Movie Recommendation System Report

Introduction

Recommendation systems are methods of providing content to users that may suit their needs and wants accurately. This is of course extremely useful in a world where there are so many choices for almost everything you do. Do you want a sandwich? There's three places you could go on your current street, each with over a dozen options. The same can be said about movies, especially in the modern day when we have a dozen streaming services and hundreds of movies.

Sifting through large quantities of options that are clearly not your taste is a pain. So, we need systems that can help provide you with a shortened list of items and/or options that are closer to what you really want. This is where recommendation systems come in, and are built precisely to solve this problem.

Recommendation systems, particularly movie recommendation systems, are typically talked about in the context of automatic or semi-automatic systems, but they can also be manual. Many decades ago, a movie recommendation system was a physical person you talked to that could help shorten the list of options for you. Nowadays, they are usually algorithms that analyze your habits and present you with the shortened list. Of course, as the scale of content rises, we need more efficient, faster, better, and smaller algorithms to sift through the billions upon billions of entries of data that each person produces each day.

Through this assignment, we explored a small subset of these recommendation algorithms. User-based and Item-based collaborative systems and graph-based, random-walk, pixie-inspired algorithms. Each has a much different way of handling the same data, and provides different results that have different properties.

Dataset

This project uses the MovieLens 100k dataset. This is essentially a large database of users, movies, and user ratings of said movies. Not much preprocessing was done, but essentially duplicates were removed from the user and movie datasets, and non-existent ratings for movies are considered to be zero. Additionally, for readability timestamps were converted from integers to readable strings.

I only used columns user_id, movie_id, rating, and title of the datasets. However, the timestamp of the rating, user name, age, gender, and occupation are in this dataset. This could be extremely useful in other recommendation systems, particularly machine learning where the age and gender could potentially be used when calculating recommendations. Additionally, they could be useful in making more aesthetically pleasing recommendations in a given app, like presenting their name.

In the end we ended up with:

- 943 Users
- 1682 Movies
- 10000 User Ratings of Movies

Methodology and Implementation

Collaborative

Collaborative recommendation systems use the already existing data that you and other users have provided to aid in their recommendations to you. User based and Item based are both subsets of this type of system.

User Based

Say you have been an active user on TubeYou.tv and have rated many movies that you have watched out of 5 stars. The people at TubeYou.tv could use that data to calculate how similar your taste is to other users on the platform (through a variety of ways) and choose the top most similar users. Then, they can see what those people have rated highly, that you haven't seen/rated, and then recommend the top n of these movies that were enjoyed by these similar users to you.

User-User similarity was calculated with simple cosine similarity. Each user was considered to be a vector of their ratings of each movie, and this vector was calculated against every other user using cosine similarity, and that similarity was stored in a user similarity matrix where each data point was the similarity between two users. (rows=users, cols=users).

Using cosine similarity is simple, but it does have its downsides. For instance, it considers missing ratings, 0, as being negative ratings. A fix for this would be to use the Pearson Correlation coefficient which essentially compares each user's rating of a movie as some signed value of how much above or below their average rating they rated a given movie. However, this was out of scope for this experiment/project.

Item Based

Item based algorithms work in a very similar way, but with one crucial difference. Instead of using your data that you have built up over time to recommend your movies, they simply compute a similar computation on your highly rated movies against other movies on the platform. So instead of comparing you to other users, they are comparing your favorite movies to other movies. This has a nice benefit of not requiring you to have been active for an extended period of time, but has the drawback that maybe your taste is more sophisticated than one movie, and so capturing that taste based on your highest rated movie may lead to less accurate recommendations.

Item-item similarity was calculated in a very similar way to User-User. Each item was considered to be a vector of each user's rating of it. Cosine similarity was again used, and this time instead of a user similarity matrix, an item similarity matrix was calculated (rows=items, cols=items). The same disadvantages apply to this in terms of what kind of similarity was used. It could again be improved by the Pearson Correlation coefficient.

Graph-Based Random Walk

Pixie-algorithms, or random walk algorithms are completely different in how they work. Essentially, they create a graph that represents the user to movie space. Where nodes are movies and users, and edges are relationships between the nodes. So users are connected to movies that they have interacted with in a generally positive way, and movies are connected to users that have rated them in a positive way. The pixie-algorithm will try to simulate a given user's future interactions by traversing this movie-user space in a completely stochastic way. Pick a movie rated by the user, count it as visited, move to another random user that rated that movie, and continue to a movie rated by that user. Continue this for a specified arbitrary amount of steps, and then recommend the movies that were visited the most. This can sort of help diversify the recommendations given and broaden the captured taste of a user. However, if you increase the step size too much the captured taste can be too broad and you can end up recommending something that the user doesn't want.

Implementation for the random graph was a little different than the collaborative methods. The graph needed by the algorithm was a little more involved in its creation.

Essentially, I merged the ratings dataset with the movies dataset to have an easier time accessing a movie's title. Then, I computed the mean of each movie for each user, just in case a user rated a movie many times. The ratings were then normalized around each user's mean rating to ensure that we capture each user's bias in their rating scale. (Similar to the idea that Pearson Correlation tries to capture). Then the graph mentioned above is constructed from this ratings table. (Each node is a movie or user, and each edge is a relationship between them).

Results

Each recommendation's output looks very similar. However, here is an example from each. The accuracy of each is mostly subjective (without a more sophisticated approach, which is out of scope)

User-User Collaborative:

Input: User: 11 | Number of Recommendations: 5

Output:

Ranking	Movie Name
1	Toy Story (1995)
2	Star Wars (1977)
3	Empire Strikes Back, The (1980)
4	Raiders of the Lost Ark (1981)
5	Return of the Jedi (1983)

This appears accurate to me. Seems like user 11 is a fan of Star Wars movies and/or movies with John Williams as the composer.

Item-Item Collaborative:

Input: Movie**:** 'Empire Strikes Back, The (1980)' | Number of Recommendations: 5

Ranking	Movie Name
1	Star Wars (1977)
2	Raiders of the Lost Ark (1981)
3	Terminator, The (1984)
4	Back to the Future (1985)
5	Indiana Jones and the Last Crusade (1989)

This also seems accurate, all of these movies live in the same part of my mind as The Empire Strikes Back. Of course this is subjective, but they also all came out late 70s to mid/late 80s. These are all classics from that time period, and have a similar vibe to them.

Random Walk Pixie Algorithm:

Input: Movie: Jurassic Park (1993) | Walk Length: 15 | Number of Recommendations: 15

Ranking	Movie Name
1	Blade Runner (1982)
2	Nightmare on Elm Street, A (1984)
3	Princess Caraboo (1994)
4	My Own Private Idaho (1991)
5	Die Hard: With a Vengeance (1995)

This seems relatively accurate. The movies are from a similar time period mid 80s - mid 90s. You can definitely pick up on some sort of taste.

Through these results, I think it is interesting to point out the strengths of each algorithm. Particularly, the User-User vs Item-Item. You can clearly see a person's taste behind User-User being captured. Most of the movies recommended are of a similar style, but out of nowhere Toy Story is the top recommendation. Clearly there is something deeper that the data is able to capture that other algorithms cannot. Like Item-Item, it provided a list of movies that are indeed very similar to the input, but the taste is very contained, and there is not much diversity. However, the random walking algorithm might have too much diversity. There isn't as clear of a taste being displayed in the list of recommendations. This could potentially be tuned with the walk length parameter.

Conclusion

In conclusion, recommendation systems are vast and varying in techniques. They are getting more and more necessary everyday as the number of types of content, and even the sheer amount of content being created every year is increasing. The main thing to take away from this report though, is that the properties that different recommendation systems provide vary significantly, and each has pros and cons.

Like mentioned, user-user collaborative systems provide some great properties in that it can capture a more refined and well defined taste if done correctly, but this comes at the cost of not being able to recommend to a user until they have interacted with the platform enough.

On the other hand, you could use item-item collaborative systems. While they cannot capture a more specific user's taste, they can be used to recommend to a user just by using a single highly rated movie.

Graph based random walkers promote diversity and exploration into different categories, but can be noisy if not tuned right.

So maybe the solution lies somewhere in the middle. When a user joins the platform, start recommendations using item-item collaborative systems, and as they interact more and more and branch out on their own, you can start incorporating user-user systems more heavily in the recommendation pipeline.

Maybe you can have an “exploration” section of your movie/content site that uses the pixie algorithm to try and match your taste with a more broad and diverse taste.

I believe this is what many online platforms do nowadays. They also try to increase the data they have on you by asking you to just provide a summary of your taste by asking you to choose your preferred show/categories when making an account.