Book Recommender System Report

Tapas Mandal
Self Project

Dec 2024

1 Introduction

This project implements a personalized Book Recommender System using collaborative filtering. The goal is to suggest books to users based on their reading history and preferences. The project leverages the Books, Users, and Ratings datasets, applies preprocessing, and uses cosine similarity to recommend books.

2 Data Loading and Preprocessing

We begin by importing libraries and loading the datasets.

```
import numpy as np
import pandas as pd

books = pd.read_csv('/content/Books.csv')
users = pd.read_csv('/content/Users.csv')
ratings = pd.read_csv('/content/Ratings.csv')
```

The datasets include book metadata, user details, and ratings. We inspect the shape and check for missing values.

```
print(books.shape)
print(ratings.shape)
print(users.shape)
books.isnull().sum()
```

3 Merging and Aggregation

We merge the ratings with books on ISBN to attach book details to ratings.

```
ratings_with_name = ratings.merge(books, on='ISBN')
```

Next, we calculate the number of ratings and average rating per book.

```
num_rating_df = ratings_with_name.groupby('Book-Title').count()['
    ↪ Book-Rating'].reset_index()
num_rating_df.rename(columns={'Book-Rating':'num_ratings'}, inplace=
    ↪ True)

avg_rating_df = ratings_with_name.groupby('Book-Title')['Book-Rating
    ↪ '].mean().reset_index()
avg_rating_df.rename(columns={'Book-Rating':'avg_rating'}, inplace=
    ↪ True)
```

4 Popular Books

We filter books with at least 250 ratings and sort by average rating.

```
popular_df = num_rating_df.merge(avg_rating_df, on='Book-Title')
popular_df = popular_df[popular_df['num_ratings'] >= 250].
    ↪ sort_values('avg_rating', ascending=False).head(50)
```

5 Filtering Active Users and Books

We only keep users who rated more than 200 books and books that received at least 50 ratings.

```
x = ratings_with_name.groupby('User-ID').count()['Book-Rating'] >
    ↪ 200
known_users = x[x].index

filtered_rating = ratings_with_name[ratings_with_name['User-ID'].
    ↪ isin(known_users)]
y = filtered_rating.groupby('Book-Title').count()['Book-Rating'] >=
    ↪ 50
famous_books = y[y].index

final_ratings = filtered_rating[filtered_rating['Book-Title'].isin(
    ↪ famous_books)]
```

6 User-Item Matrix

We build a pivot table with books as rows and users as columns.

```
pt = final_ratings.pivot_table(index='Book-Title', columns='User-ID'
    ↪ , values='Book-Rating')
pt.fillna(0, inplace=True)
```

7 Similarity Computation

We apply cosine similarity to compute similarity between books.

```
from sklearn.metrics.pairwise import cosine_similarity
similarity_scores = cosine_similarity(pt)
```

8 Recommendation Function

We define a function to recommend books similar to a given book.

```
def recommend(book_name):
    index = np.where(pt.index == book_name)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index]))
       ↪ ,
                           key=lambda x: x[1], reverse=True)[1:5]

    data = []
    for i in similar_items:
        temp_df = books[books['Book-Title'] == pt.index[i[0]]]
        item = []
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book
           ↪ -Title'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book
           ↪ -Author'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['
           ↪ Image-URL-M'].values))
        data.append(item)
    return data
```

9 Example Output

```
recommend('1984')
```

This returns a list of similar books with their titles, authors, and cover images.

10 Model Persistence

We save important data and model objects for deployment.

```
import pickle
pickle.dump(popular_df, open('popular.pkl', 'wb'))
pickle.dump(pt, open('pt.pkl', 'wb'))
pickle.dump(books, open('books.pkl', 'wb'))
pickle.dump(similarity_scores, open('similarity_scores.pkl', 'wb'))
```

11 Conclusion

This project successfully builds a collaborative filtering-based Book Recommender System. It preprocesses user and book data, filters meaningful ratings, computes similarities using cosine similarity, and provides book recommendations. The final system is ready for deployment through a Streamlit application.

12 Results & Impact

- **Objective:** Build a personalized book recommendation system to suggest relevant books to users based on their past ratings, improving book discovery and engagement.
- **Approach:** Preprocessed book, user, and rating datasets, applied filtering for active users and popular books, built a user-item matrix, and used cosine similarity to generate recommendations. Deployed via a simple Streamlit interface.
- **Impact:** Helps users discover books more effectively, demonstrates practical application of collaborative filtering and data preprocessing, and showcases deployment of machine learning models into interactive applications.