Report

With the wide variety of movies to choose from in modern day, there needs to be a way to easily access movies that each user independently enjoys, that does not include endlessly combing through a catalogue of movies that have nothing to do with the user's interests. This issue can be solved with the introduction of movie recommendation systems, an algorithm that uses a user's previous interest in other movies to find new movies of a similar kind. The three main approaches to building a recommendation system are User-based, Item-based, and Random Walk-based. User-based filtering recommends movies by grouping users based on common habits and preferences, and introduces users in that group to the movies that other users in that group liked. Item-Based Filtering recommends movies based on shared user ratings of any given item. Lastly, Random Walk-based Filtering uses a graph structure to model the relationships between users and items, and traverses the graph randomly to find recommendations. These methods help discover potential new interests for each given user.

In this particular example, we utilized the MovieLens 100k dataset, a dataset that contains 943 users, 1682 movies, and 100 thousand ratings that use a scale from 1 to 5 (hence why the dataset mentions 100k in the title). This dataset also lists the most important features of any given movie in the dataset, which include User ID, Movie ID, Movie Title, and Rating. Some of the steps that were performed before processing the data were: Merging the ratings with movie titles, aggregating and normalizing the ratings, and building a bipartite graph for the random walk to occur on.

There were three recommendation techniques implemented to create recommendations for users. The first one is User-Based filtering, where the idea is to group users together based on

common interests, and recommend movies that similar users liked, but the target user has not watched. This is done by using the cosine similarity to calculate user-user similarity. The next method used is Item-Based filtering, where the cosine similarity is instead used between movies based on a user's ratings. The last method is a Random-Walk-based algorithm where a graph is built based on users and movies they have rated, and a random walk is implemented that starts from any node to gain recommendations based on it. Movies are ranked based on how frequently they are visited during a walk, and the top N most visited movies (that are unrated) are recommended. This method is the most effective because it accurately models associations in user preferences that may not be obvious at first.

The Implementation of these recommendation systems begins with building similarity matrices using cosine_similarity, as well as replacing missing values with 0s to denote a value has not been given by the user. To make the graph, we linked user nodes to movies they rated and vice versa to show the relationships between the two types of nodes. Lastly was the Random Walk algorithm that selected a random node, and if that node was a neighbor, it moved to that new node. The Random Walk would repeat a set number of times and keep track of how many times each movie is visited. Lastly, the movies with the most visits were recommended to the target node.

The results of each approach showed that for User-Based filtering, it was easily personalized and intuitive, however struggled with a low amount of data and had a hard time recommending items to new users. Item-based filter showed to be efficient for multiple users, but had limited novelty if items are closely related. Lastly, for Random Walks, the main advantage was that it catches hard-to-notice relationships between users and other nodes, but the results may vary due to the innate randomness of the algorithm, as it is very sensitive to the parameters

set before running the algorithm. Overall, the biggest issues with these algorithms show up when there is a lack of data, an influx of new users, and unseen preferences with the use of cosine similarity

At its core, this project focused on implementing three recommendation strategies using the MovieLens 100k dataset. Each method had its strengths and weaknesses and proved to be worth considering on a case-by-case basis. Some potential improvements that could have been made were things such as creating a hybrid model that utilizes methods from multiple recommendation strategies, utilizing teleports/restarts more efficiently, and integrating more features that would appeal to people utilizing this algorithm. Some real world applications of this would be streaming platforms such as Netflix, E-commerce websites like Amazon, and other recommender systems like Pinterest and Spotify.