Movie Recommendation System Using Genome Tags and Content-Based Filtering

Syed M. Ali, Gopal K. Nayak, Rakesh K. Lenka and Rabindra K. Barik

Abstract Recommendation system has become of utmost importance during the last decade. It is due to the fact that a good recommender system can help assist people in their decision-making process on the daily basis. When it comes to movie, collaborative recommendation tries to assist the users by using help of similar type of users or movies from their common historical ratings. Genre is one of the major meta tag used to classify similar type of movies, as these genre are binary in nature they might not be the best way to recommend. In this paper, a hybrid model is proposed which utilizes genomic tags of movie coupled with the content-based filtering to recommend similar movies. It uses principal component analysis (PCA) and Pearson correlation techniques to reduce the tags which are redundant and show low proportion of variance, hence reducing the computation complexity. Initial results prove that genomic tags give the better result in terms of finding similar type of movies, and give more accurate and personalized recommendation as compared to existing models.

Keywords Movie recommendation · Genome tags · Content-based filtering Vector space

S. M. Ali · G. K. Nayak · R. K. Lenka (✉)
IIIT Bhubaneswar, Bhubaneswar, India
e-mail: rakeshkumar@iiit-bh.ac.in

S. M. Ali
e-mail: syedmohdali121@gmail.com

G. K. Nayak
e-mail: gopal@iiit-bh.ac.in

R. K. Barik
KIIT University, Bhubaneswar, India
e-mail: rabindra.mnnit@gmail.com

M. L. Kolhe et al. (eds.), *Advances in Data and Information Sciences*, Lecture Notes in Networks and Systems 38, https://doi.org/10.1007/978-981-10-8360-0_8

1 Introduction

With the rapid development of Internet, data is growing at a very high pace, having said so many online movie platforms are exploding with new content everyday. Recommendation systems have proved to be one of the successful information filtering system. In general, recommendation systems are used to predict how much user may like a certain product/service, compose a list of N best items for user, and compose a list of N users for a product/service. With this growth of media, users have to spend significant amount of time searching for the movies in which they are interested [1]. Here, the task of recommendation system is to automatically suggest users what movie to watch next based to the current movie and thus saving their time searching for the related content.

Movie recommendation is the most used application on the media streaming Web sites, both in academics and as well as commercially research has been done extensively in this topic. The Netflix Prize challenge [2] is one such example where a prize of one million dollar was at stake. The aim of the competition was to beat the Netflix's very own recommender system by ten percent. This attracted various researchers and companies and more than forty thousand entries were submitted for the same. Most of these recommender use collaborative filtering mechanism which has been developed in recent few years [3–5]. First the ratings of movies are collected given by each individual and then the recommendation is given to the users based on similar type of people with similar taste in the past. Many popular online services like netflix.com, youtube.com have used this collaborative filtering technique to suggest media to users.

This work is an attempt at implementation of a recommender system which uses genome tags to find out similar types of movies and then basic content-based filtering technique to further enhance the results. MovieLens dataset [6] is used for the development of this recommender engine and is accessed through its public FTP interface [7].

2 Related Work

2.1 Recommendation Using Collaborative Filtering

Collaborative filtering recommender relies heavily on the users data or some contribution in order to make recommendation. Contributions may include users to give ratings, like, dislikes, or other kinds of feedback which can cluster similar type of users together. As the name suggest, is a way of recommendation for a user in "collaboration" with other users. The fundamental of this filtering lies in the fact that if a person A shares same interest as of B, for certain object(here movie), then A will more likely share the same interest as of B, for a different object, than that of a randomly chosen person [8]. Collaborative filtering is easy to implement, it works well in most of the cases and has ability to find links between items which are otherwise considered dissimilar [9]. One of the drawbacks that this system suffers from "cold start" problem, this happens when either a new user comes or new item is added and we do not have much information/feedback for the item/user [10, 19]. To overcome this problem, recommender user content-based recommendation techniques coupled with collaborative recommendation.

2.2 Recommendation Using Content-Based Filtering

Recommendation is purely made on the attribute of the item, hence avoiding "cold start" problem. The attribute of an item, for example, in a movie can be its genre, year, running time, rating, starring actors can be used by the content-based recommender to make a movie recommendation. This concept has its root from the information retrieval theory where a document representation methods can abstractly encapsulate features of an item for potential recommendation [11, 20].

Over the period of time content-based recommender starts building profile for a person, it includes the taste of an individual which is extremely helpful for highly personalized recommendation [3]. This type of recommender does not require community data as it solely relies on the individual's preference, hence explanation can be given why a certain item/media was recommended. One major disadvantage of this is that it requires contents which can be broken down into meaningful attributes.

3 Proposed Approach

The dataset which was downloaded for the research contained 24404096 ratings and 668953 tag applications across 40110 movies. These data were created by 259137 users between January 09, 1995 and October 17, 2016. And was generated on October 18, 2016. The dataset contained the following files "genome-scores.csv", "genome-tags.csv", "links.csv", "movies.csv", "ratings.csv", and "tags.csv". Description of who's is as follows

- genome-scores.csv: Contains the genome score of the movies corresponding to the tags.
- genome-tags.csv: Contains genome tag id and its corresponding string.
- links.csv: Contains the link to the other sources of movie data.
- movies.csv: Contains information about movie like its title, movie id, and genres.
- ratings.csv: Each row of this file contains rating of one movie by one user.
- tags.csv: Each row of this file represents one tag applied to one movie by one user.

3.1 The Genome Tags

Netfilx and Youtube are using hybrid of collaborative and content-based filtering, the prominent feature of which is genre of the video or movie. The major problem with the genre is that they are binary in nature, i.e., they do not tell till what extent that genre applies to the certain content. A user may apply the tag ‘violent’ to ‘Fight Club’, indicating that it is a violent movie, but they might not indicate how violent the movie is. Just like in Human Genome Project where all the genes in human DNA were identified and mapped, the researchers were inspired to find and index the building blocks of their media. Pandora has developed their own Music Genome Project [12], similarly for movie recommendation GroupLens research laboratory developed tag genome [13]. The genome tag extends the traditional tagging system to give the enhanced user interaction. Genome tag contains item and its relationship to the set of tags. These range between 0 and 1, where 1 being the most relevant and 0 being the least. This creates a dense matrix in which every movie in the genome has a value for every tag. This can be used to recommend similar type of content.

3.2 Data Preparation

Since genome score does not consist all the movies present in the dataset, first task was to select only those movies who’s genome score we have. After filtering, we were left with around 10,000 movies. Next we transformed genome score which was stored like in Table 1 to like in Table 2.

After the transformation of genome scores average rating and number of users who rated the particular movie was calculated. This will come in handy while coupling our model with content-based filtering. Average rating for a particular movie *i* was calculated simply using the formula, total rating given by each user to that movie divided by the total number of users rated that movie, i.e.,

$$avg\_rating_i = \frac{\Sigma user\_rating_i}{\Sigma user_i} \tag{1}$$

Table 1 Before transformation

MovieId	TagId	Relevance
1	1	0.02400
1	2	0.02400
1	3	0.05475
1	4	0.09200

Table 2 After transformation

MovieId/tagId	1	2	3	…	1128
1	0.024	0.024	0.0548	…	0.0252
2	0.038	0.0418	0.037	…	0.0202
3	0.042	0.0525	0.0272	…	0.0200
4	0.036	0.0385	0.035	…	0.0140

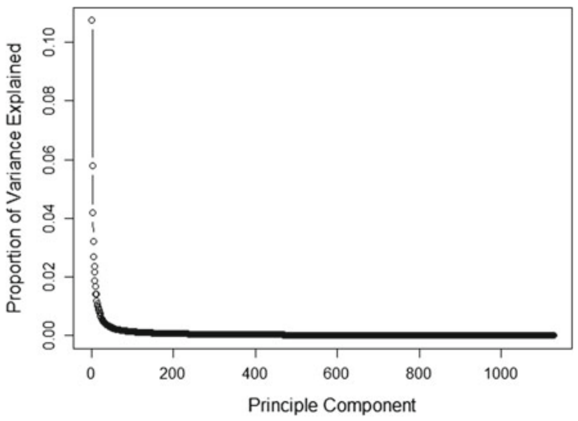
3.3 Feature Reduction

We have total of 1128 genome tags, which are very large and many of them are redundant, this will increase the computational complexity. There is need to reduce the number of features, it will not only remove redundancy in data but will also increase the performance of the model [14]. Principal component analysis (PCA) was run on the complete genome score in order to the variance explained in data by the tags available [15].

As it is quite clear from the Fig. 1 that most of the tags show very low variance in data and can be reduced. We used correlation-based feature selection technique, for which Pearson correlation method was preferred [16]. Pearson correlation find out linear correlation between two variable X and Y. It results in the value between −1 and +1 where −1 is the total negative correlation and +1 being the total positive. Correlation between all the tags were calculated, if the tags were to be correlated they should have a value between 0 and +1. We were suppose to choose the optimal threshold value above which we will say tags are correlated else not, lets say this to be cutoff value. PCA was ran after selecting the cutoff to be 0.6, 0.5, 0.4, and 0.3 (Figs. 2 and 3).

Choosing cutoff to be 0.6 and 0.5 will still leave some of the redundant tags, whereas if cutoff is set to be 0.3 we might loose some important tags, so for this

Fig. 1 PCA on compete set of tags



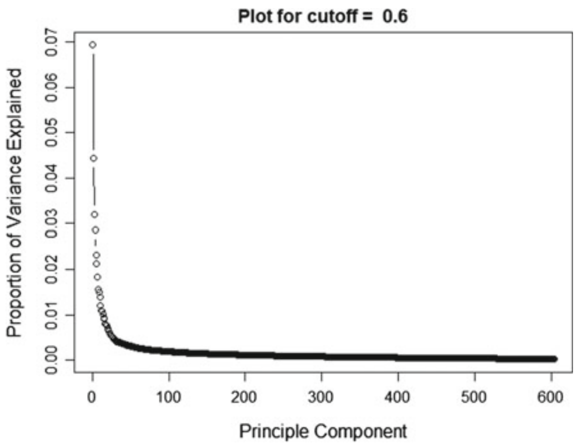


Fig. 2 PCA after cutoff was 0.6

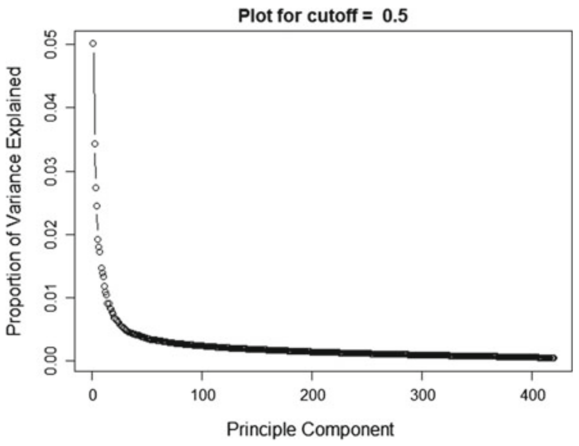


Fig. 3 PCA after cutoff was 0.5

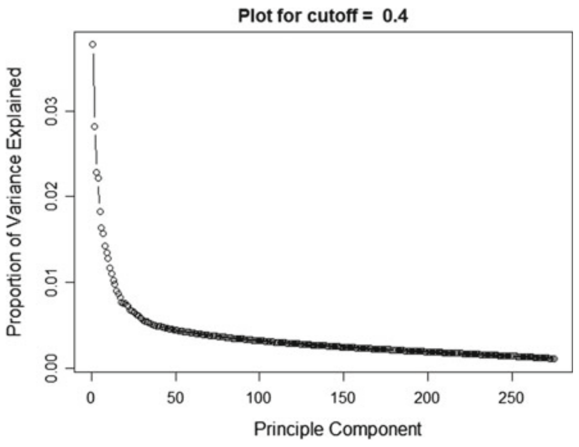


Fig. 4 PCA after cutoff was 0.4

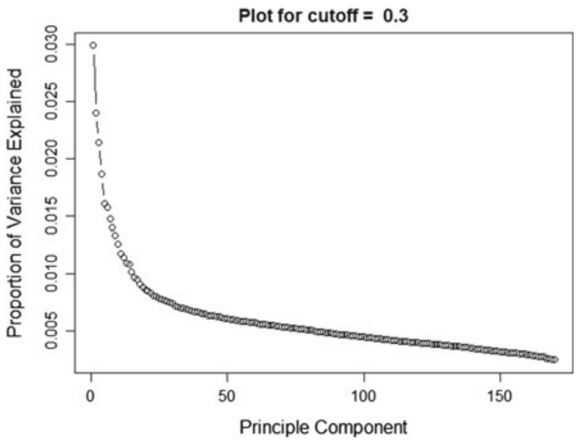


Fig. 5 PCA after cutoff was 0.3

model cutoff was set to be 0.4. With this cutoff, the number of tags was dramatically reduced from 1128 to 275 (Figs. 4 and 5).

3.4 Distance Between Movies

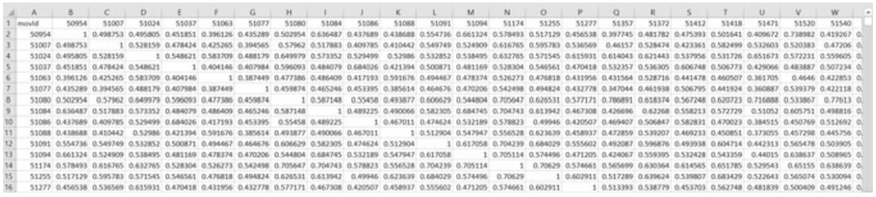
Now we are left with 275 attributes and over 10,000 rows, we can use this to find out which movies are similar to whom. Vector space model approach was used to achieve this task [17]. In vector space model, we represent documents (or any object in general, here movie) as vectors of identifier. Relevance between these vectors can be find by comparing the deviation between then angles of all the vectors [18]. In practice, cosine of the angle between vector is calculated (Fig. 6).

$$\cos(\theta) = \frac{a_1 \cdot a_2}{\|a_1\| \|a_2\|} \tag{2}$$

where $a_1 \cdot a_2$ is the dot product.

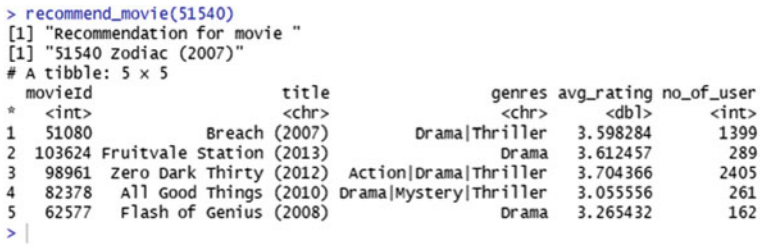
$$\|a_1\| = \sqrt{\sum_{i=1}^{n} {q_i}^2} \tag{3}$$

In our dataset, every row can be treated as a vector, hence we can find out the cosine distance between each of them. For the demonstration purpose, we have chosen sample 2000 movies in this experiment. We obtained the 2000 * 2000 matrix M, where M_{ij} will contain the value of the cosine angle between the $movieId_i$ and $movieId_j$ (Fig. 7).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
1	movid	50954	51007	51024	51037	51063	51077	51080	51084	51086	51088	51091	51094	51174	51255	51277	51357	51372	51412	51418	51471	51520	51540	
2	50954	1	0.498753	0.495805	0.451851	0.396126	0.435289	0.502954	0.636487	0.437689	0.438688	0.554736	0.661324	0.578493	0.517129	0.456538	0.397745	0.481782	0.475393	0.501641	0.409672	0.738982	0.419267	0.
3	51007	0.498753	1	0.528159	0.478424	0.425265	0.394565	0.57962	0.517883	0.409785	0.410442	0.549749	0.524909	0.616765	0.595783	0.536569	0.46157	0.528474	0.423361	0.582499	0.532603	0.520383	0.47206	0.
4	51024	0.495805	0.528159	1	0.548621	0.583709	0.488179	0.649979	0.573352	0.529499	0.52986	0.532852	0.538495	0.632765	0.571545	0.615931	0.614043	0.621443	0.537956	0.531726	0.651673	0.572231	0.559605	0.
5	51037	0.451851	0.478424	0.548621	1	0.404146	0.407984	0.596093	0.484079	0.684026	0.421394	0.500871	0.481169	0.528304	0.546561	0.470418	0.532357	0.536305	0.606748	0.506773	0.429066	0.483887	0.507234	0.
6	51063	0.396126	0.425265	0.583709	0.404146	1	0.387449	0.477386	0.486409	0.417193	0.591676	0.494467	0.478374	0.526273	0.476818	0.431956	0.431564	0.528716	0.441478	0.460507	0.361705	0.4646	0.422853	0.
7	51077	0.435289	0.394565	0.488179	0.407984	0.387449	1	0.459874	0.465246	0.453395	0.385614	0.464676	0.470206	0.542498	0.494824	0.432778	0.347044	0.461938	0.506795	0.441924	0.360887	0.539379	0.422118	0.
8	51080	0.502954	0.57962	0.649979	0.596093	0.477386	0.459874	1	0.587148	0.55458	0.493877	0.606629	0.544804	0.705647	0.626531	0.577171	0.786891	0.618374	0.567248	0.620723	0.716888	0.533867	0.77613	0.
9	51084	0.636487	0.517883	0.573352	0.484079	0.486409	0.465246	0.587148	1	0.489225	0.490066	0.582305	0.684745	0.704743	0.613942	0.467308	0.426696	0.62268	0.558213	0.572729	0.51052	0.605751	0.498816	0.
10	51086	0.437689	0.409785	0.529499	0.684026	0.417193	0.453395	0.55458	0.489225	1	0.467011	0.474624	0.532189	0.578823	0.49946	0.420507	0.469407	0.506847	0.582831	0.470023	0.384515	0.450769	0.512692	0.
11	51088	0.438688	0.410442	0.52986	0.421394	0.591676	0.385614	0.493877	0.490066	0.467011	1	0.512904	0.547947	0.556528	0.623639	0.458937	0.472859	0.539207	0.469233	0.450851	0.373055	0.457298	0.445756	0.
12	51091	0.554736	0.549749	0.532852	0.500871	0.494467	0.464676	0.606629	0.582305	0.474624	0.512904	1	0.617058	0.704239	0.684029	0.555602	0.492087	0.596876	0.493938	0.604714	0.442313	0.565478	0.503905	0.
13	51094	0.661324	0.524909	0.538495	0.481169	0.478374	0.470206	0.544804	0.684745	0.532189	0.547947	0.617058	1	0.705114	0.574496	0.471205	0.424067	0.559395	0.532428	0.543359	0.44015	0.638637	0.508965	0.
14	51174	0.578493	0.616765	0.632765	0.528304	0.526273	0.542498	0.705647	0.704743	0.578823	0.556528	0.704239	0.705114	1	0.70629	0.574661	0.565699	0.630364	0.614565	0.651785	0.529543	0.65155	0.638639	0.
15	51255	0.517129	0.595783	0.571545	0.546561	0.476818	0.494824	0.626531	0.613942	0.49946	0.623639	0.684029	0.574496	0.70629	1	0.602911	0.517289	0.639624	0.539807	0.683429	0.522643	0.565074	0.530094	0.
16	51277	0.456538	0.536569	0.615931	0.470418	0.431956	0.432778	0.577171	0.467308	0.420507	0.458937	0.555602	0.471205	0.574661	0.602911	1	0.513393	0.538779	0.453703	0.562748	0.481839	0.500409	0.491246	0.

Fig. 6 Cosine angle between movies

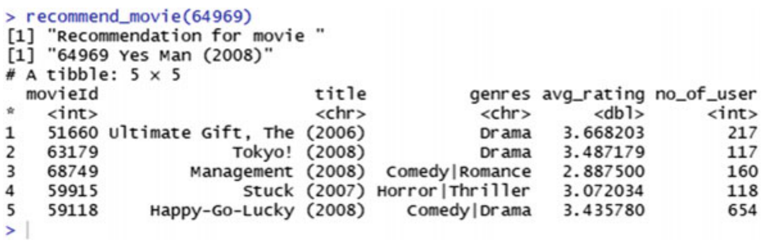


```
> recommend_movie(51540)
[1] "Recommendation for movie "
[1] "51540 Zodiac (2007)"
# A tibble: 5 × 5
  movieId                     title               genres avg_rating no_of_user
*   <int>                     <chr>                <chr>      <dbl>      <int>
1   51080            Breach (2007)       Drama|Thriller   3.598284       1399
2  103624 Fruitvale Station (2013)                Drama   3.612457        289
3   98961  Zero Dark Thirty (2012) Action|Drama|Thriller   3.704366       2405
4   82378   All Good Things (2010) Drama|Mystery|Thriller  3.055556        261
5   62577   Flash of Genius (2008)                Drama   3.265432        162
> |
```

Fig. 7 Recommendation for “Zodiac”

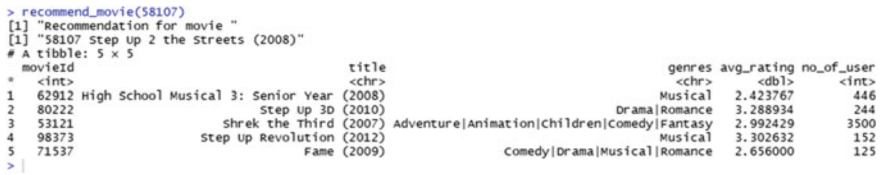
3.5 Recommending Movies

Now when we have obtained the cosine matrix, the recommendation is fairly easy from here. Suppose a user watching a movie with movieId *n*, then we will go to the *n*th of the matrix and sort out the row in decreasing order. Greater the cosine value, smaller the angle, smaller the angle more close the movies are in vector space, more closer the movie more similar it is. We will pick up top K results from the same. Now using content-based filtering on average rating of the movie, we will recommend top N movies to the users (Fig. 8).



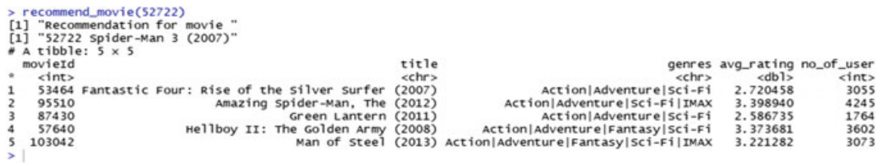
```
> recommend_movie(64969)
[1] "Recommendation for movie "
[1] "64969 Yes Man (2008)"
# A tibble: 5 × 5
  movieId                        title          genres avg_rating no_of_user
*   <int>                        <chr>           <chr>      <dbl>      <int>
1   51660 Ultimate Gift, The (2006)           Drama   3.668203        217
2   63179               Tokyo! (2008)           Drama   3.487179        117
3   68749           Management (2008)  Comedy|Romance   2.887500        160
4   59915                Stuck (2007)  Horror|Thriller   3.072034        118
5   59118       Happy-Go-Lucky (2008)    Comedy|Drama   3.435780        654
> |
```

Fig. 8 Recommendation for “Yes Man”



```
> recommend_movie(58107)
[1] "Recommendation for movie "
[1] "58107 Step Up 2 the Streets (2008)"
# A tibble: 5 x 5
  movieId                                     title                                        genres avg_rating no_of_user
*   <int>                                     <chr>                                         <chr>      <dbl>      <int>
1   62912 High School Musical 3: Senior Year (2008)                                       Musical   2.423767        446
2   80222                        Step Up 3D (2010)                                 Drama|Romance   3.288934        244
3   53121                   Shrek the Third (2007) Adventure|Animation|Children|Comedy|Fantasy   2.992429       3500
4   98373               Step Up Revolution (2012)                                       Musical   3.302632        152
5   71537                             Fame (2009)                  Comedy|Drama|Musical|Romance   2.656000        125
> |
```

Fig. 9 Recommendation for “Step Up 2”



```
> recommend_movie(52722)
[1] "Recommendation for movie "
[1] "52722 Spider-Man 3 (2007)"
# A tibble: 5 x 5
  movieId                                           title                              genres avg_rating no_of_user
*   <int>                                           <chr>                               <chr>      <dbl>      <int>
1   53464 Fantastic Four: Rise of the Silver Surfer (2007)           Action|Adventure|Sci-Fi   2.720458       3055
2   95510                    Amazing Spider-Man, The (2012)      Action|Adventure|Sci-Fi|IMAX   3.398940       4245
3   87430                             Green Lantern (2011)           Action|Adventure|Sci-Fi   2.586735       1764
4   57640                 Hellboy II: The Golden Army (2008)    Action|Adventure|Fantasy|Sci-Fi   3.373681       3602
5  103042                              Man of Steel (2013) Action|Adventure|Fantasy|Sci-Fi|IMAX   3.221282       3073
> |
```

Fig. 10 Recommendation for “Spider Man 3”

4 Results

The system takes movieId as an input and recommends top five similar movies based on it. Recommended movies rating should be above than 2.4 and at least 100 people should have rated that movie. The complete experiment is performed in R 3.2.2 (Fig. 9).

5 Conclusion

This hybrid model seems to perform good in all the early testing and gives more personalized and accurate results. Genome tag is the key driver for this model along with the content-based filters (Fig. 10).

References

1. Wei S, Zheng X, Chen D, Chen C (2016) A hybrid approach for movie recommendation via tags and ratings. Electron Commer Res Appl 18:83–94
2. Bell RM, Koren Y, Volinsky Chris (2010) All together now: a perspective on the netflix prize. Chance 23(1):24–29
3. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749
4. Linden G, Smith B, York J (2003) Amazon. com recommendations: item-to-item collaborative filtering. IEEE Internet comput 7(1):76–80

5. Sarwar BM, Karypis G, Konstan J, Riedl J (2002) Recommender systems for large-scale e-commerce: scalable neighborhood formation using clustering. In Proceedings of the fifth international conference on computer and information technology, Vol 1
6. Harper FM, Konstan JA (2016) The movielens datasets: history and context. ACM Trans Interact Intell Syst (TiiS) 5(4):19
7. MovieLens Latest datasets. http://files.grouplens.org/datasets/movielens/ml-latest.zip
8. Schafer JHJB, Frankowski D, Herlocker J, Sen S (2007) Collaborative filtering recommender systems. The adaptive web. Springer, Berlin, pp 291–324
9. Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp. 230-237
10. Schein AI, Popescul A, Ungar LH, Pennock DM (2002) Methods and metrics for cold-start recommendations. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp. 253-260
11. Balabanovi M, Shoham Y (1997) Fab: content-based, collaborative recommendation. Commun ACM 40(3):66–72
12. About The Music Genome Project, https://www.pandora.com/about/mgp
13. Vig J, Sen S, Riedl J (2012) The tag genome: encoding community knowledge to support novel interaction. ACM Trans Interact Intell Syst (TiiS) 2(3):13
14. Sarwar B, Karypis G, Konstan J, Riedl J (2000) Application of dimensionality reduction in recommender system-a case study (No. TR-00-043). Minnesota Univ Minneapolis Dept of Computer Science
15. Abdi H, Williams LJ (2010) Principal component analysis. Wiley Interdiscip Rev Comput Stat 2(4):433–459
16. Benesty J, Chen J, Huang Y, Cohen I (2009) Pearson correlation coefficient. Noise reduction in speech processing. Springer, Berlin, pp 1–4
17. Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. Commun ACM 18(11):613–620
18. Lee DL, Chuang H, Seamons K (1997) Document ranking and the vector-space model. IEEE softw 14(2):67–75
19. Panigrahi S, Lenka RK, Stitipragyan A (2016) A hybrid distributed collaborative filtering recommender engine using apache spark. Proced Comput Sci 83:1000–1006
20. Lenka RK, Barik RK, Panigrahi S, Panda S (2017) An improved hybrid distributed collaborative model for filtering recommender engine using apache spark. I.J. Intell Syst Appl