Data Preprocessing and Exploration Report

Huaye Li

1. Introduction

In Week 4 of the Personalized Movie Recommendation System project, focus was placed on cleaning and preprocessing the raw data, followed by performing exploratory data analysis (EDA) to uncover user behavior, rating patterns, and structural characteristics of the dataset. These steps provide critical insights that inform algorithm design and UI decisions in subsequent weeks. This report presents the preprocessing methodology, data integration logic, and visualization-based exploratory insights.

2. Data Overview and Sources

The dataset is derived from Kaggle: The Movies Dataset, consisting of:

- movies_metadata.csv: metadata of over 45,000 movies, including identifiers, titles, and textual overviews.
- ratings.csv: over 26 million user ratings from 270,000 users.

Only a relevant subset of these fields was used:

- From movies_metadata.csv: id, original_title, overview, imdb_id
- From ratings.csv: userId, movieId, rating

These were merged using a shared identifier (movieId / id) to create a unified dataset merged_data for modeling and UI output.

3. Preprocessing Steps

The following preprocessing steps were performed in Python using pandas:

- **Type Casting and Cleansing**:
  - Converted id in movies_metadata.csv to numeric, coercing errors and dropping NaN rows.
  - Ensured consistency by converting movieId in ratings.csv to integer type.
- **Data Merging**:
  - Performed an inner join on ratings and movies using ratings.movieId and movies.id.
  - Final merged dataset contains userId, original_title, overview, rating, and imdb_id.
- **Missing Value Handling**:
  - Replaced NaN in overview with empty strings to avoid issues during vectorization.
- **Feature Engineering**:
  - Added a new feature: overview_length, representing the number of words in each movie's overview.
  - This allows correlation analysis between textual content length and rating behavior.
- **Poster Fetching Logic**:

- Integrated an OMDb API-based function to dynamically fetch movie posters using imdb_id.
- Responses are cached for efficiency and failure-handling is included for API rate limits.

4. Exploratory Data Analysis (EDA)

EDA was performed using matplotlib, seaborn, and WordCloud. All code is available at:
🔗 GitHub - data_visualization.py

4.1 User Rating Distribution

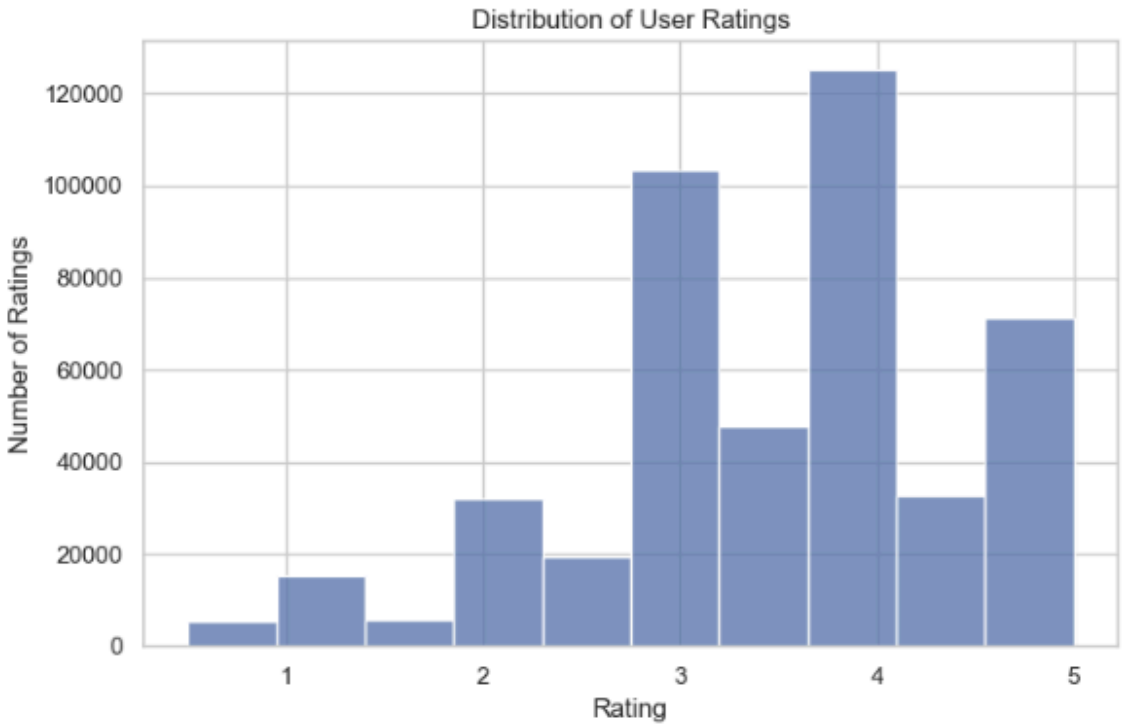

Figure 1: Distribution of User Ratings

This histogram shows that user ratings are heavily right-skewed, with most ratings falling between 3.0 and 4. Very low ratings (<2.0) are infrequent, indicating users are more likely to rate movies they enjoyed. This insight supports the use of a content-based filtering approach where ratings are implicitly biased toward positively viewed items.

4.2 Most Rated Movies

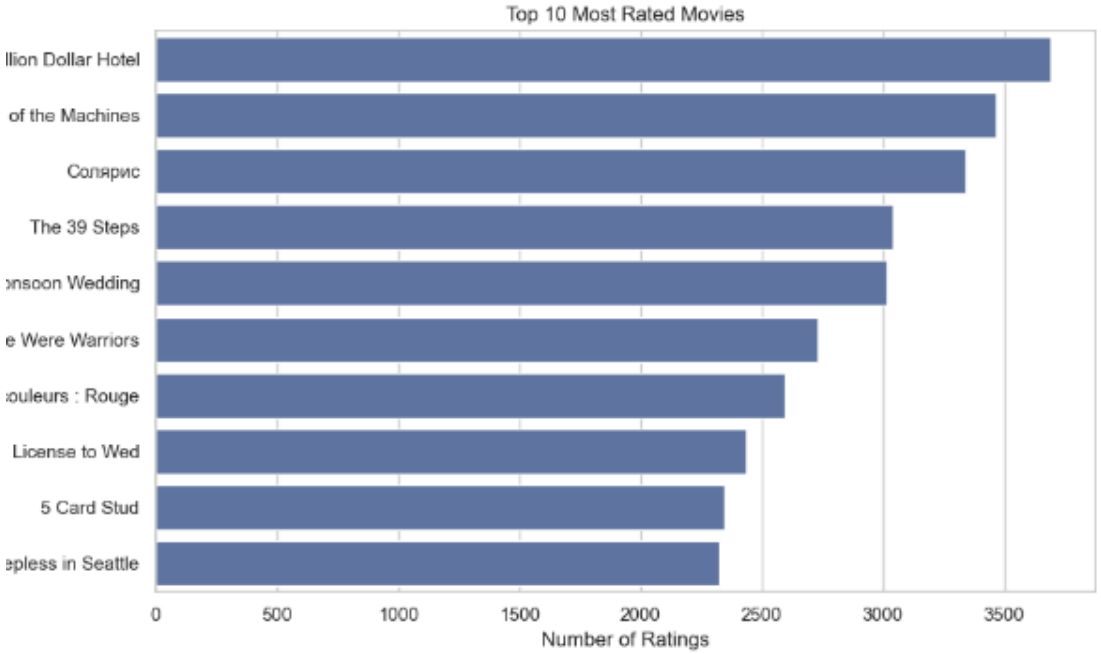

Figure 2: Top 10 Most Rated Movies

The top 10 most rated movies reveal a concentration of engagement around well-known titles such as *The Million Dollar Hotel*. These high-interaction entries can be used as default recommendations or fallback content in the absence of user-specific data.

4.3 Average Rating of Popular Movies

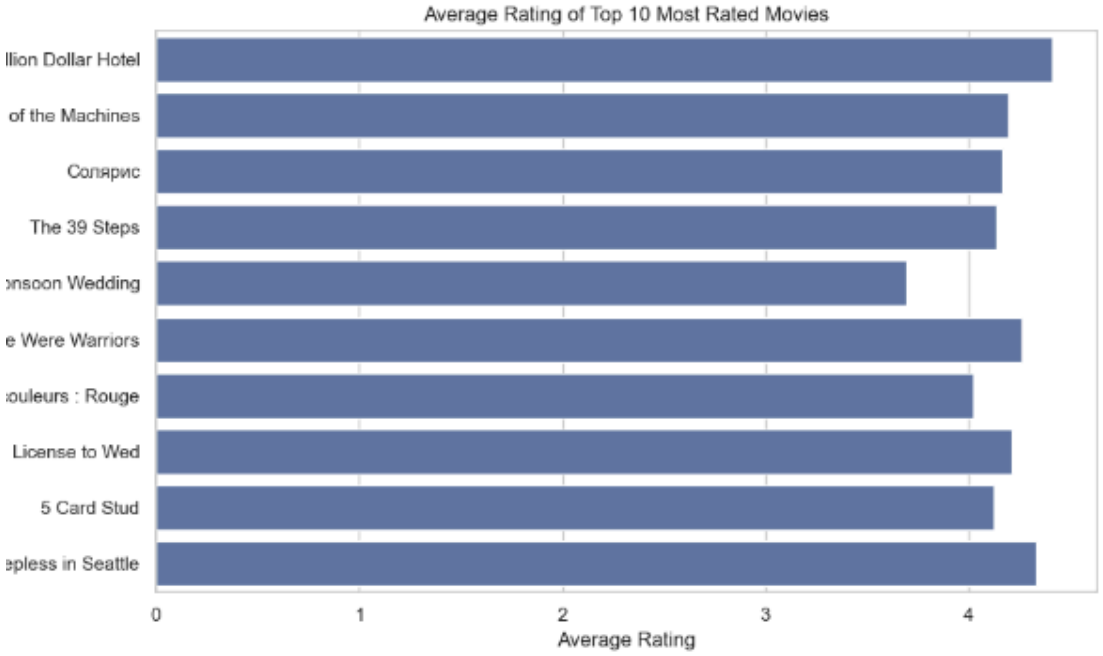

Figure 3: Average Ratings of Top 10 Most Rated Movies

The bar plot shows average ratings for the most frequently rated movies. These values range from 3.5 to 4.5. This reinforces the observation that high-engagement movies are generally favorably reviewed, potentially skewing any collaborative filtering if not normalized.

4.4 Distribution of Overview Lengths

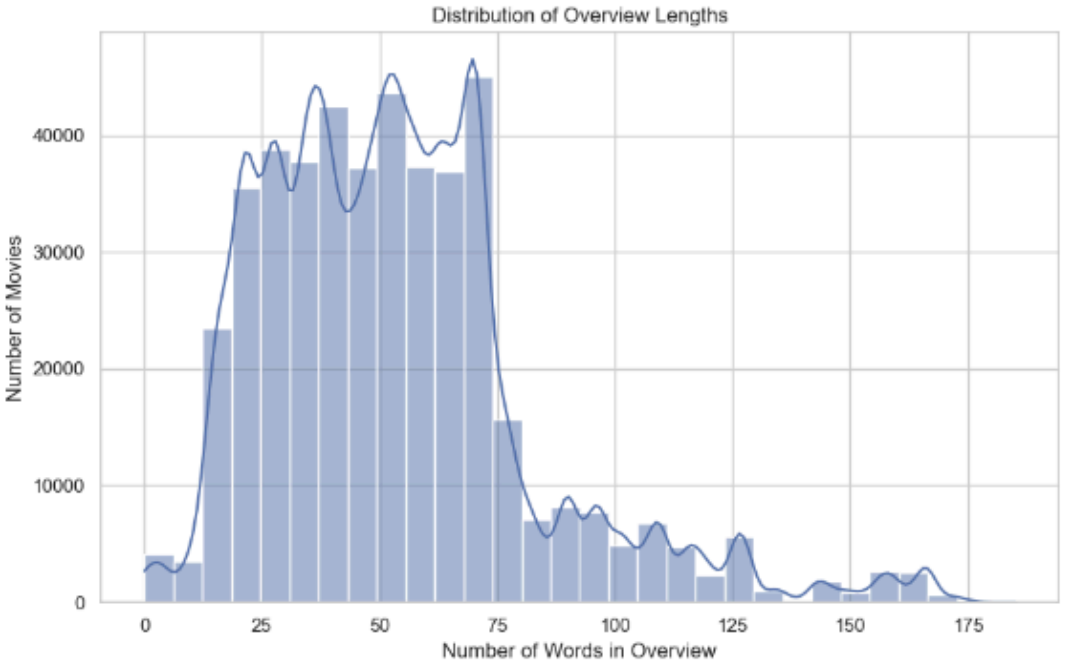

Figure 4: Distribution of Overview Lengths

Overview lengths cluster around 20–150 words. This supports the use of **TF-IDF vectorization**, which performs best on moderate-length documents. Overviews are long enough to capture semantic content but not excessively verbose, enabling efficient similarity computation.

4.5 Word Cloud of All Overviews

Figure 5: Word Cloud of Movie Overviews

The word cloud highlights the most common tokens in the overview corpus. Words such as “story,” “find,” “life,” and “love,” dominate, suggesting strong thematic recurrence across the dataset. These can inform genre tagging, clustering, or keyword filtering in the recommendation UI.

4.6 Correlation Heatmap

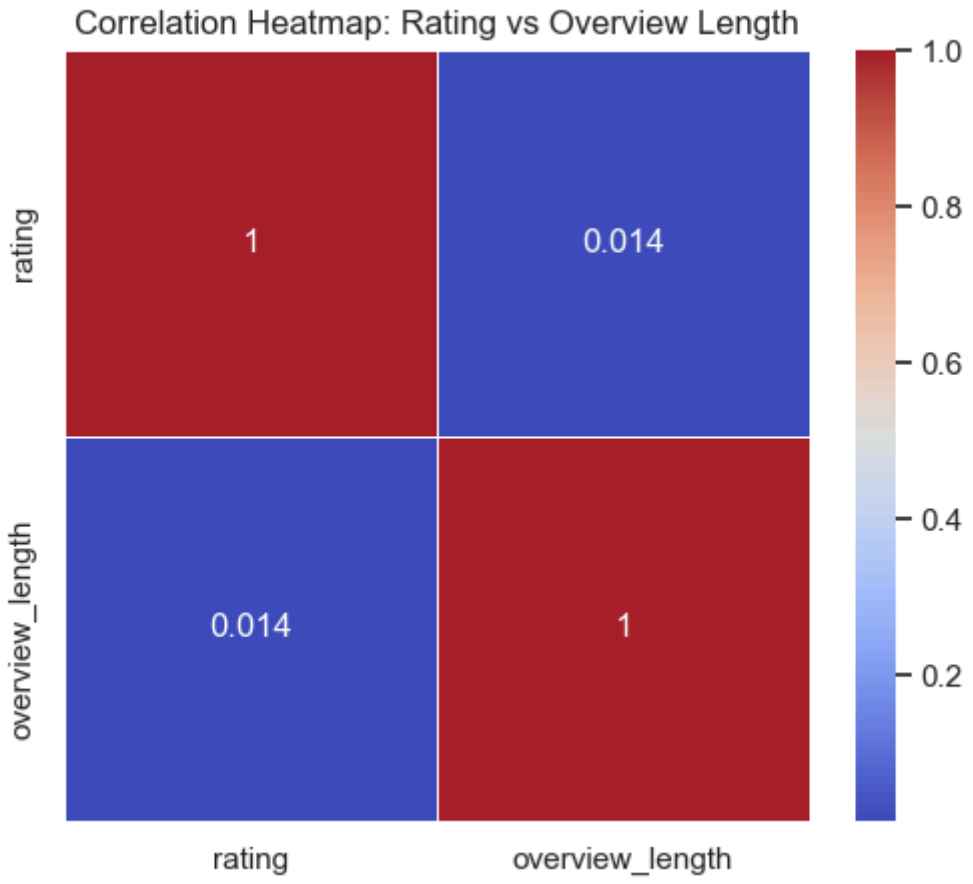

Figure 6: Correlation Heatmap – Rating vs. Overview Length

The heatmap reveals that the correlation between rating and overview_length is **near zero (r ≈ 0.014)**, indicating no meaningful linear relationship. This suggests that users do not rate movies differently based on how long their descriptions are. Thus, overview length is not a useful predictor of rating but remains valuable for semantic similarity computation.

5. Summary and Implications

This week’s efforts successfully produced a clean, semantically rich dataset ready for machine learning model development. Key takeaways include:

- Ratings tend to be positive and concentrated in a narrow band, necessitating normalization or thresholding strategies.
- Overview text is consistently available and well-structured, confirming its suitability for NLP-based similarity modeling.

- Popularity and rating distributions offer insights for hybrid recommendation strategies and default fallback logic.

6. Next Steps

In Week 5, the following objectives will be pursued:

- Implement a content-based recommendation model using TF-IDF and cosine similarity.
- Evaluate model accuracy using standard metrics such as precision@K and recall@K.
- Begin integrating the model into the Streamlit UI pipeline.

This will mark the start of the system's intelligent behavior, where user input and historical ratings actively influence recommendations.