Adam Gumieniak
Aayush Sheth

Latent Vector Based Movie Recommendation System

Introduction

Think back to the advent of the internet and how someone would have used it. One might have looked through the web searching far and wide for only a few results that they were looking for. Finding information was difficult because there was not much information available on the web. As time has progressed this lack of information has disappeared so much so that we are now faced with a new problem. Someone might not be able to find the information that they want because there is too much information to search through, there is an information overload. An information overload creates a problem where it is difficult to find the most relevant information that we want. Recommender systems are one solution to this issue and in the context of searching for movies it is a strong solution.

In this paper we outline how we built a recommendation system using a kNN collaborative method and using Singular Value Decomposition (SVD) method. We compare the differences between the SVD method to common recommender system approaches in how they generate their recommendation results.

Related Work

The ten results are obtained through using Jumanji as the movie input using the Classic Collaborative Method.

Classic Collaborative	Mean Rating
Casper	2.81
Stargate	3.38
The Nightmare Before Christmas	3.55
Home Alone	3.00
Beauty and the Beast	3.77
Aladdin	3.15
Jurassic Park	3.75
Mrs. Doubtfire	3.39
The Mask	3.19
The Lion King	3.94

Global Means

The global mean rating for the movies dataset is 3.50 while the mean of the results returned by the Classic Collaborative method is 3.39. Through the very marginal difference of 0.11 in rating, we can assume that the Classic Collaborative method would be a good method if we wanted to only provide recommendations to the user that do not have high nor low ratings.

Item Means

The movie Jumanji has a mean rating of 3.43 and with the mean rating of 3.39 through the Classic Collaborative method, it appears to be very good in determining movies that are similar in rating.

User Means

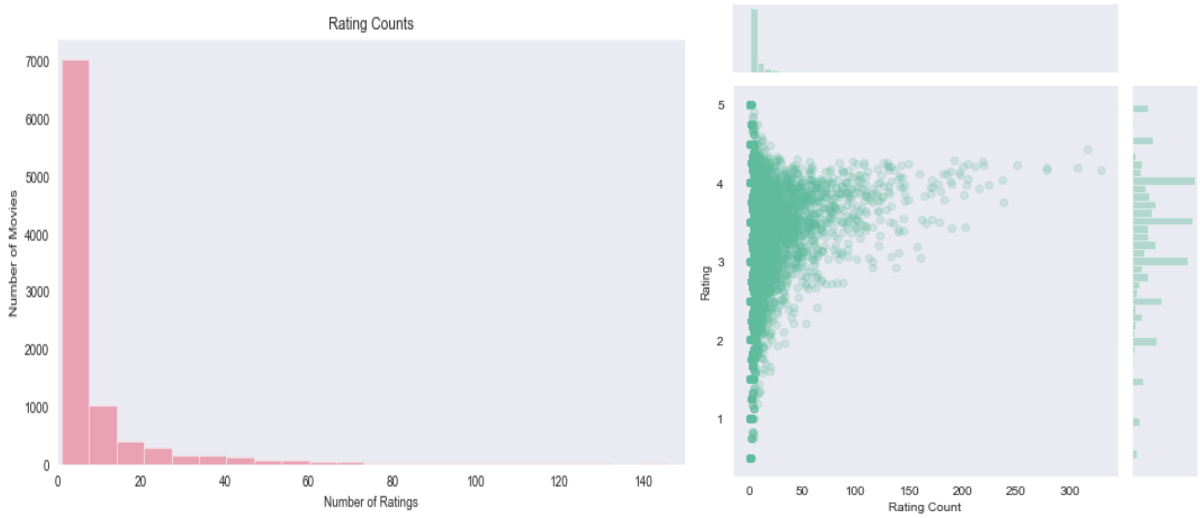
The User with the ID of 1 in the movies dataset has a mean rating of 4.36 and compared with the Classic Collaborative method using ratings as the benchmark, we can see that it would not provide very relevant recommendations to this user.

However, recommending to a user based on ratings alone is not enough to provide good recommendations. Thus, we also decided to use the SVD approach to provide more relevant results through similarity of content as well as ratings.

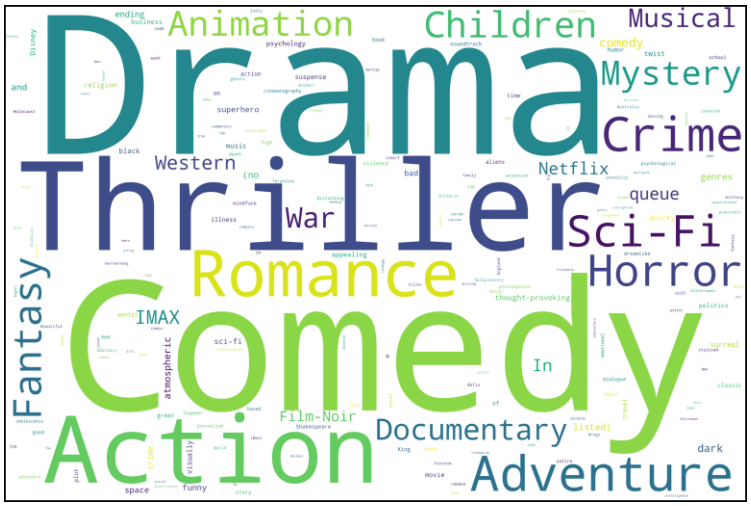
Methodology

Data Preprocessing

To make good recommendations to a user we must have an understanding of how our data looks like. Since the movies ratings will be a critical component in making our recommendations we wanted to see the number of ratings each movie had and also if the number of ratings plays a role in the actual rating of a movie.



In the chart on the left we find a large tail indicating that most movies do not have many ratings. We do not want to recommend movies with a low number of ratings since those movies have less meaningful data associated with them. To further support this, the chart on the right shows us that as a movie gets more ratings it tends to have a higher overall rating. Based on this we concluded to only consider movies with more than 50 ratings.



The adjacent word cloud helps visualize the data that is used to create our tf-idf vectors which are used for the SVD content based system. As we can see there are many terms which appear only once and a few which appear many times. User tags that appear a few times are interesting to us since different users described a movie in the same way. We capture this relation in the tf-idf vectors.

KNN Collaborative Based Approach

We toyed around with different methods such as global mean, user mean but ultimately chose the classical method of collaborative filtering. More specifically our first approach was to build a simple KNN user-movie rating based collaborative filtering system. We first filter our data and make a user-movie rating matrix but only for movies with greater than 50 ratings and only with users who have given more than 50 ratings to filter noisy data. Next the data is put into a scipy sparse matrix because this representation of data allows for faster and more efficient computation which is imperative for scalable datasets. Next we built our KNN model through SciPy using a brute force algorithm and the cosine similarity metric. The model is designed to take in the name of a movie and then recommend its 10 closest neighbors.

However, this is where we encountered our first problem. Movies within the dataset have movieId and title parameters to make it easier to index them. Unfortunately for us the naming conventions of many movies are inconsistent and grammatically incorrect as the dataset also uses the movie's release year within the title. For example, the movie titled "The Shawshank Redemption" shows up within the dataset as "Shawshank Redemption, The (1994)" for its title value. It is irrational to assume that everyone using the recommendation system would know the movieId or the dataset title convention for the movie they want recommendations based on. To overcome this problem, we used a python library called FuzzyWuzzy. It uses fuzzy matching to calculate the Levenshtein distance between strings. We take the user input movie name and use FuzzyWuzzy to find the five closest strings and pick the first one from the list which during testing was the correct movie 97% of the time. We then take this movie and use it as the centroid

for the KNN model and find its 10 closest neighbors which are then returned to the user as the movie recommendations.

SVD Content & Collaborative Based Approach

Although the KNN based system gave good results it could still be vastly improved upon. The first thing that could be improved upon would be the movie indexing within the dataset. We originally recommend movies based on the name and user movie rating matrix and nothing else. The string matching only gave correct results 97% of the time so that still gave us the possibility that incorrect movies were being recommended. The other problem is that KNN does not scale well for larger datasets and for datasets that have higher proportions of missing data. Our method to overcome these challenges was to implement a Matrix Factorization approach, more specifically a Singular Value Decomposition Matrix Factorization recommender because of their ability to work better with preprocessed data such as ours. With a SVD approach we are able to build a user movie profile for the user and compare it against other profiles which are generated during the matrix rank reduction step in the latent space. This is a more robust approach because we are able to recommend movies based on user preferences rather than just using a distancing approach. The important concepts here are the latent features which emerge during rank factorization which essentially, we break down our N items into K factors. For example, we may reduce from 25000 movies into 64 latent factors where each factor is a linear combination of movies that can be assigned to a user's profile. A factor can mean something such as "Action movies with Tom Cruise" and another might be "Movies that have a sequel". The key is that recommending based on factors is more reliable than just using movie similarities, maybe a user hasn't seen "The Hangover" but the user might have seen other movies that are related to "The

Hangover” via some other latent factors and those can be used in the recommendation process. These factors are always there in our data but they aren’t discovered until we start reducing which is when they start to emerge and hence the “latency”.

When applying SVD to the content-based approach we must represent our movies based upon some aspect of the movie content. To do this we created a metadata feature which combines a movie's genres and the user given tags of that movie. This is a good representation of a movie's contents because the genre is the main subject matter of a movie and the user generated tags give a more personal and unique description of the movie. These aspects are represented as a tf-idf vector which then is projected into latent space and we find similarity between movies using the cosine similarity metric. The similarity between movies depends on the latent factors that are found between tf-idf vectors and the more factors found between movies the greater possibility of finding very similar movies. Note that in our experiments we considered 400 factors which is large in comparison to our dataset however this is only feasible as our dataset is less than 3MB of data. As our dataset increases to much larger amounts, say 1 terabyte, then we would put greater consideration on the number of latent factors in respect to computation time. In this hypothetical scenario we would choose a number of latent factors that gives us good recommendation in respect to cosine similarity while also running within a reasonable amount of time.

Evaluation and Results

Qualitative Results for Jumanji

Classic Collaborative	SVD Collaborative
Casper	The Lion King
Stargate	The Mask
The Nightmare Before Christmas	Beauty and the beast
Home Alone	Home Alone
Beauty and the Beast	Mrs. DoubtFire
Aladdin	Aladdin
Jurassic Park	Jurassic Park
Mrs. Doubtfire	The Santa Claus
The Mask	The Nightmare Before Christmas
The Lion King	Casper

Given an input movie of Jumanji these are the recommended movies for both of our applied methods. Both recommendation systems return similar results but if we take a closer look we find an interesting difference. The SVD approach recommends movies with animal based titles higher. Based on this observation it seems that the SVD system found an animal factor between those movies which has a higher weight in latent space thus these movies are more similar to Jumanji. Whereas the classic collaborative only looked to what other similar users rated highly and recommended based on that. Since Jumanji is a movie revolving around adventure and animals, we can conclude that the SVD approach provides better recommendations when compared against Casper which doesn't have those similarities with the input movie.

SVD Cosine Similarity Result for Jumanji

Recommended Movie	Content	Collaborative
The Lion King	0.694639	0.040187
The Mask	0.665238	0.204113
Beauty and the Beast	0.649356	0.151191
Home Alone	0.635470	0.072924
Mrs. Doubtfire	0.632596	-0.002196
Aladdin	0.630469	0.102540
Jurassic Park	0.627272	0.081641
The Santa Claus	0.609470	0.131938
The Nightmare Before Christmas	0.605445	0.128918
Casper	0.596458	0.224936

To further show that our SVD system produces relevant results to the query movie we refer to the chart above. Evidently the content based tf-idf approach gave us worse results in comparison to the collaborative approach in terms of similarity. This is likely from the greater variance in the tf-idf vectors when represented in latent space which then makes movies less similar. Furthermore, the variance in comparing user movie profiles is much less and this produces a better cosine similarity result. The reduced variance of user movie profiles is due to the "wisdom of the crowd" notion where similar users are more likely to enjoy the same movies which is supported by our results.

Conclusion

Providing recommendations based upon user and content data is not a trivial task as we must examine the data we are working with. Through the use of data visualization we concluded

that users which have rated many movies and movies which have many ratings are great data points to base recommendations off of. Once our data points are selected we considered different approaches to make these recommendations which were a KNN collaborative approach and an SVD content and collaborative approach. From our experiments we get relevant recommendation results from a reasonably simple KNN collaborative approach which were then improved by implementing SVD content and collaborative approaches. Within the two SVD approaches we found that they produced impressive results with collaborative being the better of the two.