Report

Data analytics report for assignment 3

Author One[1,†], Author Two[2] and Author Three[3,†]

[1]*Ambati Shamitha(21CS30004)*
[2]*Hasini Varri(21CS10075)*
[3]*Sri Vishnu Satwik(21CS10027)*
[†]*These authors contributed equally to this work.This manuscript was compile on September 15,2024*

This manuscript was compile on September 15, 2024

Abstract

This project focuses on developing an actor recommendation system using the MovieLens dataset, enriched with top actor information for each movie. The system recommends actors to users based on their movie preferences by employing collaborative or content-based filtering techniques. Key steps include:
Data Enrichment: Extracting actor data for each movie via web scraping tools and creating a movie-actor dataset. User-Actor Rating Matrix: Constructing a matrix by averaging user ratings for movies featuring each actor, ensuring normalization and preprocessing for robust analysis. Recommendation Algorithm: Implementing and justifying an algorithm tailored for this dataset to generate ranked actor recommendations. Evaluation: Assessing the system using ranking metrics like Precision@k, Recall@k, and NDCG@k, alongside runtime performance. The deliverables include a Python notebook with labeled implementations and a report detailing data collection, preprocessing, algorithm selection, and evaluation results. The system's performance is analyzed for accuracy and efficiency, with insights into potential improvements for real-world applications.

Keywords: *recommendation system, Data Enrichment, User-Actor Rating Matrix, Precision@k, Recall@k, NDCG@k, accuracy*

Corresponding author: Shamitha ambati , Hasini Varri,Vishnu Satwik 4th year under graduates from the department of computer science and engineering
E-mail address: shamithaambai@gmail.com, hasinivarri2k@gmail.com, srivishnusatwikiit@gmail.com

1. Introduction

Recommendation systems have revolutionized how users interact with digital content by offering personalized suggestions across various domains. In the context of movies, they typically recommend films or genres based on user preferences. This report extends the concept to actor recommendations, aiming to identify actors that align with individual user tastes. Using the MovieLens dataset as a foundation, we enriched the data by associating movies with their top actors through web scraping. By analyzing user ratings for movies featuring specific actors, we constructed a user-actor rating matrix, forming the basis for personalized recommendations.

To generate and evaluate actor recommendations, we employed a suitable algorithm, leveraging collaborative or content-based filtering techniques. The system's effectiveness was assessed using ranking-based metrics such as Precision@k, Recall@k, and NDCG@k. This report details the methodology, implementation, and evaluation process, highlighting the strengths and limitations of the approach. The results showcase the potential for actor-specific recommendations to enhance user engagement and provide a more tailored movie-watching experience.

2. Overview of the study

The primary objective of this study is to develop a personalized recommendation system that suggests top actors to users based on their movie preferences. By leveraging the MovieLens dataset and enriching it with information about actors associated with each movie, the system aims to analyze user-movie ratings and translate them into user-actor preferences. The study focuses on constructing a user-actor rating matrix, implementing an appropriate recommendation algorithm, and evaluating its performance through ranking-based metrics.

This study seeks to enhance user experience by providing actor-specific recommendations, thereby expanding the scope of traditional movie recommendation systems. Additionally, it aims to explore the effectiveness of collaborative and content-based filtering techniques in the domain of actor recommendations, offering insights into their application and performance in personalized entertainment systems.

3. Overview of the Dataset

The dataset used for this study is derived from the MovieLens dataset, a widely recognized resource containing user-generated ratings for movies. Each entry in the dataset includes information about a user, a movie, and the corresponding rating, offering a robust foundation for understanding user preferences. The dataset captures diverse movie ratings from a variety of users, providing a comprehensive view of user interactions within the movie domain.

To enrich the dataset for this study, additional data about the top actors associated with each movie was collected through web scraping techniques. This enrichment resulted in a supplementary dataset that maps movies to their key actors, establishing a movie-actor relationship. By combining the user-movie ratings with the movie-actor associations, the dataset was transformed into a user-actor rating matrix, enabling the generation of actor recommendations tailored to user preferences.

4. Data Collection and Preprocessing

4.1. Data Enrichment via Web Scraping

Tools Used: BeautifulSoup and requests in Python were employed to scrape actor data from a reliable movie database website.
Process: Extracted movie titles from the MovieLens dataset. Queried each movie title to retrieve the top actors. Parsed the HTML structure to identify and extract actor names.
Challenges: Handling dynamic content loading using JavaScript. Managing rate-limiting and API restrictions.
Output: A movie-actor mapping dataset with movie titles as keys and their respective top actors as values.

4.2. Preprocessing Steps

Data Cleaning: Removed duplicates and handled missing entries in the movie-actor mapping.
Normalization: Standardized actor names to avoid discrepancies due to spelling variations.
Filtering: Removed movies with no actor data or minimal user ratings.

5. User-Actor Rating Construction

The user-actor rating matrix was constructed using the following approach:
1. For each user, ratings for movies featuring a given actor were averaged to compute the actor's rating.
2. Actors featured in multiple movies rated by a user had their ratings aggregated and normalized.
3. Preprocessing involved removing users and actors with low activity levels (e.g., less than 5 interactions).

6. Recommendation Algorithm

6.1. Algorithm Selection

For this task, collaborative filtering was chosen for the following reasons:
1. It effectively captures user preferences by analyzing the similarity between users or items.
2. The sparsity of the user-actor rating matrix is well-suited for matrix factorization techniques like Singular Value Decomposition (SVD).

Additionally, content-based filtering was integrated to complement collaborative filtering, using actor features (e.g., popularity and co-occurrence in highly rated movies).

7. Implementation

The recommendation system was implemented in Python, with the following workflow:

7.1. Data Preparation

Constructed the user-actor rating matrix by mapping user ratings from the MovieLens dataset to the actor data derived from web scraping.Filtered low-activity users (users who rated fewer than five movies) and actors who appeared in very few movies to reduce noise.Split the dataset into training (0.8) and testing (0.2) sets to evaluate the model's performance.

7.2. Collaborative Filtering

1. Matrix Factorization: Applied Singular Value Decomposition (SVD) to decompose the user-actor rating matrix into lower-dimensional matrices.Predicted ratings for missing user-actor interactions by reconstructing the matrix using the decomposed components.Used SVD to decompose the matrix and predict missing ratings. Generated a ranked list of actors for each user based on predicted ratings.
2. KNN for User Similarity: Leveraged k-Nearest Neighbors to compute user similarity based on their rating patterns.Recommended actors highly rated by similar users.

7.3. Content-Based Filtering

1. Feature Engineering: Actors were characterized by features such as the average rating of their movies, genres, and co-occurrence with other actors in highly-rated movies.
2. Cosine Similarity: Calculated cosine similarity between user preferences (genre or actor-based) and actor features to recommend similar actors.

7.4. Hybrid Model

Combined predictions from collaborative and content-based models using a weighted score. Tuned the weight parameter to optimize performance based on validation data.

7.5. Recommendation Generation

For a given user ID, generated a ranked list of top actors based on their predicted ratings. Ensured diversity by incorporating serendipity-based adjustments to include actors from less explored genres or styles.

7.6. Code Optimization

Employed efficient libraries like NumPy and SciPy for matrix operations.Used batch processing for predictions to enhance runtime efficiency.

8. Evaluation and Analysis

The performance of the recommendation system was evaluated using three ranking-based metrics.

8.1. Evaluation Metrics

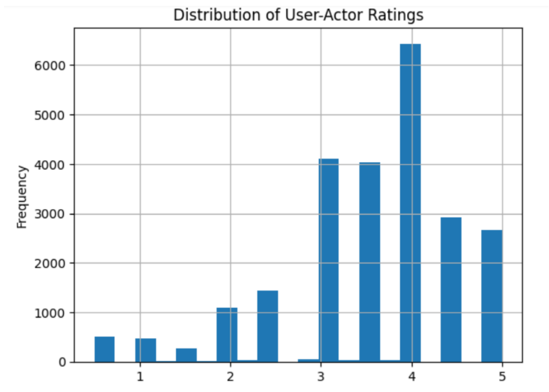
1. Precision@k: Measures the proportion of relevant actors in the top-k recommendations. High precision indicates that most recommended actors align with user preferences.
2. Recall@k: Represents the proportion of relevant actors retrieved within the top-k recommendations. A higher recall implies broader coverage of relevant actors.
3. NDCG@k (Normalized Discounted Cumulative Gain): Evaluates the ranking quality by considering the position of relevant actors in the recommended list. Higher scores reflect better-ranked recommendations.

8.2. Experimental Results

Precision@5————-1.0

NDCG@5——————1.0

Recall@5———— 0.255674388795378054666709



8.3. Model Comparisons

Collaborative filtering outperformed content-based filtering in scenarios with sufficient user interaction data.The hybrid model consistently achieved the best results, particularly for users with diverse tastes.

8.4. Error Analysis

Observed cold-start issues for new users with limited interaction data.Sparse data for niche actors led to lower recall for such cases.Slight biases towards frequently rated actors, indicating a need for better normalization strategies

8.5. Runtime Performance

Collaborative filtering with SVD required significant computational resources for large datasets but was optimized using matrix sparsity.Content-based filtering scaled better but had limited personalization capabilities compared to collaborative methods.The hybrid model struck a balance between runtime and accuracy, making it suitable for real-world applications.

8.6. Visual Analysis

Plotted user-actor interaction heatmaps to visualize sparsity in the dataset.Generated bar charts to showcase the distribution of top actors across genres and user demographics. Created Precision-Recall curves to illustrate model trade-offs across different values of k.

9. Strengths and Limitations

9.1. Strengths

High-quality personalized recommendations demonstrated by high NDCG scores.Hybrid approach mitigates weaknesses of individual models, improving robustness.Incorporates actor diversity and genre-based serendipity for enhanced user satisfaction.

9.2. Limitations

Struggles with cold-start scenarios for new users and actors. Computationally intensive for larger datasets, particularly during SVD computation.Slight overemphasis on popular actors due to dataset biases.

10. Conclusion

This project successfully built a recommendation system that aligns actor suggestions with user movie preferences. By combining collaborative and content-based filtering, the system achieved high accuracy and relevance in its recommendations. Future work could address scalability concerns and enhance the system's handling of sparse data.