ITCS 6122: DATA MINING

REPORT ON

RECOMMENDER SYSTEMS

Submitted by:
Priyal Shah
801357570

INTRODUCTION:

In today's digital age, recommendation systems play a vital role in helping users discover content that aligns with their interests. From streaming platforms to e-commerce websites, recommendation engines enhance user experience by offering personalized suggestions. In this project, we explore and implement three major recommendation approaches – user-based collaborative filtering, item-based collaborative filtering, and a Pixie-inspired random-walk-based algorithm – using the MovieLens 100K dataset.

Each approach offers a unique perspective: user-based methods find similar users to make recommendations, item-based methods identify similar movies, and Pixie-inspired algorithms simulate how users naturally navigate content in a graph-based environment. Together, they form a robust framework for building intelligent and adaptive movie recommendation systems.

Dataset Description:

The dataset used in this project is the MovieLens 100K dataset, which consists of:

- **943 users**
- **1682 movies**
- **100,000 ratings**

Each rating ranges from 1 to 5 and is accompanied by a timestamp. The dataset includes:

- U.data: Contains user-movie ratings with columns – **user_id**, **movie_id**, **rating**, and **timestamp**.
- U.item: Provides movie metadata like **movie_id**, title, release date, and **IMDb URL**.
- U.user: Contains demographic information such as **user_id**, **age**, **gender**, **occupation**, and **zip_code**.

Preprocessing Performed:

- Inspected dataset format and used appropriate delimiters.
- Converted timestamps to human-readable datetime format
- Removed missing values (none found)
- Normalized user ratings for graph-based approaches.
- Saved cleaned datasets as **ratings.csv**, **movies.csv**, and **users.csv**.

METHODOLOGY:

1. **User-Based Collaborative Filtering**

This method looks at users who have similar movie tastes. The idea is: if two users rated many movies similarly, they probably like the same kinds of movies. I used cosine similarity to find the most similar users and then recommended movies that those similar users liked – but the target user hasn't seen yet.

2. **Item-Based Collaborative Filtering**

Instead of comparing users, this approach compares movies. For example, if a lot of people who liked Jurassic Park also liked Indiana Jones, those movies are considered similar. Using the transposed user-movie matrix, I again used cosine similarity to recommend similar movies.

3. **Pixie-Inspired Random Walk**

Inspired by Pinterest's real-time recommendation engine, this approach models user-movie interactions as a bipartite graph. Each node (user or movie) connects to the other type based on ratings. We perform random walks starting from a user or a movie, simulating how users naturally explore content. Recommendations are based on how frequently each movie node is visited during these walks.

IMPLEMENTATION DETAILS:

1. Function Development Steps

- Constructed a user-movie rating matrix using **pivot()**.
- Filled missing ratings with **0** to ensure compatibility with similarity functions.
- Implemented cosine similarity computations using **scikit-learn**.

2. **Adjacency List Graph**

- Created a bipartite graph using a dictionary structure.
- Each user node links to a rated movie, and each movie node links to users who rated it.
- Graph built from normalized ratings to remove user-specific bias.

3. **Random Walk Simulation**

- Starting from a user/movie, the algorithm randomly selects a neighbor at each step.
- Visit frequencies of movie nodes are tracked.
- The most visited movies are considered most relevant.
- Final recommendations are the top-N visited movies excluding the starting node.

RESULTS AND EVALUATION:

User-Based Filtering (User id – 200)

Sample Output:

Ranking	Movie Name
1	Great Day in Harlem, A (1994)
2	Someone Else's America (1995)
3	Marlene Dietrich: Shadow and Light (1996)
4	They Made Me a Criminal (1939)
5	Entertaining Angels: The Dorothy Day Story (1996)

Item-Based Filtering(Jurassic Park (1993))

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 Pixie graph based

Sample Output:

In [29]:
```
# Exploring the Graph

user_id = 1
print(f"Movies rated by User {user_id}:", graph.get(user_id, "user not found"))

movie_id = 50
print(f"Users who rated Movie {movie_id}:", graph.get(movie_id,"Movie not found"))
```

```
Movies rated by User 1: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119,
120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 14
5, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170,
171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 19
6, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221,
222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 24
7, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272}
Users who rated Movie 50: {1, 9, 268, 15, 276, 286, 288, 544, 547, 823, 1084, 319, 324, 325, 327, 475, 100, 124, 1008, 1010, 12
5, 246, 123, 508, 253}
```

Weighted Random Walks:

Movie-based Random walk recommendations

:

Ranking	Movie Name
1	Little Princess, A (1995)
2	Hoop Dreams (1994)
3	Cool Hand Luke (1967)
4	Return of the Pink Panther, The (1974)
5	Schindler's List (1993)

User based random walk recommendations

Ranking	Movie Name
1	101 Dalmatians (1996)
2	Toy Story (1995)
3	Lion King, The (1994)
4	Shawshank Redemption, The (1994)
5	American Werewolf in London, An (1981)

Comparison & Thoughts:

- User-based Filtering gives good results but might struggle with users who have’nt rated much (cold start)
- Item-based filtering is stable and scales better – especially when we have lots of users.
- Random walk (Pixie) if fun and fast, and it’s great at mimicking how people naturally explore content. However, its randomness can sometimes lead to quirky results unless properly tuned.

CONCLUSION:

This project gave me hands-on experience with how modern recommendation systems work. I got to try three different strategies – from math-heavy collaborative filtering to graph-powered random walks – and each one has its own strengths.

What could be improved

- Combine all three approaches into a hybrid model for better performance.
- Use genres, tags, or user demographics to make context-aware suggestions.
- Incorporate more advanced techniques like matrix factorization or deep learning for smarter predictions.

Real-World Use Cases

- Streaming platforms: Suggest movies , music, or shows
- E-commerce: Recommend products based on browsing/purchase history
- Social media: Show users' relevant people or posts they might enjoy.