Movie Recommendation System Report

1. Introduction

Movie recommendation systems are algorithms that suggest movies that may be of interest to a user, based on analyzing patterns in data and the preferences of similar users. They are important to increase engagement, improve user satisfaction, help users discover content that they are likely to enjoy, and they create a personalized experience for each user.

User-user collaborative filtering works by recommending items to a user based on preferences of similar users, using cosine similarity or Pearson correlation to measure the similarity of each user based on their ratings.

Item-based collaborative filtering recommends items to a user that are like ones they already liked, measuring similarity between items and deciding items that are rated similarly by similar users are similar.

Random walk uses a bipartite graph of users and items, starting at a random user or item node, following edges randomly with certain probabilities, and recommending movie nodes that are visited most frequently.

2. Dataset Description

The MovieLens 100K dataset includes details of each user, details of each movie, and each user rating. There are 943 total users, 1682 total movies, and 100,000 total ratings. The main features used are user id, movie id, rating, and title. There seemed to only be one missing release date for the movie, which was filled with a 0, but since that was likely not a relevant feature used for the system, the overall impact of this missing value on the recommendation system is little to none.

3. Methodology

In user-based collaborative filtering, user similarity was calculated using the cosine similarity to measure how similar each pair of users is based on their movie ratings.

For item-based collaborative filtering, cosine similarity was used to measure how similar each pair of movies is based on their user ratings and additionally transposes the user-movie matrix to get the movie similarity, rather than user similarity. Both approaches fill in missing values in the user-movie matrix with zeroes, since cosine similarity does not handle missing values.

For random-walk-based pixie algorithm, graph approaches are effective because they handle sparse data well, are fast and scalable, and are able to explore the graph through short, biased walks to quickly find items closely connected to the users' interests, even if they are not directly related.

4. Implementation Details

User-based collaborative filtering was implemented by computing user-user similarity using cosine similarity, filling missing values in the user-movie matrix with 0, finding similar users, sorting in descending order so the most similar users appear first in the list, and removing the target user by the given 'user_id'. The movie ratings from similar users are then extracted, the average rating of each movie for each user is computed, and the average ratings are then sorted in descending order so the highest rated movies are first. The top 'num' recommended movies from this list are then retrieved, with the movie id's mapped to their respective titles, and the results with rankings are returned as a Pandas DataFrame.

Item-based collaborative filtering was implemented by using cosine similarity to measure how similar each pair of movies is based on their user ratings. Again, missing values in the user-movie matrix are filled with 0, but this time, the matrix is also transposed to get movie similarity, rather than user similarity. The movie ID corresponding to the given 'movie_name' is found, returning an error message if it is not found. The similarity scores from the movie are then extracted, and sorted in descending order, removing the target movie itself. The top 'num' similar movies are retrieved, the movie ID's are mapped to their respective titles, and a similar Pandas DataFrame to the user-based method, with rankings and recommended movie titles, is returned.

For random-walk-based pixie algorithm, random walks were performed by starting at the given movie ID, based on the movie title passed to the function, and for the specified number of steps randomly chooses one of that movie's neighbors in the graphs to go to. It increments that movie's visit count if it is a valid movie ID that is in the graph. It then sorts the visited movies by descending visit count, picks the top 'num' movie IDs from that, maps the IDs to their titles, and returns a similar Pandas DataFrame again, with the recommended movie titles and their rankings.

5. Results and Evaluation

Example outputs:

User-Based

recommend_movies_for_user(10, num = 5)

Ranking	Movie Name
1	In the Company of Men (1997)
2	Misérables, Les (1995)
3	Thin Blue Line, The (1988)
4	Boys, Les (1997)
5	Braindead (1992)

Item-Based

recommend_movies("Jurassic Park (1993)", num=5)

Ranking	Movie Name
1	Top Gun (1986)
2	Speed (1994)
3	Raiders of the Lost Ark (1981)
4	Empire Strikes Back, The (1980)
5	Indiana Jones and the Last Crusade (1989)

Random-Walk-Based Pixie

weighted_pixie_recommend("Jurassic Park (1993)", walk_length=100, num=10)

Ranking	Movie Name
1	Wag the Dog (1997)
2	Con Air (1997)
3	Mighty Aphrodite (1995)
4	Alien (1979)
5	Flubber (1997)
6	It's a Wonderful Life (1946)
7	Belle de jour (1967)
8	Tomorrow Never Dies (1997)
9	Four Weddings and a Funeral (1994)
10	Beverly Hillbillies, The (1993)

User-based approach is less accurate and less scalable. It is effective for smaller systems but not as effective on sparse data. It works well when many users have similar tastes, and in sparse data it becomes harder to find reliable user similarities. It is also more potentially prone to noise due to changes un user behavior. Some limitations to this algorithm include sensitivity to the choice of k, for the top k most similar users, affecting the accuracy of the results, cold start problems with new users or movies, and bias caused by zero-filling the matrix.

Item-based is typically more accurate than user-based since item similarities remain more stable and user similarities. It performs better with sparse data and is highly scalable. It is effective at recommending similar items, but the recommendations may lack diversity and lean more toward recommending the same items. Limitations to this algorithm include no consideration of rating values or user bias, strong popularity bias, and tending to recommend too much of the same.

Random-walk-based pixie algorithm is often the most accurate especially in large and sparse real-world datasets. Short, biased walks allow it to capture deeper relationships and find relevant items even when there is no direct similarity, which helps improve

personalization quality. It is very fast and scalable and works well even for users with few interactions.

6. Conclusions

Key takeaways from implementing and observing each of the recommendation system methods include the importance of data density, scalability, and the ability to account for indirect relationships when designing recommendation systems. Traditional collaborative filtering methods, like user and item based, work well for smaller or denser datasets where similarities are easier to identify, but graph-based methods like random-walk-based pixie are more effective in complex datasets because they capture multi-hop relationships and have better scalability overall.

Potential improvements to the approaches could include using hybrid models, integrating various contextual signals like timestamps or item metadata, and exploring more advanced graph embeddings or neural recommendations to further boost accuracy. For random-walk in particular, adding weighted transitions based on ratings to make probability proportional to rating strength would be a significant improvement to just randomly selecting nodes with equal probability, adding a restart probability can also help to increase stability and prevent getting stuck in clusters, and having multiple independent walks can help to reduce randomness.

For real-world applications, user-based filtering works on smaller platforms with strong community patterns such as Reddit. Item-based filtering is used in e-commerce (Amazon) and streaming services (YouTube, Spotify) to recommend similar items, and random-walk-based pixie algorithms are used for large-scale platforms like Pinterest to enable fast, personalized, and diverse recommendations by efficiently exploring the underlying user-item graph.