Movie Recommendation System with Sentiment Filtering — Explanation Document

---

Overview

This system recommends movies based on a hybrid approach combining: - Collaborative Filtering (user rating similarities) - Content-Based Filtering (genre similarities) - Sentiment Analysis on user-generated reviews

It uses cosine similarity to compute relationships between movies and incorporates sentiment polarity to refine the recommendations.

Step-by-Step Logical Explanation

1. Loading and Cleaning Data

- The system loads three CSV files: `movies.csv`, `ratings.csv`, and `reviews.csv`.
- Duplicate movie titles are removed to ensure indexing consistency.
- Ratings and movie metadata are merged using `movieId`.

2. Creating User-Movie Matrix

```
user_movie_matrix = movie_data.pivot_table(index='userId', columns='title', values='rating')
```

- This matrix holds each user's ratings across all movies.
- NaN values (unrated movies) are replaced with 0.

Collaborative Filtering (Mathematical)

Formula Used: Cosine Similarity Let A and B be two movies represented as vectors of user ratings:

```
cosine_similarity(A, B) = (A · B) / (||A|| * ||B||)
```

- Measures the angle between two vectors.
- High similarity (closer to 1) means both movies received similar ratings from users.

```
movie_similarity = cosine_similarity(user_movie_matrix.T)
```

- Computes pairwise cosine similarity between all movie vectors.
- Results stored in a DataFrame indexed by movie titles.

Content-Based Filtering using Genres

Genres are tokenized using `CountVectorizer`, then transformed into binary frequency vectors: - E.g., “Action Adventure” becomes: `[1, 1, 0, 0, ...]`

Cosine similarity is again used:

```
genre_similarity = cosine_similarity(genre_matrix)
```

- Resulting matrix shows how close two movies are in terms of genre.

Sentiment Score via TextBlob

- The `review` text is passed into `TextBlob`:

```
TextBlob(review).sentiment.polarity
```

- Returns a sentiment score between **-1 (negative)** and **+1 (positive)**.

The average sentiment per `movieId` is calculated and merged with movie metadata. - Scores are then **normalized to [0,1]** to ensure fair weighting with other similarity scores.

Recommendation Logic

In `recommend(movie_title)` function:

1. **Retrieve similarity vectors** for collaborative and genre filtering.
2. **Compute base hybrid score:**

```
base_scores = (collab_scores + genre_scores) / 2
```

3. **Normalize sentiment scores** and align with base score index.
4. **Combine all:**

```
final_scores = (0.7 * base_scores) + (0.3 * sentiment_scores)
```

- 70% weight to similarity-based scores
- 30% weight to sentiment score

5. Top 5 movies (excluding input movie) are returned based on final scores.

Streamlit UI

- Displays a dropdown of movie titles.
- On clicking “Recommend”, the app shows:
    - The 5 best movies based on hybrid logic.
    - Sentiment scores for added transparency.

Summary

This system blends behavioral patterns (ratings), content metadata (genres), and opinion-based text (sentiments) to deliver personalized and intelligent movie recommendations.

Files Created in This Project

1. `movie_recommendation_without_sentiment_score.py` - Basic recommendation using collaborative + genre filtering.
2. `movie_recommendation_with_sentiment_score.py` - Hybrid recommendation with added sentiment analysis .**(Runs slower than the model without sentiment scoring)**
- **Reason:** The sentiment model performs additional computations by analyzing textual review data using sentiment analysis (TextBlob). Even though the sentiment scores are precomputed, combining them with other metrics and managing more data slightly increases the runtime compared to the simpler model.
3. `generate_reviews_from_ratings.ipynb` - Converts numeric ratings into synthetic review texts.
4. `movie_recommendation_without_sentiment_score.ipynb` - Jupyter notebook version of the basic recommender.

How to Run the App

To launch any `.py` Streamlit app, open terminal and run:

```
streamlit run file_name.py
```

Examples:

```
streamlit run movie_recommendation_with_sentiment_score.py
```

```
streamlit run movie_recommendation_without_sentiment_score.py
```