Hybrid Movie Recommendation System

Abhijit Singh,Yousef Fahim, Yurong Jia, Eric Corbu, Benson Leung, Arhan Ikram
Data Mining CP421, Department of Physics and Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada

Abstract

In this paper we describe an approach to implement a hybrid movie recommendation system that aggregates results based on content based and collaborative filtering and recommends movies based on the user's personal taste and watch history.

1. Introduction & Context

For this project, we developed a recommendation system to suggest similar movies to a target user. Given a user's movie ratings and other user ratings, we found other users with similar preferences by analyzing all movie ratings across all users. In this model, we built a hybrid of content based and collaborative filtering approaches. We combined the results of both approaches to get precise predictions. We used the movie metadata from the given dataset and other that was obtained from internet sources such as the IMDb database. Alongside the above mentioned techniques. natural language processing techniques were applied club under content based filtering to obtain similar movies to the input movie. Based on those similarities, we found movies similar to the user's most recently viewed movies in accordance to other rated movies by the user.

We showcase how our model's predictions are preferable than other models' predictions by measuring to baseline predictive approaches. In order to measure the accuracy of our predictions, we calculated the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) were used to find the difference in predicted and original ratings. This provided an idea on how accurate our measurements are. The pipeline is depicted in details on the presentation material and can be referenced for a general overview of the model.

2. Related Work

This model incorporates both the content-based and the collaborative filtering approaches. By incorporating the two approaches, we processed the movie data given and trained our model with that data to predict the ratings of other users, therefore relying more on the data than the algorithm. As a result, more data will result in better accuracy for the model. We also solved multiple problems that come with each approach alone; for instance, the cold start problem in the content based approach. The content-based approach alone can give us semi-accurate predictions; however, it does not take into account the ratings of other users and thus can result in users getting false recommendations of movies. The collaborative filtering approach alone can also result in semi-accurate predictions; however, it does not take into account any of the movie metadata being analyzed by our approach and thus can result in new users not getting any accurate recommendations. Collaborative filtering when applied to the available dataset of movies results in inaccurate and unrelated recommendations. Generally speaking, the collaborative filtering depends on content based similarities for the direction.

The proposed hybrid model to a content based only approach, our model gives more personalized results than a content based only approach capturing the "personal taste" aspect of recommendation systems. Moreover, our model is able to make recommendations outside our similarity metrics, which helps users develop new tastes thus incorporating 'diversity' and 'long-tail' into the results which are not captured by either baselines independently. The proposed hybrid model, when compared to a collaborative filtering only approach, avoids the cold start problem by making use of content based filtering and avoids the scalability issue by making use of the latent factor model.

3. Methodology

Our solution will be following a similar approach mentioned in the study[1] to provide a hybrid model consisting of underlying collaborative and content based filtering techniques. The first half of our pipeline consists of content based filtering. This allows us to extract similarity between different movies from the given dataset. Second half of the solution uses direction given by similarity of movies based approach and ranks movies according to personal preference of the given user.

3.a. Content Based filtering

The movies are given a similarity score based on the similarity of plot, genre and director of the movie. The choice of the above mentioned features triumphs over other feature sets for calculating similarity scores because most similar movies have similar plots, they usually belong to the same genre and are most often directed by the same director.

Table 1: Depicting similarity in plotlines

MovieId	Plotline
519	Robocop saves the day once more. This time the half….
3672	A stray dog saves two kidnapped children.
7354	When shy, soft-spoken Chicago detective Wayne….saves the life …
60046	British journalist, George Hog,.., saves a group of orphaned children during..
…	…

Having additional plot data for the movies provides a great advantage in recommending similar movies. As can be seen in the table above, the common theme between the movies is that the protagonist saves someone. This aspect of movie similarity will be missed if only contextual information for the movies was obtained from tags.

Thus the content based filtering part of our solution takes in account of metadata provided by the movielens database which is genres but in addition to that it uses directors and plots obtained from IMDb database, making the use of links.csv file provided by the input data.
The metadata is combined and vectorized. The content of the improved meta data is further discussed under preprocessing of data.

Using the improved metadata we make use of Natural language processing technique and convert the movies and metadata matrix into a TF-IDF matrix. The model makes use of the TF-IDF vectorizer from the standard Scikit-learn library. Generally speaking, this gives the overview of all the movies in the terms of defining keywords from the plots. As a result a matrix of size [9742,24601] is obtained. This refers to the 9742 movies projected over 24601 distinct keywords.

Using the Linear Kernel from the Sklearn-Metrics package we then obtain the cosine similarity matrix for all the movies making use of the TF-IDF matrix obtained in the previous step. Thus a final cosine similarity matrix of size [9742,9742] is obtained. The similarity between two movies can be achieved by accessing the appropriate row and column of this matrix.

The content based filtering part of the model makes top 35 recommendations to the hybrid part, based solely on the cosine similarity of the input movie. The choice of this number is further discussed under the experiment section.

3.b. Collaborative Filtering

There are two types of collaborative filtering methods, user-based and item-based. User-based collaborative filtering measures similarity between target user and other users, while item-based collaborative filtering measures similarity betweens items the target user liked and other items. The downside for user-based collaborative filtering is that user preferences can change over time. This method could result in bad performance since the matrix is precomputed based on neighboring users. The downside for item-based collaborative filtering is scalability since the computation complexity of the algorithm is O(m*n) where m is the number of users and n is the number of items. Inaccurate predictions could be caused by the sparsity of the data. For our model, we decided to use the user-based collaborative filtering method since this will result in more accurate predictions.

Our model uses user based collaborative filtering. Since the movies would be recommended based on the user's behaviour including previously watched movies and their respective rating, better predictions are expected as “person taste” will be captured under this part. To implement the collaborative filtering, the model makes use of SVD. The Singular Value Decomposition (SVD), a method from linear algebra that has been generally used as a dimensionality reduction technique in machine learning. SVD is a matrix factorisation technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where K<N). In the context of the recommender system, the SVD is used as a collaborative filtering technique. It uses a matrix structure where each row represents a user, and each column represents an item. The elements of this matrix are the ratings that are given to items by users. The factorisation of this matrix is done by the singular value decomposition. It finds factors of matrices from the factorisation of a high-level (user-item-rating) matrix. The singular value decomposition is a method of decomposing a matrix into three other matrices as given below:

$$A = USV^T$$

Where A is a m x n utility matrix, U is a m x r orthogonal left singular matrix, which represents the relationship between users and latent factors, S is a r x r diagonal matrix, which describes the strength of each latent factor and V is a r x n diagonal right singular matrix, which indicates the similarity between items and latent factors.

SVD turns the recommendation problem into an optimization problem by minimizing RMSE on the known entries in the utility matrix. To achieve minimal RMSE, Singular Value Decomposition (SVD) adopts the following approach:

$$\underset{m \times n}{\overset{\hat{X}}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}} \approx \underset{m \times r}{\overset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}} \underset{r \times r}{\overset{S}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}} \underset{r \times n}{\overset{V^{\mathsf{T}}}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}}$$

X denotes the utility matrix, and U is a left singular matrix, representing the relationship between users and latent factors. S is a diagonal matrix describing the strength of each latent factor, while V transpose is a right singular matrix, indicating the similarity between items and latent factors.
The model essentially maps each user and each movie into a latent space with dimension r. it helps us better to understand the relationship between users and movies as they become directly comparable.

The model uses the Surprise library[2] which is built on top of Scikit for building and analyzing recommendation systems that deal with explicit rating data. Thus the algorithm model is provided a user-rating matrix and made to develop testing and training dataset on to the given data and provides predicted ratings for movies that are suggested to match the input movie based on the given input.

3.c. Hybrid Approach

When predictions are based solely based on collaborative filtering, several false-positives were encountered. For example a movie which was rated by only two users, would appear very high in the ranking if the predictions was based on collaborative filtering only. In such cases very little similarity is found and these false positives usually outrank legitimately similar movies because of their lower number of vote counts. To avoid such false positives and make accurate predictions the model uses a hybrid approach.

The hybrid part of the proposed model aggregates the results achieved from the independent approaches discussed above. It takes input as [user, Movie title]. This input is passed onto the content based section to find the top similar movies based on techniques discussed above. The return object is a list of top movies i.e. 35 in the given context, based on similarity scores. Thus the list is then provided to the SVD trained model to make predictions for the input user and list of movies obtained from the content based section. Hence, the model uses the hybrid pipeline to cover shortcoming of individual baseline approaches and makes meaningful predictions.

4. Data and Experiment

The data provided by the given dataset was from MovieLens.org. A brief overview of the data presents us with several metrics about the data. These are as follows:

Table 2: The distribution of movie ratings across the given datasets

Metrics	Value
Mean	3.50156
Median	3.5
Mode	4.0

Further analysis of the rating data shows that ratings are left-skewed as approximately 60% of the ratings are between the range 3.0 to 4.0 on a scale of 5.0. Another interesting number is the lack of rating data for approximately 50% of the movies. This is in line with how rating systems work, with only 610 users rating 9472 movies, there is expected to be a lot of data sparsity.

Turning our attention to genre metadata we find that most frequent genres are [Drama, Comedy, Thriller, Action, Romance] whereas [Film-noir, IMAX, Western, War] are the least frequent.

4.a. Preprocessing Data

The data provided by the dataset needed preprocessing in order to be used by the model. The major data preprocessing stages are as follows:
1)The genre data provided by the data set is in list form. In order to be used in the TF-IDF matrix, the entire genres list for each movie was converted to string and commas(,) stripped from the resulting string. This transformation can be visualised as follows: [Romance, Action, Comedy] → Romance Action Comedy. The resulting string is then joined alongside additional metadata obtained from external resources, as discussed below.

2) In order to provide better contextual metadata to the recommendation system, the model uses the links.csv to obtain information about the movies in the IMDb database. This process takes particularly long as a large number of calls are made to external sources. To overcome this, the input consisting of movieId and links for the respective movieId were divided in 17 equal parts. Thus all these 17 parts were run in parallel to save computation time. The IMDB data when accessed returns a "movie" object. Only the required fields from the movie object are accessed and added to the dataframe with improved metadata. These fields are [director, plotline]

3) There are chances that different directors with similar first or last names can be added to the metadata. This could lead to many false-positives based on director similarity. The issue can be seen as below:

Table 3: Director sharing 1st name

MovieId	Director Name
8	Peter Hewitt
10	Peter Hyams
126	Peter MacDonald
219	Peter Horton
304	Peter Yates

As clear from the table above, although different directors but all of them share the common first name, "Peter". Thus these would hurt the similarity scores between movies in a negative way and lead to incorrect predictions. In order to mitigate this problem, we decided to apply the classic text manipulation technique of clubbing the first and last name together, with the entire string converted to lowercase and stripped of any whitespaces. The transformation looks as follows:

Director name		Transformation
Peter Hewitt	→	peterhewitt
Peter Hyams	→	peterhyams
Peter Macdonald	→	petermacdonald
Peter Horton	→	peterhorton
Peter Yates	→	peteryates

This removes any kind of mismatching while calculating the similarity scores and ensures the accuracy of the method is maintained.

4) Since the tag information is only provided for a small subset of the movies provided (about 35%), the tag information is purposefully excluded from the metadata. This is to avoid any inconsistencies across the prepared metadata as only a small number of movies will benefit from the additional tag information.

5)The external extracted data was cached in forms of csv files named pd_movieId_director and pd_movieId_plotline. This step ensures that during experimentation, numerous repeated calls to external data sources are avoided and computation time is saved.

4.b. Experimentation

Experimentation with different parameters to achieve best results. The experimentation state can be divided into two parts. The first part involved fine tuning the content based section to return a suitable number of movies to be ranked. The second part involved fine tuning the SVD model in order to achieve lowest RMSE value and most accurate predictions.
This involved tuning the hyperparameters for the model. Both the approaches and methods are mentioned in details below:

1) Returning the right number of movies for the content based section has a great effect on the final recommendation. Various numbers were tried and different runs were logged and manually verified to ensure the least RMSE values. The intuitive approach of passing the entire row from the similarity matrix after sorting fails to produce any accurate results. This is because when provided with the entire movie list, the pipeline is reduced to basically collaborative filtering. This gives a lot of false-positives because movies with low vote count are rated high and outrank valid predicted movies. Hence this approach is discarded. Similarly, if it returns only a small number of movies, say 5-10, the personal taste of the model is not captured because the recommendations now are mainly cosine similarity dominated with very little room for rearrangement by collaborative filtering. During the experimental phase, several runs indicated using the range 30-40 yields the best results. The provided model uses 35 as the number of movies returned.

2)Experimentation for the collaborative filtering involved fine tuning the hyperparameters of the SVD model and creating test and train splits on user-ratings data using different proportions. In order to automate the process and achieve the optimal hyperparameter values, the model makes use of GridSearchCV from the model_selection package. This function helps to loop through predefined hyperparameters and fit the model on your training set. The best hyperparameters are selected at the end. The parameters list for GridSearchCV to work with was follows:

Table 4: Hyperparameter tuning for model

Parameter	Values
n_epochs	10,20,30
lr_all	0.004, 0.008, 0.01
reg_all	0.4, 0.6

The following results were chosen as the best set of hyperparameters:

```
{'n_epochs': 30, 'lr_all': 0.01,
'reg_all': 0.4}
```

The given set of hyperparameters is able to achieve RMSE score of 0.8773 and MAE score of 0.6776 which for our prediction purposes is reasonable accuracy.

Cross-validation across the training data was used in order to create a robust model against underfitting and overfitting. It makes use of 20-80 test-train splits. The results from the cross validation process were as follows:

Table 5: Results from cross-validation

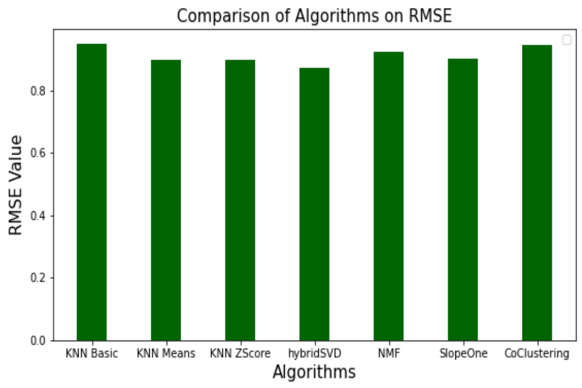
	RMSE	MAE	Test Time
Fold 1	0.8771	0.6744	0.21
Fold 2	0.8625	0.6631	0.16
Fold 3	0.8689	0.6681	0.26
Fold 4	0.8771	0.6718	0.21
Fold 5	0.8812	0.6773	0.16
Mean	0.8733	0.6709	0.20
Std	0.0067	0.0049	0.04

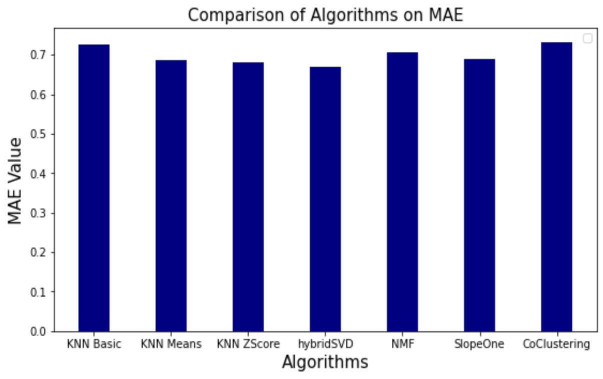
5. Evaluation & Results

The evaluation is performed using Root Mean Squared Error (RMSE) and MAE (Mean absolute error). The lower the RMSE, the better our model performs. Since SVD treats MF as an optimization problem for minimizing, SSE(Error Sum of Squares) which in turns minimizes the RMSE for better prediction results. To evaluate the model further we compare the accuracy of the proposed model against other baseline prediction models. For this evaluation KNN Basic, KNN Means, KNN ZScore, NMF(non-negative matrix factorization), Slope One and CoClustering and their respective RMSE values were used.

Algorithm	RMSE	MAE
HybridSVD	0.8773	0.6776
KNN Basic	0.9485	0.7264
KNN Means	0.8974	0.6851
KNN ZScore	0.8974	0.6809
NMF	0.9226	0.7065
SlopeOne	0.9013	0.6887
CoClustering	0.9459	0.7325

As evident from the evaluation, the proposed model performs the best. The results are depicted by the bar plots below:





As evident from the graphs the proposed model provides the best result when compared to other baseline prediction models. This results matches with our conclusion that hybrid model provides better accuracy than similarity based models and at the same time provides prediction based on user's personal taste by incorporating diversity and long tail into the final predictions.

Acknowledgement

We would like to thank Dr. Yang Liu for continued support and guidance throughout the process of development of this project.

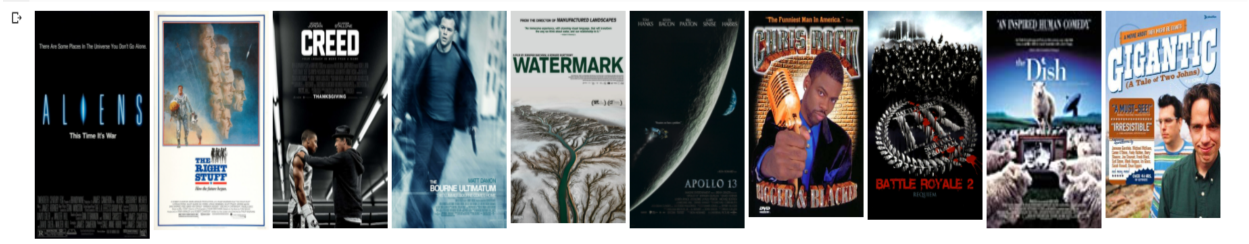
References:

1. S. Salmani and S. Kulkarni, "Hybrid Movie Recommendation System Using Machine Learning," *2021 International Conference on Communication information and Computing Technology (ICCICT)*, 2021, pp. 1-10, doi: 10.1109/ICCICT50803.2021.9510058.

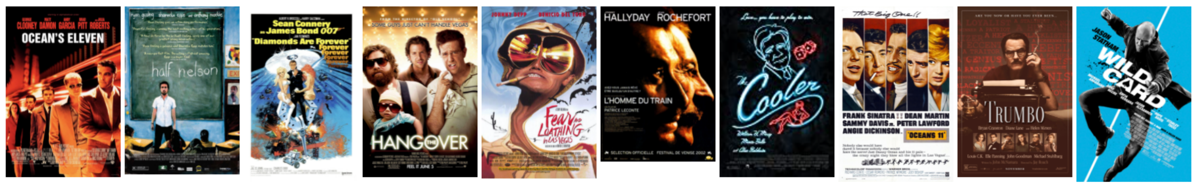
2. https://surprise.readthedocs.io/en/stable/index.html

Output

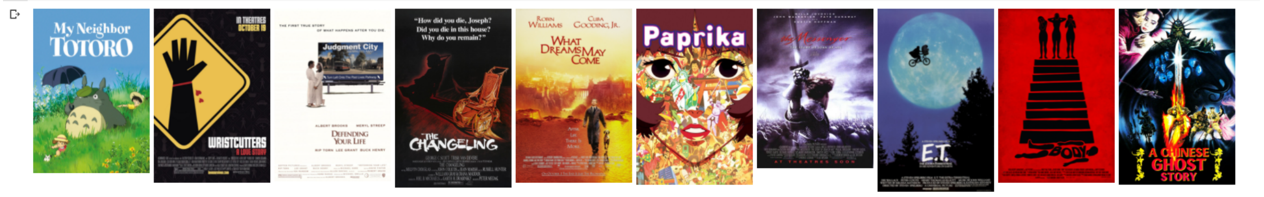
Screenshot of the recommended movies for user =146 and movie = From the Earth to the Moon (1998)



Screenshot of the recommended movies for user = 495 and movie = Leaving Las Vegas (1995)



Screenshot of the recommended movies for user = 321 and movie = Casper (1995)


My Neighbor TOTORO
WRISTCUTTERS
Judgment City
DEFENDING YOUR LIFE
"How did you die, Joseph? Did you die in this house? Why do you remain?"
THE CHANGELING
ROBIN WILLIAMS CUBA GOODING, JR.
WHAT DREAMS MAY COME
Paprika
E.T.
BODY
A CHINESE GHOST STORY