Movie Recommendation System Report

Aman Waghchoure

UNC Charlotte

ITCS 6162 Data Mining

Dr. Siddharth Krishnan

April 15, 2025

Introduction

In the age of technology, a faulty recommendation system can lose millions of dollars in profits. You are fighting for each customer's attention, it is so limited. Each movie you incorrectly recommend may lead to a customer getting bored and leaving the platform. If you cannot keep up with your competitor's recommendation system, you will not compete. A company specializing in movie or TV streaming services needs to be able to keep users on their platform for a long period of time which will result in high profit. A good recommendation system can ensure a user stays subscribed to a service and you can stay afloat.

There are a few types of recommendation systems including user-based, item-based, and random-walk. User-based recommendation systems use user information to detect the most similar users. Then based on the most similar user rating, we will find the best-rated movie and recommend it. Movie-based recommendation system uses the user's movie rating. Then, we would find the most similar movie to high favorite/highest rated movies and output similar movies. Random-walk recommendation systems randomly select edges to move towards and find the most frequent movies from the current node and output them.

Dataset Description

The data is composed of the MovieLens 100k dataset. This contains 943 users, 1682 movies, and 100,000 ratings. The features include user IDs, movie IDs, ratings, timestamps, movie titles, release dates, and user information such as age, gender, and occupation. We used features such as ratings and user/movie cosine similarity to determine recommendations. The preprocessing steps included merging tables, converting timestamps to dates, and removing and detecting empty entries.

Methodology

Below are the explanations for each of the recommendations techniques implemented:

- **User-based collaborative filtering**: The user was compared to other users to find the most similar users using cosine similarity. Then, these most similar users were sorted to find the top (N) number of similar users for each movie that did not have empty entries. The movie values were determined based on the weighted average of the top (N) most similar users and their ratings. Finally, the top (num) highest-rated movies were selected and output.
- **Item-based collaborative filtering**: The item ID needed to be determined with the movie title. Then, the item was compared to the most similar items. Finally, we would select the top (num) similar items and output them.
- **Pixie Random-walk-based algorithm**: The algorithm would start at a node (user_id). The algorithm would randomly select an edge to move toward for a variable length (walk_length). Since we know the algorithm uses a bipartite graph, we understand the properties of the next node will be different from the current node, meaning if we are at a user node, the next node will be a movie node. We move 2 nodes away from the start node and we end up in the same node set we started in. We would add each movie we walk over. For a variable amount (totSteps), the algorithm would restart the random walk with the knowledge of the previous walk history. When the algorithm finishes, we will have an understanding of the histories of the random walks, understanding the most frequent movies. Finally, we would sample the top (num) movies walked over.

Implementation Details

The user-based collaborative filtering function takes the user_id and num as variables and outputs the most similar items based on user similarity. The item-based collaborative filtering function took the movie_name and num as variables and output the most similar items based on item similarity. Finally, the weighted pixie random walk function took the graph, user_id, walk_length, and num. Further explanations of the function and code are provided in the markdown text field below the code cell.

Below is the description of the input variables used:

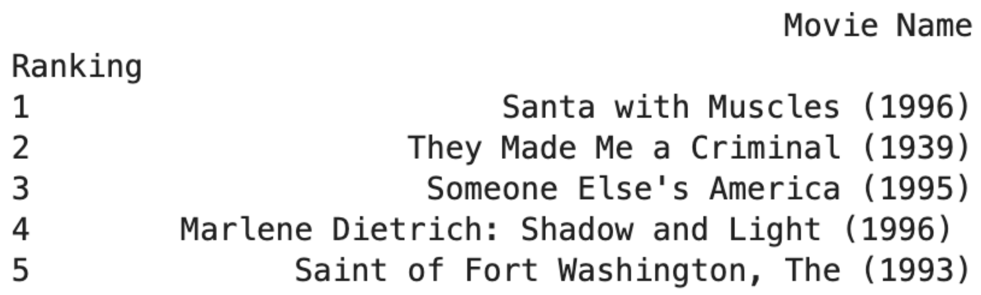
- **User_id**: The id value of a given user.
- **Num**: The top number of items to output.
- **Movie_name**: The name of a given movie
- **Graph**: The bipartite representation of the items and user data
- **Walk_length**: The length to walk before resetting to the start node.

The adjacency list graph was created as a bipartite graph using a Python dictionary. The Python dictionary would either have a key as "user_x" or "movie_x" to differentiate between the items and users. If the additional string was not added the numbers for both graphs would overlap and be incorrectly representing the data. For each pair (user, item) in the row of ratings, we would add this pair to the adjacency list for each node.

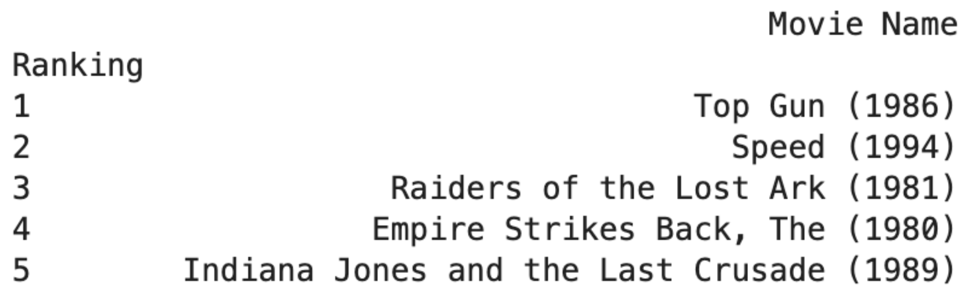
The random walks were performed by randomly selecting a node in the adjacency list corresponding to the key, the current node. It moved twice, once from user_id to movie_id back to user_id. This is how it was implemented in the Pixe paper. Finally, each time it went to a movie_id the value would stored in a dictionary. If the movie_id was very frequent it would appear high in the output dictionary.

Results and Evaluation

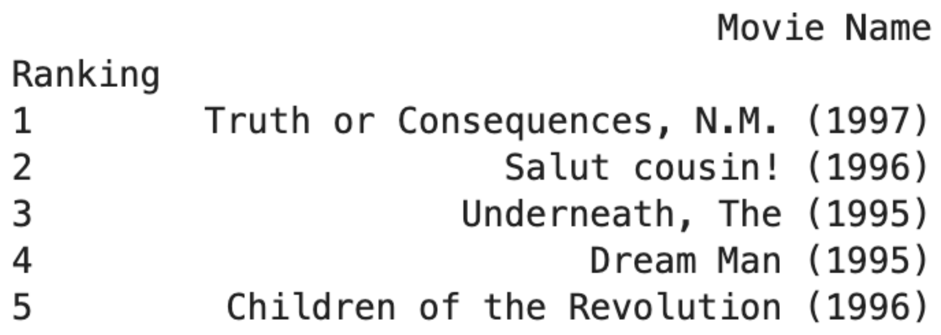
Below are the sample outputs for user-based, item-based, and Pixie random walk collaborative filtering, respectively:



```
                                            Movie Name
Ranking
1                           Santa with Muscles (1996)
2                       They Made Me a Criminal (1939)
3                        Someone Else's America (1995)
4           Marlene Dietrich: Shadow and Light (1996) 
5                  Saint of Fort Washington, The (1993)
```



```
                                            Movie Name
Ranking
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)
```



```
                                    Movie Name
Ranking
1         Truth or Consequences, N.M. (1997)
2                      Salut cousin! (1996)
3                    Underneath, The (1995)
4                          Dream Man (1995)
5          Children of the Revolution (1996)
```

I found that user-based collaborative filtering performed the worst. Some movies were comedies and others were documentaries or action movies. Since the recommendations are based on the most similar users' best-rated movies, it has a very diverse set of movies that it recommends. The diversity can be a plus side, but it likely confuses the user as they never showed interest in that particular genre or time period. Item-based collaborative filtering recommendations performed the best. The item-based collaborative filtering results are very similar to one another. Each of the movies was an action movie from 1980-1995. However, these recommendations lack diversity, but precisely understand which movies the user will enjoy. Finally, the Pixie random walk performed well. It was able to determine the good movies that similar users enjoy that could be useful. If the user has already watched the recommended items, item-based and user-based recommendation number variables would need to be extended. Adversely, random walks would need to be rerun to determine a new set of outputs. This is because item-based and user-based recommendations are deterministic, always finding the top elements. Random walks have a much higher chance of finding new elements each time the algorithm is run.

Conclusion

Overall the project demonstrates the strengths and weaknesses of the different approaches to creating a recommendation system. User-based collaborative filtering allows for more diverse items. Item-based collaborative filtering is more specialized in finding similar items and random-walk-based algorithms are a more holistic approach that finds the underlying neighbor-to-neighbor connections. Future improvements could involve creating a hybrid model that uses weighted averages from all 3 recommendation systems or latent space recommendations for underlying feature detection. These recommendation systems are commonly used in companies like Netflix, Amazon, Pinterest, and many more social networking applications. So every time you try to find recommendations based on a pin on Pinterest or a new movie you enjoy, remember the underlying recommendation system that went into curating it.