INTELLIPAAT

Netflix Movie Recommendation System

Personalized Movie Recommendations: A Netflix-Inspired Approach Using Collaborative and Content-Based Filtering

Anuj Kumar

11/1/2024

This project focuses on the development of a personalized movie recommendation system, inspired by the recommendation mechanisms employed by platforms like Netflix. By leveraging **Content-Based Filtering (CBF)** and **Collaborative Filtering (CF)** techniques, the system aims to suggest movies based on both the attributes of the movies themselves and the historical preferences of similar users

Contents

1. Project Overview

Recommendation systems play a critical role in personalizing user experiences on platforms like Netflix, Amazon, and YouTube. This project aims to create a personalized movie recommendation system using two popular approaches: Content-Based Filtering (CBF) and Collaborative Filtering (CF).

By leveraging user behavior and movie metadata, this system provides movie suggestions tailored to user preferences, enhancing engagement and satisfaction.

---

2. Objectives

The objectives of the project are:

1. **Content-Based Filtering:** Recommend movies based on features like genres and release year by analyzing the textual similarity of movie descriptions.
2. **Collaborative Filtering:** Suggest movies based on user preferences and the behavior of similar users.

Address challenges like data sparsity and the cold start problem in recommendation systems.

---

3. Data Description

The project uses two datasets from the **MovieLens** database:

- **Movies Dataset (movies.csv)**
  - Contains metadata about movies, including titles, genres, and release years.
  - Key columns: movieId, title, genres, year.
  - Example:

movieId	title	genres	year
1	Toy Story (1995)	Animation,Children's	1995

- **Ratings Dataset (ratings.csv)**
  - Contains user ratings for movies.
  - Key columns: userId, movieId, rating.

- Example:

  userId movieId rating

  1 1 4.0

---

4. Data Preprocessing

To prepare the data for analysis, several preprocessing steps were performed:

1. **Formatting Movie Titles**: Extracted the release year from titles and cleaned unnecessary text.
   - Example: "Toy Story (1995)" → "Toy Story"
2. **Handling Missing Values**: Removed rows with missing genres or ratings.
3. **Tag Creation**: Combined genres and year into a composite tag column for better feature representation.
4. **Merging Datasets**: Merged the movies and ratings datasets on movieId.

---

5. Methodology

5.1 Content-Based Filtering (CBF)

1. **Feature Representation:**
   - Used **TF-IDF Vectorization** to convert the tag column into numerical vectors.
   - Example Tags: "Animation,1995", "Comedy,2001".
2. **Similarity Calculation:**
   - Calculated cosine similarity between movie vectors to measure their textual similarity.
3. **Recommendation Logic:**
   - For a given movie, identified the top 10 most similar movies.

5.2 Collaborative Filtering (CF)

1) **User-Item Matrix:**
    - Constructed a matrix with users as rows, movies as columns, and ratings as values.
2) **User Similarity:**
    - Computed cosine similarity between users to find similar profiles.
3) **Weighted Ratings:**
    - Aggregated ratings from similar users, weighted by their similarity scores.
4) **Recommendation Logic:**
    - Recommended the top 10 movies with the highest weighted scores for each user.

---

6. Implementation Details

1. Tools and Libraries

- **Python Libraries**: Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn.
- **Algorithms**: TF-IDF Vectorization, Cosine Similarity, Matrix Operations.

2. Code Snippets

Content-Based Filtering:

python
Copy code
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(stop_words='english')
movie_tfidf_matrix = tfidf.fit_transform(movie['tag'])
similarity_scores = cosine_similarity(movie_tfidf_matrix)
```

Collaborative Filtering:

python
Copy code

```
from sklearn.metrics.pairwise import cosine_similarity
user_similarity = cosine_similarity(user_item_matrix)
weighted_ratings = user_item_matrix.T.dot(similar_users)
cf_scores = weighted_ratings / similarity_sum
```

---

7. Results and Analysis

1. Content-Based Filtering

Example Recommendations for "Toy Story":

1. A Bug's Life
2. Finding Nemo
3. Monsters, Inc.
4. The Incredibles
5. Up

2. Collaborative Filtering

Example Recommendations for User 1:

1. The Godfather
2. The Dark Knight
3. Pulp Fiction
4. Schindler's List
5. Forrest Gump

---

8. Challenges Faced and Solutions

1. Data Sparsity:

- Issue: Many users rated only a few movies.
- Solution: Used cosine similarity and weighted aggregation to account for sparse data.

2. Cold Start Problem:

- Issue: Difficulty in recommending for new users/movies.
- Solution: Suggested popular movies as a fallback.

---

9. Future Enhancements

1. **Hybrid Models**: Combine CBF and CF for better performance.
2. **Deep Learning:** Use neural networks for feature extraction and collaborative filtering.
3. **Deployment**: Build a web application for real-time recommendations.

---

10. Conclusion

This project demonstrates the potential of recommendation systems in enhancing user experience. The combination of Content-Based and Collaborative Filtering offers a robust solution for personalized recommendations. With further optimizations, the system can handle larger datasets and provide more accurate suggestions.