Movie Match

Haley Lautenbach

There are already many movie recommendation systems or streaming services that allow for users to set up a profile. These profiles then collect information about the user, what type of films they enjoy, preferred actors, watching habits, etc. This allows for the service to make personalized recommendations to the user based on their behaviour. However, there doesn't seem to be a system in place to recommend to more than one user, which would allow for couples, families or groups of friends to have personalized recommendations.

This project began with a question, how do you choose what film to watch when more than one person is watching? The goal of this project then, was to build a system that will take into account a second user, and recommend films that both users would enjoy.

DATA CLEANING & EDA

The datasets used came from MovieLens as well as IMDB. From MovieLens, the 27M, 20M and 1M datasets were used. The 1M dataset, which has one million ratings, was used for training the models and making predictions. The smaller dataset had to be used because the model optimization had to be done on a local machine. The MovieLens 27M dataset was used for the 'links' table, which gave ids for the MovieLens and IMDB datasets, so that movie information could be linked along these keys. The 20M dataset was used for the Movies and Tags tables. The IMDB datasets were used for auxiliary information on the runtime, number of votes and average rating as well as which directors and writers were involved with the films.

Pre-processing involved taking all of the data in its raw tabular form and transforming it into utility matrices. The movie-feature utility matrix requires each row to be a different movie and each column to be a different feature. Each field where the movies and features intersect is filled with either a 1 or a 0, 1 meaning that the movie does have that feature and 0 meaning the movie does not. The user-rating matrix is a similar format, where each row is a user and each column is a movie. However, for this matrix, the fields where the user and movies intersect is filled with a rating, on a scale of 1-5. If the user has not yet rated a film, then the field will be blank. To get the user-rating matrix into this form, the Python package Surprise was used to convert data into a "surprise dataset". This dataset is a form of utility matrix best suited for the models in the package.

The movie-feature utility matrix involved many complicated steps of pre-processing. The Movies table had lists of genres in each row that then had to be pulled out of each field and vectorized. The films year of release was within the title string, at the end, which had to be extracted and formatted into its own column. This column was then one-hot encoded and re-merged with the original table.

The tags table had tags that were input by users, meaning they were all formatted differently with different language styles. The tags had to be normalized, vectorized and then

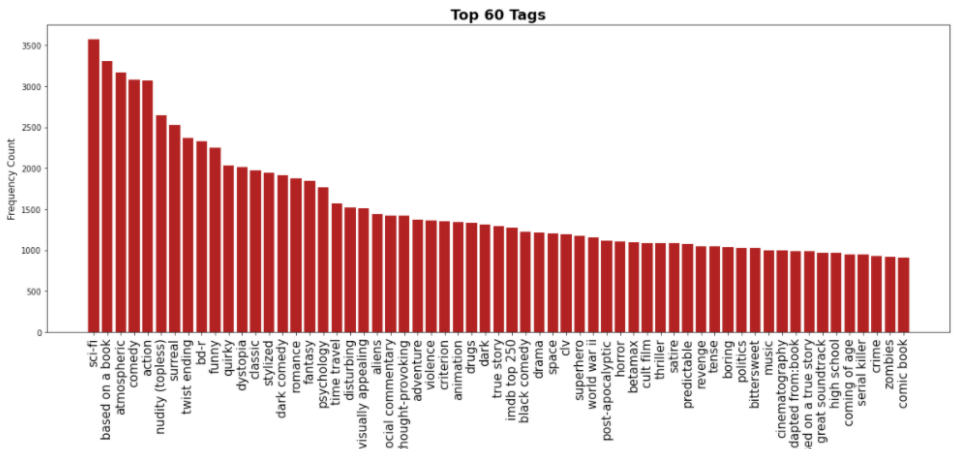

Figure 1: The top 60 tags in the dataset, after normalization and filtering.

filtered so that tags that appear infrequently as well as tags that are too common are removed from the dataset. Having all of these included could potentially cause overfitting when optimizing the models. As you can see in the figure on the previous page, some of the tags after those steps were taken are still quite similar.

Similar steps were taken for the directors and writers provided in the IMDB datasets. However, with these, each film had an embedded list of writer and director ids associated with them. The lists then had to be pulled out of each field and vectorized with the appropriate binary indicators in the fields of intersection. The column headers were then id codes for the names instead of the actual names provided. While this would not be an issue really for the creation of the similarities matrix, it would be an issue for human visibility if the matrix was ever needed for any forms of reference. The column headers then had to be matched up with the actual names of the actors in a different table and renamed.

MODEL SELECTION

The movie-features matrix was used to create a movie similarities matrix. Cosine similarity was used to calculate how similar the movies are to one another, which is the dot product of the two vectors divided by their lengths multiplied. This is why the matrix had to be in that specific format, so that the cosine similarity scores would be as accurate as possible.

The first model used was the Funk SVD, or singular value decomposition. This is a method of matrix factorization that strives to find the values for two decomposed matrices U and M so that the dot product values of these two matrices is as close to the original values as possible. The matrices U and M are then used to calculate the empty spaces in the original matrix, resulting in predicted rating estimations. The second model used was a KNN (Nearest Neighbours) style of recommender that took into account baselines. The model calculates the similarity of "neighbours" using the Pearson correlation coefficient, modified to use the user and item baselines instead of means. The models were optimized using the error metrics RMSE, MSE and MAE as well as the FCP score. RMSE and MSE penalize single large errors over many small errors, while MAE penalizes all errors equally. The FCP score calculates the fraction of the ratings which are in the correct order, or the fraction of concordant pairs. For the majority of the optimization process, all error metrics moved together, as shown in the figure below.

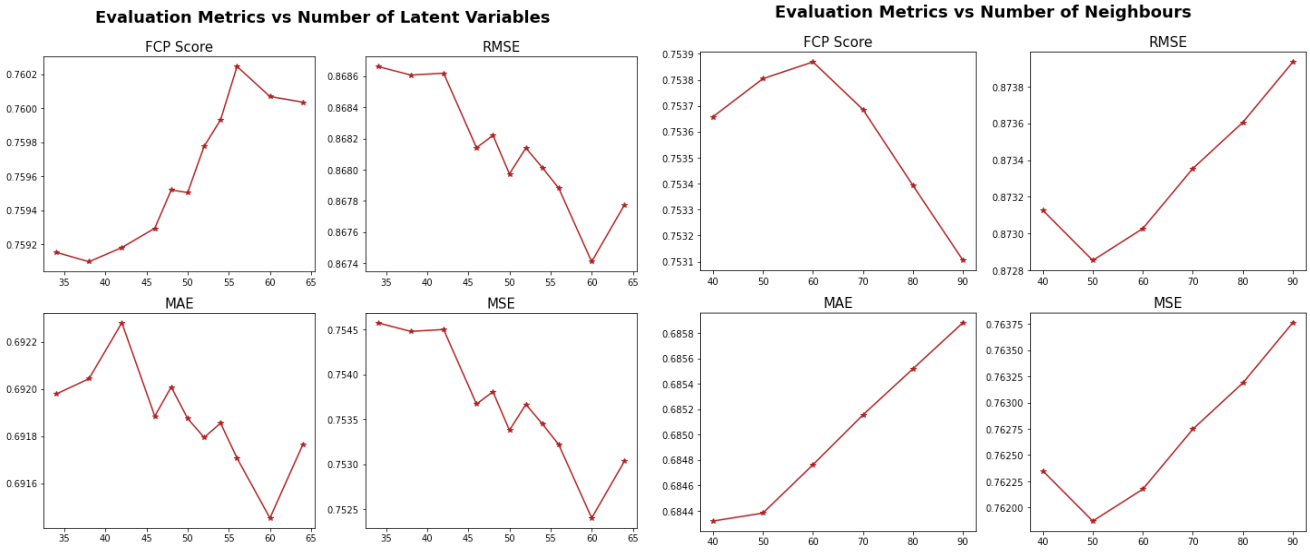

Figure 2: The scores for the evaluation metrics during the optimization of number of latent variables for the SVD model (left) and the optimization for the number of neighbours (K) for the KNN model (right).

Once the optimal parameter values were found that returned the lowest levels of error, or highest FCP score, the models were then re-fit on the entire dataset. These new models were used to predict ratings for all empty values in the ratings matrix, using a surprise datatype known as the anti-testset. Once all predictions were generated they were averaged together, to create a hybridized rating set. This was done because the precision and recall scores for the SVD alone showed high precision, low recall. With the influence of the ratings from the KNN, which alone had lower precision, but higher recall, the overall precision and recall scores would be more even. Both high precision and recall are necessary with recommender systems so that not only are all of your recommendations relevant to the user, but also that you are recommending all relevant films. With low recall, the system misses a lot of relevant recommendations.

OUTCOMES AND THE FUTURE

Within the project timeframe, a system was created that recommends films to two users that were matches on both of their top rated lists. These recommendations can then also be used to find top movies most similar, allowing for a wider breadth of recommendations if the users do not have many points of commonality between them. The functions are computationally and memory intensive at the moment and would benefit from only using and storing information that is necessary at the time of the function call. The system also suffers from issues where a lot of movies that are recommended are similar across all users, which leads to a lot of repeat recommendations. This is not ideal because it shows that the system is not picking up on specific user preferences and choosing a significant portion of the recommendations based on general popularity.

Similar To:	Movie	Year of Release	Similarity Score
Saving Private Ryan	Schindler's List	1993	0.563621
	Boot, Das (Boat, The)	1981	0.477214
	Downfall (Untergang, Der)	2004	0.468293
	Braveheart	1995	0.444878
	Gladiator	2000	0.430690
Toy Story 2	Toy Story	1995	0.629941
	Monsters, Inc.	2001	0.521168
	Pooh's Heffalump Movie	2005	0.512823
	Toy Story 3	2010	0.501739
	Pinocchio	1940	0.491354
Green Mile, The	Shawshank Redemption, The	1994	0.383326
	Godfather, The	1972	0.361403
	American History X	1998	0.347440
	Forrest Gump	1994	0.344318
	American Beauty	1999	0.332900
Life Is Beautiful (La Vita è bella)	Celeste and Jesse Forever (Celeste & Jesse Forever)	2012	0.412479
	City Lights	1931	0.408248
	Forrest Gump	1994	0.379049
	Train of Life (Train de vie)	1998	0.375000
	Stalag 17	1953	0.375000
Apollo 13	1492: Conquest of Paradise	1992	0.375239
	Captain Phillips	2013	0.371391
	Tracks	2013	0.352332
	Contact	1997	0.352332
	Kon-Tiki	2012	0.350931

Figure 3: A sample output of the top 5 for two users, as well as the top 5 films most similar.

The next steps could be to have the system created put in to an already existing streaming service, so that user profiles could be mixed together when more than one person is watching. It could also be made into a new app or website that you can add 'friends' profiles together when deciding to watch a film, with the streaming services each member subscribes to being taken into account for the recommendation.

Recommendation systems like this, in general, would be very useful especially in today's world. More than ever, people are in long distance relationships, long distance friendships and having to connect online. There are apps that allow you to watch films simultaneously over multiple profiles within a streaming service, but not ones that recommend items to users based on multiple profiles. Implementation of this system to the current services is beyond the scope of this current project, but definitely a subject of future investigations.