A Survey to Build a Movie Recommender System Using Hybrid Filtering

T. Suvarna Kumari[1] , K.Sagar[2]
[1]Assistant Professor, CBIT, Hyderabad, Telangana
[2]Professor, CBIT, Hyderabad, Telangana
suvarnakumarit@gmail.com, kadapasagar@cbit.ac.in

Abstract:

Now a days, machine learning algorithms are playing a very important role recommender system. In this paper, we propose a survey to build a movie recommender system using Hybrid Filtering.

Keywords: Machine Learning Algorithms, Recommender System, Hybrid Filtering.

1. Introduction

In today's world, due to the continuous development of the internet technology, overload of the information has become a serious issue. Huge amount of data is being generated each hour. To get the information that we need, the search engines are useful up to some extent. But when we do not know what exactly we are looking for, they cannot provide the results which satisfy the user. The recommender systems help in regard to this problem by providing the user with the information based on his preferences, search patterns, etc. These help the user to find the information which is otherwise anonymous to him. Recommender systems are now playing a major role in various online applications like Amazon, YouTube, etc. During this survey we got to know how a recommender system works, what methods are used for the generation of the recommendations and we also have gone through some of the existing systems.

Recommender Systems

Recommender systems are used to provide personalized recommendations based on the users past behavior and his preferences. The services like YouTube, Amazon continuously monitor their users' actions, keep a record of them, mine the data to find any patterns and use this information to generate recommendations which are unique to each and every user. There are several ways to mine these patterns. These are called the filtering techniques, which are generally of three types-

a) Collaborative filtering
b) Content Based Filtering
c) Hybrid Filtering

These are explained in detail the next section.

1. Filtering Techniques

Collaborative Filtering:

This filtering technique is totally based in the users' previous history, that is, the movies that he has already watched and rated are taken into consideration. Now these ratings are stored. Similarly we have the ratings given to the movies by thousands of users. Now for a particular user a set of similar users are found based on the similar movies that they have watched and how similar their ratings are. Simply put, the logic here is that if an User A and another User B like an item then the item that are liked by user A may also be liked by the user B and vice versa[1].

Content Based Filtering:

In this technique, the attributes of the movies such as their genre, cast etc. are used to find the similar movies. That is if the user watches a movie then based on this movie's attributes a list of similar movies is generated. This may include the movies that are of same genre or include one or more of the cast members. Since the attributes are being used, the recommendations that are generated maybe repetitive and limited in scope[2].

Hybrid Filtering:

To overcome the disadvantages of a single method, two or more methods are combined together which may result to generation of better recommendations. These methods maybe combined in any manner. If two methods are being used then they both maybe applied separately and the combined result may be presented or after applying one method, the results of this method can be taken and the second method can be applied to these results[3].

Any of the methods above can be chosen to generate the recommendations but the final goal is to give the recommendations that may be useful to the user.

2. Finding Similarities

There are several methods to find the similarities among the users or the movies-

Cosine Similarity:

This method is most commonly used method in

collaborative filtering in recommender systems. Cosine similarity finds how two vectors are related to each other using measuring cosine angle between these vectors. For the user-based algorithm, Cosine Similarity is,

$$\text{Cos\_Sim(u,v)}=\frac{\vec{R}_u \bullet \vec{R}_v}{|\vec{R}_u| \cdot |\vec{R}_v|}$$

where, R_u, R_v are rating vectors of users u and v.
The major drawback with cosine similarity is that it considers null preferences as negative preference

Adjusted Cosine Similarity:

Cosine similarity measure does not consider the scenario in which different users use different rating scale. Adjusted cosine similarity solves it by subtracting the average rating provided by the user .Adjusted cosine similarity considers the difference in rating scale used by each user. Adjusted cosine similarity is slightly different from Pearson Correlation; Pearson Correlation considers the average rating of user u for co- rated the items. Adjusted cosine similarity subtracts the average rating of user u for all the items rated by user u.

$$Adjusted\_COS\_Sim=\frac{\sum_{i\in I}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_v)}{\sqrt{\sum_{i\in I}(r_{u,i}-\bar{r}_u)^2+\sum_{i\in I}(r_{v,i}-\bar{r}_v)^2}}$$

where, r_i is the ratings of the movies rated by user I itself.

Pearson Correlation:

This method is most commonly used method. This method is used to find linear correlate between two vectors. PCC results a value between -1 and +1. -1 represents a negative co relation while +1 represents high positive correlation. The value 0 shows no relation sometimes called zero order correlation. For the user-based algorithm, Pearson correlation is.

$$PCC\_Sim(u,v)=\frac{\sum_{i\in I}(r_{u,i}-\bar{r}_{u,I})(r_{v,i}-\bar{r}_{v,I})}{\sqrt{\sum_{i\in I}(r_{u,i}-\bar{r}_{u,I})^2+\sum_{i\in I}(r_{v,i}-\bar{r}_{v,I})^2}}$$

Where $\bar{r}_{u,I}$ and $\bar{r}_{v,I}$ represents Average rating of user u and user v ,

There are also several matrix factorization methods such as the Single Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF), Normal Prediction, Single Value Decomposition++ (SVDpp). These came to trend during the Netflix prize competition and these were also very efficient in generating the recommendations. The evaluation metrics such as Root Mean Square Value (RMSE), Mean Absolute Error (MAE), Fit time can be used to know which among the above method is more efficient.

In our survey, we have used the MovieLens100k small dataset. Upon performing the five cross validation for the above methods the results were as follows-

	RMSE	MAE	Fit Time
SVD	0.9349	0.7361	4.88
NMF	0.9613	0.7554	5.39
Normal Predictor	1.5187	1.2190	0.15
SVDpp	0.9207	0.7219	188.68

Table-1: Comparing different Matrix Factorization Techniques

$$AC(i,j)=\frac{\sum_{U\in u_{ij}}(r_{ui}-\bar{r}_u)(r_{uj}-\bar{r}_u)}{\sqrt{\sum_{U\in u_{ij}}(r_{ui}-\bar{r}_u)^2\sum_{U\in u_{ij}}(r_{uj}-\bar{r}_u)^2}}$$

Based on the above results we can see that the more effective methods (methods with less RMSE and MAE values) are SVD and SVDpp. But, in between these two, SVD has far less fit

Where in this r_{ui} representing the rating that is given by the user u to the item i and r_{uj} represents the rating that is given by user u to item j, r_u indicates the average ratings that are given by user u as a whole in the rating matrix.

3. Existing Systems

Recommender Systems using Item based collaborative filtering technique

The Dataset that is used in [4] is the Netflix user item ratings Data set maintained by the Group Lens Research Organization from the Movie lens website. These datasets were gathered over various intervals of time. And for our present system dataset used consists of one million ratings as preferences that are given by 6040 users over for 3952 movies. In contrast with the User based Collaborative filtering approach in which we will be looking for the most similar users, for the current user in Item based collaborative filtering approach we will be using the items that are most similar to the current item for which we are going to predict the rating by using the item similarity weights and using the K most similar items and predicting the unknown rating. The top n items with the highest predicted rating is returned as recommendations. MAE is used for evaluation.

Step 1:

To compute the similarity weight the adjusted cosine

formula is used Selection of neighbors has to be done more carefully so as to not affect the quality of recommendations generated. Hence we will be choosing the K most similar neighbors which are having the highest similarity compared to others.

Step2:

Selection of neighborhood: time than SVDpp and hence we came to a conclusion that SVD is more effective

Step3:

Prediction of unknown ratings is done as follows

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u(i)} W_{ij} r_{uj}}{\sum_{j \in N_u(i)} |W_{ij}|}$$

Where W_{ij} represents similarity weight between items i and j. r_{ij} will be representing the rating that is given by the user u to the item j. $N_{u(i)}$ represents users that have rated item i.

Step 4:

In the last step, the top N values with higher predicted ratings are recommended to the user.

$$MAE(f) = \frac{1}{|R_{test}|} \sum_{r_{ui} \in R_{test}} (f(u,i) - r_{ui})$$

Implementation of Recommender System Using Graph database

Graph Database is a new NoSQL database based on graph theory. Its data storage structure and data query methods are based on graph theory. In graph theory, the basic elements of a graph are nodes and edges. In graph database, nodes and relationships correspond to each other. The output of the system shows 50 recommendations by default. The result is depicted in the form of nodes and edges. Larger the node and larger the thickness of the edges, more the movie is recommended. Different colours are used to depict the nodes with different weightages. In a relational database, the relations between two tables or multiple tables are mutually referenced by using foreign key constraints. In graph databases, through relationships, we can associate nodes together to build complex models that are closely related to the problem domain. Each node in the graph database model directly contains a relational list, which stores the relationship records between the node and other nodes in the relational list.

The data model obtained by using graph database to organize data in [3] is simpler and more expressive than using traditional relational database or other NoSQL database. It can be used to model and manage data applications in a simple and intuitive way, and it can also make data units smaller and more standardized. At the same time, it can realize rich relational links.

Based on the above advantages of the database, the database technology used for the movie recommender system in [3] is Neo4j. Neo4j is an open source NoSQL graph database implemented by Java, which realizes the storage of graph data model at the professional database level.

Py2neo is a library using which you can build, query, modify, and delete nodes and relationships, no need to use the cypher language. And flask framework is used to render the front pages. Ajax encapsulated in jQuery implements the transfer of parameters between front and back. The front end uses Bootstrap framework. And the graph is drawn using E-charts.

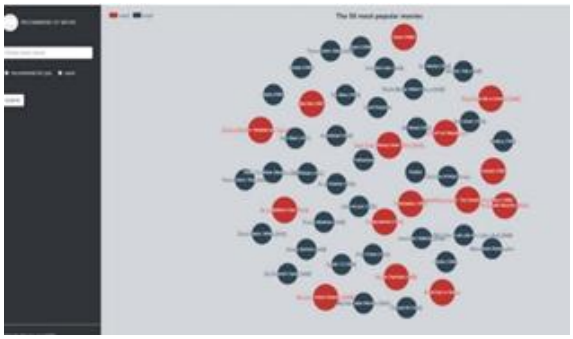

Fig 1: Graph Showing Weightages of all movies

From these only the top movies are given to the user.

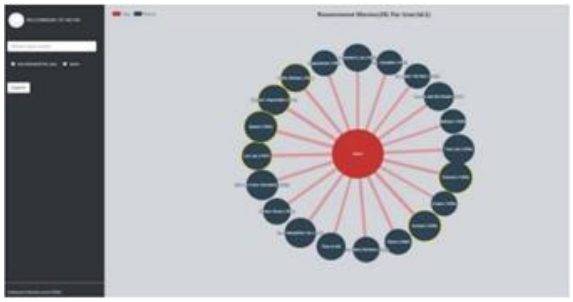

Fig 2: Graph Showing top movies recommended to the user

Weight based movie recommendation system using K-means Algorithm

In [2], the dataset that is used is the IMDB dataset. The movie recommendation system that has been developed depends on five movies attributes to make a recommendation. It does so by balancing the five

attributes in a proper way and then providing the user with a recommendation.

The user will go to a web page and will input some attributes like the genre, actor, year and rating. It means that the type of genre that the user likes, or the actor that they prefer, or the year which they would like and the rating of the movie they would want to see, will be input by them. After that, based on the inputs, the system built in [2] will recommend movies.

In the user input stage, the user has to first come to the recommendation page and enter five attributes which are based on the user's favorite actors, directors, genre, year and ratings; and based on the entries, the movies are recommended according to the similarities in the attributes of the movies and the user's input.

In the data fetched stage, data is fetched from the dataset based on the user input and an array of suitable movies is prepared. The movies included in the array have at least one matching attribute value with the input value of the user.

Then the total weight of each movie is calculated for each attribute

Mt= Total no. of movies in dataset

Ma = No. of movies of Actor in data set *Md*= No. of movies of Director in data set *Mg*= No. of movies of Genre in data set *My* = No. of movies of Year in data set

Actor (Wa) : If actor name for a particular movie match with the user preferences actor name from the user activity log or the user input then the weight of the actor will be (Ma/Mt)+.4

else (Ma/Mt)

Director (Wd) : If director name for a particular movie match with the user preferences director name from the user activity log or the user input then the weight of that director willbe

(M_d/M_t) +.3 else (Md/Mt)

Genre (Wg) : If genre for a particular movie match with the user preferences genre from the user activity log or the user input then the weight of the genre will be

(Mg/Mt).2 else (Mg/Mt)

Year (Wy) *:* If year for a particular movie match with the user preferences year from the user activity log or the user input then the weight of the year will be

(Mg/Mt).1 else (Mg/Mt)

Total weight of a particular movie is given by

$Wm = Wa + Wd + Wr + Wg + Wy$

After the total weight of the particular movie is calculated, the no. of movies in array is counted with the help of a counter. If the counter value is less than or equal to twenty the movie list sorted according to the weighted value associated with the movies is displayed. If a number of movies are greater than twenty then a pre-filter is applied and the top twenty movies according to weighted values are selected. When both movies have a same rating and the same weighted value than the priority is given to the movie having a large number of votes.

After this, k-means algorithm is applied on the weighted values. We assume k to be equal to 4 so that in average every k has five movies, where k is the number of clusters. Euclidean Distance is used to calculate the distance between data points and centroid.

Once the final clusters are formed then the mean movie rating of all clusters is computed. Then according to the input user query the cluster of movies having highest mean cluster rating are displayed.

A Generic Recommender System based on Neural Networks

Artificial Neural Networks are classification algorithms inspired by the functioning of the neurons in the human brain. It consists of layers of neurons, with each layer depending on the next layer. Their versatility has been of great advantage in solving in many problems in the learning sphere and outside. There exist many structures, of which Feed forward is the most popular. Method of trial and error is used to randomly initialize weights in the network to converge to an optimum value. Many learning algorithms can be employed to fulfill the task. One such algorithm is Back propagation

Artificial Neural Network with feed forward architecture can be trained for supervised learning using Back Propagation (BP) technique. BP is used heavily in classification and prediction problems, mapping a non-linear relationship from input to output.

Here a feed forward network is used to learn from user-rating matrix (R_{UI}) and content of items C_{IF} and build a user profile for each user u. The number of neurons in the input layer is F, i.e., the number of features used to represent items I, along with a bias unit $f_0 = 1$. Since the system is content boosted, there is no need to train the system for generating item profile, as in the case of model based CF. Training for only the user profile reduces the computation time and thus makes the process faster and scalable. The accuracy of the system is tested by varying the number of hidden layers and

neurons in them, starting with 1 hidden layer with F+1 neurons, including the bias. The output layer will be a 5-neuron layer corresponding to the rating1-5.

The training of the network is done through Back propagation. After training the network for each user in U by using $C_{IF,}$ the set of minimized weights is considered as the user profile. The building of the user profile constitutes the model based collaborative filtering technique. The network first runs through the training data, whose test error is calculated. Then the test error of the test data training is calculated, which should be minimum or equal to the previous test error.

These user profiles are used in the memory based collaborative filtering technique instead of the normal user-item rating data. Doing this helps in eradicating the data sparsity problem, since the user profiles would be dense matrices and thus the similarity between users would be derived more efficiently. To produce item recommendations for user u, the similarity of u's profile with the rest of the other users' profiles in U is calculated. Cosine similarity is used to find the correlation between the users as depicted as

$$sim(u,v) = \frac{u'v'}{\sqrt{\sum_{i=1} u'^2_i}\sqrt{\sum_{i=1} v'^2_i}}$$

The results of this similarity ranges from [-1, 1] where -1 means exactly opposite, 1 means exactly same, and 0 represents independence. Ten users' profiles with highest similarity value, or all profiles with a similarity value ≥ 0.5, whichever is lower, are chosen to help in producing recommended items. Then predicted rating for user u is generated for all the items not rated by him. The rating predicted for item i by user u is given as

$$p_{ui} = r_u + \frac{\sum_{i=1} sim(u,v)(r_{vi} - r_v)}{\sum |\, simm(u,v)\,|}$$

where $$r_u = \frac{\sum_{i=1}^{l} r_{ui}}{|\,I\,|}$$

The predicted ratings for all i is sorted out with highest rating first and the first 10 items is provided to u (user) along with the predicted ratings.

4. Conclusion

Recommender systems are very much useful to the users in finding the items in which they may be interested in. There are several methods to generate the recommendations of which the matrix factorization methods provide better results when compared to the similarity measures.

The methods mentioned in [2], [3], [4], [5] are successful but, they have their disadvantages. The method in [3] has a problem of cold start. The method in [2] takes explicit feedback from the user and also euclidean distance is used to measure the similarity here. The system in [4] may provide repetitive results. Thus to eliminate these drawbacks, hybrid altering approaches can be used. Future Work We have limited maximum part of our survey in the data mining field. There are also several methods that make use of Natural Language Processing techniques, Neural Network techniques and Deep Learning techniques and generate more effective set of recommendations. Further study can be done on such methods.

References

[1] Ajay Agarwal, Minakshi Chauhan, "Similarity Measures used in Recommender Systems", International Journal of Engineering Technology Science and Research, Volume 4, Issue 6, 2017.

[2] M. T. Himel, M. N. Uddin, M. A. Hossain and Y. M. Jang, "Weight based movie recommendation system using K-means algorithm," *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, 2017, pp. 1302-1306.

[3] N. Yi, C. Li, X. Feng and M. Shi, "Design and Implementation of Movie Recommender System Based on Graph Database," *2017 14th Web Information Systems and Applications Conference (WISA)*, Liuzhou, 2017, pp. 132-135.

[4] L. T. Ponnam, S. Deepak Punyasamudram, S. N. Nallagulla and S. Yellamati, "Movie recommender system using item based collaborative filtering technique," *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*, Pudukkottai, 2016, pp. 1-5.

[5] A. Gupta and B. K. Tripathy, "A generic hybrid recommender system based on neural networks," *2014 IEEE International Advance Computing Conference (IACC)*, Gurgaon, 2014, pp. 1248-1252