Movie Recommendation Report

Alexander Mesa

Movie recommendation systems provide movie suggestions for users based on their similarities to other users or movies. This has many applications in streaming platforms and consumer-based websites. Customers can enjoy recommended movies they haven’t seen and purchase products they may want but never hear of. Recommender systems keep users engaged in these subscriptions. The user-based approach idea is that if user likes a movie, then a similar user will probably also like that movie. The item-based approach idea is that if a user likes a movie, then that same user will probably also like a similar movie. The random-walk approach is a graph with users or movies as nodes, and you can hop between random nodes with weighted probability for more similar users/nodes. Each time you hop, you recommend the corresponding movie. We can compute the similarities between two users or two movies with cosine similarity measure, though it’s only effective if the users actively rate the movies. These are the approaches implemented in the Movie Recommendation Programming Assignment

The MovieLens dataset has 100K entries of all the movie ratings. There are 943 users and 1,681 movies. The features are: user\_id, movie\_id, rating, timestamp. The user\_id and movie\_id are numeric strings that represent a specific user and movie, respectively. The rating is an integer that shows how much that user liked that movie, ranging from 1 to 5. The timestamp is an integer for the time of the user rating of that movie. The most important preprocessing step was converting the timestamp values from an Epoch time format to an understandable datetime format. We also had to be mindful of how the feature values were separated in the user, item (movie), and data (ratings) datasets when converting them to CSV files. We used the user\_id, movie\_id, and rating feature values from the ratings dataset to pivot them into a new table with User IDs organized by rows and Movie IDs organized by columns. The data shows the user ratings of each movie with default value of 0.0 if that user did not rate that movie. This dataset is the precursor for the movie recommendation implementation.

The user-based collaborative filtering is calculated by finding the cosine similarities of the input user with respect to every other user. The cosine similarities were sorted in descending order to easily find the top 10 most similar users, excluding the input user themself. We then found the average movie ratings from each similar user and sorted the movies from highest average rating to lowest average rating. We picked the top N movies from that list as recommendations for input user. The item-based collaborative filtering process is similar to the user-based collaborative filtering process with the difference being that we found the cosine similarities of the input movie with respect to every other movie. We also don’t need to compute average ratings of the movies since we already have the list of movies. We just sort the list and pick from the top N movies. The random-walk-based pixie algorithm initializes the connected nodes list by finding the users with at least 100 common movie ratings with the user input. Once we have the nodes, we can use the Python random package to generate a random number for each node (user) where users that are more similar to the input user have a higher weight. The node with the highest number is the node we walk to. We repeat this random walk walk\_length times. We then return the top N movies list sorted by most frequently visited Each user has a movie that corresponds to them, which is the first movie in the movies list that the visited user gave 5 star to. It may be more effective if we randomly choose a 5-star movie from that user to get more random movies, but this is a good start. The graph-approach is effective for helping users continuously find new movies they may be interested by randomly generating movies while still taking into account user or item similarities so that the user would enjoy most of the movies in the list.

The result for user-based filtering for user 1 returns the top five movies in order: Star Wars, Raiders of the Lost Ark, The Empire Strikes Back, The Silence of the Lambs, Aliens. This list makes sense since these movies except The Silence of the Lambs are science-fiction and adventure movies. Still, all of these movies are from the 20th century, so there’s similarity among them. The results for user 100 return: Titanic, Good Will Hunting, As Good as it Gets, Air Force One, Apt Pupil. These movies are romantic or based on real-life events, so they’re very similar. The result for item-based filtering for Jurassic Park are: Top Gun, Speed, Raiders of the Lost Ark, The Empire Strikes Back, Indiana Jones and the Last Crusade. Just like the input movie, these movies are all action, adventure, and fiction, so the list is accurate. The result for Home Alone is: Mrs. Doubtfire, Ace Ventura: Pet Detective, The Santa Clause, Batman, Jurassic Park. Just like the input movie, the top three are family-friendly or comedy movies with a Christmas movie. The last two are still family-friendly but are more action than comedy. It’s not perfect since not everyone who likes Christmas comedies will enjoy superhero or science fiction movies, but the top three movies are very similar to Home Alone. For the random walker with 50 hops, the top five results for User 1 are: Babe, Toy Story, Star Wars, Twelve Monkeys, Seven (Se7en). These movies are kind of all over the place with genre. The top five movies for User 100 are: Contact, Air Force One, Dead Man Walking, Good Will Hunting, Jungle2Jungle. This is a more accurate list since most of these movies have more similar genres like science-fiction and thriller. There’s some romantic or Disney movies mixed in. These are more randomized, so it’s expected that we will occasionally see outliers. For a more randomized movie selection, we could randomly choose a 5-star movie the user we walked to rated instead of just using the first 5-start movie. That’s the one big improvement I would make to the random walker implementation.

This assignment reinforced my knowledge of Python and taught me the DataFrame function from Pandas module. I also learned how to convert datasets into a CSV file so that we can transform the data and pivot so that it’s in a format where we can perform relevant calculations like cosine-similarity. Some improvements include context-filtering like using the history of the recently watched movies by a user for more personalized recommendations. This is great for trilogies, movie sequels, or spinoffs. Instead of cosine similarity, it would be better to use pearson correlation coefficient because that similarity measure is not skewed by missing reviews. Also, users that have similar tastes in movies but different rating standards will have a higher similarity score with pearson correlation coefficient measure than cosine similarity measure. This has real-world applications in virtually every streaming service like Netflix and consumer-based companies like Amazon. Recommendation systems are great for engaging users with unfamiliar products and movies that they will usually enjoy so they can spend money. Part of recommendation system is a business that benefits the consumers and entrepreneurs.