Movie Recommendation AI using KNN

Jack Shang, Electrical and Computer Engineering, University of British Columbia

On many occasions, I would find myself wanting to watch a movie, but not quite sure what to pick. I would turn to peers and family for recommendations, but they often miss the mark compared to what I want to see. I don't like a specific genre of movies, nor do I like a specific actor. I find lots of people in similar spots, so I decided to make a movie recommendation AI that would take a movie I just watched and liked, and recommend similar movies. Different implementations of these AIs are used in many online streaming platforms, such as Netflix, Prime Videos and YouTube, where their algorithms will push for different movies and shows to one's recommended page after you finish watching a video. However, their implementation includes all past videos that one has watched, whereas mine will be a more simple version, where one enters a movie name, and it would recommend five movies for one to enjoy. To do this, my machine learning model would have to take different parameters of different movies, process them to find similarities, and produce movies that are similar to ones entered as prompts. I will be using the MovieLens Small dataset, a dataset that includes movie titles, genres, user ratings and many more metrics recorded in 2018.

I began by comparing different models for the task. Since I wanted a model that would find movies that one would enjoy based on a movie they did enjoy, then I must create a model that aims to compare potential recommendations with the prompt movie

for a large similarity between them, such that it is most likely that the user will also enjoy similar movies. Since this is a regression problem and not classification, I began thinking of using different regression models such as linear regression, logistic regression and Lasso regression. I decided to use a Kth Nearest Neighbor(KNN) model, since it takes parameters, compares them with the nearest neighbors and finds similar data points based on their calculated distance. This KNN model, logically, would be a great fit for my movie recommendation AI, since I want it to compare it to similar movies and find the best-fit movies to recommend. To do this, however, I have to first filter out noise from the MovieLens data. Since many user ratings do not contribute to every movie, it will be hard to interpret and analyze the data. To deal with this, I pivoted the data frame such that each user rated every movie, and replaced the instances where a certain user did not rate the movie with a 0.0 rating. I then removed this sparsity using CSR matrix function from scipy, such that the 0.0 ratings will not affect my KNN model. Finally, I fitted the model with the MovieLens data, and created a function called “recommend” that takes a string (movie name) and the list of movies as input, producing five recommended movies as output based on cosine distance calculated by the brute algorithm from the nearest neighbor class in scipy.

The results for this model were quite difficult to display. Out of the sample tests that ran in the Google Colab file, they seemed to be recommending similar movies, both in genre and by rating. However, it is difficult to measure such a recommendation on accuracy, as there is no classification for “most accurate recommendation” or some quantifiable justification for the accuracy of the KNN model trained to perform this task.

It can be said that the model performs well, since it is able to recommend well-rated, similar movies by a qualitative analysis, and I believe that it solves the problem that it aims to deal with.