International Journal of Advanced Science and Technology
Vol.110 (2018), pp.65-76
http://dx.doi.org/10.14257/ijast.2018.110.07

Recommendation Engine for Predicting Best Rated Movies

Vivek P. Khadse, Akhil P, Syed Muzamil Basha, N.Ch.S.N. Iyengar[2]
and Ronnie D. Caytiles[3]

VIT, Vellore-632014
[2]Sreenidhi Institute of Science and Technology, Ghatkesar, Hyderabad, India
[3]Multimedia Engineering Department, Hannam University, Daejeon, Korea
srimannarayanach@sreenidhi.edu.in

Abstract

Recommendation systems are a subclass of information filtering system that seek to predict the "rating" or "preference" that a user would give to an item. Recommendation systems have become extremely common in recent years, and are utilized in a variety of areas: some popular applications include movies, music, news, books, research articles, search queries, social tags, and products in general. For this study, we are considering movie recommendation system based on content from a movie dataset. Generally, two types of filtering methods used in recommendation system are content based filtering and collaborative based filtering. For this study, we are using these two methods not only on single but on different contents of movie information like rating, genre. Our recommendation engine would consider previously stored ratings and genre of the movie selected by user, to train the system and project movie name list that the user may like. In this study, for building the recommendation engine we have used content based algorithms and collaborative filtering algorithms available in GraphLab package in python. Finally, these two methods are compared to show which recommendation engine algorithm works better compared to other for most of the time.

Keywords*: Recommendation systems, Content based filtering. Collaborative filtering*

1. Introduction

Recently internet charges became cheaper and then the users of internet became increasing day by day. As internet users became increasing the data contents also get increased. This causes the user overloaded with lots of information. Recommendation engine helps to deal with this overload, by categorizing this large list of information, which can later be used for recommending. These days recommendation systems are found in variety of services, such as news, music, movies, online shopping sites and many others. Recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" that a user would give to an item. Recommender systems have become increasingly popular in recent years, and are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts, collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners, and Twitter pages. Recommender systems typically produce a list of recommendations in one of two ways – through collaborative and content-based filtering or the personality-based approach. Collaborative filtering approaches building a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by

Received (September 26, 2017), Review Result (January 4, 2018), Accepted (January 18, 2018)

other users. This model is then used to predict items (or ratings for items) that the user may have an interest in. Content-based filtering approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties. These approaches are often combined. Recommender systems are a useful alternative to search algorithms since they help users discover items they might not have found by themselves. Interestingly enough, recommender systems are often implemented using search engines indexing non-traditional data.

2. Existing Products and Services

Social networking sites began to emerge with www.classmates.com in 1995, and it gained popularity among people. Different specialized website only for review like Yelp and Tribe allow user to give review on different types of products. Different blogs website like Windows Live Spaces, Vox and Blogger started using reviews for different blogs posted by its users. Some academic communities also emerged in 90's which started using this concept to rate the research work or different cited work on their website. Later on, some more complex procedures are adopted by Professional website like LinkedIn, Ecademy and ActiveRain. These website not only considered star ratings but also different user parameters like work experience, technologies known and domain knowledge. In early 2000, many match making website emerged like Match.com and Multiply which started using this concept of recommendation for suggesting most appropriate bride or groom. Professional photographic website for posting and sharing like Picasa also joined the race recommending different photos based on user search. There were are other organization in this sector like Webshots, MyPhotoBucket and Flickr. Social Networking websites like Orkut, Google+ and Facebook used same concept of recommendation with new name as chat, where user were allowed to give real time open reviews along with images. But recommendation is extensively used by one of the video sharing major YouTube and then used by different websites like Last.fm, Pandora, Netflix. After this Recommendation Engine started serving variety of purposes, targeting all types of users as shown in Figure 1.

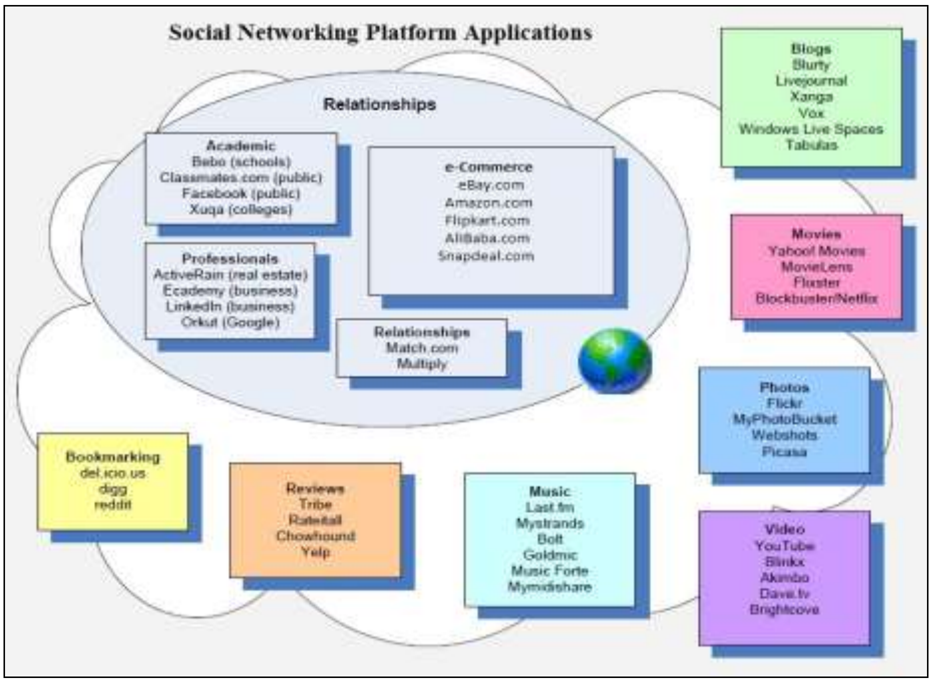


Figure 1. Association of Different Websites with Social Network [9]

3. Social Movie Platforms

Here we have taken Movie Lens – 100k dataset for testing and evaluating recommendation engine. Our project has significant improvements when compared with existing products and services. Existing movie websites such as IMDB, AOL functions by providing global user ratings of each movie on their database. The Movies are categorized by metadata information such as era, genre, and directors and so on. So, this system lacks personal recommendation system and doesn't have the advantage of crowd wisdom and social networking communities' opinions. Yahoo! Movies uses personal ratings and suggest movies currently playing on TV, theatre and whose DVD is out. Other movie site, such as Flixster, uses a different approach. Flixster makes web based communities among movies and suggest the movies what their friends have rated.

4. Related Work

Netflix is another popular website which is used by people to watch movies and TV channels. Netflix also provides the video that the user may like to watch based on his previously watched videos. For this recommendation Netflix mainly uses two algorithms. Algorithms used are Restricted Boltzmann Machines (RBM) and a form of matrix Factorization. Restricted Boltzmann Machines uses neural networks, while form of matrix Factorization uses SVD++. This is an asymmetric form of SVD which uses implicit information.

Movie Recommendations from User Ratings is a paper by Hans Byström which used k-means clustering and softmax regression classification for movie recommendation. Root mean square error (RMSE) is used for evaluation of results. In this study, whole movie data set is considered and classified using Softmax Regression. Regression is performed on numeric values. So, here only rating column from the data set is used. This will recommend all movies that come under certain star rating. This is a first level of search criteria where user wants all best movies irrespective of its genre or year of released.

Movie recommendation system based on user's similarity is a paper by Gaurav Arora. In this study, the recommendation designed compared the taste of users and arrange the users who have the same taste. Recommendation will suggest the movies to other persons who are in same category. When user search any movie name, similar genre movies which is reviewed previously are suggested. In this study, Data Science techniques like City Block Distance and Euclidean Distance is used. Similarity measure is calculated between searched genre and respective genre from data set. Based on this similarity measure top rated movies are recommended.

4.1. Research Gap

Movie recommendation is still have many drawbacks. For movie recommendation some uses comparison with others who have the same taste, which is not an effective method for recommendation, while some others used global ratings of movies to suggest best movies. Here we have taken the dataset of movies which is rated by individuals and also taken the genre of the movie while predicting. So the user can select the best movie based on genre and rating.

5. Problem Considered

Existing movie websites like IMDB, AOL movies *etc.* provides information about the movies we selected only. Also the ratings are fixed by them. It has no connection how the people rated the movies on various social networking sites. Here we planned to add this feature to the users. Here the users can rate the movies after logging in. Based on each users rating the data will be trained.

6. Design

One approach to the design of recommender systems that has wide use is collaborative filtering. Collaborative filtering methods are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself. Many algorithms have been used in measuring user similarity or item similarity in recommender systems. For example, the k-nearest neighbor (k-NN) approach and the Pearson Correlation as first implemented by Allen.

6.1. Collaborative Filtering

Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. When building a model from a user's behavior, a distinction is often made between explicit and implicit forms of data collection as shown in Figure 3.

Examples of explicit data collection include the following:

- Asking a user to rate an item on a sliding scale.
- Asking a user to search.
- Asking a user to rank a collection of items from favorite to least favorite.
- Presenting two items to a user and asking him/her to choose the better one of them.
- Asking a user to create a list of items that he/she likes.

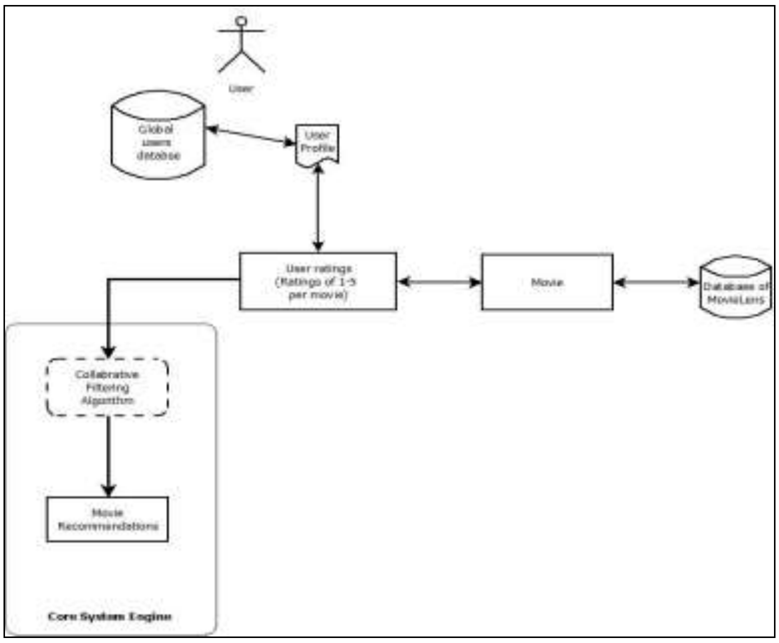


Figure 2. Collaborative Filtering Algorithm for Movie Recommendation

Examples of implicit data collection include the following:

- Observing the items that a user views in an online store.
- Analyzing item/user viewing times.[23]
- Keeping a record of the items that a user purchases online.

- Obtaining a list of items that a user has listened to or watched on his/her computer.
- Analyzing the user's social network and discovering similar likes and dislikes.

Collaborative filtering approaches often suffer from three problems: cold start, scalability, and sparsity.

- **Cold start:** These systems often require a large amount of existing data on a user in order to make accurate recommendations.
- **Scalability:** In many of the environments in which these systems make recommendations, there are millions of users and products. Thus, a large amount of computation power is often necessary to calculate recommendations.
- **Sparsity:** The number of items sold on major e-commerce sites is extremely large. The most active users will only have rated a small subset of the overall database. Thus, even the most popular items have very few ratings.

6.2. Content Based Filtering Algorithm

Content-based filtering methods are based on a description of the item and a profile of the user's preference. In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past (or is examining in the present). In particular, various candidate items are compared with items previously rated by the user and the best-matching items are recommended. This approach has its roots in information retrieval and information filtering research as shown in Figure 3.

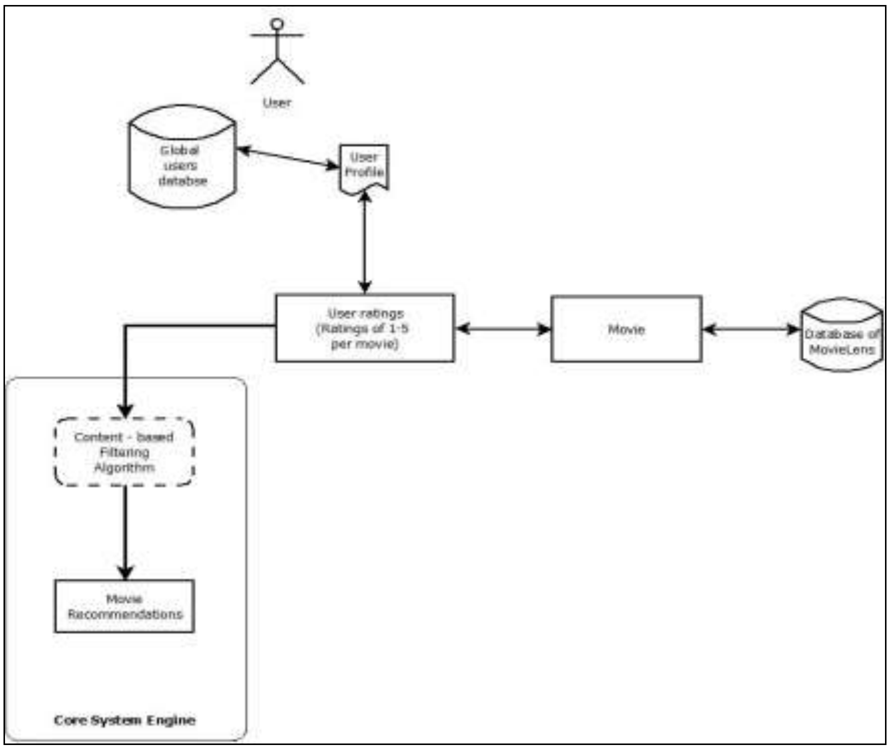


Figure 3. Content Based Filtering Algorithm for Movie Recommendation

To abstract the features of the items in the system, an item presentation algorithm is applied. A widely used algorithm is the tf–idf representation. A key issue with content-based filtering is whether the system is able to learn user preferences from users' actions regarding one content source and use them across other content types. When the system is limited to recommending content of the same type as the user is already using, the value from the recommendation system is significantly less than when other content types from other services can be recommended. For example, recommending news articles based on browsing of news is useful, but would be much more useful when music, videos, products, discussions *etc*. from different services can be recommended based on news browsing.

7. Methodology

7.1. Graphlab

As The GraphLab project was started by Prof. Carlos Guestrin of Carnegie Mellon University in 2009. It is an open source project using an Apache License. While GraphLab was originally developed for Machine Learning tasks, it has found great success at a broad range of other data-mining tasks; out-performing other abstractions by orders of magnitude. GraphLab provides a high level programming interface, allowing a rapid deployment of distributed machine learning algorithms. The main design considerations behind the design of GraphLab are:

- Sparse data with local dependencies
- Iterative algorithms
- Potentially asynchronous execution

Main features of GraphLab are:

- A unified multicore and distributed API: write once run efficiently in both shared and distributed memory systems
- Tuned for performance: optimized C++ execution engine leverages extensive multi-threading and asynchronous IO
- Scalable: GraphLab intelligently places data and computation using sophisticated new algorithms
- HDFS Integration
- Powerful Machine Learning Toolkits

GraphLab is a paid package for python. But for educational purposes they give it free to students for 1 year. Those who need can register in their site to get free access to all its features for 1 year. We used two main functions from GraphLab to implement the project.

The functions are:

- graphlab.popularity_recommender.create() – Which is used to implement content based recommendation system.
- graphlab.item_similarity_recommender.create() – Which is used to implement collaborative filtering recommendation system.

7.2. Dataset Description

For this study, we have chosen Movie Lens – 100k dataset for testing and evaluating recommendation engine. This data setcontains dataconsisting of total 90570 number of observation, of 943 users along with their ratings and reviews for at least 20 movies. This data set contains different columns such as Movie_Id, Movie_Name, Genre, Rating, Review, Release_Year and imdb_link. For designing recommendation engine, we will be using Movie_Id, Genre, Rating and Review columns.

7.3. Building the Engine

Recommendation system is built with the help of GraphLab and python. The entire dataset is loaded into python. Secondly, data set is divided intraining data and testing data for popularity recommendation algorithm and content-based filtering algorithm. Finally, this divided data sets are used for designing recommendation engine.

7.4. Making Recommendations

Now use the trained data for building popularity recommendation and content based filtering algorithms. The engine will provide appropriate recommended movies to the user according to genre he specified.

7.5. Evaluating the Algorithm

After recommending movies, each algorithm isevaluated separately using Precision-Recall method. Precision and Recall helps us to know that how much the predicted result is precise and how much correctly it recalled.

Precision is the fraction of retrieved results that are relevant and is given as Equation 1.

$$precision = \frac{Total\, no.of\ correct\ predicted\, records}{Total\, no.of\ records} \tag{1}$$

Recall is the fraction of relevant instances that are retrieved and is given in Equation 2,

$$recall = \frac{Total\, no.of\ correct\ predicted\, records}{Total\, no.of\ corrected\ records} \tag{2}$$

8. Implementation

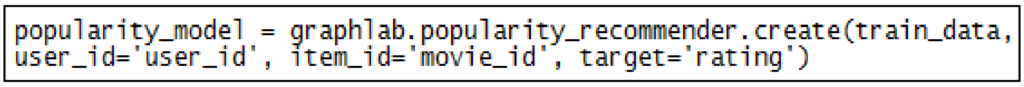
Use First download GraphLab from this site “https://turi.com/download/install-graphlab-create.html”. Register for free trail or as academic user for free use of the software. Follow the given procedures in the site for the successful installation of software. Before that ensure that latest version of python is installed and configured the latest pip version in your system.

Now download the Movie Lens data set for testing and analyzing the algorithm. Dataset can be downloaded from site “http://grouplens.org/datasets/movielens/100k/”. This data set consists of:

- 1,00,000 ratings (1-5) created from 943 users for about 1682 movies.
- Each user has rated a minimum of 20 movies.
- It contains genre information of movies.
- It also contains details of users like their age, gender, occupation and zip.

Load this dataset into python.

Next train the Simple Popularity model on the loaded dataset. The code is given below:

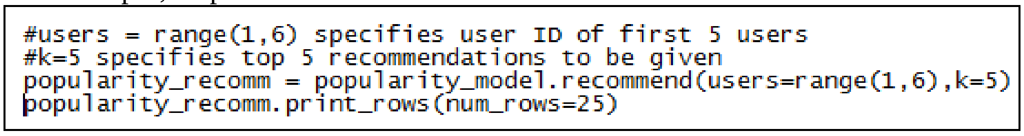


```
popularity_model = graphlab.popularity_recommender.create(train_data,
user_id='user_id', item_id='movie_id', target='rating')
```

Parameters:

- Train_data: contains required training data.
- User_id: column which represents each user id.
- Item_id: here we mention the item to be recommended.
- Target: column representing ratings given by user.

For example, Top 5 recommendations for first five users:



```
#users = range(1,6) specifies user ID of first 5 users
#k=5 specifies top 5 recommendations to be given
popularity_recomm = popularity_model.recommend(users=range(1,6),k=5)
popularity_recomm.print_rows(num_rows=25)
```

Our next method to implement is Collaborative Filtering Model. The training data is created by:

```
item_sim_model = graphlab.item_similarity_recommender.create
(train_data, user_id='user_id', item_id='movie_id', target='rating',
similarity_type='pearson')
```

Using this trained data movie recommendation can be done as follows:

```
item_sim_recomm = item_sim_model.recommend(users=range(1,6),k=5)
item_sim_recomm.print_rows(num_rows=25)
```

9. Results and Discussions

For evaluating recommendation engines, we have used two concepts *i.e.* Precision and Recall. This helps us to compare two models more accurately.

Precision:

- Out of all the recommended items, how many the user actually liked?
- If 5 items were recommended to the user out of which he liked say 4 of them, then precision is 0.8.

Recall:

- What ratio of items that a user likes were actually recommended.
- If a user likes say 5 items and the recommendation decided to show 3 of them, then the recall is 0.6

If we simply recommend all the items, it will definitely cover the items that any particular user liked but we think about precision then it would not be maximized in this case for every instance. For example, if we recommend say 1000 items and user like only say 10 of them then precision is 0.1%. This is really low. Our aim is to maximize both precision and recall.

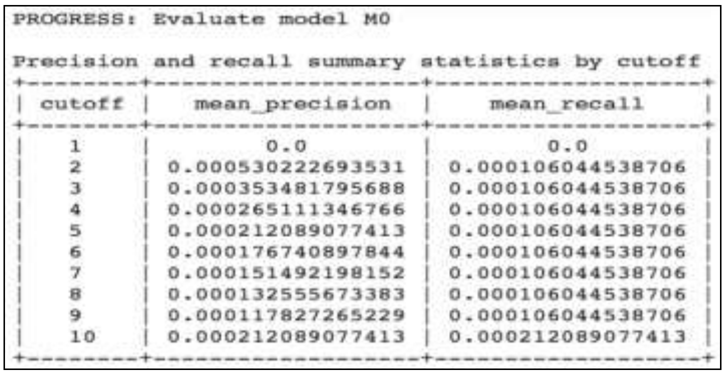
Model Performance based on precision and recall can be used as

```
model_performance = graphlab.compare(test_data, [popularity_model, item_sim_model])

graphlab.show_comparison(model_performance,[popularity_model, item_sim_model])
```

Results for Popularity Recommendation Model are as given in Table 1.

Table 1. Precision – Recall Table for Popularity Model

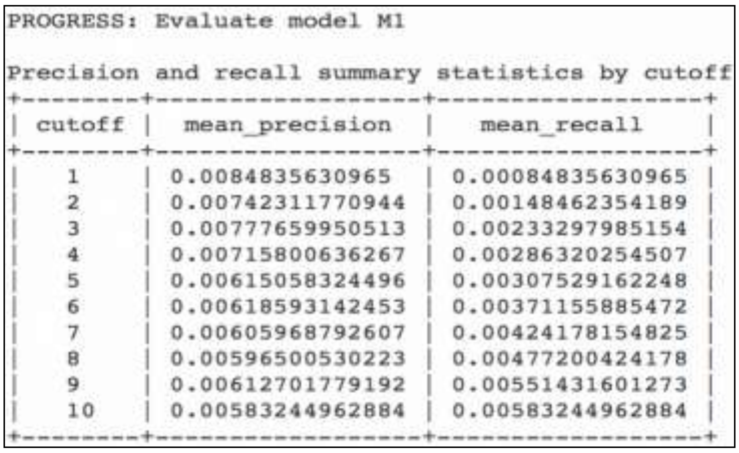


```
PROGRESS: Evaluate model M0

Precision and recall summary statistics by cutoff
+--------+----------------------+----------------------+
| cutoff |    mean_precision    |     mean_recall      |
+--------+----------------------+----------------------+
|   1    |         0.0          |         0.0          |
|   2    | 0.000530222693531    | 0.000106044538706    |
|   3    | 0.000353481795688    | 0.000106044538706    |
|   4    | 0.000265111346766    | 0.000106044538706    |
|   5    | 0.000212089077413    | 0.000106044538706    |
|   6    | 0.000176740897844    | 0.000106044538706    |
|   7    | 0.000151492198152    | 0.000106044538706    |
|   8    | 0.000132555673383    | 0.000106044538706    |
|   9    | 0.000117827265229    | 0.000106044538706    |
|   10   | 0.000212089077413    | 0.000212089077413    |
+--------+----------------------+----------------------+
```

Results for Item Similarity or Collaborative Filtering Model are as shown in Table 2:

Table 2. Precision – Recall Table for Item Similarity Model



```
PROGRESS: Evaluate model M1

Precision and recall summary statistics by cutoff
+--------+-----------------+------------------+
| cutoff |  mean_precision |    mean_recall   |
+--------+-----------------+------------------+
|   1    | 0.0084835630965 | 0.00084835630965 |
|   2    | 0.00742311770944 | 0.00148462354189 |
|   3    | 0.00777659950513 | 0.00233297985154 |
|   4    | 0.00715800636267 | 0.00286320254507 |
|   5    | 0.00615058324496 | 0.00307529162248 |
|   6    | 0.00618593142453 | 0.00371155885472 |
|   7    | 0.00605968792607 | 0.00424178154825 |
|   8    | 0.00596500530223 | 0.00477200424178 |
|   9    | 0.00612701779192 | 0.00551431601273 |
|   10   | 0.00583244962884 | 0.00583244962884 |
+--------+-----------------+------------------+
```

Above results are obtained after implementing Popularity Model and Item Similarity Model methods on the dataset consisting of 90570 number of observation which has different reviews given by943 number of users. From the above results we can make two very quick observations that

- The item similarity model is definitely better than the popularity model (by atleast 10x)
- On an absolute level, even the item similarity model appears to have a poor performance. It is far from being a useful recommendation system. Graph plot for Popularity Recommendation Model is as shown in Figure 4.

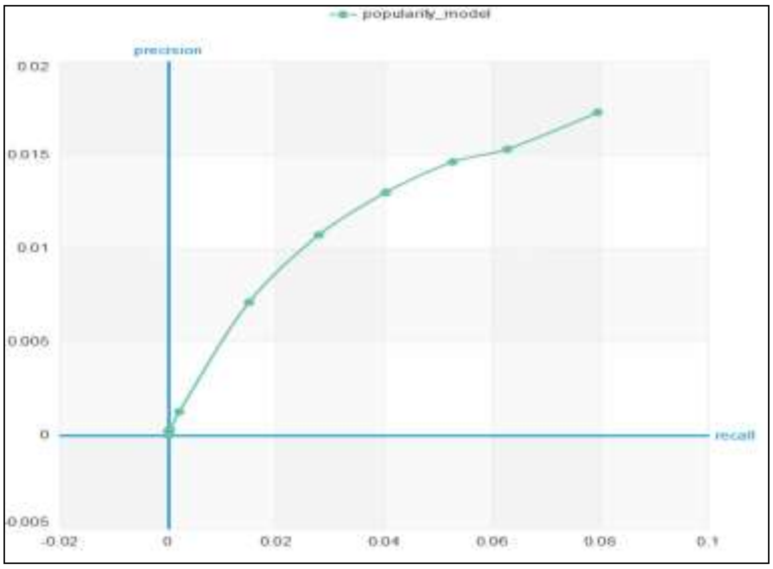


Figure 4. Precision-Recall Graph for Popularity Model

Above figure represents the graph plotted on Precision vs Recall values of Popularity Model. This graph is plotted by considering 10 points from among total 90570. If more value are considered then the graph will move downwards towards Recall more, which states that it predict correct results but probability of predicting correct results from total no. of values in less. Similarly, Graph plot for Item Similarity Mode is as shown in Figure 5.

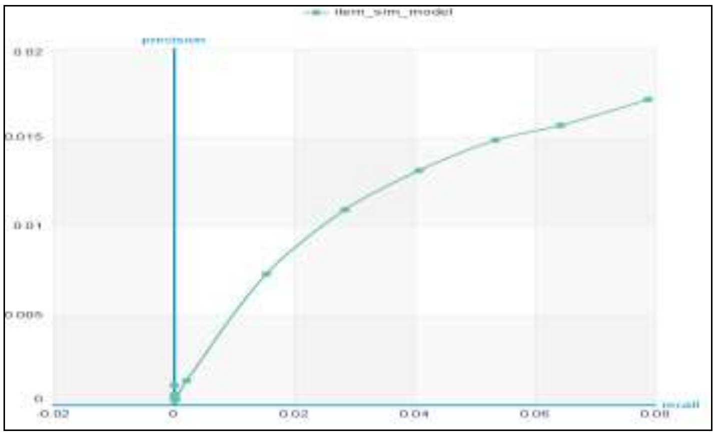


Figure 5. Precision-Recall Graph for Popularity Model

Above figure represents the graph plotted on Precision vs Recall values of Item Similarity Model. This graph is plotted by considering 10 points from among total 90570. If more value are considered then the graph will move upward towards linearly, which states that it can recall and predict correct results. Hence, increase the probability for predicting correct results. Their combined result can be represented as shown in Figure 6.

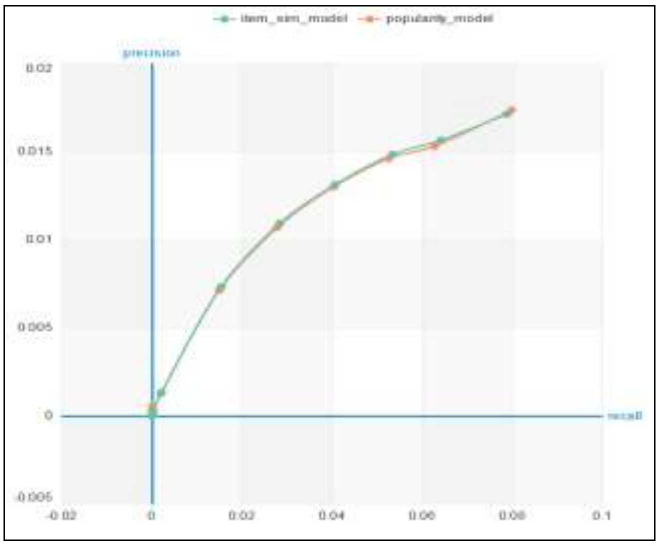


Figure 6. Combined graph of Precision-Recall for Popularity and Item Similarity Model

From the above we can clearly see that Item Similarity Model or Collaborative Filtering Model is always better compared to simple Popularity Model based on ratings given by users. The graph of Popularity Model moves downwards towards Recall more compare to Item Similarity Model. This clearly show that Popularity model will give accurate results but it will try to overfit the results *i.e.* it will try to give maximum number of movie name that user searched for. But as only rating is used there will be some movies which user doesn’t like may be displayed because it has high rating given by some other user. Here, Item Similarity play a vital role where we are comparing more than one column which helps to give precise correct results.

10. Conclusions

In this study we have used, user based and genre aspect of recommendation system. An ideal recommender system is the one which only recommends the items which user likes. So, maximizing the search result should not be an ultimate aim but it should be to maximize both precision and recall. User wants definite results and not a list of result where the user need to search for his/her required results in this case favorite movie. We have worked on the same where user gets his/her required specific results and not all possible results. Thus, maintaining both Precision and Recall. For future work, we can extend the filters to different metadata available for the dataset such as release data, actors as well as taking real time data for the movies from social networking sites. We can also increase the efficiency by leveraging the additional context information which we have. We can also use more sophisticated algorithms like matrix factorization. Again we can combine this two recommendation engine to give more appropriate results based on search filters. If filter does not specify more information, then a simple popularity model can be used or we can more for more complex content based collaborative filtering model.

References

[1] G. Adomavicius and A. Tuzhilin- “Towards the next generation of recommendation systems: a survey of the state-of-the-art and possible extensions”, Knowledge and Data Engineering, IEEE Transactions on, vol. 17, no. 6, **(2005)**, pp. 734–749.
[2] P. Melville, R.J. Mooney and R. Nagarajan, “Content-Boosted Collaborative Filtering for Improved Recommendations”, Proc. of the 18th National Conference on Artificial Intelligence, **(2002)**.
[3] G. Xue, C. Lin and Q. Yang, “Scalable Collaborative Filtering Using Cluster-based Smoothing”, Proc. of 28th Annual International ACM SIGIR Conference, **(2005)**.
[4] A. Carvalho and P. Calado, “Movie Recommendation based on User Interests”, Instituto Superior Tecnico 1049-001 Lisboa, **(2005)**.
[5] G. Stockman and L. G. Shapiro,“Computer Vision”, Prentice Hall PTR, 1st edition, **(2001)**.
[6] M. Mandl, E. Teppan and M. Schubert, “Consumer decision making in knowledge-based recommendation”, Journal of Intelligent Information Systems, vol. 37, no. 1, **(2011)**, pp. 1–22.
[7] M. Luisa Hernández-Alcaraz, W. Carrer-Neto, F. García-Sánchez and R. Valencia-García “Social knowledge-based recommender system. Application to the movies domain”, Expert Systems with Applications, vol. 39, no. 12, **(2012)**, pp. 10990 – 11000.
[8] L. Zhao and Z. Lu, “Matrix Factorization+ for Movie Recommendation”, Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16), **(2011)**.
[9] G. Arora1 and A. Kumar, “Movie Recommendation System Based On User’s Similarity”. IJCSMC, vol. 3, no. 4, April **(2014)**, pp.765 – 770.
[10] H. Byström, “Movie Recommendations from User Ratings”, Stanford University, USA. (Reference document), **(2015)**.

Authors

Vivek Khadse, he was a M. Tech. graduate student with specialization in Big Data Analytics from Vellore Institute of Technology has an interest in Machine Learning and Data Science. He had worked on many company based projects that involve innovation and product development and has a knowledge in Finance Domain. He is currently working in the field of Machine Learning, designing Recommendation engine and prediction models.

Akhil.P, he was a M. Tech. graduate student with specialization in Big Data Analytics from Vellore Institute of Technology has an interest in Machine Learning and Data Science. He had worked on many company based projects that involve innovation and product development and has a knowledge in Finance Domain. He is currently working in the field of Machine Learning, designing Recommendation engine and prediction models.

Syed Muzamil Basha, he had his Bachelor of Science in Information Technology at SITAMS, MTech in Information Technology (Networking) at VIT University and currently doing his research at VIT University. His research area are Wireless Sensor Networks, Text Mining and Big Data Predictive Analytics.

N. Ch. S. N. Iyengar (b 1961), he currently Professor at the *Sreenidhi Institute of Science and Technology (SNIST) Yamnapet, Ghatkesr, Hyderabad, Telengana, India.* His research interests include Agent-Based Distributed Computing, Intelligent Computing, Network Security, Secured Cloud Computing and Fluid Mechanics. He had 32+ years of experience in teaching and research, guided many scholars, has authored several textbooks and had nearly 200+ research publications in reputed peer reviewed international journals. He served as PCM/reviewer/keynote speaker/ Invited speaker. He received 2017 achievement award for his contributions in teaching and research by IOSRD.

Ronnie D. Caytiles, he had his Bachelor of Science in Computer Engineering- Western Institute of Technology, Iloilo City, Philippines, and Master of Science in Computer Science– Central Philippine University, Iloilo City, Philippines. He finished his Ph.D. in Multimedia Engineering, Hannam University, Daejeon, Korea. Currently, he serves as an Assistant Professor at Multimedia Engineering department, Hannam University, Daejeon, Korea. His research interests include Mobile Computing, Multimedia Communication, Information Technology Security, Ubiquitous Computing, Control and Automation