Movie Recommendation System Report

Introduction

Movie recommendation systems are essential tools used by streaming services, e-commerce platforms, and social networks to provide personalized content to users. They help users discover movies or shows based on their preferences, increasing user engagement and satisfaction. These systems rely on analyzing patterns in user data and the relationships between users and items to suggest the most relevant content. There are several approaches used in movie recommendation systems, including user-based collaborative filtering, item-based collaborative filtering, and random-walk-based algorithms.

- **User-based collaborative** filtering suggests items based on the similarity between users.
- **Item-based collaborative filtering** recommends items similar to the ones the user has previously rated highly.
- **Random-walk-based approaches**, like the Pixie algorithm, use graph-based methods to simulate how users and items interact, exploring the relationships in a more dynamic way.

Each approach has its strengths and weaknesses, but when combined, they offer a powerful way to personalize recommendations.

Dataset Description

For this project, the MovieLens 100K dataset was used, which contains 100,000 ratings from 943 users on 1,682 movies. The dataset provides a rich source of information to experiment with recommendation algorithms. The dataset consists of the following features:

- **user_id**: Identifies the user.
- **movie_id**: Identifies the movie.
- **rating**: A numerical rating provided by the user for the movie.
- **timestamp**: The time when the rating was made.

The MovieLens 100K dataset was preprocessed by merging the ratings with movie titles from a separate movies_df dataset, enabling us to recommend movies by their names rather than just movie_ids. Additionally, any missing or duplicate ratings were removed, and the ratings were aggregated when necessary.

Methodology

The following three techniques were implemented to generate movie recommendations:

1. **User-based collaborative filtering:**
    - This technique identifies users that are similar to the target user (based on the movies they have rated similarly) and recommends movies that similar users have liked.
    - Similarity between users was calculated using cosine similarity, which measures the angle between two vectors representing the users' ratings.
2. **Item-based collaborative filtering:**
    - Instead of focusing on user similarities, this approach looks at how similar movies are to each other. If a user likes a movie, the algorithm recommends movies similar to it.
    - The similarity between items (movies) was computed using the same cosine similarity approach, but this time comparing movie vectors rather than user vectors.
3. **Random-walk-based Pixie algorithm:**
    - This graph-based approach simulates a random walk starting from either a user or a movie node. The walk randomly moves between connected nodes (users and movies) and counts the frequency of each movie being visited.
    - Graph-based approaches like this are effective because they consider the broader context of relationships and can discover indirect associations between users and items that other methods might miss.

Implementation Details

User-Based Collaborative Filtering:

In user-based collaborative filtering, recommendations are made by identifying users who are similar to the target user. The similarity is calculated based on the overlap of movies they have rated and the ratings they have given. To measure similarity between users, we used cosine similarity, which computes the angle between two vectors (representing user ratings). The smaller the angle, the more similar the users are. Once similar users are identified, movies that they rated highly but the target user hasn't seen are recommended.

Item-Based Collaborative Filtering:

In item-based collaborative filtering, the algorithm recommends items (movies) that are similar to those the user has already rated highly. Similarity between items (movies) was computed using cosine similarity, where each movie is represented as a vector of ratings by different users.

Movies that are similar to each other in terms of ratings will be ranked higher. When a user rates a movie, the algorithm suggests other similar movies for that user to watch based on the similarity between the rated movie and other movies in the dataset.

Building the Graph (Adjacency List):

A graph was created where users and movies are nodes. The edges between nodes represent the ratings given by users to movies. Each user is connected to the movies they rated, and each movie is connected to the users who rated it. To efficiently traverse the graph, an adjacency list was created. This structure allows us to easily access all of a user's rated movies or all of a movie's ratings. For example, if a user rates a movie, an edge is created between that user and the movie in the graph. Similarly, movies are connected to users who rated them.

Random Walk Implementation (Pixie Algorithm):

For the random walk-based Pixie algorithm, we started the walk from either a user node or a movie node. At each step, we randomly selected a neighboring node (either another user or another movie) and moved to it. The walk continued for a predefined number of steps (walk_length). During the walk, we tracked how many times each movie was visited in a dictionary (movie_visits). After the walk was completed, the movies were ranked based on their visit frequency. The more times a movie was visited during the random walk, the higher its ranking as a recommendation for the target user.

Results and Evaluation

Example Outputs: For the user-based recommendation system, the top recommended movies for user_id = 1 might be:

Movie Name	**Ranking**
Toy Story (1995)	**1**
Jumanji (1995)	**2**
Grumpier Old Men (1995)	**3**

Similarly, for the movie-based recommendation system, for movie_name = "Jurassic Park (1993)", the output might look like:

Movie Name	**Ranking**

```
------------------------------------
Rear Window (1954)          | 1
Great Dictator, The (1940)  | 2
Field of Dreams (1989)      | 3
```

Comparison of Methods:

- **User-based** filtering works well when there is a sufficient amount of overlap between users. However, it might struggle when dealing with cold-start problems (i.e., when users have few ratings).
- **Item-based** filtering can be more stable, as it relies on movie-to-movie similarity, which tends to remain consistent over time.
- **Random-walk-based approaches** are unique because they capture latent relationships and patterns in the data, even those that aren't immediately obvious from just user or item ratings. This approach can recommend movies that are indirectly related to a user's previous preferences.

Limitations:

- User-based and item-based methods might suffer from data sparsity, especially with a large dataset like MovieLens where not every user has rated every movie.
- The random-walk method can be computationally expensive due to the randomness and the need to traverse the graph multiple times, especially for large graphs with many nodes.

Conclusion

In conclusion, this project successfully implemented three different recommendation techniques: user-based collaborative filtering, item-based collaborative filtering, and the random-walk-based Pixie algorithm. These methods provide a diverse set of tools for generating movie recommendations, each with its strengths and weaknesses. The random-walk-based Pixie algorithm stands out by considering indirect relationships and capturing the broader context of user-item interactions, making it a valuable addition to the toolkit of recommendation systems.