

**A HYBRID APPROACH TO MOVIE RECOMMENDER SYSTEM**

**FINAL REPORT**

Submitted by:

Rachit Aggarwal – M12506500

Prashant Gupta – M12506009

Ambika Kaushik – M12507845

Shashira Gunjepalli – M12505196

Machine Learning

**Prof. Anca Ralescu**

**ABSTRACT**

Recommender systems are used for predicting products that align with user's interest. These systems are capable of computing the similarities between users and products and generate efficient recommendations. The two most popular techniques for generating these systems are collaborative filtering and content-based filtering. Individually both approaches hold some limitation which degrades the final results. To overcome these problems a hybrid recommender is used that combines the best features of both methods. In this project, we aim to build a hybrid movie recommender system that performs collaborative filtering to cluster similar users and content-based filtering to cluster similar movies and finally make recommendations for a specific user.

**INTRODUCTION**

Almost every E-commerce website are focused towards providing the most suitable products to their users. These portals offer large number of products, which decreases the probability of user ever finding that product. To mitigate this issue, portals deploy a recommender system which learns the behavior of each user from their instances (ratings, views etc.) and recommends products accordingly. This reduces the effort required by users and providing good recommendations can enhance the success rate of the system.

Recommender systems are based on three common approaches collaborative filtering, content-based filtering and hybrid recommender system.

1. **Content-based Filtering**

In content-based filtering, the system uses different features of the product to make recommendations. For a movie recommender system the product is a movie and associated features can be genre, actors, directors etc., based on these features content-based filtering form clusters of similar movies. These clusters are formed according to the ratings provided a specific user. Movies with the highest ratings from these clusters will be recommended to the user. Content-based filtering has a limitation of being domain restricted. Since a fixed set of features are used in this approach, any product having different features will generate unfavorable results. For example, a movie recommender system can not be used to recommend news as features for both movies and news will be different.

1. **Collaborative Filtering**

Collaborative filtering uses preference values of each user and compute the recommendations. Users having most similar preference values as compared to a specific user are clustered together. Movies which have the highest ratings by users from the defined cluster are provided as final recommendations. Collaborative filtering overcomes the limitation of content-based filtering of being domain restricted, since instead of features about movies it emphasizes on preference value. Thus, it can be applied in any system irrespective of domain. But collaborative filtering has its own shortcomings like cold start, scalability and sparsity.

1. **Hybrid Recommender System**

A hybrid recommender system combines both collaborative filtering and content-based filtering. It can perform both approaches one after the other in any order or it may combine the results from individual approaches and compute the final results. Building a hybrid recommender reduces the probabilities of individual limitations of collaborative filtering and content-based filtering and is capable of providing more accurate results. Despite having many advantages, hybrid systems are quite complex to build.

In this project we will build a hybrid recommender system to recommend movies. Collaborative filtering will be used to cluster similar users and content-based filtering will be used to cluster similar movies with respect a specific user.

**LITERATURE REVIEW**

Recommender systems have been a prevailing topic of research since past 2 decades. Since these systems implements different techniques to provide better recommendations to users. These systems aim to reduce the human effort and recommend products which aligns with user interest. Many researchers have presented their different approach towards building these systems to provide improved results.

Kaushik & Tomar [1] presented the evaluation of different similarity functions using collaborative filtering. Different similarity functions generate different recommendations. These outcomes were analyzed and results were displayed. Sarwar et al. [2] experimented with different similarity techniques (item-item correlation vs. cosine similarity) and different techniques for obtaining recommendations (weighted average vs. regression model) and compared their result with simple KNN approach.

Davodi et al. [3] exploits the social aspects of expert’s behavior and constructed a kernel using background knowledge from *Wikipedia* repository. Using this they discovered the hidden relations between individuals and applied a content-based filtering for recommendations.

A healthy living program recommender system was developed by Zang et al. [4], which uses a hybrid approach for making recommendations. The recommendation system takes as input patient clinical data such as health conditions and vital observations and wellness data. They used Pearson’s similarity to measure the similarity between patients and using patient’s attributes as features k-nearest neighbor was used to make final recommendations.

Sharif & Raghavan [5] proposed a hybrid recommender system based on clustering of items using co-occurrence information of web pages and content information of pages. They used cosine similarity for computing similarities and used a hierarchical clustering for producing recommendations.

Modi & Narvekar [6] published a paper presenting an architecture that integrates product information with user’s access log data and generates a set of recommendations for a particular user. They deployed k-Means clustering algorithm in their model and Bayer Moore pattern matching algorithm was used to discover client interest items around the current client exercise.

**PROCEDURE**

The procedure is divided in two major parts, user similarity in which collaborative filtering will be implemented to obtain a cluster of similar users and in the other part using content-based filtering a set of movies will be identified which are similar to movies preferred by a specific user.

1. **User Similarity**

For user similarity, unsupervised learning will be used to form clusters of similar user. The data consists of *user\_id, movie\_id and ratings* where *user\_id* denotes the unique Id assigned to the user, *movie\_id*is the unique id for movies and *ratings* are the preference value provided by the user for each movie. The similarity measure will be computed with respect to one specific user.

|  |  |  |
| --- | --- | --- |
| **User\_Id** | **Movie\_Id** | **Ratings** |
| *User1* | *Movie1* | 4 |
| *User1* | *Movie2* | 3 |
| *User2* | *Movie2* | 4 |
| *User2* | *Movie4* | 2 |
| *User3* | *Movie1* | 3 |
| *User3* | *Movie2* | 3 |
| *User3* | *Movie4* | 5 |
| *User4* | *Movie1* | 2 |

**Table1: User-rating matrix.**

Let’s say we have to compute similarity for *user1*, we will first observe the movies rated by him and respected ratings. *User1* have rated movies *Movie1* and *Movie2*, only, users who have rated these movies will be considered and their preference instances for all other movies will be extracted.

|  |  |  |
| --- | --- | --- |
| **User\_Id** | **Movie\_Id** | **Ratings** |
| *User1* | *Movie1* | 4 |
| *User1* | *Movie2* | 3 |
| *User2* | *Movie2* | 4 |
| *User3* | *Movie1* | 3 |
| *User3* | *Movie2* | 3 |
| *User4* | *Movie1* | 2 |

**Table2: Relative instances**

Now users who have provided most similar ratings as *user1* for movies *Movie1* and *Movie2* will be clustered together to form the neighborhood of *user1*. Neighborhood means the set of most similar users for a specific user. According to example *user2* and *user3* will classified as the most similar user since the distance between their ratings on same movies is relatively less. Final cluster would be

|  |
| --- |
| **Cluster** |
| *User2* |
| *User3* |

**Table3: Similar user cluster**

1. **Product Similarity**

For product similarity, again unsupervised learning will be applied to extract the set of most similar movies. Features associated with each movie will be exploited to obtain the similarity measure and movies with highest similarity value will be clustered. Our data will possess *movie\_id, genre, frequency* and *over\_all\_rating.* We will count *genre, frequency* and *over\_all\_rating* as three features for each movie.

|  |  |  |  |
| --- | --- | --- | --- |
| **Movie\_Id** | **Genre** | **Frequency** | **OVR** |
| *Movie1* | *Action* | 30 | 8.5 |
| *Movie2* | *Comedy* | 40 | 7 |
| *Movie3* | *Romance* | 25 | 6.5 |
| *Movie4* | *Action* | 35 | 9 |
| *Movie5* | *Comedy* | 60 | 5 |

**Table4: Movie matrix.**

Referring to Table1, *user1* have provided preferences for two movies *movie1* and *movie2*. Since we are to make recommendations for *user1*, movies preferred by him will be considered. Now we have to compute the most similar movies to *movie1* and *movie2*. Based on the features, it is clear that *movie4* and *movie5* are relatively more similar and will be clustered together.

|  |
| --- |
| **Cluster** |
| *Movie4* |
| *Movie5* |

**Table5: Similar movie cluster**

**3. Recommendations**

Once we have obtained clusters from Table3 and Table 5, we can compute the recommendations. Users from the similar user cluster, having highest preference value for movies in similar movie cluster will be recommended to *user1*. Since similar user cluster contains the most similar user to *user1* and similar movie cluster contains most similar movies to the ones which *user1* have rated. Therefore, the most similar movies with highest ratings from most similar users will be the final recommendations.

**EXPERIMENT**

To perform different similarity measures and clustering in both user similarity and product similarity we have used specific similarity metrics. The experimentation was performed on a real data set.

1. **Dataset**

The data we used in this project is ml-latest-small provided by MovieLens [7]. It consists of user rating matrix and movie attributes. The ratings.csv contains 100,000 instances of 700 users, providing a rating for a movie. The format of the data is *<user\_id>, <movie\_id>, <rating>, <timestamp>*, where we will ignore the timestamps where *user\_id* and *movie\_id* is the unique ID assigned to each user and movie respectively, *ratings* is the preference value of user on each movie. File movie.csv contains the data for 9,000 movies in format *<movie\_id>, <title>, <genre>*, where *movie\_id* is unique ID assigned to each movie, *title* denotes the name of the movie and *genre* identifies the category of movie.

We have also scrapped data from IMDB to retrieve more attributes about each movie to use them as features to train our model. The attributes we extracted were *frequency, genre* and *overall\_ratings.* *Genre* are the specific genres associated with each movie, *frequency* is the number of times a movie have been rated b users and *overall\_ratings* consists of the average rating for each movie provided by IMDB. We have combined the data from both the sources and used it for our model.

1. **Similarity Measurement**

For computing the similarity between users and movies we have used cosine similarity metrics. The metrics was used to determine the similarity between different users and also the similarity between different movies. Cosine similarity measures the cosine angle between two vectors in normalized space. Lesser the angle between the vectors more will be the similarity measure. Advantage of this metric is not to be inclined towards the magnitude of vectors rather the orientation between them. To calculate the similarity measure between users, each user will be treated as a vector and the users having minimum cosine angle between them will be considered similar to each other. Similarly, for movies, each movie along with its features will be represented as vectors in normalized space and movies having minimum cosine distance will be the most similar. The value of cosine similarity ranges between 0 to 1, and is given as:

1. **Unsupervised Learning**

For forming clusters of similar users and movies we have deployed k-Nearest Neighbor method. kNN will be used to perform unsupervised learning that usually identifies the nearest neighbors of data point. Nearest neighbor are the data points which is at the least distance from a specific point. Distance between the point is directly proportional to the similarity between points. To judge the distance between two points a distance metrics is required by kNN.

In our model kNN will decide the similarity based on value of cosine similarity metric as discussed in previous section. To obtain the cluster of similar users, input to kNN algorithm will be the user-rating matrix as shown in Table1. Based on this input and the similarity metric the clusters of similar users will be created. For movies, all the features from Table4 will be provided to kNN along with the similarity metric do decide the most similar movies.

While using kNN we need to have a value for ‘k’, which means number of neighbors should be identified. In our case, it would denote the number of similar users or similar movies to be included in the final cluster. For both cases the value of k = 5, i.e. the cluster of 5 similar users and similar movies will be build.

The biggest advantage of kNN is to have a flexible decision boundary rather than a concrete one. Other advantages could be less computational time and high predictive power also interpreting the output value is quite easy.

**EVALUATION**

Evaluation was performed on the recommendations generated by our model. Once we have obtained the cluster of similar users and similar movies we can recommend movies to a user. For each movie in the similar movie cluster, we will consider the preference value of each user present in the similar user cluster. Movies with the highest preference values will be the final recommendations. We have designed the model to generate 10 best recommendations.

To evaluate our model, we must somehow compare the actual value for a movie provided by a user and value which is predicted by the model. To begin evaluation first step was to divide the dataset in to a ratio 75% - 25%, where 75% data was used for training and 25% was used for testing. To compute the predicted rating, we used the maximum rating provided by a similar user to a similar movie. Every recommended movie will be presented along with a predicted value. This predicted value is nothing but the preference value a user is expected to give for a particular movie.

The actual preference values from the test data will be compared against the predicted preference values for all the right recommendations. To calculate this comparison, we have used an evaluation metric called Normalized Root Mean Squared Error (NRMSE). It is a simple standard deviation between the actual values and the estimated values. It is given by:

Here denotes the actual preference value and denotes the preference value predicted by the system. RMSE will give the minimum error between the two values. To get the values of RMSE between 0-1 we will use a normalization technique, that is by dividing the RMSE value by the difference of maximum rating possible and minimum rating possible.

**RESULTS AND CONCLUSIONS**

We computed the cluster of similar users by using the user-rating matrix by applying cosine similarity to determine the similarity measure and kNN to create clusters. Similarly, we build a cluster of similar movies based on features and by using only the movies viewed by a specific user. We used the value of k = 5 each time we performed clustering Once we obtain these two metrics we provided the final recommendations as the similar movies which have highest preference values by each user in the similar user cluster. Total 10 most relevant movies were recommended to the user.

Finally, based on the recommendations we evaluated our model by using the actual preference values and the predicted preference values for the recommended movies. We used Normalized Root Mean Square Error to calculate the error. The final values of NRMSE we obtained was:

**FUTURE WORK**

The proposed model can be implemented using different datasets from different domains to observe the performance of the model. Different clustering algorithms could be implemented, and the results could be compared with proposed approach. Also distinct similarity metrics and evaluation metrics can be exploited to identify the changes and the quality of the results.

**REFERENCES**

[1]. Kaushik & Tomar. 2015. *Evaluation of Similarity Functions by Using User Based Collaborative Filtering approach in Recommendation Systems*. International Journal of Engineering Trends and Technology (IJETT) 2015. ISSN: 2231-5381.

[2]. Sarwar et al. 2001. *Item-Based Collaborative Filtering Recommendation Algorithm*. WWW10, May 1-5, 2001, Hong Kong. ACM 1-58113-348-0/01/0005.

[3]. Davoodi et al. 2012. *A Semantic Social Network-Based Expert Recommender System.* 12 October 2012 © Springer Science+Business Media, LLC 2012. Appl Intell (2013) 39:1–13 DOI 10.1007/s10489-012-0389-1.

[4]. Zang et al. 2014. *Automatically Recommending Health Living Programs to Patients with Chronic Diseases through Hybrid Content-Based and Collaborative Filtering.* 2014 IEEE International Conference on Bioinformatics and Biomedicine. 978-1-4799-5669-2/14.

[5]. Sharif & Raghavan. 2014. *A Cluster Based Scalable Hybrid Approach for Web Page Recommendation.* 2014 IEEE International Conference on Big Data. 978-1-4799-5666-1/14.

[6]. Modi & Narvekar. 2015. *Enhancement of Online Web Recommender System Using A Hybrid Clustering and Pattern Matching Approach.* 2015 International Conference on Nascent Technologies in the Engineering Field (ICNTE-2015). 978-1-4799-7263-0/15.

[7]. F. Maxwell Harper and Joseph A. Konstan. 2015. *The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4, Article 19 (December 2015), 19 pages. DOI=<http://dx.doi.org/10.1145/2827872>