Predicting Through The Lens: A Look At Recommendation Systems

Kit Delling Piepkorn
Knox College
Galesburg, Illinois
kdpiepkorn@knox.edu

Abstract— **Streaming services are constantly looking for ways to increase their viewer interaction with their system. One of the primary ways that these companies (e.g. Netflix, Hulu etc.) accomplish this, is through the use of recommendation systems. These systems take existing user data, and through the use of machine learning algorithms, determine a user's preferences. This paper will examine applications of Content-Based and Collaborative filtering systems, primarily through the use of similarity matrices and matrix factorization. These systems will be trained and tested on the MovieLens dataset. The goal of this research is to test several models in order to achieve the best level of accuracy and prediction when estimating a user's preferences and future viewing needs. Finally, I will discuss the results to show the effectiveness of fitting and training a model with Content-Based and Collaborative Filtering algorithms in order to increase user interaction within a streaming service.**

I. Introduction

Recommendation systems are a prevalent part of our everyday lives. They can be found in a wide variety of companies such as Netflix, Hulu, Amazon, Spotify etc. (Takaes et al., 2009) Recommendation systems' primary focus is to try and predict what a particular user will want next from the current database, (e.g. A movie for Netflix or a song for Spotify). Recommender systems are an incredibly popular and powerful tool as those who are able to accurately predict what their customers will want next are able to capitalize on the precious commodity known as the user's time.

For this research I decided to use a popular dataset in the recommender world known as the MovieLens dataset. This clean and easy to use set has a storied history and I deemed it to be an appropriate dataset to use for my research.

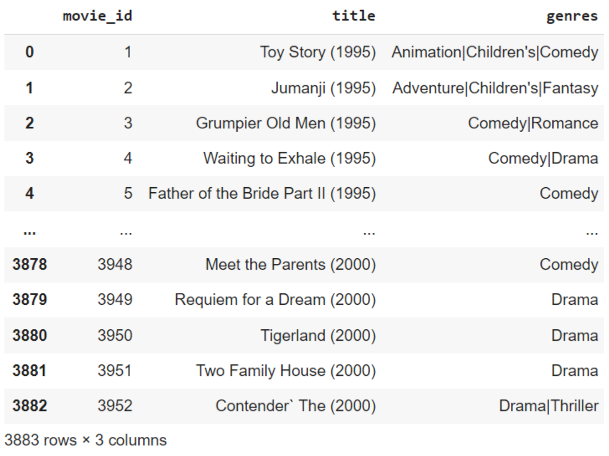
There are many different techniques utilized in creating recommender systems and in this research I will be focusing on two of the major categories, Content-Based and Collaborative Filtering. Using 3 different systems I will evaluate their ability to perform on the MovieLens dataset to see which system performs most accurately while giving worthwhile predictions.

II. MovieLens

The movielens dataset has played an important part in the landscape of recommendation systems. A group at The University of Minnesota called GroupLens created the MovieLens database in the summer of 1997. (Harper & Konstand, 2015) MovieLens is a database that was originally developed with the purpose of building recommendation systems for movies that rely on user preferences in order to make recommendations. The first MovieLens dataset was released in 1998 with information gathered from the MovieLens website. The first dataset is called the 100k dataset because 100k is the number of ratings included in the dataset. One important thing of note is that only users who rated a minimum of 20 movies were included in the data. This is because the data is already incredibly sparse so this requirement is put in place to help manage that, in addition many recommender systems need a user history in order to make accurate predictions. There have been 3 other versions released since 1998, those being the 1M, 10M and 20M. Each version number represents the number of user ratings included within the dataset. I chose to use the MovieLens dataset because the set is clean, easy to use data that has provided stable data to researchers for many years. The release of multiple data sets over a period of time all expanding off each other has cemented MovieLens in the history of recommendation systems. There are many systems that used MovieLens to train and develop many models that have advanced recommendation systems, "including item-item collaborative filtering, dimensionality reduction collaborative filtering, trust-based recommenders, recommenders on rapidly changing sets of items, and cold-start algorithms." (Harper $ Konstan, 2015) In addition to this, the datasets have been downloaded hundreds of thousands of times since their release and have been a staple in the world of research, education and industry.

A. *Looking At The Data*

There are multiple tables included in the MovieLens dataset, but for the purposes of this research only 3 were used. These being the movies, users and ratings tables. The attributes included in the movies table are a unique ID for each movie, the title of the film along with the year of release and a genre attribute which labels all the genres associated with a particular movie. There are roughly 4000 unique users contained in the 1M dataset.

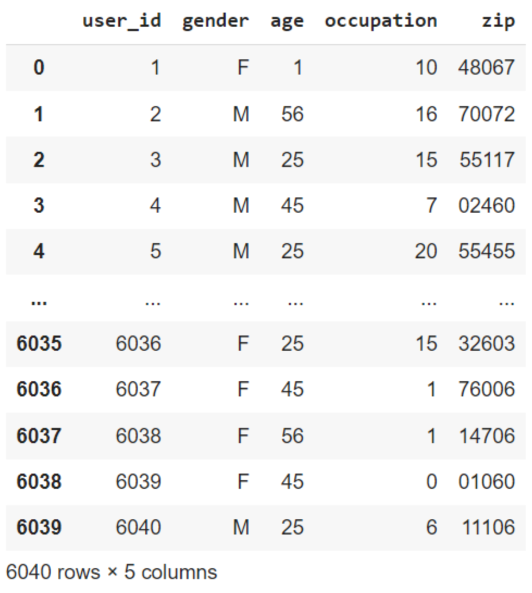


	movie_id	title	genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
3878	3948	Meet the Parents (2000)	Comedy
3879	3949	Requiem for a Dream (2000)	Drama
3880	3950	Tigerland (2000)	Drama
3881	3951	Two Family House (2000)	Drama
3882	3952	Contender` The (2000)	Drama|Thriller

3883 rows × 3 columns

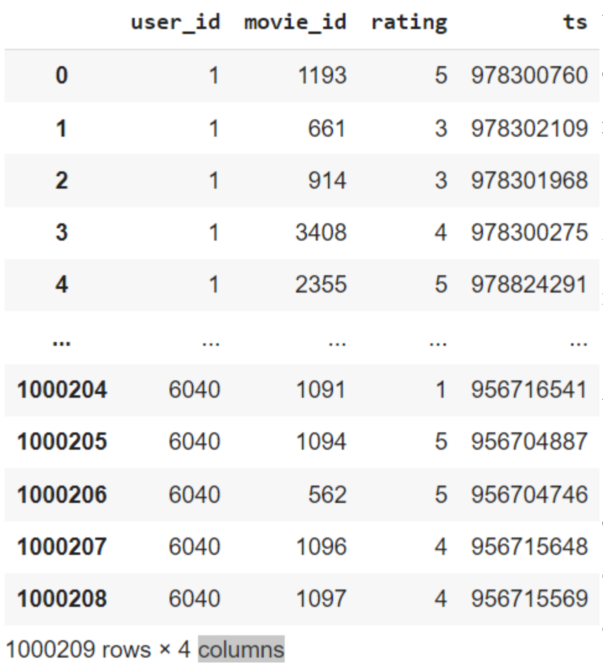
The users table includes a unique identifier for each user along with their gender, age, occupation and zip-code. The occupation attribute is only contained in the 100k dataset as it was discontinued from the MovieLens website before the 1M set was released. However, for the occupation attribute, each number is a unique identifier which is tied to a specific job or field of work.

The last table needed for this research is the ratings dataset. The ratings table includes the unique identifiers for users and movies. The rating attribute represents the rating given to a specific movie by a specific user.



	user_id	gender	age	occupation	zip
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455
...	...	...	...	...	...
6035	6036	F	25	15	32603
6036	6037	F	45	1	76006
6037	6038	F	56	1	14706
6038	6039	F	45	0	01060
6039	6040	M	25	6	11106

6040 rows × 5 columns



	user_id	movie_id	rating	ts
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291
...	...	...	...	...
1000204	6040	1091	1	956716541
1000205	6040	1094	5	956704887
1000206	6040	562	5	956704746
1000207	6040	1096	4	956715648
1000208	6040	1097	4	956715569

1000209 rows × 4 columns

III. Content-Based

In this section I will be discussing the first of two primary types of recommendation systems, Content-Based filtering, as well as the challenges and opportunities of implementing such systems.

A.

Content based filtering systems rely on using the characteristics of items in a dataset in order to make predictions. In the example of recommending movies, some examples of characteristics that could be used are the genre, cast, director and run time. Content-Based systems draw similarities found between these characteristics to form similarity matrices so that similar items can be recommended. (Karagiannakos, 2021) For example, when predicting movies using genre, if a user watches several comedy movies in a short period of time the system would think the user would want more comedy movies, and from there recommend to them a list of mostly comedy movies. Content-Based systems are relatively easy to implement when compared to their Collaborative counterpart and in general take less computation resources to execute.

Content-Based systems are good at dealing with very sparse data as items do not need user interaction in order to find their similarity to all other items in the set. Content-Based systems are also a good solution to the Cold Start problem.

The Cold Start problem is a term that refers to cars. During cold temperatures, when a car is left outside overnight, the automobile struggles to turn on and run efficiently. However, once the vehicle has been on and running, the internal temperature becomes optimal, and the engine will run smoothly. (Milankovich, 2017) This analogy directly relates to many recommender systems. Many recommender systems rely on user ratings to make their predictions. When a new user is introduced to a system, the user has no previous interaction with any of the items within the data. So, when a system that relies on ratings is given a new user, it has no item interaction history to go off and therefore can only make very inaccurate predictions. With that being said, Content-Based systems are a good solution to this problem because, as stated earlier, they do not make their predictions based on user interaction but rather similarities between items Themselves. (Milankovich, 2017) A Content-Based system would only need a list of preferences from a new user to make recommendations, or for a user to interact with a single item.

The biggest weakness of Content-Based systems quickly becomes apparent. These systems are unable to take into account how a user interacts with a film, only that they did interact with it. A movie could be watched that the user did not actually like but all the system would see is that the user had seen it. From there, the system would then go on to recommend more movies to the user similar to the disliked movie. (alextsoft, 2021) Another weakness is that Content-Based systems are unable to pick up on hidden patterns in the data and are unable to expand upon a user's interests and recommend more out of the box items. The last big weakness of Content-Based is that it does require extensive domain knowledge in order to build datasets that Content-Based will be effective on.

IV. Collaborative Filtering (CF)

CF takes a different approach than Content Based and has many more techniques and implementations than Content-Based. This is because CF systems make recommendations based on similarities between user history with target items. CF strives to predict how a user will interact with items they have not used before. (Kang et al., 2018) So, for the purposes of this research, predicting what a user will rate a given movie.

A. Memory Based CF

There are two main types of CF systems, Model and Memory Based. Memory based CF systems assume predictions can be made on the memory of the data. Memory based models can be further split into two types of models, User-User and Item-Item.

Item-Item models operate very similarly to Content-Based models in the sense that similarities are found between movies and recommended to a user, the big difference though is that Item-Item models are dependent upon how a user interacts with a specific item. (Zheng et al., 2016) These systems take into account how a user rated a movie, and if it is a positive rating, would then recommend movies that are similar to the liked movie.

User-User models take a different approach. In these models, similarities are drawn between users. This is done by comparing the ratings of every user and finding those users who tend to rate the same or similar items highly. (Han et al., 2021) Once similar users are found and paired, movies are recommended that have been seen or viewed by a similar user to their pair.

B. Model Based CF

Model-Based systems are another type of CF. These models utilize machine learning and or data mining techniques to extract latent features and find patterns within the data. (Chen, 2017) There are many techniques which can be used to accomplish this, but for the purposes of this research I focused on evaluating models that incorporate matrix factorization to help deal with the data sparsity of the MovieLens dataset. Techniques like matrix factorization are utilized to fill in missing data and predict user ratings for items they have no interaction with.

1) Singular Value Decomposition (SVD): In this research a specific matrix factorization method is utilized, this method being singular value decomposition. SVD takes a rectangular matrix of data (nxp) and decomposes it into 3 matrices in the form of $A = U_{nxn}\Sigma_{nxp}V_{pxp}^T$ where U has the left singular vectors, E has singular values and is diagonal and V^T has rows which are right singular vectors. The eigenvectors of A^TA make up the columns of V, the eigenvectors of AA^T make up the columns of U, and S is composed of the square roots of eigenvalues from AA^T or A^TA and are ordered according to importance. When reconstructing A from the decomposed state, each latent factor has equal influence power to the final rating. (Chen, 2017) SVD allows us to deal with the data sparsity of the MovieLens dataset and build predictions for items which have not yet been rated by each particular user.

C. CF Strengths and Weaknesses

CF is typically the more commonly used type of system compared to Content-Based. I believe this is due to the fact the CF systems take into account a user's interests rather than just what they have seen. Model-Based systems such as matrix factorization are able to pick up on hidden patterns within the data. These hidden patterns allow these systems to make recommendations that are otherwise not obviously apparent. Lastly, CF systems do not require domain knowledge as most of them operate solely on the ratings attribute within the data.

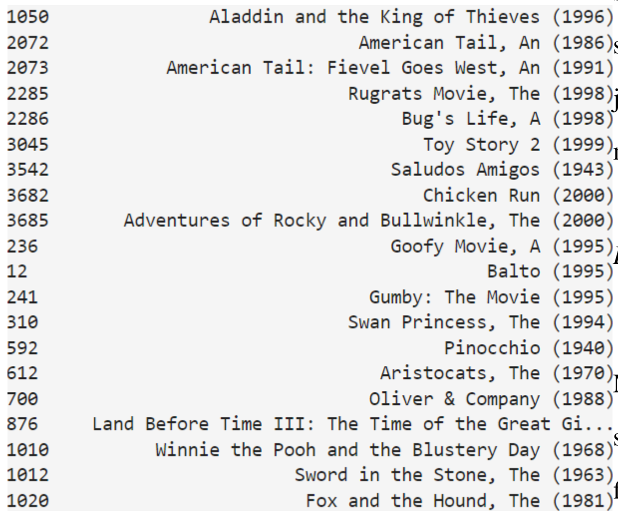
A big weakness for CF systems is that they struggle with the cold start problem on two fronts. (Milankovich, 2017) When a new movie is added to a system, it has no ratings which means no user interaction. Because of this the system struggles to recommend the film to users as it has no history. The other front is on the user end. When a new user is entered into the system they will have no movie interactions and therefore the system will have nothing to base recommendations on. This problem is often solved by creating a hybrid system that uses collaborative and content filtering. Another disadvantage that CF systems struggle with is the issue of data sparsity. This means that if a system does not have enough information in general, the systems will give less than ideal prediction estimate scores. Certain techniques help to address this problem such as Matrix Factorization. However, the limited accuracy of these techniques may be attributed to the sparsity level of the MovieLens Dataset (95.5%)

V. Models Evaluated

For this research I decided to implement and review 3 different models, those being one Content-Based, one Memory-Based CF model which contains both a Item-Item and User-User model side by side, and one Model Based model which utilizes the matrix factorization technique known as Singular Value Decomposition. Each of these models were compiled with the specifications provided from (Namle) which is included in the bibliography.

A. Content-Based Model

The first model I will be evaluating is a Content-Based Filtering system. I thought it would be best to start with this because they are generally regarded as the easier of the two systems to implement. For this particular model, the genre attribute was used as the characteristic that would be used for finding similarities between movies. This was done by breaking up the genre strings using a tokenizer and then using the TfidVecotrizer library from sklearn.(Namle) The genres were broken down into feature vectors that could then be compared using Cosine Similarity.(Namle) This system was pretty straightforward with its predictions as movies of similar genres were recommended for a given movie. A good example of this is seen when looking at the top 20 movies generated from a user watching Toy Story.



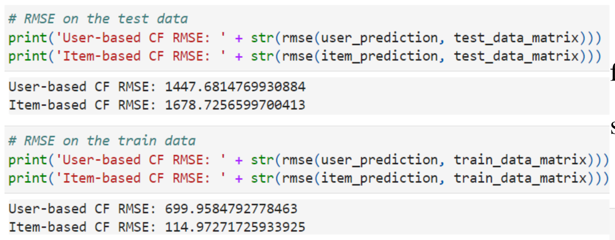
```
1050              Aladdin and the King of Thieves (1996)
2072                           American Tail, An (1986)
2073           American Tail: Fievel Goes West, An (1991)
2285                          Rugrats Movie, The (1998)
2286                              Bug's Life, A (1998)
3045                                Toy Story 2 (1999)
3542                             Saludos Amigos (1943)
3682                                Chicken Run (2000)
3685         Adventures of Rocky and Bullwinkle, The (2000)
236                              Goofy Movie, A (1995)
12                                        Balto (1995)
241                            Gumby: The Movie (1995)
310                          Swan Princess, The (1994)
592                                   Pinocchio (1940)
612                              Aristocats, The (1970)
700                            Oliver & Company (1988)
876     Land Before Time III: The Time of the Great Gi...
1010           Winnie the Pooh and the Blustery Day (1968)
1012                        Sword in the Stone, The (1963)
1020                        Fox and the Hound, The (1981)
```

Looking at the results it's easy to see that all of these movies are also animated movies, meaning that the most importance for movie similarity was put on the animated feature. The drawbacks to using this kind of recommendation system become apparent quickly. Since this system does not use any user input like ratings, it cannot take into account how a user felt about a movie. This means that we are unable to really test for any kind of effectiveness, and are only able to make very surface level recommendations. For these reasons, this was the only Content-Based system that I decided to implement as the effectiveness can be very limiting especially in the context of movie predictions. Being able to find movies that are of a similar genre can be a useful feature, and it becomes apparent why this is a good system to use when dealing with the cold start problem. (Milankovich, 2017) While we don't know how a user felt about the movies, the ability to generate recommendations despite the lack of user input is a great way to handle new users who have not rated enough movies to utilize Collaborative Filtering Systems.

I did not run any evaluation metrics on this system as I did not find it necessary to do so. This system does not predict any kind of user ratings and just gives the movies most similar to the watched movie.

B. Memory-Based Collaborative Filtering

The second system that I implemented was a Memory-Based Collaborative Filtering system. This system creates a User-User and an Item-Item based filtering. Typically, one of three similarity metrics are used when creating the initial similarity matrices between users and their ratings, also between movies. These are the Jaccard, Cosine and Pearson Similarity measures. The system I evaluated chose to go with Pearson as the similarity measure. (Namle) For this system the user-user predictions took into account the similarity between user ratings while also adjusting for the average rating of a user. This helps take into account a user tending to rate on a higher or lower score and helps to balance that out. This model was measured using Root Mean Squared Error on both the User and Item based models.



```
# RMSE on the test data
print('User-based CF RMSE: ' + str(rmse(user_prediction, test_data_matrix)))
print('Item-based CF RMSE: ' + str(rmse(item_prediction, test_data_matrix)))
```

```
User-based CF RMSE: 1447.6814769930884
Item-based CF RMSE: 1678.7256599700413
```

```
# RMSE on the train data
print('User-based CF RMSE: ' + str(rmse(user_prediction, train_data_matrix)))
print('Item-based CF RMSE: ' + str(rmse(item_prediction, train_data_matrix)))
```

```
User-based CF RMSE: 699.9584792778463
Item-based CF RMSE: 114.97271725933925
```

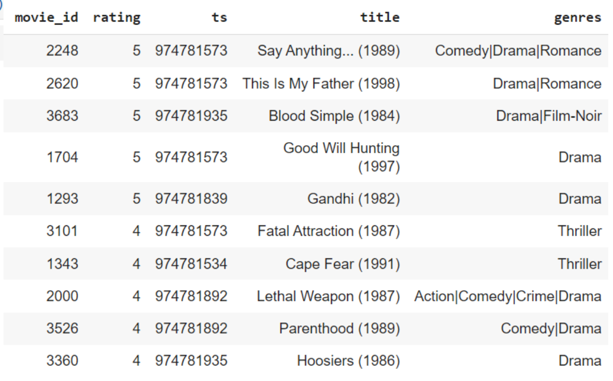
Looking at the RMSE comparatively between the test and train data, I am led to believe that this system is overfitted considering the disparity between the test and train, also the size of the RMSE for the test data.

Looking at the results of this system, while it may be overfitted, the quality of the recommendations are reasonable but the system could definitely be fleshed out more. A large issue with the implementation of this system was that it struggled in dealing with a dataset as sparse as the MovieLens set. This is something that can be addressed with Model-Based Collaborative Filtering systems which will be looked at next.

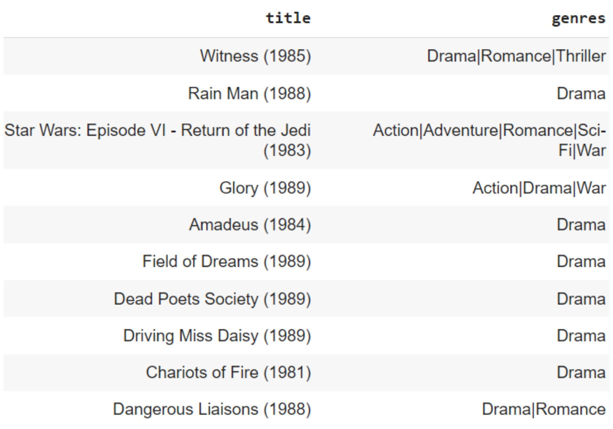
C. Model Based CF: Singular Value Decomposition

I chose to explore this next model based on the finding from the Memory-Based system examined earlier due to the issues that it had dealing with sparsity. Using a Matrix Factorization technique such as Singular Value Decomposition seemed like a good way to handle that obstacle and look at a more effective recommendation model. On the initial implementation of this model I chose to pick 50 for the rank approximation of Sigma (Namle), this was the recommended number to choose and I thought it best to stick with it. I also ran the system multiple times using different values to see how it would change the evaluation metrics.

The table below shows the top 10 rated movies for a specific user. Looking at the table we are able to see that the user tends to like a lot of drama movies.



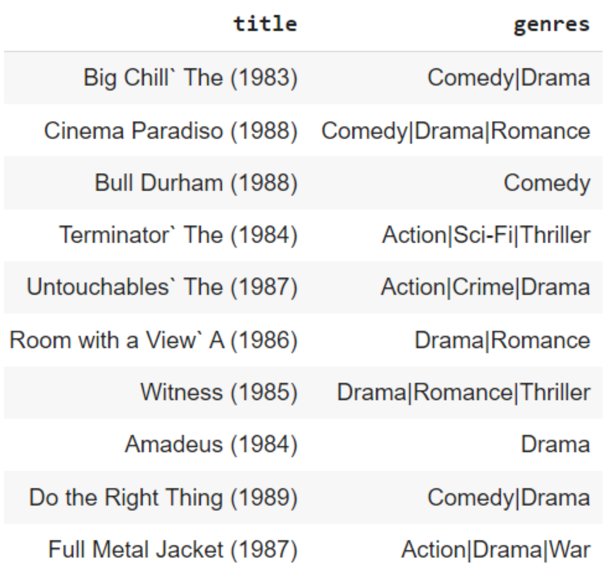
movie_id	rating	ts	title	genres
2248	5	974781573	Say Anything... (1989)	Comedy|Drama|Romance
2620	5	974781573	This Is My Father (1998)	Drama|Romance
3683	5	974781935	Blood Simple (1984)	Drama|Film-Noir
1704	5	974781573	Good Will Hunting (1997)	Drama
1293	5	974781839	Gandhi (1982)	Drama
3101	4	974781573	Fatal Attraction (1987)	Thriller
1343	4	974781534	Cape Fear (1991)	Thriller
2000	4	974781892	Lethal Weapon (1987)	Action|Comedy|Crime|Drama
3526	4	974781892	Parenthood (1989)	Comedy|Drama
3360	4	974781935	Hoosiers (1986)	Drama



title	genres
Witness (1985)	Drama|Romance|Thriller
Rain Man (1988)	Drama
Star Wars: Episode VI - Return of the Jedi (1983)	Action|Adventure|Romance|Sci-Fi|War
Glory (1989)	Action|Drama|War
Amadeus (1984)	Drama
Field of Dreams (1989)	Drama
Dead Poets Society (1989)	Drama
Driving Miss Daisy (1989)	Drama
Chariots of Fire (1981)	Drama
Dangerous Liaisons (1988)	Drama|Romance

Within this system it is important to note that when the prediction matrix is generated for the predicted ratings for users, the values outputted are in decimal form, meaning a user can have a predicted rating of 3.48. I found this unusual as users are only able to rate movies on an integer scale. The table above shows the top 10 predicted ratings for the same user as above. An interesting feature that we see here is that the system recommended quite a few drama movies, this exemplifies how these model based systems are able to pick up on patterns within the data. Genre similarity is a feature that is not explicitly implemented into the code at any point, and is a pattern that was picked up innately by the system.

Revisiting how the prediction matrix is generated in decimal form, I thought it would be interesting to see how the predictions would be affected if the predicted ratings were rounded off into whole numbers. We can see from the results below that by rounding off the predictions we get a much different looking set of recommendations. The types of movies that are recommended to the user encompass a much more robust selection of genres, and to me form a healthier list of recommendations.



title	genres
Big Chill` The (1983)	Comedy|Drama
Cinema Paradiso (1988)	Comedy|Drama|Romance
Bull Durham (1988)	Comedy
Terminator` The (1984)	Action|Sci-Fi|Thriller
Untouchables` The (1987)	Action|Crime|Drama
Room with a View` A (1986)	Drama|Romance
Witness (1985)	Drama|Romance|Thriller
Amadeus (1984)	Drama
Do the Right Thing (1989)	Comedy|Drama
Full Metal Jacket (1987)	Action|Drama|War

Within this system I decided to see what would happen when certain genres were removed from the data. My goal in this was to see how it would affect the recommendations, the sparsity level and the evaluations. I removed the Drama and Comedy genres separately as they were the two most popular genres in the data with Drama at 1600 and Comedy at 1100 of the 4000 total movies. The results of removing these genres was lackluster and provided no real change to the output, evaluation or sparsity level.

D. Evaluations

The primary means of evaluating recommendation systems is done with Root Mean Squared Error (Rendle et al., 2019), which is the average distance between the actual ratings and the predicted ratings. All of the RMSE values shown in this section were obtained using a 5-fold cross validation.

For the initial base model I got a RMSE of .8739, for the rounded prediction matrix the RMSE was .8734 and when removing Drama and Comedy, a RMSE of .8766 and .8710 were found respectively. Now while the changes made did not affect the RMSE that much, an evaluation of .8739 is actually a very good error rate for a recommendation system. This is due to a variety of factors such as the high data level sparsity and the fact these systems are trying to predict human preferences which is a near impossible task. When comparing this RMSE to that of the highest rated peer reviewed recommendation systems, we find that even the top systems are only able to achieve an RMSE of around .77. (Rendle et al,. 2019).

VI. 6.0 Conclusion

Recommendation systems are a vital part of our everyday lives and are incorporated into so many products and services that we use such as Netflix, Tik Tok, Amazon, Spotify etc.. For this reason it is important for these systems to be accurate. If the knowledge of what goes into creating recommendations for users was common known, we would be able to convince people to utilize these systems more effectively for themselves which would also give more accurate data that could build better recommendation systems.

When evaluating these systems side by side I found that Model-Based CF systems were able to perform at the highest level while also giving good recommendations. While these systems performed the best on an individual level, the real answer is to have a hybrid system which combines and blends Content-Based and CF filtering as they are able to compliment each other's weaknesses. (Ling et al., 2014)

I would like to give a special thanks to the Richter Fund for providing me with a grant so that I was able to have access to a wider variety of sources and information that I would not have had access to otherwise.

VII. Bibliography

Bhadani, S. (2021). *Biases in Recommendation Systems.* ACM digital library. *https* : //*dl.acm.org*/*doi*/*pdf*/10.1145/3460231.3473897

Chen, H.-H. (2017, October 2). *Weighted-SVD: Matrix Factorization with Weights on the Latent Factors.* Papers With Code. *https* : //*arxiv.org*/*pdf*/1710.00482*v*1.*pdf*

Cheng, T. (2019). *Product Recommendation System Design.* ACM digital library. *https* : //*dl.acm.org*/*doi*/*pdf*/10.1145/3357292.3357314

Davidson, J., Liebald, B., Lio, J., Nandy, P., & Vleet, T. *The YouTube Video Recommendation System.* (2010). ACM digital library. *https* : //*dl.acm.org*/*doi*/*pdf*/10.1145/1864708.1864770

Dziugaite, G., & Roy, D. (2015, December 15). *Neural Network Matrix Factorization.* Papers with Code. *https* : //*arxiv.org*/*pdf*/1511.06443*v*2.*pdf*

Grouplens, 2023. (2003). Retrieved March 24, 2023, from *https* : //*grouplens.org*/*datasets*/*movielens*/1*m*/.

Han, S., Lim, T., Long, S., Burgstaller, B., & Poon, J. (2021, August 27). *GLocal-K: Global and Local Kernels for Recommender Systems.* Papers with Code. *https* : //*arxiv.org*/*pdf*/2108.12184*v*1.*pdf*

Harper, M., & Konstan, J. (2015, December). *The MovieLens Datasets: History and Context.* ACM digital library. *https* : //*dl.acm.org*/*doi*/*epdf*/10.1145/2827872

Kang, W.-C., & McAuley, J. (2018, August 20). *Self-Attentive Sequential Recommendation.* Papers with Code. *https* : //*arxiv.org*/*pdf*/1808.09781*v*1.*pdf*

Krishnan, V., Narayanashetty, P., Nathan, M., Davies, R., & Konstan, J. (2008). *Who Predicts Better? - Results from an Online Study Comparing Humans and an Online Recommender System.* ACM digital library. *https* : //*dl.acm.org*/*doi*/*pdf*/10.1145/1454008.1454042

Ling, G., Ryu, M., & King, I. (2014). *Ratings Meet Reviews, a Combined Approach to Recommend.* ACM digital library. *https* : //*dl.acm.org*/*doi*/*pdf*/10.1145/2645710.2645728

Rendle, S., & Zhang, L. (2019, May 4). *On the Difficulty of Evaluating Baselines.* Papers with Code. *https* : //*arxiv.org*/*pdf*/1905.01395*v*1.*pdf*

Stein, S., Weiss, G., Chen, Y., & Leeds, D. (2020). *A College Major Recommendation System.* ACM digital library. *https* : //*dl.acm.org*/*doi*/*pdf*/10.1145/3383313.3418488

Takaes, G., Pilaszy, I., Nemeth, B., & Tikk, D. (2009, September 3). *Scalable Collaborative Filtering Approaches for Large Recommender Systems.* ACM digital library. *https* : //*dl.acm.org*/*doi*/*epdf*/10.5555/1577069.1577091

Varudkar, H. (2016). *Hadoop based collaborative recommendation system.* ACM digital library. *https* : //*dl.acm.org*/*doi*/*pdf*/10.1145/2905055.2905353

Zheng, Y., Tang, B., Ding, W., & Zhou, H. (2016, May 31). *A Neural Autoregressive Approach to Collaborative Filtering.* Papers with Code. *https* : //*arxiv.org*/*pdf*/1605.09477*v*1.*pdf*

A. Models Evaluated

Namle, K. (n.d.-a). *Movie Recommendation with Content-Based and Collaborative Filtering.* Jupyter Notebook Viewer. *https* : //*nbviewer.org*/*github*/*khanhnamle*1994/*movielens*/*blob*/*master*/*C*

Namle, K. (n.d.-b). *SVD for Movie Recommendations.* Jupyter Notebook Viewer. *https* : //*nbviewer.org*/*github*/*khanhnamle*1994/*movielens*/*blob*/*master*/*S*

B. Other Sources

Deep Learning, D. I. (n.d.). 21.1. *overview of recommender systems Colab* [*pytorch*] *open the notebook in colab colab* [*mxnet*] *open the notebook* *in colab colab* $[jax]$ *open the notebook in colab colab* $[tensorflow]$ *open the notebook in colab sagemaker studio lab open the notebook in SageMaker Studio Lab.* 21.1. Overview of Recommender Systems - Dive into Deep Learning 1.0.0-beta0 documentation. Retrieved April 27, 2023, from $https : //d2l.ai/chapter_recommender - systems/recsys - intro.html$

Deutschman, Z. (2023, April 19). *Recommender Systems: Machine Learning Metrics and business metrics.* neptune.ai. Retrieved April 27, 2023, from $https : //neptune.ai/blog/recommender - systems - metrics$

Editor, altexsoft. (2021, July 27). *Recommender systems: Behind the scenes of machine learning-based personalization.* AltexSoft. Retrieved April 27, 2023, from $https : //www.altexsoft.com/blog/recommender - system - personalization/$

Karagiannakos, S. (2021, May 20). *An introduction to recommendation systems: An overview of machine and Deep Learning Architectures.* AI Summer. Retrieved April 27, 2023, from $https : //theaisummer.com/recommendation - systems/$

Milankovich, M. (2017, July 5). *The cold start problem for Recommender Systems.* Medium. Retrieved April 27, 2023, from $https : //medium.com/@markmilankovich/the - cold - start - problem - for - recommender - systems - 89a76505a7$

P., V. (2022, May 20). *Recommendation Systems explained.* Medium. Retrieved April 27, 2023, from $https : //towardsdatascience.com/recommendation - systems - explained - a42fc60591ed$