MINI PROJECT 3 & 4

SC6613(51) Recommender System

Presented By

0615971 Min Hein Khant

2024-09-28

Outline

Introduction

- This project focuses on building a book recommendation system using Goodreads data, which includes book metadata and user interactions.

- In Mini-project 3, I develops a basic recommendation model by splitting the dataset and applying an algorithm to predict ratings.

- The model’s performance is evaluated using RMSE to assess prediction accuracy.

- Mini-project 4 enhances this approach by improving preprocessing and using hybrid techniques for more accurate predictions.

- Finally, the enhanced model’s performance is compared to the initial version.

Overview

- **Dataset:** Goodreads data consisting of book metadata and user interactions.
- **Mini-project 3:** Develop a basic recommendation model, split the data into training and test sets, and predict user ratings.
- **Mini-project 4:** Improve data preprocessing and implement a hybrid recommendation technique.
- **Evaluation:** Both models are evaluated using RMSE to measure prediction accuracy.
- **Comparison:** The enhanced model from Mini-project 4 is compared to the initial model from Mini-project 3 to assess improvements.

Problems

Main Problem(In project-3)

```
<ipython-input-2-48c3882b0a6b>:12: PerformanceWarning: The following operation may generate 13794952686 cells in the resulting pandas object.
  user_item_matrix = interactions.pivot(index='user_id', columns='book_id', values='rating').fillna(0)
```

•Encountered a **Performance-Warning** due to creating a large user-item matrix with 13.8 billion cells.

•The pivot operation fills missing values with zeros, leading to potential memory and performance issues.

•Optimization or using sparse matrix techniques could help address this.

Subproblems

- **Data Issues:** Missing or inconsistent information can reduce model accuracy.
- **Performance Problems:** Large datasets may slow down processing and require more memory.
- **Sparsity:** Few user ratings can make it hard to generate accurate recommendations.
- **Algorithm Limits:** The chosen method may not work well with the specific data.
- **Overfitting:** The model might work well on training data but not on new data.
- **Evaluation Difficulties:** Understanding and using RMSE correctly can be challenging.

Project Works

Project-3

What is SVD and How it work?

•**Singular Value Decomposition (SVD)** is a mathematical technique

•**Matrix Factorization**: SVD breaks down a complex matrix (like user ratings for items) into three simpler matrices: U, S, and V.

•**Identifying Patterns**: It uncovers hidden patterns in data by identifying latent features that represent user preferences and item characteristics.

•**Making Predictions**: By combining these features, SVD helps predict missing ratings, enabling better recommendations in systems like movie or book suggestions.

What is SVD and How it work?

1. User-Item Matrix Creation

user_id	book_id	rating
1	101	5
1	102	3
2	101	4
2	103	2
3	102	4
3	103	5

SVD

User →	Book 101	Book 102	Book 103
User 1	5	3	?
User 2	4	?	2
User 3	?	4	5

What is SVD and How it work?

2. Matrix Factorization

User →	Book 101	Book 102	Book 103
User 1	5	3	?
User 2	4	?	2
User 3	?	4	5

User Feature Matrix (U)   Item Feature Matrix (V)   Singular Values (S)

What is SVD and How it work?

3. Learn User and Item preferences

- Now that the matrix has been broken into smaller parts, the SVD algorithm learns how users and items are connected through these hidden features.
- For each user, it learns their preferences for different types of books (like action, romance, etc.).
- For each book, it learns which types of users might like that book based on its characteristics.

What is SVD and How it work?

4. Predicting Missing Ratings

- After learning the hidden patterns from the training data, the model can now predict the missing ratings.

$$\hat{r}_{ui} = U_u \cdot V_i^T = \sum_{f=1}^{F} (U_{u,f} \cdot V_{i,f})$$

Where:

- $\hat{r}_{ui}$: Predicted rating for user u on item i.
- $U_{u,f}$: Feature f of user u.
- $V_{i,f}$: Feature f of item i.
- F: Total number of features.

Project-3

```
from surprise import SVD, Dataset, Reader, accuracy
from surprise.model_selection import train_test_split
import pandas as pd


# Load the dataset
interactions = pd.read_csv('/content/drive/MyDrive/ColabNotebooks/Mini project 3 data/interactions_large.csv/interactions_large.csv')
books_metadata = pd.read_csv('/content/drive/MyDrive/ColabNotebooks/Mini project 3 data/books_metadata_large.csv/books_metadata_large.csv')

# Define the Reader and load the dataset into Surprise
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(interactions[['user_id', 'book_id', 'rating']], reader)

# Use train and test splits from Mini-project 3 (80% train, 20% test)
trainset, testset = train_test_split(data, test_size=0.2)

# Collaborative filtering using SVD (Mini-project 3)
algo_svd = SVD()
algo_svd.fit(trainset)

# SVD Predictions on test set
svd_predictions = algo_svd.test(testset)

# Calculate RMSE for SVD (Mini-project 3)
rmse_svd = accuracy.rmse(svd_predictions)
```

Project Works

Project-4

Improvements in project-4

•**Top-N Popular Books**:

•Added a mechanism to select the most popular books based on user ratings, using average ratings and the number of ratings.

•**Hybrid Recommendation System**:

•Combines predictions from the SVD algorithm with the popularity of books to improve recommendations.

•**Evaluate the Hybrid Model**:

•Converts hybrid predictions for RMSE evaluation.

Improvements in project-4

1. Top N Popular Books

```
# Step 1: Generate Top-N Popular Books based on ratings count and average rating

top_books = books_metadata[['book_id', 'average_rating', 'ratings_count']]

popular_books = top_books[top_books['ratings_count'] > 100]

popular_books = popular_books.sort_values(by=['average_rating', 'ratings_count'], ascending=False)

N = 30

top_n_books = popular_books.head(N)
```

Improvement: Selects widely liked books based on ratings.
Reason:
•**Filtering Popularity**: Only includes books with significant ratings, ensuring recommendations are based on community preferences.
•**Balanced Ranking**: Combines average rating and ratings count, enhancing recommendation relevance.

Improvements in project-4

2. Hybrid Recommendation System

```
# Hybrid predictions: Combine SVD predictions with popular book recommendations
hybrid_predictions = []
for uid, iid, true_r in testset:
    # Get SVD prediction
    svd_pred = algo_svd.predict(uid, iid).est

    # If the book is in the top N popular books, average its rating with the SVD prediction
    if iid in top_n_books['book_id'].values:
        popular_rating = top_n_books[top_n_books['book_id'] == iid]['average_rating'].values[0]
        hybrid_rating = (svd_pred + popular_rating) / 2  # Average of SVD and popular book rating
    else:
        hybrid_rating = svd_pred  # If not in popular books, use only SVD prediction

    hybrid_predictions.append((uid, iid, true_r, hybrid_rating))
```

Improvement: Combines SVD predictions with popular book ratings.

Reason:

•**Enhanced Accuracy**: Merges personalized and popular ratings, reducing prediction bias.

•**User Engagement**: Offers a mix of tailored and community-favorite recommendations.

Improvements in project-4

3. Hybrid Recommendation System

```
# Step 3: Evaluate the Hybrid Model Using RMSE
# Convert hybrid predictions to Surprise's Prediction object format for RMSE evaluation
hybrid_pred_objs = [Prediction(uid, iid, true_r, est, None) for (uid, iid, true_r, est) in hybrid_predictions]

# Calculate RMSE for the hybrid model (Mini-project 4)
rmse_hybrid = accuracy.rmse(hybrid_pred_objs)
```

Improvement: Converts hybrid predictions for RMSE evaluation.

Reason:

•**Performance Measurement**: RMSE quantifies prediction accuracy, enabling effective comparison with the SVD model.

•**Informed Improvements**: Helps determine the effectiveness of the hybrid approach for future model enhancements.

Conclusion

SVD RMSE(Project3):1.739

Hybrid Model RMSE(Project4):1.740

- **Popular Books Overlap**: Many test set books were not in the "Top-N popular books," so the hybrid model often relied on the same SVD predictions.
- **Averaging Effect**: Combining SVD predictions with popular book ratings did not always enhance accuracy, as popular ratings may not align with individual user preferences.
- **Similar Approaches**: The hybrid model predominantly used SVD predictions, resulting in only minor RMSE differences.

THANK YOU