Project 1: Recommender system

Decision Support Systems

Kramer, Bastian Aron
201908709post.au.dk

Fisker, Jens Villadsen
201908671@post.au.dk

April 29, 2023

Contents

1 Introduction

Recommendation systems are a huge part of the modern person's life. It is all around us from when we see advertisements and are on social media, to when we are sitting at home ready to watch a movie. Here, the use of recommendation systems decides what you want to see, buy or interact with. This is done with the help of endless amounts of data generated from the internet and describes users' habits, tendencies, or interests. That data is what has given the possibility to predict a user's interest by comparing them to similar users' interests. Recommender systems have a range of different techniques such as collaborative filtering, content-based filtering, and hybrid approaches. This report will look into a collaborative filtering recommender system that can be used to predict which movies a user would want to see based on the "MovieLens" dataset.

2 Methods

While this report will be focusing on collaborative filtering, a brief introduction to content-based and hybrid approaches will be presented in this section as well.

In general, recommender systems fall into two categories depending on the approach of the filtering algorithm used. First, the content filtering approach use attributes such as genre, participating actors, box office popularity, and so forth to characterize the nature of products. Likewise for users, the content filtering approach will use attributes such as age, gender, country, religion and so forth to characterize them. The content filtering approach then associate the characteristics of users with products including the same or similar characteristics. For this approach to work, a lot of external data is required, which could prove difficult to acquire.

Alternatively, the collaborative filtering approach, allows matching of users and products without requiring explicit profiles. This can be done because collaborative filtering relies on past user behavior such as prior transactions or product ratings.

Lastly, hybrid approaches combine recommendation techniques to provide more accurate and diverse recommendations. These approaches aim to overcome the limitations of individual recommendation techniques and provide a better user experience.

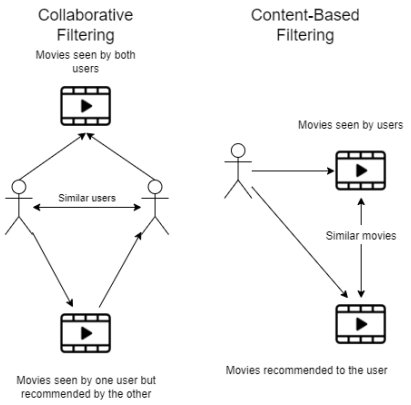

Figure 1: Collaborative and content based filtering methods visualization

2.1 Collaborative filtering

Collaborative filtering analyzes connections between users and products to discover new user-product associations. Its versatility and ability to address data aspects that are challenging to profile with content filtering are some major appeals of collaborative filtering. Although collaborative filtering has better accuracy than content based filtering, it is not perfect. One flaw of the collaborative approach known as the "cold start problem" happens when information is very sparse, such as when dealing with new products and users.

Collaborative filtering comprises two primary areas: neighbourhood methods, and latent factor models.
Neighbourhood methods compute relationships between items or relationships between users. When computing relationships between items, the algorithm evaluates a user's

affinity for a given item based on the user's rating of "neighbouring" items. In this case, being a "neighbour" to an item, means being similarly rated when rated by the same user. As such, this approach will identify like-minded users, and use this pattern to help predict good recommendations[1].

Latent factor models tries to explain the ratings by defining items and users on factors inferred from the rating patterns. These factors can represent various dimensions that can be more or less obvious to us as humans. Examples of such dimensions for a movie could be genre, character development, amount of action, or orientation to children. For users, each factor is a measurement of the users inclination towards movies that score high in the given factor.

2.2 Matrix Factorization techniques

Matrix factorization techniques can be broadly classified into two categories: deterministic and probabilistic methods. Deterministic approaches, such as singular value decomposition (SVD) and non-negative matrix factorization (NMF), directly minimize the reconstruction error between the original matrix and its approximation. Probabilistic methods, on the other hand, introduce latent variables to represent hidden factors and model the observed data in a probabilistic framework[2].

The most common matrix factorization technique in collaborative filtering is SVD, which decomposes a matrix into the product of three matrices: two orthogonal matrices and a diagonal matrix containing singular values. The truncated version of SVD, also known as low-rank approximation, is often employed to capture the most significant latent features and reduce overfitting.

Another popular technique is NMF, which decomposes a non-negative matrix into two non-negative matrices. It is advantageous in cases where the latent factors should have non-negative values, such as when modeling user preferences or content features [3].

Even though this report does not use Probabilistic matrix factorization (PMF), it is still worth briefly mentioning as it is a Bayesian approach that introduces priors on the latent factors and optimizes the posterior distribution of the latent variables. It has been shown to produce more accurate predictions and better generalization performance compared to deterministic methods, especially when dealing with sparse data [2].

3 Implementation, Results, and Discussion

In this section the implementations of the recommender system will be described. This will be followed by the presentation of the results from the different implementations. To conclude this section, the results will be analyzed in the discussion.

3.1 Implementation

The recommender system is implemented in python. Here, the "MovieLens Latest Dataset" is used as data and the surprise library used for training testing and getting some results. This implementation is can be found by following this reference[4].

3.1.1 Dataset

The dataset used is the small version of the MovieLens dataset as it was recommended for education and development including 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. This dataset can be found at grouplens.org. The reason for the small dataset is that when tested with the large dataset the computation timer was too big. The dataset has two files a movies.csv and a ratings.csv. The movies has the movieID , title and the genres of the movies rated by the users. Then the ratings has the userID the movieID rating and a timestamp.

3.1.2 Surprise

The surprise library is used for doing most of the implementation regarding learning algorithms, training, testing and getting the results in the form of the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

3.2 Results

The model that scored the best will be the first item to be examined in this results section, followed by a discussion of the outcomes of the anti-testing.

Algorithm	RMSE	MAE
SVD	0.8834	0.6781
NMF	0.9295	0.7131
KNNBasic	1.4288	1.1426
NormalPredictor	0.9519	0.7299

Table 1: Show the different Root Mean Square Error and Mean Absolute Error for the tested Algorithm

In the Table 1 above it can be seen the results of the RMSE and MAE for the four tested algoritems SVD, NMF, KNNBasic and NormalPredictor. Here it can be seen that the best algorithm is SVD on both parameters. The test was also done with respect to training and test time, however, the measured times were so small on the tested dataset, which meant it had an insignificant impact. However if a bigger dataset was to be used, these parameters could have made a lighter model more suitable, and thus better, in some cases.

When the SVD model was trained with anti testing, meaning the test data is data the user has not yet recommended, the RMSE and the MAE performed better than without it giving an RMSE of 0.4858 and MAE of 0.3768. This was a surprisingly amount better than expected, and as such there seamed to be something wrong. It was later discovered that RMSE and the MAE can not be used as evaluation metrics when using anti testing. Therefore, a more manual evaluation was done in the discussion section.

3.3 Discussion

In this discussion section the manual evaluation will be discussed. Furthermore, the limitations of the surprise library and dataset will be discussed. Finally, a subsection will be dedicated for a comparison with state of the art models for recommender systems.

```
Drama                2998
Comedy               2402
Thriller             1220
Action               1216
Romance              1118
Adventure             888
Crime                 801
Sci-Fi                655
Fantasy               521
Horror                495
Animation             437
Children              437
Mystery               387
Documentary           310
War                   287
Musical               217
IMAX                  133
Western               109
Film-Noir              62
(no genres listed)     18
Name: genre, dtype: int64
```

```
Drama          61
Comedy         31
Action         22
Thriller       20
Crime          19
Romance        17
War            17
Adventure      16
Fantasy        11
Animation      10
Sci-Fi         10
Mystery         6
Children        6
Horror          6
Musical         4
Film-Noir       3
Documentary     2
Western         2
IMAX            1
Name: genre, dtype: int64
```

Figure 2: Show users most rated genres on the right. And on the recommend genres are show on the left

As seen in 2, the top 5 genres are almost identical, as 4 out of 5 genre are shared. This can be used as an indicator that the model build is somewhat accurately recommending movies that have an interest for the user. The problem however with this recommender system, is the fact that it can be somewhat hard to prove they are recommending the best movies for a user. So if an accurate test wanted to be done to determine how good the recommender system actually performed, there would have to be real world testing with it. In our case a user could rate 50 movies and then come up with a recommended rating for 5 movies that the user then can confirm or deny and give a different rating. This test can then be done for some amount of people and then when most people agree with four out of the five recommended ratings you can call it a success. However, this was not feasible for this project.

Some of the problems regarding the surprise library were that when recommendation based on more than just the ratings of the movies the code would not compile. At the

end of the code there is a code block where an implementation using the genres of the movies too. However this piece of code does not compile. Here was spent a lot of hours trying to figure out how to implement and debugging before the decision to cancel it was made. So we wanted it as proof that we tried but it wouldn't work.

The dataset used was the smaller of the two datasets available and it may have been interesting to have seen that dataset would have performed. But due to limitations in machine power the training times of models with that dataset it was decided not to be done. Another problem with the data set is that it seems not to have been optimized for doing recommender systems on more than the ratings. This gave trouble trying to rearrange the dataset to have genres in the same dataset that had the rating of the users.

4 Conclusion and Perspectives

In this project, a recommender system was developed using the MovieLens dataset and the Surprise library in Python. The primary goal was to analyze the performance of different algorithms in predicting user preferences. The SVD algorithm performed best in both RMSE and MAE. However, it should be noted that these metrics could not be used for evaluating the performance of the recommender system when anti-testing was used.

Instead, the manual evaluation provided some insights into the model's ability to recommend movies based on the user's interests. Nonetheless, a more accurate assessment would require real-world testing where users rate movies recommended to them - how well it fit - which was not feasible within the scope of this project.

There were some limitations identified in the implementation process. The Surprise library proved challenging to work with when incorporating additional features like movie genres, which could have potentially improved the performance of the recommender system. Additionally, the smaller dataset was used due to computational constraints, which may have affected the overall results.

In future work, several aspects could be explored to enhance the recommender system. Firstly, incorporating additional features such as movie genres, actors, and directors could improve the model's ability to make more personalized recommendations. This would require overcoming the limitations found in the Surprise library or finding an alternative approach.

Another perspective would be to test the model using the larger MovieLens dataset to investigate how the increased data size impacts the performance of the algorithms. Additionally, using more sophisticated evaluation methods, such as precision and recall, could provide a better understanding of the system's performance in real-world scenarios.

Finally, exploring other state-of-the-art models and techniques, such as deep learning and reinforcement learning, could potentially lead to more accurate and reliable recommender systems. This would require further research and experimentation to determine the most suitable approach for this application. Overall, this study provides a solid foundation for future work in the field of recommender systems and highlights areas for potential improvement.

References

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[2] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., vol. 20. Curran Associates, Inc., 2007. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf

[3] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization." *Nature*, vol. 401, no. 6755, pp. 788–791, October 1999.

[4] B. A. Kramer and J. V. Fisker, "Recommender$_s ystems_p roject.''[Online].Available$: