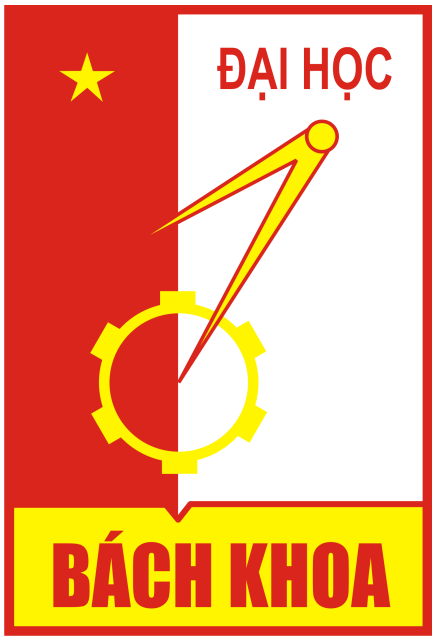
HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY
FACULTY OF COMPUTER SCIENCE

o0o



PROJECT REPORT:
MOVIE RECOMMENDATION SYSTEM

Course: Project 1 - IT3910E
Class Code: 733506
Supervisor: Associate Prof Le Thanh Huong
Student: Vu Lam Anh - 20214876

Ha Noi 2024

Contents

1 Introduction to Movie Recommendation System

1.1 Short description

A movie recommendation system, or a movie recommender system, is an ML-based approach to filtering or predicting the users' film preferences based on their past choices and behavior. It's an advanced filtration mechanism that predicts the possible movie choices of the concerned user and their preferences toward a domain-specific item (movie).

1.2 Input and output

There are two main elements in every recommendation system: users and items. In this case, the system generates movie predictions for its users, while items are the movies themselves. The movie recommendation system takes input is the data about the user's past choices and also the features of the movie, then it generates the output is the filtration or prediction for the user's film preferences.

2 Datasets

2.1 Introduction to Datasets

In this project, I use the datasets **The Movies Dataset**[1] for training and testing below model. This dataset has containing 26 million ratings from 270,000 users for all 45,000 movies, ratings are on a scale of 1-5 and have been obtained from the official GroupLens website. Moreover, it also has metadata for all 45,000 movies released on or before July 2017 including cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

Below are the all the files I use in this datasets for the project:

1. **movies_metadata.csv**: The main Movies Metadata file contains information on 45,000 movies features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
2. **ratings.csv**: Containing 26 million ratings from 270,000 users for all 45,000 movies.
3. **ratings_small.csv**: The subset of 100,000 ratings from 700 users on 9,000 movies.

2.2 Data Pre-Processing

Since in the content-based filtering approach, we need the good item profiles for movie, I clean and pre-process the files **movies_metadata.csv** and also tokenize some features for using in the Word2Vec approach discussed below.

Firstly, I decide to build item profiles with three features: title, genres and overview so in the file movies_metadata.csv i only keep 4 columns containing id of movie and three features above while droping the others. Secondly, I clean the the datasets by removing the Null value and also the records with error attribute such as: it has no genres or no overview. Following that, I transforms the genres of each movie from the original format in datasets into format which is the list contains all the genres of the movie. Here is the metadata of movie after the transformation:

	id	title	genres	overview
0	862	toy story	[family, comedy, animation]	led by woody, andy's toys live happily in his ...
1	8844	jumanji	[family, adventure, fantasy]	when siblings judy and peter discover an encha...
2	15602	grumpier old men	[comedy, romance]	a family wedding reignites the ancient feud be...
3	31357	waiting to exhale	[romance, comedy, drama]	cheated on, mistreated and stepped on, the wom...
4	11862	father of the bride part ii	[comedy]	just when george banks has recovered from his ...
...	...	...	...	...
45459	222848	caged heat 3000	[science fiction]	it's the year 3000 ad. the world's most danger...
45460	30840	robin hood	[drama, romance, action]	yet another version of the classic epic, with ...
45461	439050	subdue	[family, drama]	rising and falling between a man and woman.
45462	111109	century of birthing	[drama]	an artist struggles to finish his work while a...
45463	67758	betrayal	[drama, thriller, action]	when one of her hits goes wrong, a professiona...

42143 rows × 4 columns

Figure 1: The metadata of movie after transforming

From the movie data above, I can start with TF-IDF approach and K-Means approach. But to implement with Word2Vec approach, I need the tokenization step. In this step, I tokenize the features title and overview so that from the string format I will have the list of title and overview token. Follow by that, I also remove some tokens which are not necessary such as: regular english word and the punctuation.

After processing the data about movie, I process data about ratings. With content-based approach, I keep only records with rating larger than three, which means I keep only movie the user like. With matrix factorization approach, I normalize the ratings by subtracting the ratings with average of all the ratings of each user, from that I can remove some bias where the hard user only give very low ratings or the easy user only give high ratings.

3 Content-Based Filtering

3.1 Introduction to Content-Based Filtering

Content-based filtering is a type of recommender system that attempts to guess what a user may like based on that user's activity. Content-based filtering makes recommendations by using keywords and attributes assigned to objects in a database and matching them to a user profile. In this project, I define the user profiles by specifying the nearest movie that user loves, then I use some approachs below to recommend movies which are the most similar to that movie. Moreover, to evaluate the quality of the recommendation, I will use the recommendation above and the actual liked movie of the user(except the nearest one).

3.2 TF-IDF Approach

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. Here is the formula to compute TF-IDF for word x in document y:

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Figure 2: TF-IDF formula

In this project, I use only the overview features to compare one film with other film. I consider the overview of each movie as the document then I can compute TF-IDF formula above to compute relevant each word in the overview.From that, I can build TF-IDF matrix where each row is the vector represent the relevant of words in one overview. Therefore, I can compare them together and get the list of movie which is similar to one movie.

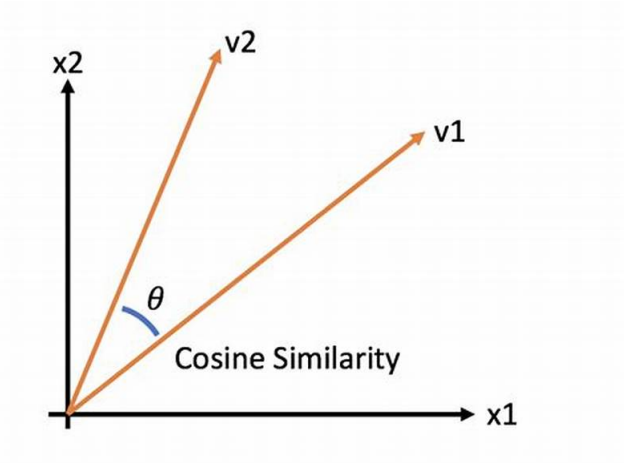


Figure 3: TF-IDF formula

To evaluate how similarity of each vector to other vector, I use the cosine similarity measurement. The range of value of cosine similarity measurement go from -1 to 1. If the value is closer to 1, it means that two vector seems similar to each other. Otherwise, if the value is closer to -1, it means that two vector are different. Below is the formula for the cosine similarity between two vector u_i and u_j:

$$sim(u_i, u_j) = \frac{u_i \cdot u_j}{\|u_i\|\|u_j\|} \tag{1}$$

From the TF-IDF matrix and the cosine similarity measurement, I can recommend the list of movies which are the most similar to the movie that user likes.

3.3 K-Means Approach

In this approach, I use TF-IDF matrix where each row is the vector represent the relevant of words in one overview above. But, instead of measuring the similarity between vector, I use those vectors from the matrix to classify each vector(or can say movie) into the clusters. So when I want to recommend movie for user, I will recommend the movies which are in the same cluster with the movie that the users likes.

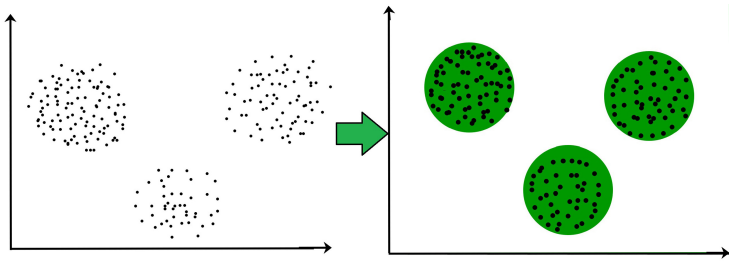


Figure 4: Clustering

3.4 Word2Vec Approach

Word2Vec is a widely used method in natural language processing (NLP) that allows words to be represented as vectors in a continuous vector space. Word2Vec is an effort to map words to high-dimensional vectors to capture the semantic relationships between words, developed by researchers at Google. Words with similar meanings should have similar vector representations.

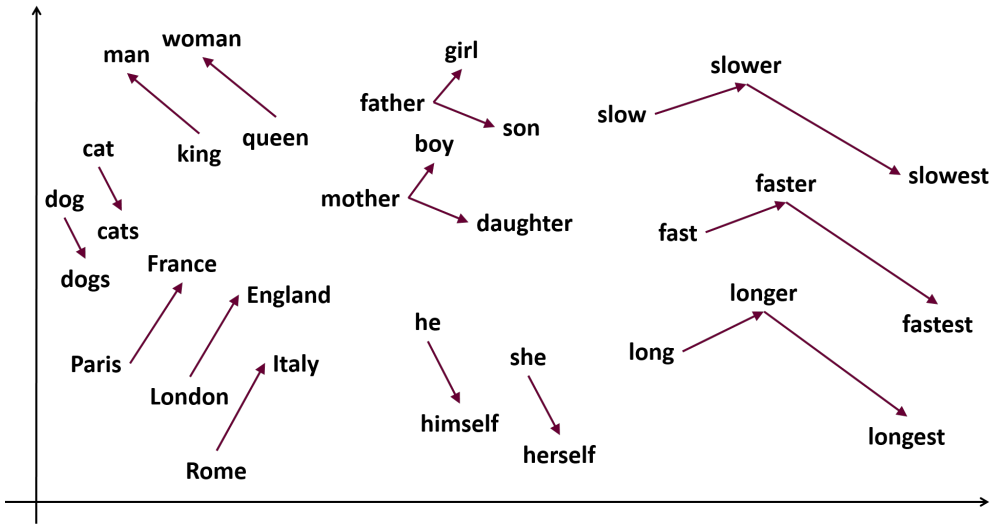


Figure 5: Word2Vec continuous vector space example

After the data pre-processing above, we have the profile of movie data which has the list of tokens of three features: title, overview and genres. Therefore, based on the Word2Vec model, I can represent those tokens as vectors in a Word2Vec continuous vector space, then compute the similarity between each vector by using the cosine similarity measurement as I discuss above. In the specific, I will compute the cosine similarity of each features between movies, then sum it together and ranks the how similarity of each movie to others by ranking the summation. If the summation in some cases equals, then I will continue compare the cosine similarity of features in order: genre, overview and title. Based on that, I can recommend the list of movies which are the most similar to the movie that user likes.

4 Matrix factorization Singular Value Decomposition

4.1 Introduction to MF Collaborative Filtering

In the real life, the preferences of each users has the connections to the preferences of other users through features. Similarly, some movies have identical feature with other movies. But the problem is we just identify the limited feature and a lot of other connections we don't actually know. The matrix factorization help us to deal with it very well.

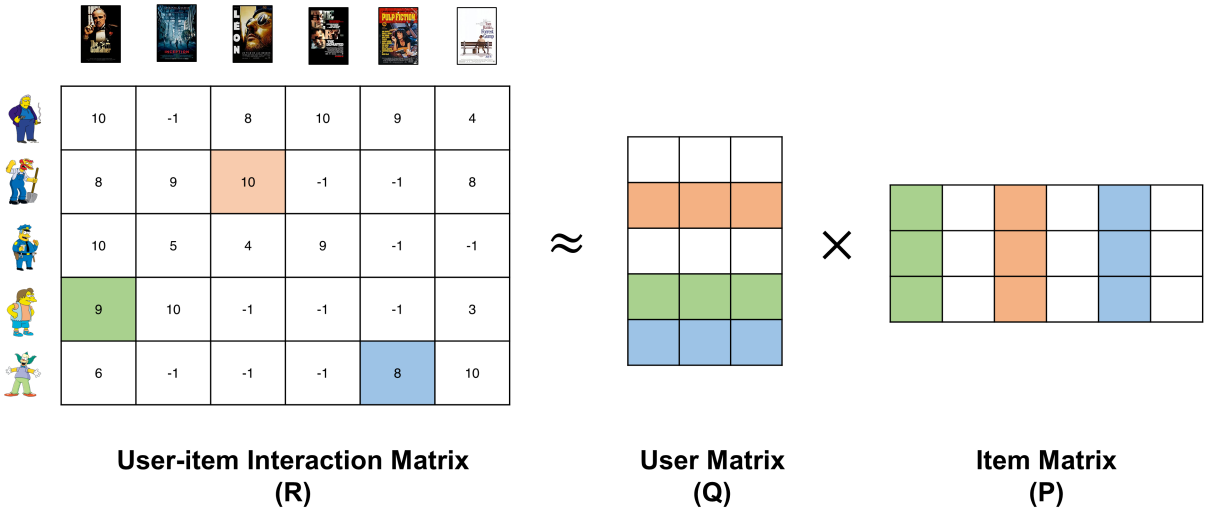


Figure 6: Matrix Factorization

The main idea behind the matrix factorization is representing users and items in lower dimensional latent space. That means we expressed the original matrix into 2 lower dimensional matrix is Users matrix which will tell us the connections between users and Items matrix which tell us the connections between movies through K latent features. We call it the latent features since we don't actually know what exactly K features is, we only know that those will help us to express the relationship between Users and between Movies. Therefore, matrix factorization will identify the connections between users and between movies based on known ratings and use it to figure out the unknown ratings, then make a recommendation to user. The high coefficient corresponds to one latent feature in both item and user matrix will also be high in the user-item matrix. That means the item has latent feature which the user likes, it will be suggested to the user.

4.2 Matrix factorization with Truncated Singular Value Decomposition

There are many variations of the matrix factorization implementation. In this project, I use Matrix factorization with Truncated Singular Value Decomposition. The singular value decomposition of a matrix A is the factorization of A into the product of three matrices:

$$A = U\Sigma V^T$$

where the columns of U and V are orthonormal and the matrix Σ is diagonal with positive real entries whose i^{th} diagonal entry equals the i^{th} singular value σ_i for i = 1, . . . , r and all other entries of Σ are zero.

Truncated SVD is the lower-rank approximation. That means we use low-rank matrix to approximate the original matrix. In the diagonal matrix Σ the singular values σ_i in the diagonal is non-negative and decreasing: $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq ... \geq \sigma_r \geq 0$, show how important the various columns of U and rows of V are. Therefore, U_1 is more important than U_2 and U_2 is more important than $U_3, ...$ Similar to V_1, $V_2, ...$

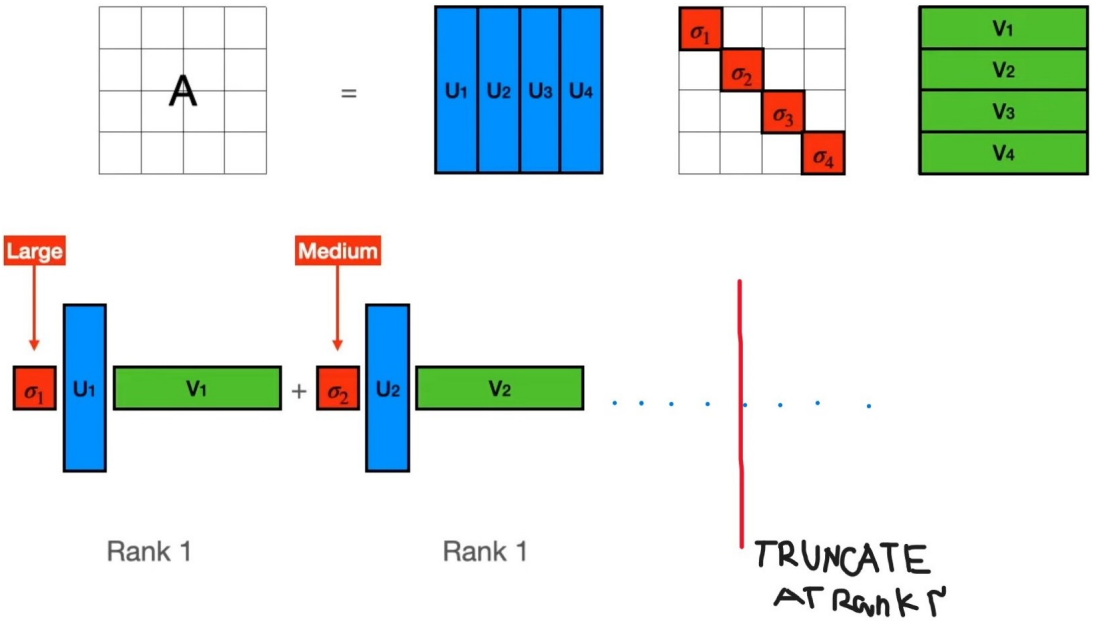


Figure 7: Truncated SVD

The word "important" here express which amount of values those columns or rows contribute to the original matrix. To be clearly, we can see the picture above. We can express the original matrix as the sum of many rank-1 matrices and the more to the right side, value of sigma is decreasing so it contribute value less to the original matrix.And maybe some last matrices, value of $\sigma \approx 0$. Now, truncated is applied, truncated is the action that we only keep the most important $r(r < rank(A))$ matrices or r matrices are left to right and all matrices after r we discard.So, from that, by using lower-rank matrix, we almost still have important information about the original matrix.

From the original ratings matrix, we use the truncated SVD to generate the low-rank r matrix which still have almost information of original matrix, then use it make the recommendation.

5 Experiment and result

5.1 Evaluation Strategy

With content-based approach, as I discuss above that I will define specify the nearest movie that user loves, then I recommend movies which are the most similar to that movie and to evaluate the quality of the recommendation, I will compute how many portion of the recommendation actually matches the actual liked movie of the user(except the nearest one).

With Matrix Factorization SVD, I will split the ratings data into training datasets which is 66% of ratings data and testing datasets is the remaining. From the training datasets, I will build the approximate user-movie matrix then from that matrix, I can compute the ratings prediction of user for movie in the testing datasets. Based on that, I will compute the Mean Square Error of ratings prediction with actual prediction. Moreover, to evaluate the quality recommendation and also the quality of ranking, I try to recommend with two way: One is no order and the other is follow descending order of rating. Then I will compute Precision@K, Recall@K, F1@K, MAP@K and nDCG@K for each way.

5.2 Content-based approach

From the evaluation strategy above, we compute for each user then takes the average over all user. Below is the result I achieve:

Approach	Precision
TF-IDF	0.00095
K-Means	0.00084
Word2Vec	0.00834

We can see that although Word2Vec approach has the better result with two others. But, in overall the result of three approachs in content-based filtering is too low. It suggest that the content-based approach is not suitable for this datasets or maybe in general, is not suitable for the movie recommendation.

5.3 Matrix Factorization Singular Value Decomposition

Firstly, I try to compute the Root Mean Square Error (RMSE) with from range value of rank and try to choose the number of rank that give us the lowest RMSE.

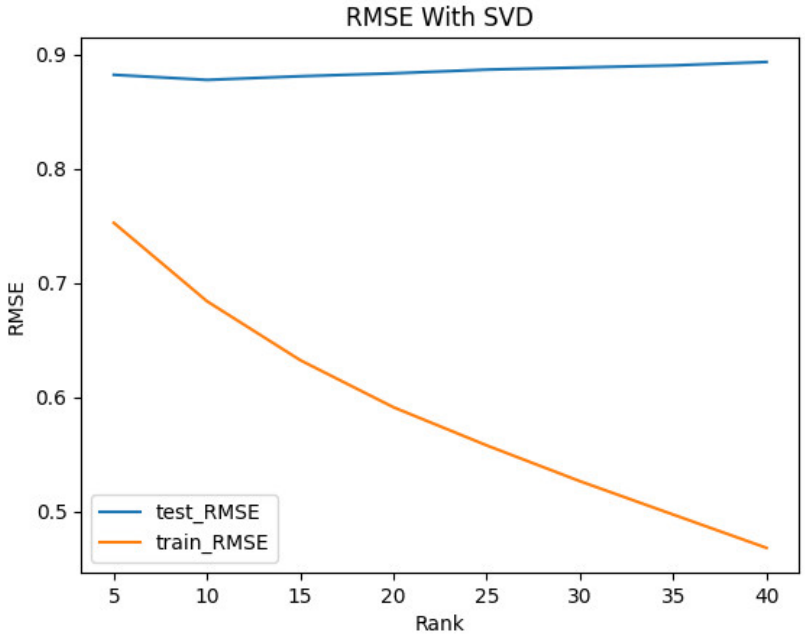


Figure 8: RMSE with each rank

I find that with rank equals 10, then the model give us the lowest RMSE approximates 0.878. After that, I compute Precision@K, Recall@K, F1@K with two ways: no order and descending order of rating.

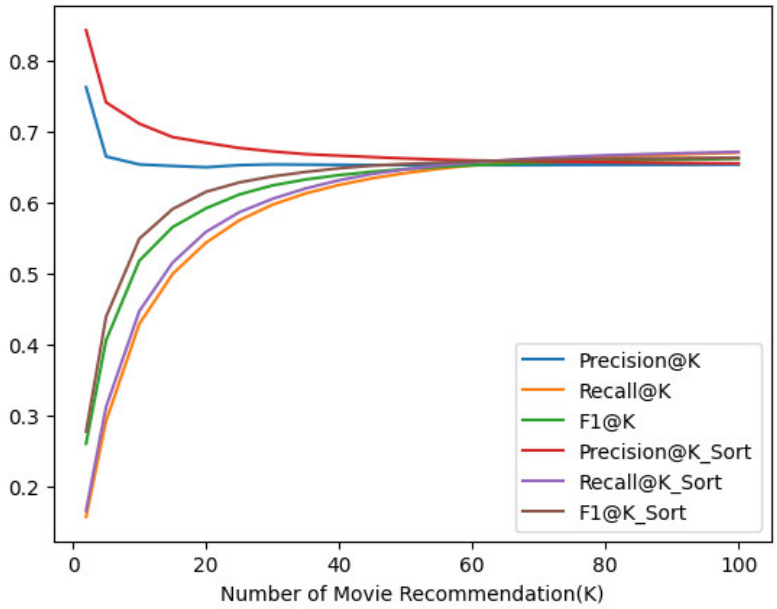


Figure 9: Precision@K, Recall@K, F1@K with each K

From the graph above, with matrix factorization approach, we have the precision score much better than the content-based approach and also we have quite high F1-score. Moreover, we can see that recommend with descending order of rating give us the measurement slightly

better than no order.

To evaluate the quality of recommendation ranking, I compute MAP@K and nDCG@K with two ways: no order and descending order of rating.

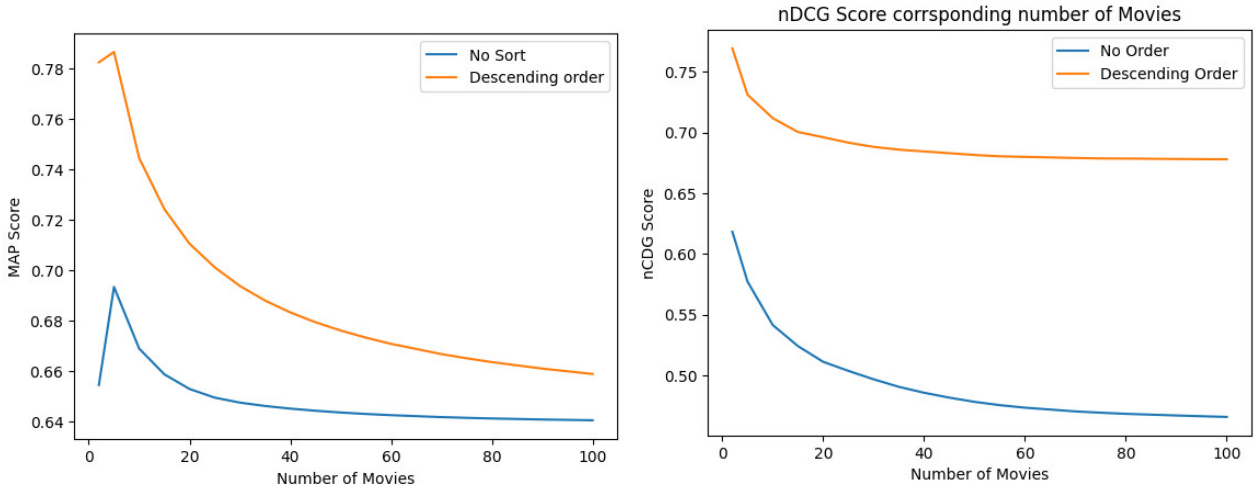


Figure 10: MAP@K and nDCG@K with two ways: no order and descending order of rating

We easily see that recommend with the way: descending order of rating is give us better results than when we recommend with no order

6 Conclusion and Future work

The content-based approach actually give us the similar movie but the problem is in this datasets I observe, it seems that user doesn't like the movie which are similar to each other so the evaluation of the Content-based is very bad. Alternatively, Matrix factorization SVD perform better so much. Moreover in this approach, from the evaluation above, we can see that we should recommend movie based on the descending order of rating since it give us the better results in both quality recommendation and also the quality of ranking.Throughout this project, we were able to have a better insight on various kinds of approach to recommendation and specifically, movie recommendation that it seems like content-based approach is not a good option in movie recommendation, but in stead, collaborative Matrix factorization give us the much better result.

There are also the problem in Matrix factorization SVD is that I can't do this method with large datasets. Therefore in future, I want to apply some techniques to solve this problem. Moreover, I can work on others approaches like Non-negative Matrix Factorization, Fast Incremental Matrix Factorization, Neural Collaborative Filtering, etc to obtain better performance.

7 Reference

References

[1] Kaggle: The Movies Dataset

[2] Kaggle: Recommender Systems in Python 101

[3] Machine Learning co ban: Matrix Factorization Collaborative Filtering

[4] neptune.ai: Recommender Systems: Machine Learning Metrics and Business Metrics

[5] Evidently AI: Precision and recall at K in ranking and recommendations

[6] Evidently AI: Normalized Discounted Cumulative Gain (NDCG) explained

8 Appendix

8.1 RMSE

Root Mean Square Error (RMSE): RMSE is a common metric in prediction and recommendation problems, used to measure the average error between the actual value and the predicted value. In this instance, the RMSE is determined by comparing the actual rating to the rating predicted by the model. The following are the steps to compute RMSE in the movie recommendation problem:

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}(x_i - \hat{x}_i)^2}{N}} \tag{2}$$

where N is the number of data observed, x_i is the actual value, $\hat{x_i}$ is the predicted value.

8.2 Precision,Recall and F1

Precision helps evaluate the movie recommendation model's ability to recommend movies that are quality and tailored to the user's personal preferences. Precision focuses on ensuring that the movies recommended to the user are precise and relevant.

The formula for calculating Precision is:

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

Recall: Recall is a crucial measurement for assessing how well a model can identify all the appropriate movies for the user. Recall concentrates on finding as many appropriate movies as possible for users, avoiding missing important movies. To calculate the Recall, we need to know the following values: True Positive (TP): Number of movies suggested by the model and actually relevant to the user. False Negative (FN): Number of movies that match the user but are not suggested by the model.

The formula for calculating Recall is:

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

Where:

1. True Positive (TP): Number of movies suggested by the model and actually relevant to the user.
2. False Positive (FP): Number of movies suggested by the model but not suitable for the user.
3. False Negative (FN): Number of movies that match the user but are not suggested by the model.

F1-score: F1-score combines Precision and Recall into a single statistic to assess how well model accuracy and coverage are balanced. The F1-score aids in evaluating the movie recommendation model's overall effectiveness, ensuring that the model achieves high accuracy and coverage.

The formula for calculating F1-score is:

$$F1_score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5}$$

However, the use of F1-score should note that it is a composite metric and only gives an overview of performance. We need to consider the need to combine Precision and Recall individually to get a more detailed view of the model's performance.

8.3 Normalized Discounted Cumulative Gain (NDCG)

The NDCG evaluates the quality and priority of the movie recommendations given by the model. NDCG measures the priority of suggested movies by considering the distribution and position of movies in the suggested list relative to the actual rating of the user.

The NDCG calculation formula will include the following steps: Calculate Discounted Cumulative Gain (DCG): The premise of DCG is that highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result.

The traditional formula of DCG accumulated at a particular rank position r is defined as:

$$DCG_r = \sum_{i=1}^{r} \frac{rel_i}{log_2(i+1)} \tag{6}$$

where rel_i is the actual rating of the movie at position i in the recommended list.

To stronger emphasis on retrieving relevant documents, we utilize an alternative formulation of DCG:

$$DCG_r = \sum_{i=1}^{r} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{7}$$

Calculate Ideal Discounted Cumulative Gain (IDCG): IDCG is the ideal DCG value calculated from a list of actual ratings sorted in descending order of rating. The IDCG is the maximum DCG value that can be obtained for an evaluation list.

Calculate Normalized Discounted Cumulative Gain (NDCG): To make DCGs directly comparable between users, we need to normalize them. We divide the raw DCG by this ideal DCG to get NDCG, a number between 0 and 1.

$$NDCG = \frac{DCG}{IDCG} \tag{8}$$

8.4 Mean Average Precision (MAP)

Precision@K (P@K): Normal Precision metric does not consider the rank of the recommendations. However, Precision@K calculate the precision for the top K recommended movies:

$$P@K = \frac{\#\text{Relevant items in top K recommendations}}{\#\text{Items in top K recommendations}} \tag{9}$$

Average Precision@K (AP@K): The Average Precision@K is the sum of Precision@K where the item at the k_{th} rank is relevant divided by the total number of relevant items (r) in the top K recommendations

$$AP@K = \frac{1}{r}\sum_{k=1}^{K} P@K \cdot rel(k) \tag{10}$$

$$rel(i) = \begin{cases} 1, & \text{if items at } k_{th} \text{ rank is relevant} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

Average Precision@K is higher if the most relevant recommendations are on the first ranks. Hence, AP@K shows the goodness of the top recommendations.

Mean Average Precision@K (MAP@K): MAP@K calculate the mean value of AP@K for all users.

$$MAP@K = \frac{1}{M}\sum_{i=1}^{M} AP@K_i \tag{12}$$

where M is the number of users

MAP@K not only provides insights if your recommendations are relevant but also considers the rank of your correct predictions.