Hyunjin Kim
5/25/2024

In today's ever evolving digital landscape, users consume a variety of content across multiple platforms. Providing personalized recommendations has become essential for enhancing user engagement. However, most recommendation systems are confined to a single platform which limits their ability to leverage user preferences across different platforms.

This project aims to develop a cross-platform recommendation system that can predict user preferences and provide personalized recommendations for books, movies, and music. By utilizing various modeling techniques, we seek to compare their performance and identify the best model for this task. The ultimate goal is to uncover actionable insights that can enhance user engagement and satisfaction across these different media platforms.

Our approach involves several key steps:

1. **Data Collection**
2. **Data Cleaning and Preprocessing**
3. **Exploratory Data Analysis (EDA)**
4. **Model Development**
5. **Model Evaluation**
6. **Recommendations for Improvement**

By combining collaborative filtering techniques and leveraging the strengths of each model, we aim to create a robust recommendation system that provides high-quality recommendations across multiple content platforms.

Data Sources:

Books Dataset:
Source:
The book dataset was sourced from Kaggle (https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset?select=Books.csv).

Content:
This dataset contains information about user interactions with books, including user ratings for each book.

Size:
200,000+ rows
14 columns

Key Features:
Book Title: Title of the book

Book Rating: Rating given by the user (scale of 1-5)

Movies Dataset:
Source: The movie dataset was sourced from movielens (https://grouplens.org/datasets/movielens/).

Content:
This dataset contains anonymous ratings of movies made by MovieLens users.

Size:
1,000,000+ Rows
10 columns

Key Features:
Movie Title: Title of the movie
Movie Rating: Rating given by the user (scale of 1-5)

Music Dataset:
Source: The music dataset was sourced from zenodo (https://zenodo.org/records/6090214)

Content: This dataset contains user interactions with music artists, including the number of times a user has listened to a particular artist.

Size:
17,000,000 rows
4 columns

Key Features:
Artist Name: Name of the artist the user listens to
Plays: Number of times the user has listened to that particular artist

Each dataset was selected to provide a comprehensive view of user preferences across different types of media. The diversity of these datasets will enable us to build a recommendation system that works for varied user interests and behaviors.

Data Cleaning:

In this project we worked with three different datasets. Each dataset required specific cleaning steps to make sure the data was consistent, complete, and prepared for analysis. Below are some common steps as well as specific steps taken, along with the specific steps taken for each dataset.

Common steps:

- Loading Data: Loaded each dataset into a pandas Dataframe.
- Missing values: Identified missing values by either dropping the row containing the missing value or filled with “Unknown” if specific values were not needed for analysis.
- Removed Columns: Dropped any columns that were not relevant to the analysis.
- Data Types: Made sure all columns had appropriate data types for analysis.
- Standardized Values: Standardized text values, like country names, to ensure consistency.
- Standardized the ratings and number of plays. Specifically, all ratings and play counts were binarized to indicate whether a user interacted with an item, 1 for interaction, 0 for no interaction.

Specific Steps:

Books Dataset:

- Removed columns related to image URL’s, , as they were not needed for the recommendation analysis.
- Dropped rows with missing age values.
- Merge the three books dataframes first on ISBN then User-ID columns.

Movies Dataset:

- Used latin1 encoding to read the .dat files.
- Merged Dataframes: Merged the dataframes on user_id and movie_id columns.

Music Dataset:

- Converted Type: Converted the plays column to integer
- Missing values: Dropped rows with missing age and gender values

Creating Synthetic Users:

To simulate a real-world scenario and handle the different sizes of the datasets, we created synthetic users by merging the datasets. This approach allowed us to generate a combined dataset that captures interactions across all three platforms. Synthetic users help in balancing the datasets and providing a comprehensive view of user preferences.

Creating User-Item interaction matrices:
We created a user-item interaction matrix for books, music, and movies. This matrix captures the interactions between users and items, normalized to a range of 0-1 for analysis. The sparsity of the final matrix was calculated to understand the proportion of zero values, resulting in a matrix with 99.97% sparsity.

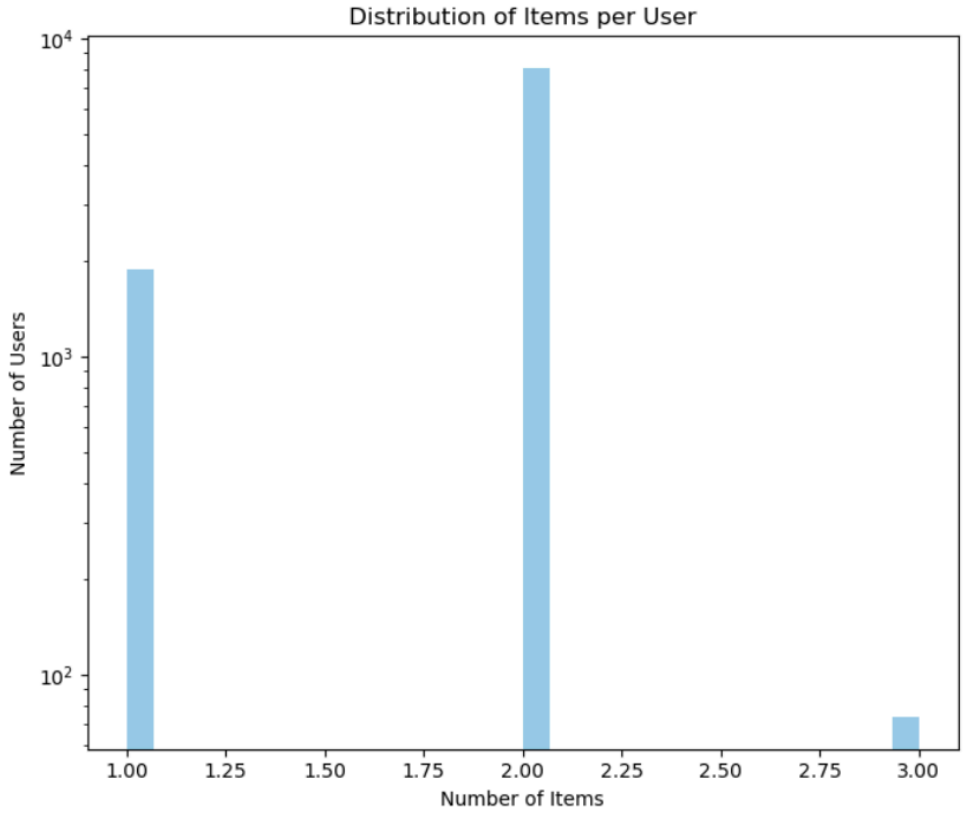
Note on Data Processing:
Due to hardware constraints and the different sizes of the datasets, we limited the number of rows processed to 10,000 for each dataset. This approach helped simulate a realistic scenario while managing computational resources.

Exploratory Data Analysis (EDA):

We conducted EDA on the combined user-item interaction dataset in order to gain insights and prepare the data for building the recommendation system.

Key insights and Visualizations

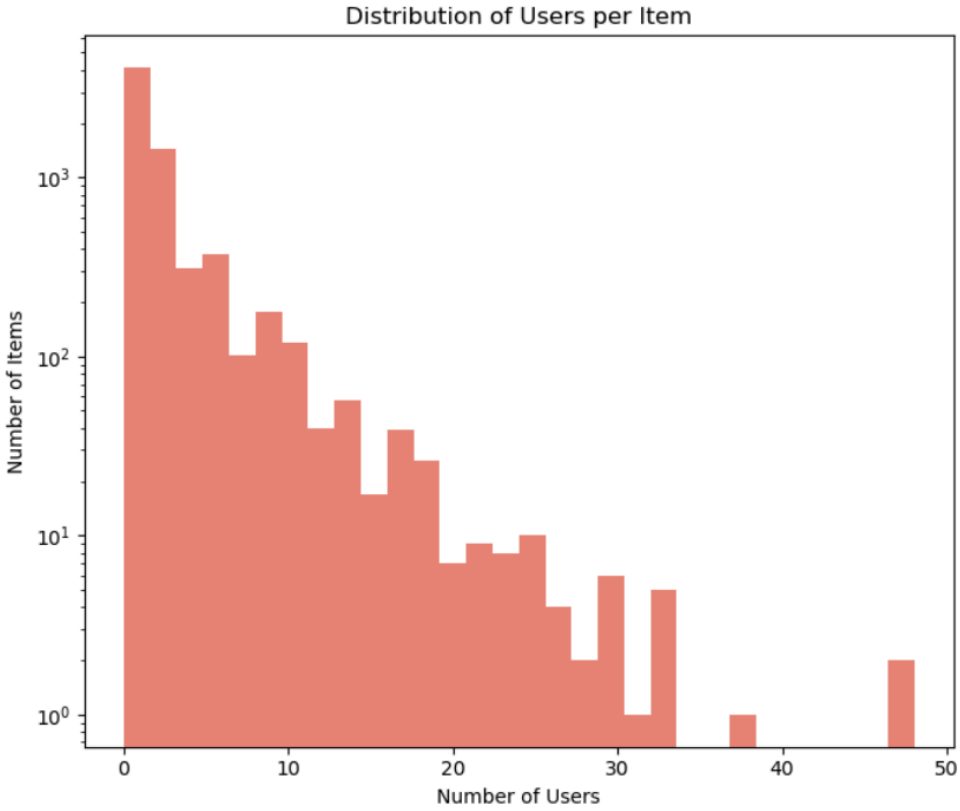
Distribution of items per user:



The histogram shows the distribution of the number of items per user. The distribution shows that most users have interacted with a limited number of items. A significant number of users

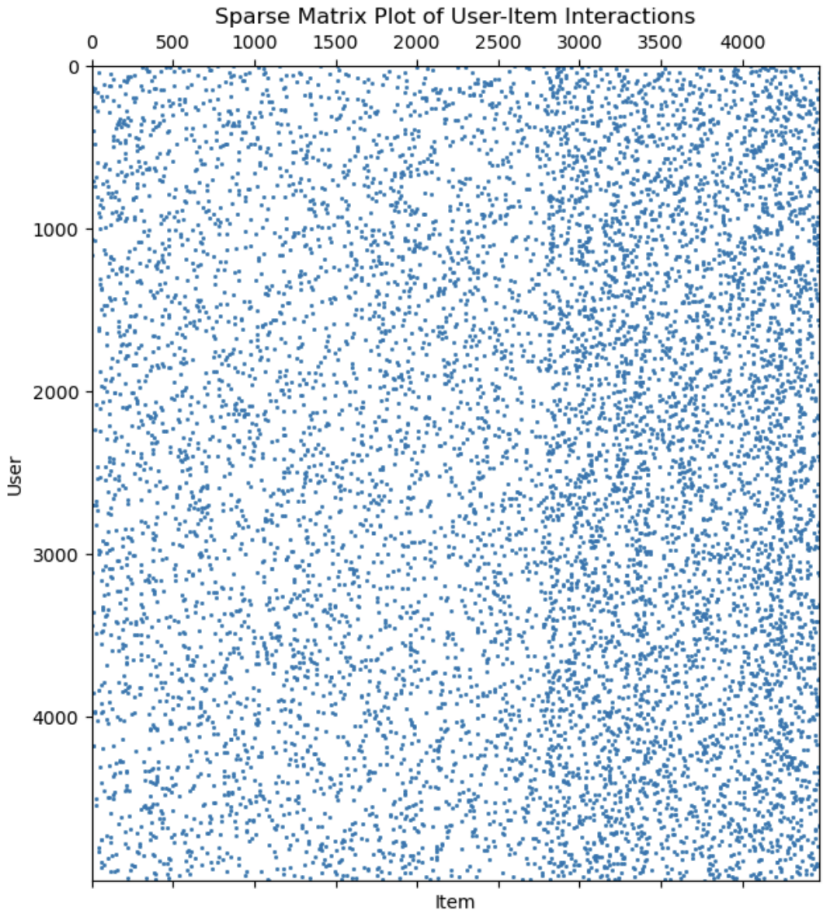
have interacted with one or two items, which suggests a high level of sparsity in the user-item matrix.

Distribution of users per item:



The histogram shows the distribution of the number of users per item. The distribution shows that most items are interacted with by very few users. A few items have high interaction counts indicating these items are popular.

Matrix Plot:



This plot shows the user-item interactions. The plot clearly shows the high sparsity of the dataset, with very few non-zero interactions spread across the matrix.

Summary Statistics:
Users: 5,000
Items: 6,855
Total Interactions: 9,123
Average Interactions per User: 1.82
Average Interactions per Item: 1.33
Sparsity: 99.97%

Rationale for Filtering the Matrix:
Given the high level of sparsity in the matrix with more than 99% of the entries being zero, it was essential to attempt to filter the data to enhance the quality of the recommendation system.

The distribution showed that a significant number of items had no interactions, while some items had very high interactions. This skewness can introduce bias in the recommendation model.

Filtering strategy:
Items that had no interactions were removed to reduce the sparsity and focus on users that had at least some engagement. Items with a high number of interactions were filtered out to avoid bias towards overly popular items that could dominate the recommendations.

Post-Filtering Statistics:
Users: 5,000
Items: 4,478
Total Interactions: 9,123
Average Interactions per User: 1.82
Average Interactions per Item: 2.04
Sparsity: 99.96%

Benefits of Filtering:
By focusing on users and items with meaningful interactions the model can be trained more effectively. Filtering out extremely popular items and inactive users helps to mitigate bias and ensure that the recommendations are more personalized. Lastly, reducing the size of the matrix helps in managing computational resources allowing us to evaluate the models. The filtering process was a crucial step in preparing the data for our model. We attempted to address the high sparsity and skew of the distributions.

Model Development:

In this project, we explored three different models to develop a cross-platform recommendation system: Singular Value Decomposition (SVD), Alternating Least Squares (ALS), and a Deep Learning model. Each model was implemented, trained, and evaluated to determine its effectiveness in predicting user preferences.

Before developing our models, we split the user-item interaction matrix into training and testing subsets. We used an 80, 20 split for training and test sets.

Model Selection:

1.) **Singular Value Decomposition (SVD):**
Rationale: SVD is used to decompose user item interaction matrices into latent factors. It is useful for handling sparse matrices and capturing latent features that describe user

preferences and item characteristics.

Evaluation: The model's performance was evaluated using the Root Mean Squared Error (RMSE), Precision, Recall, and F1-score. The results indicated that the model struggled to accurately predict user preferences with an RMSE of about 0.9996. The model also scored 0 for all other metrics, indicating poor performance.

Hyperparameter tuning: I experimented with different values for the latent factors (k), in an attempt to balance model complexity and performance. However, tuning was quickly abandoned as the initial results were not promising.

2.) **Alternating Least Squares (ALS):**
Rationale: ALS is another matrix factorization technique that can handle collaborative filtering tasks. It is very effective in handling implicit feedback data by assigning confidence levels to interactions.

Evaluation: The ALS model showed better performance with an RMSE of 0.00199. This result appeared to be promising, however, when looking into other metrics we found that precision was 1.0 while the recall and F1-score were both 0. These scores indicated that the model identified few relevant items but missed many actual relevant items, this is most likely due to the model underfitting due to the high sparsity of the matrix.

Hyperparameter tuning: We used a grid search to find the optimal combination of hyperparameters for our model. Even after tuning the hyperparameters the issue of underfitting seemed to still be present suggesting that I need to take a more complex approach or step back into data preprocessing.

3.) **Deep Learning:**
Rationale: Deep learning models can capture nonlinear relationships between users and items, making it suitable for recommendation tasks. By using a neural network we can learn more complex representations of users and items.

Evaluation: The deep learning model achieved the lowest RMSE of the models so far with a score of 0.0183. However, despite this, precision, recall, and F1-score were all 1.0, suggesting severe overfitting. This limits the model's practicality for generalization to new data.

Hyperparameter tuning: We manually tuned several hyperparameters, including the number of layers, activation function, dropout rate, and the learning rate. Despite achieving a low RMSE the model's overfitting indicated that further tuning was needed. In order to further enhance the model we could look to perform a grid search to tune the hyper parameters to attempt to fix the overfitting issue.

Model Comparison and Analysis:

Performance Comparison:

Model	RMSE	F1-Score	Precision	Recall
SVD	0.9996	0.0	0.0	0.0
ALS	0.00199	0.0	1.0	0.0
Deep Learning	0.0183	1.0	1.0	1.0

Both SVD and Truncated SVD failed to perform well due to their assumption of linear latent factors. The result was a high RMSE and scores of zero for all other metrics. The ALS model showed promising results when it came to RMSE, but the issue of underfitting in the top-k recommendations highlights the need for further tuning and addressing the sparsity of the data. The Deep Learning model had the best RMSE but its perfect scores on all other metrics indicates severe overfitting which limits its practicality.

Recommendations for improvement:

Hyperparameter Tuning:

ALS:
For our ALS model we could conduct a more extensive grid search in an effort to attempt to explore a wider range of hyperparameters for implicit feedback. Another suggestion would be to implement a k-fold cross-validation to better understand its performance across different subsets of the data.

Deep Learning Model
We can attempt to implement regularization techniques in order to deal with the overfitting issue of the deep learning model. We can also experiment with different dropout rates in order to find a balance between underfitting and overfitting.

Dealing with Sparsity
The sparsity of our matrix plays a role in the results our models are exhibiting. In order to help deal with the sparsity we could introduce more synthetic interactions based on similar users or items to reduce sparsity. We might also try to reduce the dimensionality of the matrix so it is more manageable for the models.

Hybrid Approaches

A new approach we could attempt would be to create a pipeline that uses multiple models into an ensemble in order to leverage the strengths of each approach and potentially improve the accuracy of recommendations overall.

Feature Engineering
We could return to our data cleaning and EDA in order to incorporate additional user and item features, we would need to be careful to not introduce bias in this case into our model.

By implementing these recommendations we can address the current limitations and enhance the performance of the cross platform recommendation system.

Final Model Selection:
Based on our evaluation, the Deep Learning model demonstrated the most promise with the lowest RMSE of 0.0183. Despite its overfitting issues, it showed potential for capturing complex user-item interactions more effectively than the other models. Therefore, we selected the Deep Learning model as our final model. Future work will focus on mitigating overfitting through regularization techniques, dropout, and further hyperparameter tuning.

Conclusion:

We aimed to develop a cross-platform recommendation system. Our goal was to predict user preferences and provide personalized recommendations across different media types. We used various modeling techniques in order to compare their performance and identify the best approach for our recommendation system.

We gathered datasets from reliable sources and performed data cleaning to ensure consistency, and completeness for analysis. Through our EDA we gained insight into the user item interactions, highlighting the high level of sparsity in the data. We created visualizations of the interactions in our matrix which helped us to determine how to filter the data to enhance model performance. Our next step was to implement and evaluate three models. SVD was the first model we implemented. The SVD model struggled with the high sparsity of our user-item matrix. The ALS model showed better results but experienced underfitting. The last model was our Deep Learning model which managed to achieve the lowest RMSE but showed signs of severe overfitting. We proposed several strategies for improvement including further hyperparameter tuning, handling sparsity through data augmentation and dimensionality reduction. Lastly we suggest exploring hybrid recommendation approaches.

The high level of sparsity in the matrix poses a significant challenge. Future work should focus on advanced techniques to handle sparsity. Model optimization should also be a major focus for any future work, further hyperparameter tuning and cross validation techniques will improve the models performance and generalization capabilities. Regularization techniques and dropout should be applied to try to stop the overfitting in the deep learning model. Lastly we can

incorporate additional features from our datasets to enhance the recommendation accuracy and personalization.

This project has demonstrated the complexity of building a cross-platform recommendation system. Despite the difficulties encountered with the data sparsity and model optimization, the insights we have gained and the strategies proposed give us a strong foundation for the future improvements. By iterating on these models with the suggested enhancements we can develop a more accurate, scalable and reliable recommendation system that enhances user engagement and satisfaction across different media.