Movie Recommendation System Project

Petros Grammatikoy, Giannis Doumos

January 29, 2024

1 Introduction

Let's consider this: It's a Thursday night and after a very stressful and hard day of work, you want to see a movie. You start to consider what movie you want to see depening on your personal perfernces. You start looking in the internet for your favourite actors, directors and category of movie(comedy, drama ...) you even start texting your friends what good movies are they suggesting. After all this reaserch you realise, the time has passed and you need to sleep beacuase it's late. After all this, you end up having seen no movie! This is a problem that we all have faced, and some people (like us) wanted to solve it!

1.1 Common Features of Recommendation Systems

For sure one of the most common entertainments of today's era is undeniably movies, especially nowadays with platforms like Netflix, DisneyTV that provide a huge number of movies every month to their users, in their home simply and easily. A common feature of these platforms that we have all used at some point (whether we want it or not) is the recommended movie system. Surely, as soon as you finish a movie, it will give you about 10 movies that the platform suggests you to watch. This is very useful tool that can save us time, because we don't have to make our research or text any of our friends , simply the platoforms do this for us! But how do they do that? How does this whole system work and how do they know better than me what movies I like?

1.2 Collaborative and Content-Based Filtering

This project focuses on this topic and approaching it from different ways. In order to approach the recommendatio system of these platforms we used: Collaborative Filtering: This technique analyzes user behavior and preferences to identify patterns and similarities among users. It recommends content based on the viewing history and preferences of users with similar tastes. For the collaborative filtering we used K-Nearest Neighbours and SVD Matrix Multiplication methods. Also we use deep neural networks to extract complex patterns and relationships from vast amounts of user data. Deep learning models can understand intricate details in user behavior and provide more accurate predictions by considering a wide range of features. We used the Keras Library for Deep Learning.

Content-Based Filtering: This approach recommends content by comparing the attributes of items (movies or TV shows) with the user's profile. It takes into account the user's past preferences and suggests similar content based on features such as genres, directors, actors, and themes, names of the movies. For Content-Based Filtering we used Cosine Similarity.

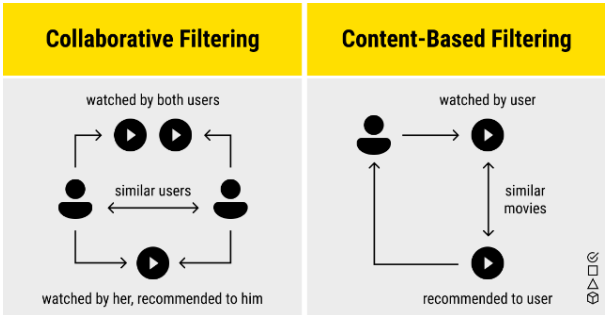


Figure 1: Collaborative and Content-Based Filtering

2 The Data set we use

The data set we will use contains 100836 ratings and 3683 tags in 9742 movies. This data was generated by 610 users between March 29, 1996 and September 24, 2018. This dataset was created on September 26, 2018.

Users were randomly selected for inclusion. All featured users had rated at least 20 movies. Demographics are not included. Each user is represented by an ID and no other information is provided. The data is contained in the 'movies.csv', 'ratings.csv' and 'tags.csv' files. Below are more details about the content and usage of all these files. It contains the Id of the movie, the name of the movie as well as the category of the movie (if there are many then the categories are separated by '—').

2.1 Movies

The categories are as follows: • Action
• Adventure
• Animation
• Children's
• Comedy
• Crime
• Documentary
• Drama
• Fantasy
• Film-Noir
• Horror
• Musical
• Mystery
• Romance
• Sci-Fi
• Thriller
• War
• Western
• (no genres listed)
Also this dataset has the titles

2.2 Ratings

It contains the user's Id, the movie's Id and the movie's rating from 0.5 to 5 stars. All the users has some ratings on movies. One rating for each pair of user and movie.

2.3 Tags

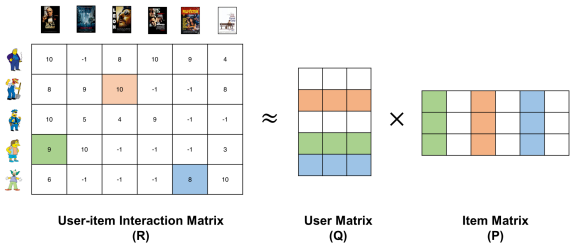
It contains the same fields as the rating (user ID and movie ID) and instead of a review it has a small comment about the movie. Some of the common comments are actors, directors, movie category and many more. Comments are 1-3 words each. The dataset does not need any kind of Scale, the ID are just int from 1-1000 and the tags are just descriptive words that we directly use them for the calculation of

cosine similarity analysis. An important catch at the data set is that the movies ID are not consequential from 1 to 10000 but there are some empty IDs like 33, 35. The data set looks like that:
31,Dangerous Minds (1995),Drama
32,Twelve Monkeys (a.k.a. 12 Monkeys) (1995),Mystery—Sci-Fi—Thriller
34,Babe (1995),Children—Drama
36,Dead Man Walking (1995),Crime—Drama
witch didn't affect or analysis after all. It would be a waste of time to transform it to a consequential countdown because we don't know IDs are missing and the users ratings depend of the IDs , so it would be risky to transform it. The only transform we did is that we know that some movies don't have enough ratings so we cut them out from our analysis.

3 Methods

3.0.1 Singular Value Decomposition (SVD) for Collaborative Filtering:

Singular Value Decomposition is a matrix factorization technique commonly used in collaborative filtering for recommendation systems. In the context of collaborative filtering, the user-item interaction matrix is decomposed into three matrices: a user matrix, a singular value matrix, and an item matrix. By representing users and items in a lower-dimensional space, SVD captures latent features that contribute to user preferences and item characteristics. This method allows for the reconstruction of the original matrix, providing predictions for user-item interactions. SVD has proven effective in capturing complex patterns and relationships in user behavior, making it a powerful tool for collaborative filtering-based recommendation systems.



3.0.2 K-Nearest Neighbors for Collaborative Filtering

K-Nearest Neighbors (KNN) is a simple yet effective algorithm for collaborative filtering in recommendation systems. In KNN collaborative filtering, users or items are treated as points in a high-dimensional space, and recommendations are made based on the preferences of similar users or items. The algorithm calculates the similarity between users or items using a chosen distance metric, such as cosine similarity or Euclidean distance. The "k" in KNN represents the number of nearest neighbors to consider. By identifying users with similar preferences, KNN recommends items that those users have liked but the target user has not interacted with. KNN is intuitive, easy to implement, and does not require training a model, making it a popular choice for collaborative filtering applications.

3.0.3 Cosine Similarity for Content-Based Filtering

Cosine Similarity is a widely used metric in content-based filtering for recommendation systems. In content-based filtering, items are represented as vectors based on their features, and the similarity between items is measured using the cosine of the angle between their feature vectors. This metric is particularly effective for handling high-dimensional data where the magnitude of the vectors is crucial. A higher cosine similarity indicates a smaller angle and, therefore, greater similarity between items. In content-based recommendation, cosine similarity helps identify items that are most similar to a given item, allowing for personalized recommendations based on the content features of the items. It is commonly used in applications where the characteristics of items play a crucial role in user preferences. The cosine similarity uses a complex formula that can identify the similarity between the comments(comments contain the cast, directors and many more useful words used by users to describe a movie) or even the names of the movies.

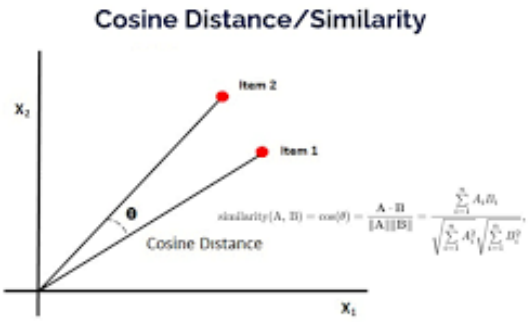
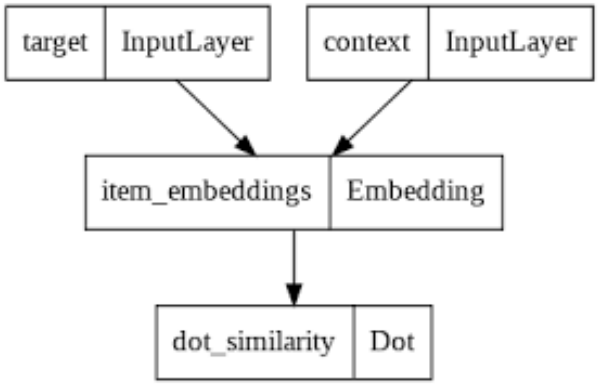


Figure 2: Enter Caption

3.0.4 Collaborative Filtering Neural Network using Keras

Collaborative filtering recommendation system using matrix factorization through neural embeddings. The model learns latent representations for movies and users and uses the dot product to predict ratings. The training process is visualized, and the model's predictions are evaluated on a test set. Adjustments to hyperparameters, architecture, and training duration can be made based on specific requirements and data characteristics.



4 Experiments/Results/Discussion

Before we start we needed to do some preprocessing on the data like removing the time stamp column and merging the tables with some SQL like operations so that we can handle and observe them better. We also removed some movies that had a really small number of ratings.

4.1 Evaluating The Movies

First we need to evaluate the movies and assign them a rating so us to be able to have a list that can be sorted. Of course the easiest way to do this is to compute the average rating of the movie. We also used a more sophisticated method to do this which is the official IMDb method which uses a weighted rating formula shown below:

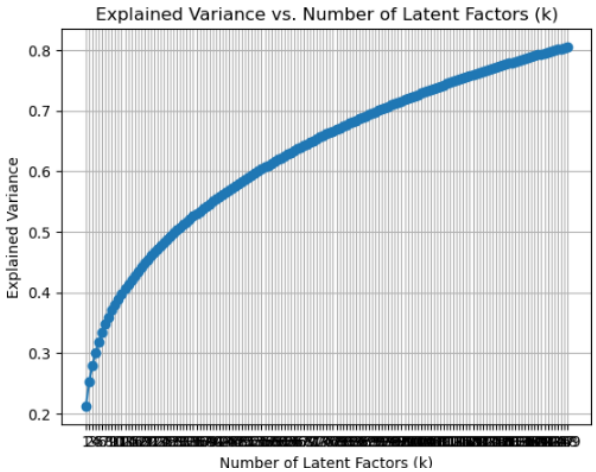
Weighted Rating (WR) $= (\frac{v}{v+m}.R) + (\frac{m}{v+m}.C)$

where: v is the number of votes for the movie ,m is the minimum votes required to be listed in the chart, R is the average rating of the movie, C is the mean vote across the whole report

In the end we decided that the better metric to rate the movies was the IMDb formula which seems to evaluate more movies with large amount of ratings cause the larger amount of people liking something is a more reliable source than a smaller one(this can be more easily observed when using the big versions of the data files, due to the lack of space and computational power we used the smaller one for most of the algorithms implemented). Just as simple as that. We also implemented a filter that can rate movies from single genre which is the simplest kind of content based recommendation, a more sophistacated version was also implemented and is descussed bellow.

4.2 Collaborative Filtering Using SVD Matrix Multiplication

The listing method is for sure a reliable one but it locks of personalization, every user asking for a movie of a single genre or not will get the exact same result as a recommendation. To capture the information of a user's taste in movies and recommend accordingly we used collaborating filtering with SVD. At first we processed our data to create a user movie Matrix which contained all the ratings every user has given to a movie while all the other not rated movies were filled either with 0 (could also be filled with the mean rating of all the novies) . In SVD the most desfining parameter is K, the number of singular values. To choose among all the values that could be chosen we used the elbow method, we plotted the variance captured from the SVD for a large amount of K values and we ended up with a diagram that showed a logarithm like trend. The results seem to be fine cause it does not seem to show any bias for very highly rated movies and the recommendations differ from one user to another!



4.3 Content Based Recommendations

Content based recommendation according to genre only was not enough! We needed to make recommendations according to the comments a user had given to the movies. Data processing was also needed here cause we had to combine all the tags the users had given to a movie to make a bin with all the description was given for every single movie. Then all the combined comments represented as strings were analyzed as vectors and using cosine similarity we were able too match movies with comments that were similar to a specific one. The results were great because we observed that the system could come to information not only like genre but also the cast and the director of the movie or characteristics like the plot , the protagonists of the movie or the time and space a movie was showing that existed in the comments and could not be accessed only by genre based recommender.
If a user has seen the Dark Knight, The (2008) we get recommedations:

- 89 302 Garden State (2004)

- 103 137 Batman (1989)
- 513 198 Batman Returns (1992)
- 522 248 Superman (1978)
- 249 Superman II (1980)

The Dark Khight is a superhero movie, so does these 5! More results in the notebook!

Figure 3: Superman and Batman

4.4 Deep Learning Model(or Neural Network) using the Keras Library:

We first set the nfactors at 30 this the number of our embedding layers, we tried also with 50 and 20 layers (the results wasn't that good), but we choose the 30 to prevent overfitting with 50 and to prevent underfitting with 20. We set the loss function as the mean squared error because it seemed to have more accurate results but this is didn't make that much of a difference, we would use other loss functions like absolute error as well. We set the epochs=7 we can see in this diagram :

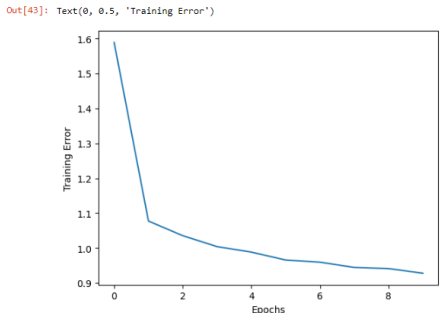


Figure 4: Diagram for Loss and Epochs

After the 6th epoch the loss is the same. The most critical part was the optimizer we used. We used Adam optimizer with Learnig Rate 0.01, the other optimizers didn't have much difference but a smaller (0.001) learning rate made the model overfit to the data, giving us the same movies for all users, and a bigger learning rate (0.1) made the model go straight to underfitting, giving us unexpected results The final results looks like this:
For user 7

- 89 Nick of Time (1995) Action—Thriller
- 103 Unforgettable (1996) Mystery—Sci-Fi—Thriller
- 513 Radioland Murders (1994) Comedy—Mystery—Romance

- 522 Romper Stomper (1992) Action—Drama
- 547 Surviving the Game (1994) Action—Adventure—Thriller

we can see the action and thriller movies are recommended.
For user 295

- 103 Unforgettable (1996) Mystery—Sci-Fi—Thriller
- 223 Clerks (1994) Comedy
- 351 Corrina, Corrina (1994) Comedy—Drama—Romance
- 476 Inkwell, The (1994) Comedy—Drama
- 542 Son in Law (1993) Comedy—Drama—Romance

we can see the comedy movies are recommended and maybe same drama.
For user 4

- 13 Balto (1995) Adventure—Animation—Children
- 325 National Lampoon's Senior Trip (1995) Comedy
- 356 Forrest Gump (1994) Comedy—Drama—Romance—War
- 382 Wolf (1994) Drama—Horror—Romance—Thriller
- 440 Dave (1993) Comedy—Romance

we can see the Romance movies are recommended. The model works perfectly

4.5 K-Nearest Neighbors

In this algorithm the implementation is provided in the sklearn library so the ONLY parametre we could change is the metric and the algorithm and the NearestNeighbors() function. We choose cosine ,but the default parameters also did show much different results. Also we chose the 'auto' argument for it to find the most appropriate algorithm based on the values passed to fit method.
For user 296

- 337 What's Eating Gilbert Grape (1993) Drama
- 430 Calendar Girl (1993) Comedy—Drama
- 432 City Slickers II: The Legend of Curly's Gold (... Adventure—Comedy—Western
- 441 Dazed and Confused (1993) Comedy
- 548 Terminal Velocity (1994) Action—Mystery—Thriller

Similar categories with the nn.

4.5.1 Hybrid KNN with High Rated Movies

With this implementations we combined the KNN with the Listing. We use a knn analysis just like the above example(with algorithm='kdtree') with the best movies in each category. The idea is to find what categories of movies the user likes and find the best movie from each category, not only the movies some users have in common but the movies that have very good ratings. For example: if the user 1 and 2 like War movies but the user 1 only saw 'bad' war, and now we want to recommend War movies to user 2, the Braveheart (witch have really good ratings overall 4.5¿) will not be recommended. The results are like this : For user 4

- 318 Shawshank Redemption, The (1994) Crime—Drama
- 356 Forrest Gump (1994) Comedy—Drama—Romance—War

- 110 Braveheart (1995) Action—Drama—War
- 527 Schindler's List (1993) Drama—War
- 480 Jurassic Park (1993) Action—Adventure—Sci-Fi—Thriller

We can see the preference to War and Drama movies (these movies are very high rated both in our system but also in official sites and magazines).We see that we have the same results with the NN.(the Forrest Gump is the same movie in both methods)

For user 295

- 356 Forrest Gump (1994) Comedy—Drama—Romance—War
- 1939 2571 Matrix, The (1999) Action—Sci-Fi—Thriller
- 593 Silence of the Lambs, The (1991) Crime—Horror—Thriller
- 260 Star Wars: Episode IV - A New Hope (1977) Action—Adventure—Sci-Fi
- 110 Braveheart (1995) Action—Drama—War

We can see the preference to drama and comedy movies (indeed these movies are very highly rated both in our system but also in official sites and magazines). The NN also spot drama and comedies movies as personal preference. We don't use any mse or loss function on this because we want to test it more practical way, we see that the movies are the ones with the highest rating like The Matrix (has 8.7/10 on IMDb). So this might be the best method so far by refering to the quality of the movies suggested for a specific user.

All of the above were studied, analysed and implemented with code on the Jupyter Notebook Files.

5 Conclusion/Future Work

In the end of the day making a recommendation system for movies is for sure not an easy task. We should always have in mind that people's preferences are very complex and hard to analyze and categorize , even more when talking about movies where the only limit is the imagination of the creator .This is the reason why we approached this problem using many different strategies and algorithms. Some of them show an appeal to diversity and do not care that much about the rating while others show an appeal to more highly rates ones witch is always the safest option. After all this journey we have surely learned a lot about recommendation systems , machine learning algorithms and theory too, people's preferences but also movies that we deeply love. This is only the start for us and we really look forward to take part in bigger and better ML projects with more resources and like minded and passionate about AI (or movies) people with us.

PETROS GRAMMATIKOU, GIANNIS DOUMOS

6 Links To References

- KNN library: `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassi`
- Kaggle - How to build a rec-system:
  - `https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109`
- Keras library: `https://keras.io/api/models/model_training_apis/`
- MovieLens dataset: `https://grouplens.org/datasets/movielens/`
- Cosine Similarity from sklearn: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pai`
- The website of our curse provided good material: https://piazza.com/class/llug8fzp1x63qa
- SVD: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html
- Sklearn: https://scikit-learn.org/stable/