Recommendation System Using Rotten Tomatoes and IMDB Data

AI: Chatbots & Recommendation Engines Course

Naif - Alzmami - Nicholas - Adrian

Overview

This project focuses on building a content-based movie recommendation system using datasets from Rotten Tomatoes and IMDB. The goal is to create a system that recommends movies to users based on the similarity of movie descriptions and metadata.

Datasets Overview

Reviews Data:

– Contains critic reviews from Rotten Tomatoes.
– Filtering out entries with missing critic names.
– Merging reviews with movie metadata to ensure consistency.

Movies Data

– Two sources: Rotten Tomatoes (basic info) and IMDB (detailed metadata).
– Merging on movie title to create a master dataset.

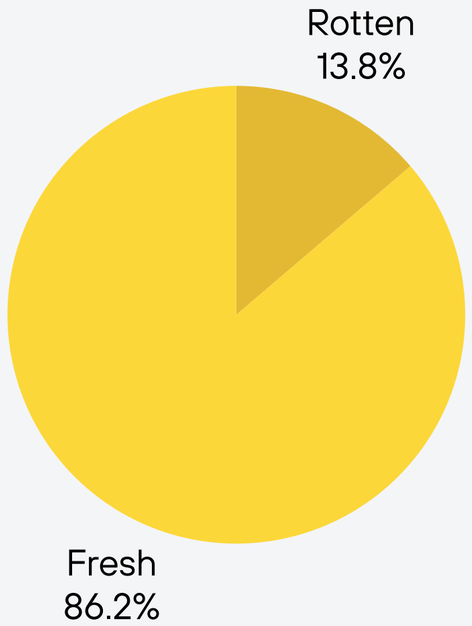
Statistics

755
Movies

6289
User

119605
Reviews

Fresh VS Rotten Reviews



Base Recommenders

Random

- Picks movies at random from the entire set of movies.
- Assigns each recommendation a random review state ("fresh" with 60% chance, "rotten" with 40%) (smoothing of true %'s to make sampling more fair)

Popular

- Sorts movies by gross earnings in descending order.
- Recommends the top movies based on their financial success.

Collaborative

- Converts user review states into numerical ratings (fresh = 1, rotten = 0).
- Uses SVD model collaborative filtering and Performs cross-validation

Preprocessing Data

Data Cleaning

– Removing commas from numerical fields and converting string values to numbers.
– Extracting relevant numerical parts from strings (Runtime).

Text Processing

– Converting categorical data (Genre, Director, Stars) into lowercase.
– Removing extra spaces and punctuation.

Creating the "soup"

– Concatenating categorical attributes into a single text string per movie.
– Enables vectorization with techniques like **CountVectorizer** for similarity calculations.

Normalization & Transformation

– Standardizing numerical features using **StandardScaler**.
– Exploding lists for genre and cast visualizations.

Similarity Calculation

01

Numerical Similarity

Computing pairwise **Euclidean distances** between movies based on normalized numerical features.

02

Categorical Similarity

Transforming the “soup” text into a count matrix. Then calculating cosine distances between movies based on categorical text data.

03

Textual Similarity

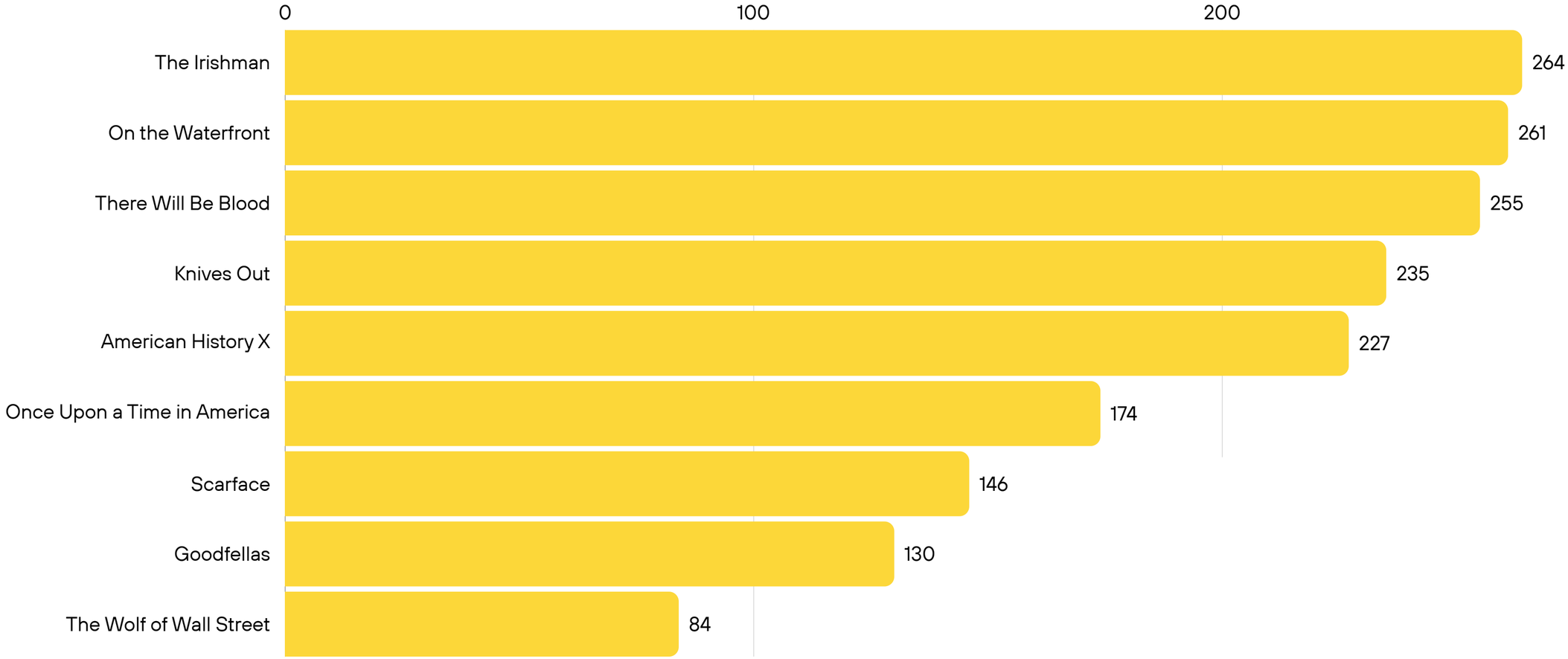
Using **SentenceTransformer** to generate embeddings from movie overviews. Then calculating cosine distances between these text embeddings.

04

Hybrid similarity

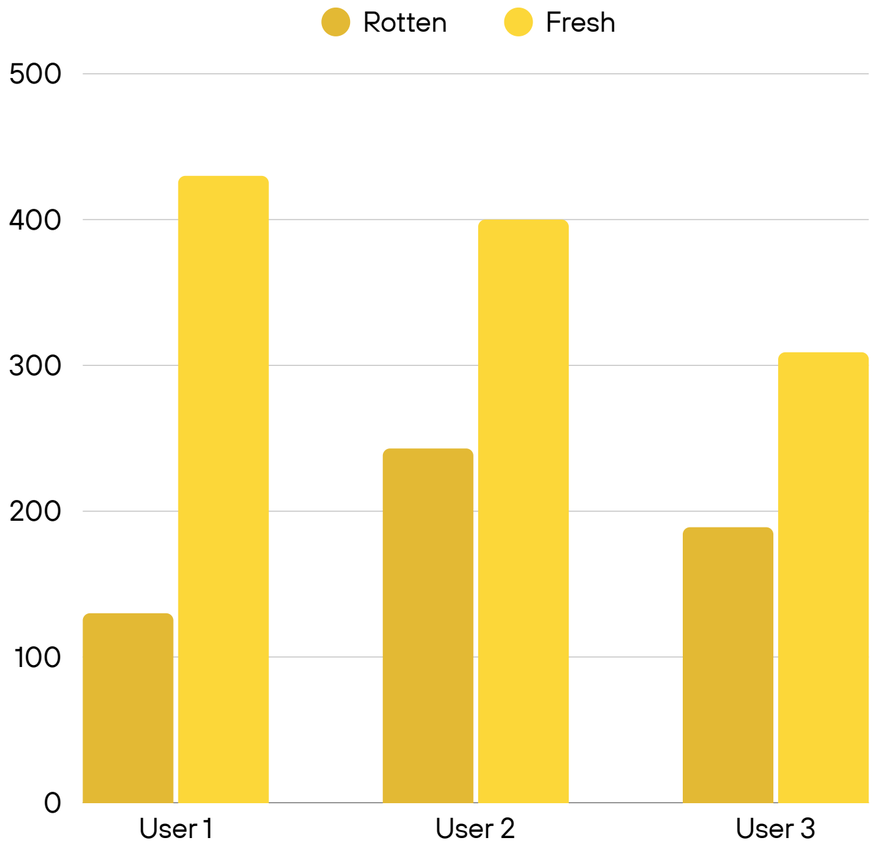
Converting each similarity measure (numerical, categorical, textual) into rank scores and summing ranks to create a hybrid similarity score.

Selecting top similar movies for “The Godfather” using the hybrid ranking.



User-Centric Recommendations

1. Using user review data to distinguish between movies the user liked (fresh) and disliked (rotten).
2. Averaging hybrid similarity scores for both liked and disliked movies.
3. Creating two recommendation lists:
    - Movies to recommend (similar to those the user liked).
    - Movies to avoid (similar to those the user disliked).



Evaluation

Coverage

Ratio of unique movies recommended to total movies ensuring diverse recommendations, not just popular items.

Coverage Score: 73%

RMSE & MSE

Quantifying the error between predicted ratings and actual ratings.

RMSE Score: 0.3835
MSE Score: 0.1471

NDCG

Measuring the ranking quality of recommendations.

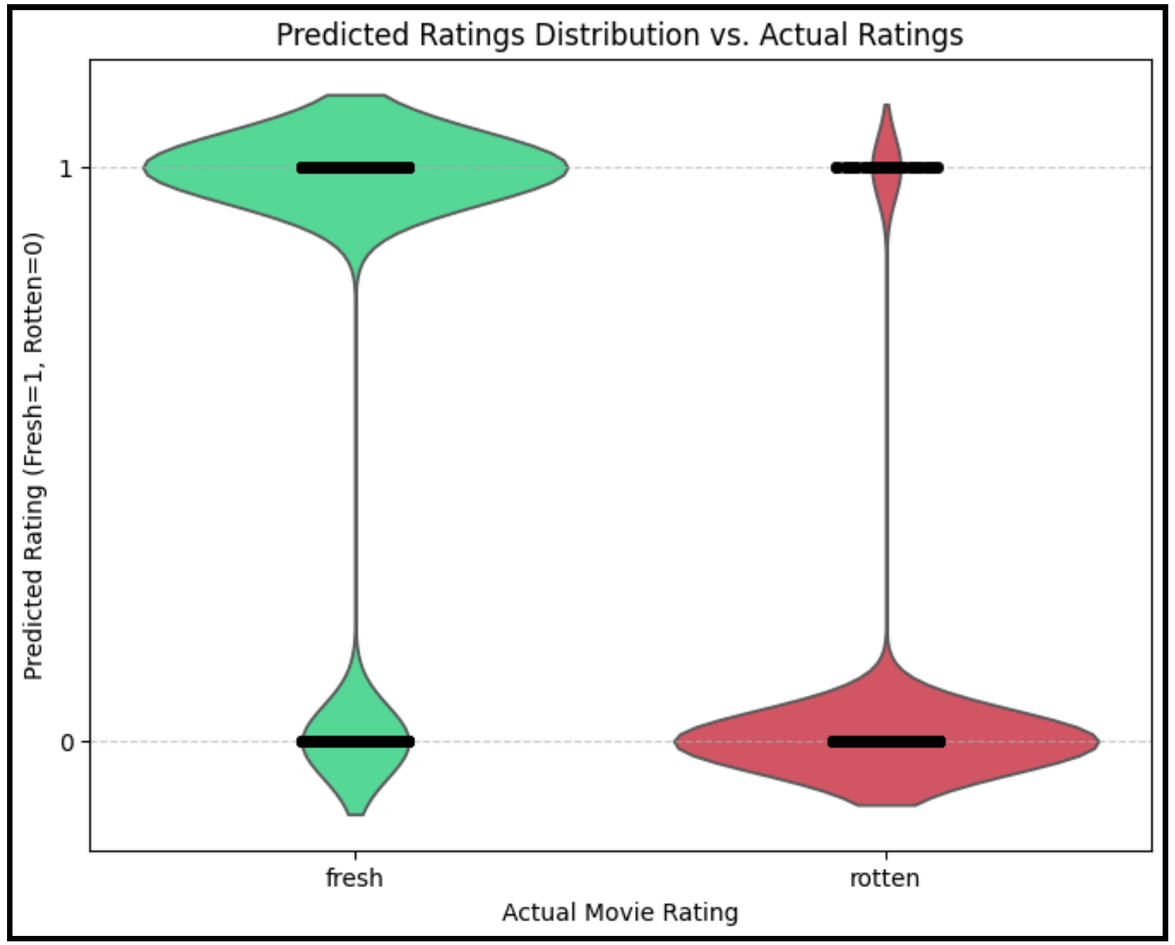
NDCG Score: 0.9956

Comparsion

3

	Random	Popular	Collabrative Filtering	Conten-Based
MSE	0.24	0.09	0.20	0.14
RMSE	0.49	0.31	0.31	0.38
Coverage	100%	1%		73%
NDCG	0.00	0.00		0.99

Content-Based Violin Plot



AI: Chatbots &
Recommendation Engines Course

Thank You

Naif - Alzmami - Nicholas - Adrian