Web Science 2023: Final Project

Taraburca Radu - whx862

Abstract

This document contains the discussions and answers for the final project.

1 Week 6

I calculated the distribution of user and film ratings, as well as average user rating and film. Also, I calculated the first 5 most popular elements based on the number of evaluations. Besides this, I also made the graphs of the distributions of user ratings and movie ratings, as well as the chart of average rating for each film.
Based on the data observed after performing statistics on items and users, we can deduce the following:
A first observation that we can easily deduce from charts, but also from statistics is that the data set is quite scatter. On average one user has rated 80 films, but we also have a standard 78.8 deviation, which indicate a great variation in the number of films that each user has evaluated. The minimum number of films evaluated by a user is 4 and the maximum is 635, indicating that some users have specific tastes, while others are more versatile and evaluate more films.
Another observation we can make is that some movies are rated more than others. On average a film has 51 ratings with a standard deviation of 66.9, indicating that the number of ratings on film varies a lot. The minimum number of ratings of a film is 1, and the maximum is 484. From the top 5 the most rated films that have at least 390 ratings, we can see that some films are more popular than others. From here comes the problem of creating a recommendation system that is not a biased. The problem with biased recommendation systems is that they tend to recommend popular items, in this case movies. Thus, the already popular movies will be seen by users and will receive more reviews, thus becoming more popular, and the other films will be recommended less often. Therefore, such a system will not be able to offer users diverse and personalized recommendations.
Another easy thing to see is that on average a movie has a rating of 3.12, and the rating range has as a minimum of 1 and maximum 5. The standard deviation of 0.7318 indicates that there is some variability in the movie ratings, some movies being evaluated much higher or lower than the average. From here we deduce that on average the films are rated above average. This is not necessarily a positive thing, because we can assume that users tend to overate movies.

Based on the calculated statistics, the data set has some important properties that should be taken into account during the evaluation.

First of all, because it is such a big difference between the number of users ratings, the minimum being 4 and the maximum of 635, the system could be biased with the more active users' preferences. Secondly, because it is a big difference between the number of film ratings, the minimum number of ratings of a movie being 1 and the maximum of 484, the recommendations could be a biased towards the more popular movies.
Lastly, due to the average rating per movie of 3.12 out of 5, there is the problem of over-rating the movies by users. This could impact the accuracy of the recommendation model. By over-rating, the accuracy of the recommendation system decreases because the data is distorted, and in reality, what the recommendation system can recommend as suitable for the user, may not be exactly to his liking. Due to the over-rating of the movies, the system will start recommending movies that are not exactly the best choice for the user, and thus a mismatch occurs between the recommended movies and the user's preferences.

Parameter	Value
count	**943**

mean	**80.003**
std	**78.797**
min	**4**
25%	**26**
50%	**49**
75%	**109**
Max	**635**

Table 1: Number of ratings per user.

Parameter	**Value**
count	**1457**
mean	**51.779**
std	**66.882**
min	**1**
25%	**7**
50%	**26**
75%	**169**
max	**484**
Average rating of all movies	**3.124**

Table 2: Number of ratings per movie

Parameter	**Value**
count	**1457**
mean	**3.124**
std	**0.731**
min	**1**
25%	**2.711**
50%	**3.20**
75%	**3.661**
max	**5**

Table 3: Number of average ratings per movie

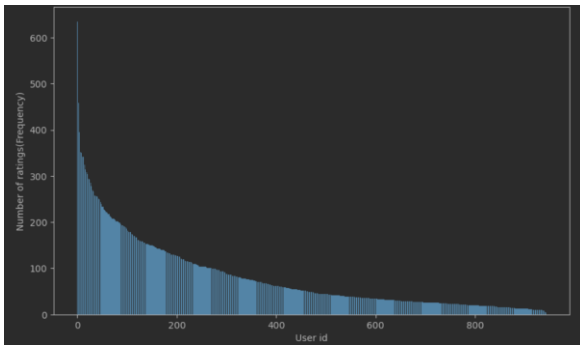

Figure 1: Number of ratings per user.

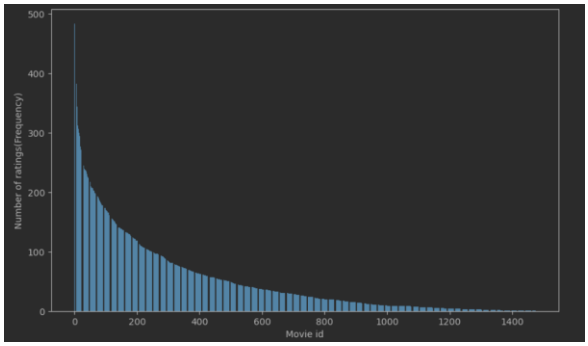

Figure 1: Number of ratings per user.

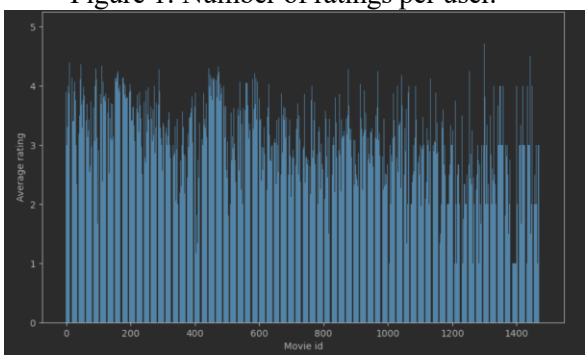

Figure 3: Average rating per movie.

2 Week 7

Since the chosen data set has a small number of movies but also a relatively small number of people, any neighborhood-based model is good to be used, but if we consider the previously mentioned observations, we can see that in this case it is better to use a user-based model. TEXT The item-based approach is more suitable for datasets where items have specific characteristics that users value, such as product features or attributes. For the "Movielens" dataset, movies are rated overall, with a rating from 1 to 5, which makes the user-based approach better in this case. Besides this, the data set contains users with different preferences, so using this method, we can make a recommendation system that provides interesting recommendations for users and thus we can minimize the bias towards popular movies.

Besides that, another factor that made me choose this model instead of the other one is the large variation in the number of movies that each user has rated.

For the latent factor model, I chose to use the SVD, because the data set contains sparse and noisy data, and this model can deal with these problems.

I chose the RMSE (Root Mean Squared Error) as the assessment metric because it is a simple and

straightforward tool for assessing the accuracy of the predicted ratings. I chose it because it is one of the most commonly used metrics for evaluating accuracy, but also because it penalizes larger errors more heavily. Thus, this measure is suitable to measure the accuracy of the predicted ratings. Although HR and MRR are good metrics to evaluate the performance of a recommender system, they are not very useful for hyper-parameter tuning, as they are binary metrics that only consider whether the recommended items are relevant or not and do not provide a measure of error between predicted and actual ratings. Pe langa asta RMSE măsoară diferența dintre evaluările prezise și cele reale, ceea ce este important pentru acuratețe.

After 5-fold cross-validation, we obtained the following hyperparameters for the user-based collaborative filtering model, which we listed in Table 4

This model achieved an RMSE of 0.98 averaged over the 5 folds

.

And for the SVD model, we obtained the following hyperparameters, which we listed in Table 5

User-based Parameter	**Value**
name	**msd**
user_based	**True**
min_support	**5**
shrinkage	**0**

Table 4: User based parameters

SVD Parameter	**Value**
n_factors	**150**
n_epochs	**30**
lr_all	**0.01**
reg_all	**0.1**

Table 5: SVD Parameters

For user-based CF, I used the KNNWithMeans algorithm of the Surprise library. The name parameter represents what value we will use for the similarity value, in this case we used "msd", which is the mean squared difference. The "min_support" parameter represents the minimum number of users who must have evaluated a movie for it to be included in the recommendation process. The "user_based" parameter determines whether to use the user-based or item-based approach. "shrinkage" is a parameter used to adjust the similarity between movies. For SVD-based CF, I used the Surprise library's SVD algorithm. The "n_factors" parameter represents the number of factors (latent dimensions). The "n_epochs" parameter represents the number of epochs (iterations) for optimizing the algorithm. The "lr_all" parameter represents the learning rate, and the "reg_all" parameter represents the regularization rate.

3 Week 8

Although RMSE is a metric generally used to predict the accuracy of predictions, it still has some limitations.

One of the limitations of this metric is the fact that it does not take into account the rank of the recommended item, so a movie, in this case, that the user might like might have a lower RMSE score.

Besides this, another limitation of this metric is that it assumes that all users give ratings in the same way. When I say that this metric assumes that all users give ratings in the same way, I mean that we cannot say that, for example, two people who give a rating to the same movie, the same rating, gave the rating according to the same criteria. Although the rating is basically the same, the criteria that led users to give that rating can be completely different. But in reality, each user has a unique way of giving ratings. Thus, the accuracy of the metric may be affected.

After calculating Hit rate averaged across users, Precision@k, averaged across users, Mean Average Precision (MAP@k), Mean Reciprocal Rank (MRR@k), and Coverage, I obtained the following results, which I noted in the table Table 6.

All these metrics have both advantages and disadvantages.

The hit rate has the advantage of being easy to calculate and comprehend. However, it does not consider the rank of the recommended item.

Precision@k takes into account the ranking of the recommended items and gives more detailed information about the recommendation. But it only considers the presence or absence of the relevant items in the top K recommended items and doesn't take into account the user's personal opinion about the relevance of the recommendation. And sometimes it might punish the system for suggesting relevant items that don't perfectly match the user's taste.

Mean Average Precision (MAP@k) takes into account both the relevance and position of recommended items so it gives more detailed feedback on the quality of the recommendations. But it takes more time to calculate and may not be suitable for big datasets. Additionally, it may not capture the diversity of recommendations and may not be as informative in cases where the user has a niche interest.

Mean Reciprocal Rank (MRR@k) measures how quickly a relevant item appears in the recommendation list, giving a higher score to systems that suggest relevant items at higher ranks. However, it may not take into account the variety of recommended items and may not reward systems that suggest multiple relevant items at lower ranks.

Coverage shows the percentage of items that are recommended to at least one user. It helps to evaluate the diversity of the recommended items. But it may not work well for systems that suggest personalized items to each user.

Comparing these metrics for the 3 systems, User-based, SVD and TopPop, we can draw the following conclusions:

User-based has a low hit rate, precision, MAP, and MRR but has a high coverage.

TopPop has the highest hit rate, precision, MAP, and MRR and a high coverage.

SVD has a higher hit rate, precision, MAP, and MRR than user-based model, but still lower than TopPop, and it also has a low coverage.

Among all these 3 systems, the one with the best overall performance is TopPop. One reason TopPop works so well is that it is a very simple and easy to implement solution that does not involve complex calculations or algorithms. It simply recommends the most popular articles that are likely to be liked by many users, leading to high hit rates and accuracy. But the main reason why it works the best of the three systems is that it recommends the most popular items so it will have an advantage in metrics that prioritize popular items, such as hit rate and Precision@k.

Each of these 3 models comes with advantages and disadvantages.

User-based collaborative filter is a quick and easy implementation without the need for initial data modeling. However, it also has limitations, such as when it encounters the "cold start" problem where new users and items have insufficient data for accurate recommendations. And another disadvantage of it is that it is not scalable, with the increase in the number of users and movies, the performance will decrease.

The SVD has the advantages that it can manage the problem of the data sparsity better than the user-based model, and it can make accurate recommendations for new users and items. On the other hand, this model may suffer from overfitting and requires more time-consuming data modeling to calculate the factorized matrices.

TopPop system is a simple and easy to use algorithm, it does not require data modeling and can be used in comparisoning the recommendation models. On the other hand, it does not take into account the preferences of each user, but only recommends what is the most popular, thus suffering from popularity bias.

4 Week 9

At the beginning, the size of the vocabulary for metadata was 172304, and after I applied all the related processing on the overview column, it decreased to 70731. After that, I realized that it is more efficient from the point of view of space to do these processings directly above the variable that contained the merge between the mapping_table with metadata, and thus at the beginning we had a vocabulary of 15754 words, which after the related processing ended up having only 9559 words. For the preprocessing part, I mainly used the nltk library.

I represented movies within the vector spaces in two ways, TF-IDF and word embeddings. After that, I calculated the cosine similarity between all the films for each method. To compare the two matrices obtained after the cosine similarity calculation, I used the Pearson correlation coefficient, obtaining a value of 0.1906. This value is relatively small, therefore it indicates that the two cosine similarity matrices are not strongly correlated. This is normal, because the TF-IDF method focuses more on the frequency of words in the overview of the films, while word embeddings has to do with the meaning of the words and how they relate to each other.

5 Week 10

Comparing these metrics between the content-based recommendation system, with the 3 previously created recommendation systems, I reached the following conclusions:

Content-Based Recommender System has a lower hit rate, Precision@5, and MRR@5, but a higher MAP@5 and coverage.

This recommender system has a hit rate which is higher than CF (User-based) but lower than CF (SVD) and CF (TopPop).

The precision@5 is also lower than all three CF models.

It has the highest MAP@5 of all models.

The MRR@5 is also higher than CF (User-based), but lower than CF (SVD) and CF (TopPop).

It has a high coverage compared to the 3 other models.

6 Week 11

I have created 3 hybrid recommendation models, each with a different strategy.

The hybrid model, created for weighted strategy, combines the recommendations given by the CF(SVD) recommendation system with the recommendations of the content-based recommendation system. I chose to use as CF, CF(SVD) because it had better metric values than CF(user-based), and TopPop is not a collaborative filter. In order to combine the recommendations of the two systems, I use two weights, alpha and beta, each related to a system. I decided to give a higher weight to alpha of 0.6 because CF(SVD) had better metrics values. The final list of recommendations is obtained by sorting the list of weighted ratings in descending order and selecting the first k movies.

The hybrid model, created for the switching strategy, combines the recommendations given by the CF(SVD) recommendation system with the recommendations of the content-based recommendation system. And in this case I chose to use CF(SVD) due to the better metric values.In this case, this combination of recommendations from the two models is based on a condition. This condition is given by Borda count, which assigns scores to each film based on its position in the list of recommendations for each model. The movies are then sorted by scores, and the list of recommendations for each user is made by alternating between the top recommendations of the two models.

Also, because I faced the problem where two or more recommendations have the same rank, I introduced a parameter that is randomly generated to arbitrarily select a movie from the two recommendation lists. And in order to have a higher degree of randomness, I decided to run the recommendation generation algorithm for this hybrid system a random number of times.

The hybrid model, created for meta level strategy, uses the recommendations made by the content-based recommendation system as input for a CF(SVD) recommendation model. I decided to use the output of the content-based system as input for CF because although it has comparatively weaker metrics, it has a high coverage and MRR, so I thought it might have some pretty good recommendations. I decided to use the SVD as CF because it had better metrics than the User-based one and thus it can compensate for the weaker performance of the content-based model. For each user, it receives the best recommendations from the content-based system and predicts ratings for unrated items using the SVD model. The top movies are then combined from both systems and presented as recommendations to the user.

Comparing these metrics between the 3 hybrid recommendation systems and the 4 previously created recommendation systems, I reached the following conclusions:

Compared to previous models, hybrid recommender systems generally perform better in Average Hit Rate, Precision@5 and Average MRR@5. However, the coverage of the hybrid models is lower than that of the Content based model.

The hybrid model (weighted strategy) performs best with higher Average Hit Rate, Precision@5 and Average MRR@5 scores than the other two hybrid models.

The hybrid model (switching strategy) performs better than the meta-level strategy in all metrics, but its scores are lower than the weighted strategy.

The hybrid model (the meta-level strategy) performs the worst among all 3 hybrid models, with very low scores on all metrics.

If we do not take into account the coverage, which is quite large, the hybrid model (strategy at the meta-level) has the worst scores among all other recommendation systems.

7 Table comparison and other informations

Model	Average Hit rate	Precision@5	Average MAP@5	Average MRR@5	Coverage
CF (User-based)	**0.07**	**0.01351**	**0.00424**	**0.02121**	**13.83%**
CF (SVD)	**0.37**	**0.11808**	**0.01244**	**0.05320**	**6.17%**
CF(TopPop)	**0.63**	**0.19259**	**0.10919**	**0.38250**	**0.39%**
ContentBased	**0.10**	**0.021**	**0.10**	**0.046**	**38.35%**
Hybrid(Weighted strategy)	**0.35**	**0.1146**	**0.0932**	**0.1639**	**6.19%**
Hybrid(Switching strategy)	**0.25**	**0.0645**	**0.0523**	**0.1062**	**26.68%**
Hybrid(Meta-level strategy)	**0.01**	**0.0039**	**0.0027**	**0.0068**	**32.47%**

Table 6: Metrics results for all models

I used Surprise library off-the-shelf implementations for the user-based CF and SVD-based CF models. Except for tuning hyperparameters and selecting the best model based on performance metrics, I didn't do much adaptation. It was successful because these models performed well. Another standard implementation I also used a standard implementation for calculating the rmse for the 2 collaborative filter models.

The TopPop model was made from scratch, being a simple algorithm to implement. It has a pretty good performance.

I used TF-IDF vectorization for the content-based model to convert movie descriptions into vectors, which we then used to calculate similarity scores between movies. I also built this model from the scratch and it performs pretty well.

I preprocessed and cleaned the dataset, such as removing irrelevant columns and dealing with missing values.

In addition, I used a preprocessing method for the overview column in the dataset. These steps were helpful in improving the data quality and reducing noise, which helped the models perform better.

A limitation of the approach we had is that we did not take into account the time at which the review was made. Because with the passage of time, the user's preferences change and thus, if we take this aspect into account, we could make a system that recommends more relevant movies. Another limitation could be that we did not take into account the genre of the films. This could have helped create a personalized recommendation system. Another limitation could be that we did not consider the demographics of the user, another aspect that could have helped in creating a more personalized recommendation system.

Another approach to automatic recommendation could have been Deep learning, to train a neural network capable of detecting patterns in user behavior and recommending new movies based on them.

A major challenge in working with this data set was to combine several data sets to obtain the required set. Besides that, I had to come back and do its processing to get rid of unimportant data or to convert columns from one type to another. But another problem with this data set is that it is quite small. It has a relatively small number of users and movies.