Recommender Systems

Module Name: COMP3607

Date: 7/11/2022

1 Introduction

For the project, the domain chosen is movie recommendations, this is because movies are inherently easy to compare to one another due to their genres and associated tags.
The overall purpose of the project is to use a movie dataset to give recommendations that are accurate, relevant and explainable to the active user. The aims of the project include making a hybrid system: the collaborative filtering system (CF) collects the predictions then the content-based filtering (CBF) and expert system evaluate and reorder them. Other intermediate aims include predicting the score a user would give to a movie accurately and making recommendations explainable where possible. Finally, an evaluation on accuracy, relevance and explainability will be conducted to judge the performance of the system against its non-personalised counterpart.

2 Methods and techniques

2.1 Dataset

The dataset chosen is the Movie Lens Dataset as it contains over 25 million data entries. Within this dataset there are 6 csv files containing every movies' information and every user's individual ratings.

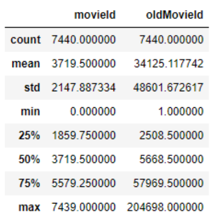
2.2 Data preparation

Data Pre-Processing

For initial data pre-processing I decided to reduce ratings.csv to one million ratings, this is because the csv file is too large to load into memory and other pre-processing on the full dataset will run into memory issues. For testing we used the 1M size however the size parameter can be used to increase the size of the dataset.

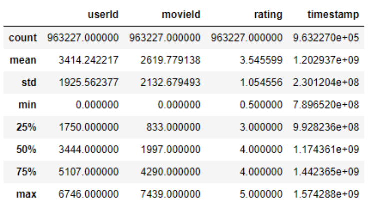
Data Wrangling

For data wrangling every movie that was rated less than 10 times was removed from all of the datasets, drastically decreasing the number of movies (Fig.1.). After this every dataset is reindexed by movieId in order starting from zero. Next every userId is subtracted by 1 so that the first userId starts at zero. By starting each movieId and userId at zero and getting rid of redundant movies, it means that future Matrix Factorization won't have any redundant rows and will take less computation as Matrix Factorization computes on a matrix of size userId.max x movieId.max (Fig.2.).



	movieId	oldMovieId
count	7440.000000	7440.000000
mean	3719.500000	34125.117742
std	2147.887334	48601.672617
min	0.000000	1.000000
25%	1859.750000	2508.500000
50%	3719.500000	5668.500000
75%	5579.250000	57969.500000
max	7439.000000	204698.000000

Fig. 2. Description of movies.csv after data preperation



	userId	movieId	rating	timestamp
count	963227.000000	963227.000000	963227.000000	9.632270e+05
mean	3414.242217	2619.779138	3.545599	1.202937e+09
std	1925.562377	2132.679493	1.054556	2.301204e+08
min	0.000000	0.000000	0.500000	7.896520e+08
25%	1750.000000	833.000000	3.000000	9.928236e+08
50%	3444.000000	1997.000000	4.000000	1.174361e+09
75%	5107.000000	4290.000000	4.000000	1.442365e+09
max	6746.000000	7439.000000	5.000000	1.574288e+09

Fig. 1. Description of Ratings.csv after data preparation

Feature Selection and Extraction

The main features used from the dataset are:

- Ratings from ratings.csv, used in the CF for individual user movie rating. Furthermore, used in the expert system to collect an average rating and rating count for all movies.
- Relevance from genome_scores.csv, used to collect the 3 most relevant tags for each movie in contributing to each movie's TFIDF description.
- IMDBid from links.csv will be collected to process the movie description using natural language processing and then collect the 3 most important words for each movie's TFIDF description
- Genres from movies.csv will be used to contribute to each movie's TFIDF description
- IDs will be used to collapse the different tables into single tables.

2.3 Methods

One common thread in recommender systems research is the need to combine recommendation techniques to achieve peak performance [1]. This project will be comprised of a hybrid model that will use CF system which is then evaluated by a CBF system and an expert system to produce an importance metric which determines the order of the CF system's recommendations [2].

Singular Value Decomposition

CF can have two different approaches, a user-based approach chooses items based on the correlation between users with similar preferences [3]. An item-based approach aims at finding similar items to those that the user has already rated, recommending other items. In this paper we will use a user-based approach as a study shows the item-based approach gives a lower mean squared error (MSE) when compared to the item-based approach on Movie lens 25M Dataset [2].

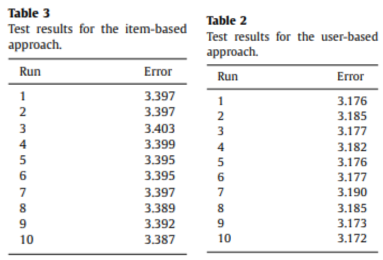


Table 3
Test results for the item-based approach.

Run	Error
1	3.397
2	3.397
3	3.403
4	3.399
5	3.395
6	3.395
7	3.397
8	3.389
9	3.392
10	3.387

Table 2
Test results for the user-based approach.

Run	Error
1	3.176
2	3.185
3	3.177
4	3.182
5	3.176
6	3.177
7	3.190
8	3.185
9	3.173
10	3.172

Fig. 1. Table of results for different collaberative filtering techniques on movieLens dataset from [2]

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. This family of methods became widely known during the Netflix prize challenge due to how effective it is, demonstrating how useful CFs are in movie recommendations [4]. We can do matrix factorization via the Singular Value Decomposition algorithm (SVD) as it is the most effective method in terms of its performance. The SVD algorithm will give us a predicted score for each predicted movie. 20 predictions are used because with more predictions the average user will get diminishing predicted ratings, which don't need to be calculated.

Movie description collection

After CF, each recommendation and already rated movie collects its relevant TFIDF description: firstly, each movies genre from movies.csv is collected, secondly. each movie's top 3 most relevant tags are collected from genome_scores.csv and finally each movie's description is collected using the IDMBid from links.csv. Each IMDB movie description then collects the 3 most relevant words using natural language processing techniques. All these words are collected and stored together.

TF-IDF algorithm and Cosine Similarity

CBF relies more on descriptions and features in the dataset over historical interactions and preferences hence why we will use it to calculate similarity rather than recommendations [5]. First a TF-IDF algorithm is used, this quantifies the importance or relevance of string representations (words, phrases, lemmas, etc) in a document amongst a collection of documents (also known as a corpus) [6]. We use every already rated movie and every predicted movie for the TF-IDF algorithm. We then use cosine similarity; this will calculate the highest level of similarity between already rated movies and every other recommendation for each recommendation. Using this we compute a similarity metric for each recommendation and store the most similar movie if it has already been rated (for explainability purposes).

Expert system

The expert system works using the average rating of the movie, the total number of movie ratings and the similarity to already rated movies and recommendations, to calculate an importance metric (1). This metric is involved in the final evaluation metric which determines the order of the recommendations to the user, reordering the predictions so more relevant results appear higher up (2).

$$importance = (similarity_i \,/\, similarity.max()) * (averageRating_i \,/\, averageRating.max()) * (count_i \,/\, count.max()) \qquad (1)$$

$$finaleval_i = Importance_i * predictedEval_i \qquad (2)$$

2.4 Evaluation methods

A non-personalised system has been made which gives the same predictions for every user based on the average user. We evaluate the personalised system by testing a sample of users with a different number of already rated movies, using half of their already rated movies, in order to try and predict 20 of the other already rated movies. We then do the same for the non-personalised system and compare their performances using metrics.

Evaluation Metrics

The first metric that will be used is the accuracy, this determines how many of the recommended movies were in the users unknown already rated movies, this tests the accuracy of ranking predictions (3). The second metric is the average mean absolute error this is done on the predicted evaluation against the actual evaluation of the correct movies, this tests the accuracy of rating predictions (4). Next is the explainability score which is a percentage of how many of the recommendations can be explained (5). The system is made and evaluated by the relevancy of its results. Unfortunately, the true relevance cannot be properly tested as this is opinionated and there is no test group.

$$accuracy = correctRecommendations \,/\, totalRecommendations \qquad (3)$$

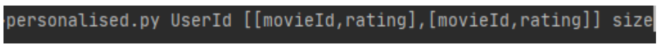
$$mae = \Sigma_i\,[abs(predictedEval_i - actualEval_i)] \,/\, correctRecommendations \qquad (4)$$

$$explainability = explainedRecommendations \,/\, totalRecommendations \qquad (5)$$

3 IMPLEMENTATION

3.1 Input Interface

The personalised recommender system can be run through the command line with 3 arguments, the latter 2 being optional.

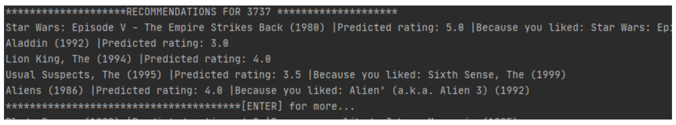


```
personalised.py UserId [[movieId,rating],[movieId,rating]] size
```

Fig. 2. Input interface format.

3.2 Output Interface

The output interface contains 5 recommendations as we want to show the best recommendations first, to show more users can press Enter (up to 20 recommendations). In each recommendation users only get the relevant information to them including, in some cases, why the movie was recommended.



```
********************RECOMMENDATIONS FOR 3737 ********************
Star Wars: Episode V - The Empire Strikes Back (1980) |Predicted rating: 5.0 |Because you liked: Star Wars: Epi
Aladdin (1992) |Predicted rating: 3.0
Lion King, The (1994) |Predicted rating: 4.0
Usual Suspects, The (1995) |Predicted rating: 3.5 |Because you liked: Sixth Sense, The (1999)
Aliens (1986) |Predicted rating: 4.0 |Because you liked: Alien³ (a.k.a. Alien 3) (1992)
****************************************[ENTER] for more...
```

Fig. 3. Output interface for the Recommender System

4 Evaluation

4.1 Evaluation metrics results

Accuracy

Personalised average accuracy: 63%.
Non-personalised average accuracy:53%.

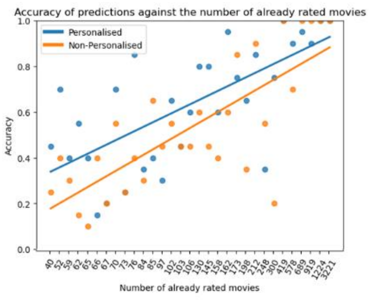


Fig. 4. Accuracy of predictions against the number of already rated movies.

Mean Average Error

Personalised average MAE: 0.64.
Non-personalised version average MAE: 0.71.

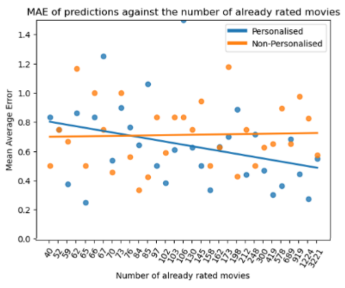


Fig. 5. Mean Average Error of predictions against the number of already rated movies.

Explainability

Personalised average explainability percentage: 73%.
Non-personalised average explainability percentage: 0%

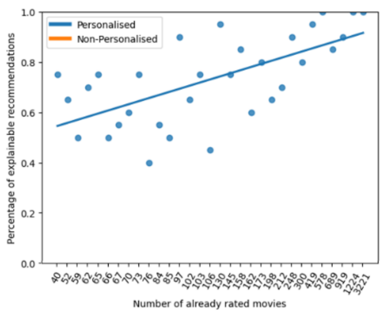


Fig. 6. Explainability of the personalised recommender system against the number of already rated movies from the test group.

We can observe that the explainability increases with the number of already rated movies as there is a better chance an already rated movie is the most similar to a predicted movie.

4.2 Comparison

In Fig.5. The personalised versions accuracy was greater than the non-personalised version when the number of already rated movies was low, this is because when the number of movies increases it means that a selected user is more likely to of rated one of the most popular movies. With lots of already rated movies, the non-personalised version works really well as the chance someone has rated the 20 most popular movies increases. However, for the average user the personalised version has a higher accuracy.

In Fig.6. we can observe that the personalised version gives a similar MAE to the non-personalised version for users who have around 76 already rated movies, however as the number of movies increases the personalised version gets a lower MAE, as it can better understand the users' rating habits. MAE is better than the non-personalised version overall, as it uses the average rating for every movie rather than personalised rating habits.

Overall, the personalised version performs better for the average user however sometimes the non-personalised version can outperform the personalised version, in some metrics, when the number of already rated movies is in the top ~5% of users.

5 Conclusion

The advantage to a hybrid system is their ability to combine different approaches which can improve on cold start and sparsity problems [7]. Disadvantages to hybrid systems and CF are that they are hard to explain where different predictions have come from, this is a problem as recent trends include transparency in recommender systems. The system developed tackles some of these problems, creating explainable, accurate, relevant results for users with not that many ratings. Future development of the system could include accounting for changes in user preferences as currently there is no metric that makes old movies "fall off" compared to newer movies that users have watched.

4 References

[1] Burke, R. (2002) Hybrid Recommender Systems: Survey and experiments - user modeling and user-adapted interaction, SpringerLink. Kluwer Academic Publishers. Available at:
https://link.springer.com/article/10.1023/A:1021240730564 (Accessed: November 6, 2022).

[2] Walek, B. and Fojtik, V. (2020) A hybrid recommender system for recommending relevant movies using an expert system, Expert Systems with Applications. Pergamon. Available at:
https://www.sciencedirect.com/science/article/pii/S0957417420302761 (Accessed: November 6, 2022).

[3] Meteren, R.van (2000) Using content-based filtering for recommendation, http://users.ics.forth.gr/. Available at: https://users.ics.forth.gr/~potamias/mlnia/paper_6.pdf (Accessed: November 6, 2022).

[4] Vidiyala , R. (2020) How to build a movie recommendation system , https://towardsdatascience.com/. Available at: https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109 (Accessed: November 6, 2022).

[5] Tompkins, T. (2020) Selecting the right recommendation system, Advancing Analytics. Advancing Analytics. Available at:
https://www.advancinganalytics.co.uk/blog/2020/5/13/recommendation-systems (Accessed: November 6, 2022).

[6] Simha, A. (2021) Understanding TF-IDF for Machine Learning, Capital One. Available at:
https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/ (Accessed: December 23, 2022).

[7] Kumar, M. (2015) A movie recommender system: MOVREC - Researchgate, https://www.researchgate.net/. Available at: https://www.researchgate.net/profile/Manoj-Kumar-643/publication/283042228_A_Movie_Recommender_System_MOVREC/links/61a0ac4407be5f31b7b82c3e/A-Movie-Recommender-System-MOVREC.pdf?origin=publication_detail (Accessed: November 6, 2022).