- The code fills missing values (users who haven't rated a book) with 0.
- The rating matrix is then converted to a sparse matrix (rating_sparse) using csr_matrix from scipy.sparse for efficient memory usage when dealing with large datasets with many missing values.

4. **KNN Model Fitting**

- The KNN model is created using NearestNeighbors from scikit-learn.
- The sparse rating matrix (rating_sparse) is used to fit the KNN model. This allows the model to learn the relationships between users based on their rating similarities.

4.1 Recommendations

- The code retrieves recommendations for a specific user (index 200 in the user-item rating matrix) using the KNN model.
- It calculates the distances and suggests the top 10 nearest neighbors (most similar users) based on their rating patterns.
- Finally, it retrieves the book titles associated with these nearest neighbors, providing recommendations for the target user.

4.2 Output Interpretation

The code outputs the top 10 book recommendations for the user at index 200 in the rating matrix. These recommendations are based on the ratings provided by the user's nearest neighbors, suggesting books that users with similar rating patterns have enjoyed.

Sample Recommendations:

- The Da Vinci Code
- The Sum of All Fears (Jack Ryan Novels)
- The Burden of Proof
- Slow Waltz in Cedar Bend
- Under the Tuscan Sun

By selecting a sample user profile from the matrix, the KNN model identified similar users and, based on their ratings, generated a list of top 10 book recommendations. This process illustrated the system's capability to personalize book suggestions effectively.

5. **Limitations:**

- Sparse Data: The challenge of data sparsity affects the system's ability to identify user similarities, particularly with users having fewer ratings.
- Cold Start Problem: New users or those with limited interaction history pose a challenge, as the system relies on existing ratings to generate recommendations.

6. **Future work:**
   - Incorporating item features to personalize recommendations beyond user similarities.
   - Exploring matrix factorization techniques for more complex user-item relationships.
   - Implementing hybrid approaches that combine user-based and item-based collaborative filtering for a more comprehensive recommendation strategy.

7. Conclusion

This project's successful implementation of a user-based book recommendation system highlights the potential of collaborative filtering and the KNN algorithm in delivering personalized content recommendations. Through detailed data preprocessing and strategic model application, the system provides a solid foundation for future enhancements and research in the domain of recommendation systems.