A personalised movie recommendation system based on collaborative filtering

V. Subramaniyaswamy*, R. Logesh, M. Chandrashekhar and Anirudh Challa

School of Computing,
SASTRA University,
Thanjavur, India
Email: vsubramaniyaswamy@gmail.com
Email: logeshr@outlook.com
Email: chandroo18@gmail.com
Email: anirudh.challa108@gmail.com
*Corresponding author

V. Vijayakumar

School of Computing Science and Engineering,
VIT University,
Chennai, India
Email: vijayakumar.v@vit.ac.in

Abstract: Over the last decade, there has been a burgeoning of data due to social media, e-commerce and overall digitisation of enterprises. The data is exploited to make informed choices, predict marketplace trends and patterns in consumer preferences. Recommendation systems have become ubiquitous after the penetration of internet services among the masses. The idea is to make use of filtering and clustering techniques to suggest items of interest to users. For a media commodity like movies, suggestions are made to users by finding user profiles of individuals with similar tastes. Initially, user preference is obtained by letting them rate movies of their choice. Upon usage, the recommender system will be able to understand the user better and suggest movies that are more likely to be rated higher. The experiment results on the MovieLens dataset provides a reliable model which is precise and generates more personalised movie recommendations compared to other models.

Keywords: data mining; collaborative filtering; movie recommendation; data acquisition; personalisation.

Reference to this paper should be made as follows: Subramaniyaswamy, V., Logesh, R., Chandrashekhar, M., Challa, A. and Vijayakumar, V. (2017) 'A personalised movie recommendation system based on collaborative filtering', *Int. J. High Performance Computing and Networking*, Vol. 10, Nos. 1/2, pp.54–63.

Biographical notes: V. Subramaniyaswamy received his BE (CSE) from Bharathidasan University of Trichy and MTech in Information Technology from the Sathyabama University at Chennai and PhD from the Anna University, Chennai in 2013. He is a Senior Assistant Professor in the School of Computing, SASTRA University, India. His specific interests include data mining, recommender systems, human computer interaction, information retrieval and social network analysis to solve computational engineering problems. He has published number of papers in all the above areas in leading journals and conferences. He has been organising conferences, serving in many conferences, technical committees and has been a reviewer for many conferences and journals.

R. Logesh obtained his BTech in Computer Science and Engineering and MTech in Networking from the Pondicherry University, Puducherry, India in 2012 and 2014, respectively. Currently, he is a PhD Scholar in the School of Computing at SASTRA University, Thanjavur, India. His research area includes recommender systems, data mining, social media analytics, etc.

M. Chandrashekhar is pursuing his BTech in Computer Science and Engineering from SASTRA University, Thanjavur, India. His research interests include data mining and data acquisition.

Anirudh Challa is pursuing his BTech in Computer Science and Engineering from SASTRA University, Thanjavur, India. His research interests include data mining and recommender systems.

V. Vijayakumar received his BE (CSE) from Periyar University of Salem and ME in Computer Science from the VIT University at Vellore and PhD from the Anna University, Chennai. He is a Professor in the School of Computing Science and Engineering at VIT University, Chennai, India. His specific interests include data mining and information retrieval. He has published number of papers in all the above areas in leading journals and has been a reviewer for many conferences and journals.

1 Introduction

The amount of information generated by the internet is growing at an overwhelming rate (Balabanovic and Shoham, 1997). The fact that more products and services are available to consumers now than ever has not only compounded the problem but has also increased its scope and diversity (Göker and Thompson, 2000). This presents a major challenge to e-commerce because consumers cannot simply explore and compare every possible product. Businesses are forced to come up with smarter solutions to prevent consumers from being engulfed by seemingly countless choices and more importantly to be still relevant to them by directing their attention to commodities they might be interested in. Recommendation systems (RSs) are introduced to e-commerce to help businesses tackle this challenge. Among all the existing techniques, collaborative filtering is one of the most promising approaches to build RSs (Goldberg et al., 1992). In this approach, user preferences for commodities are obtained using a rating system and this information is utilised to recommend products they might be interested in Bell and Koren (2007). This task is accomplished by searching the database for user profiles with similar tastes.

Businesses that sell a humongous assortment of goods on a daily basis both in real world like Walmart, Target, etc. on their e-business find RS immensely useful. RSs help businesses gauge consumer interests and retain them by actively engaging them. The system provides recommendations based on the ratings on commodities that users have previously rated (Amatriain et al., 2009a, 2009b). A RS is efficient since in practice the system handles tens of thousands of ratings and prediction is calculated in real-time. The more data the RS has to work with, more accurate the predictions done by the system are. Recommender systems can be classified based on the approach used to provide recommendations and the technique used (Balabanovic and Shoham, 1997; Ekstrand et al., 2010b; Xiao and Xie, 2015).

1.1 *Content-based recommender systems*

Content-based recommender systems are those that require the end user to describe using natural language or any other means like rating to show their preference. This content obtained from the user's history is used to make recommendations (Ekstrand et al., 2010b). Each potential recommendation is compared with the preferences of the user from the past and products that are most compatible with the perceived taste of the user are recommended (Ekstrand et al., 2010a; Subramaniyaswamy and Pandian, 2014).

1.2 *Collaborative filtering recommender systems*

These recommender systems upon obtaining initial information about the user, attempt to find other users with similar interests — users that have given similar products similar ratings. Recommendations are provided based on the preferences of similar users (Adomavicius and Tuzhilin, 2005; Ekstrand et al., 2010b).

1.3 *Demographic recommender systems*

Demographic recommender systems obtain demographic information from the users and rely on the collective demographic product preferences, i.e., if a product is generally accepted by a specific demographic, it is predicted that it would be of interest to the end user too. Demographic systems work better on products which have highly specific target group (Ekstrand et al., 2010b). Recommender systems are classified based on the approach used to provide recommendations and the technique used.

1.4 *Knowledge-based recommender systems*

There are item domains in which a lot of beforehand knowledge is required before making recommendations. This is primarily because of the complexity as there are not enough item ratings (Brand, 2003). As a result, collaborative filtering and content-based filtering would not be accurate (Ekstrand et al., 2010b). For example, items like automobile and apartments are not bought often and it requires a lot of knowledge to make intelligent suggestions.

1.5 *Hybrid recommender systems*

A hybrid recommender system is one that takes two or more of the above techniques to suit the application (Adomavicius and Tuzhilin, 2005). They are usually used to make the application more effective in its accuracy of recommendations (Ekstrand et al., 2010b). Hybridisation can be achieved in several ways.

The challenges traditional RSs face and their limitations are discussed below.

The first challenge that is to be tackled is to find neighbours (Bell and Koren, 2007; Herlocker et al., 2000). Given a large number of movies, a huge items profile, and a large number of users, the system must find recommendations extremely fast if predictions are to be made in real-time. There are two phases of computation required for recommendation. In the first phase, the neighbours of the user to whom the recommendations are made are identified. In this phase, similarity is computed between the targeted user and its neighbours. In the second phase, similarities between target users and the neighbours are computed and collected so as to make the prediction. After the prediction is computed, items which only cross a particular threshold will be recommended to the target user. The above two tasks involve complex processes and hence, it is important to have efficient algorithms to complete the tasks.

The second challenge is to compute the rating accurately. Information about the neighbours is needed before the predictions can be made. But identifying the neighbours is a challenging task. User ratings are highly individualistic. So is it possible to find parallels between users giving the ratings of movies? Even if the neighbours are identified somehow, how can prediction of similarities de done accurately?

The third challenge is the issues of cold start and data sparsity. Recommended systems might not be able to provide accurate predictions when there is not a significant amount of data to work with. Due to the scantiness of the available data, neighbour selection might become really difficult.

The fourth challenge traditional RSs have to face is that the weight of niche movies is not factored into the prediction evaluation. Two movie watchers that have only one niche movie in common have more similar tastes than those who have only one mainstream movie in common.

In this paper, an improvement to the existing approach is presented and various challenges are addressed. The efficiency of proposed algorithm is evaluated. Recent advancements in the field of collaborative filtering and recommendation engines are discussed in Section 2. In Section 3, the architecture and implementation are presented using illustrations and data acquisition, collaborative filtering and continual learning processes are explained in detail. The collaborative filtering algorithm is discussed in Section 4. The datasets obtained from MovieLens are discussed in Section 5. The experimental results are and performance analysis is performed in Section 6 and Section 7 concludes the work.

2 Related work

Collaborative filtering recommender system is a topic in which a lot of research is being conducted across the world. The first recommender systems were proposed by researchers in late 1990s (Adomavicius and Tuzhilin, 2005). The whole idea of recommender systems emerged from the Tapestry project in 1992 (Colombo-Mendoza et al., 2015). Ever since, it has been an active area of interest for data scientists because of the plethora of problems in the field and innumerable avenues of application in real world. Recommender systems root from extensive work in cognitive science, approximation theory, information retrieval, forecasting theories and even market analysis and predictions (Adomavicius and Tuzhilin, 2005).

Also with the increase in e-commerce, there has been a collection of staggering amount of user data. Recommender systems make intelligent use of big data and also helps tackle the growingly intimidating information overload (Guy et al., 2009). A lot of academic work has also been done across the world and domain literature has also flourished (Martin, 2009). Universities are offering recommender systems courses, and an annual conference dedicated only to the subject is also held. The scope of recommender systems has also widened. Even though, initial work was almost entirely done in collaborative filtering, it has grown to accommodate a host of content-based and knowledge-based methods. While such diversity in systems are significant, recommender systems that are based on collaborative filtering are of main interest (Shen and Zhou, 2008).

The giant success of the e-commerce ventures can be greatly attributed to recommender systems. As a result, they are used extensively in the e-commerce industry. Recommender systems are used to generate highly personalised suggestions for books at Amazon (Linden et al., 2003), movies at MovieLens and NetFlix (Bennett and Lanning, 2007). They help filter out humongous masses of irrelevant information, also called noise, from the information that can be leveraged to make highly intelligent suggestions and business decisions.

E-commerce recommender systems face big data and highly challenging business scenarios (Linden et al., 2003). They can be used by a conglomerate to make high-stakes decisions from comprehensive collection of data gathered from its wide customer base or applications that need recommendations to be done in a matter of seconds without compromising on the quality of suggestions.

Recommender systems usually adopt one of the three widely used approaches – conventional filtering algorithms, clustering models or content-based algorithms. Conventional filtering algorithms employ an N-dimensional vector space, N being the number of unique entities in the inventory. Positive values correspond to positive responses from the user end and negative values reflect negatively rated items. These algorithms operate on the assumption that people with similar rating patterns for similar products are an accurate indicator of products that might be of interest to the end user (Linden et al., 2003). These users that have ‘similar tastes’ as the end user are called neighbours.

Clustering models operate on the concept of segregating the sample space into clusters. These models are widely used to remedy scalability problems while still delivering recommendations of comparable accuracy (McNee et al., 2006). The aim of the clustering models is to partition items into clusters in such a way that the objects will have the minimum distance between them.

Content-based algorithms approach the task of making suggestions as a query problem (Balabanovic and Shoham, 1997). They look at the history of ratings, descriptions of products by the user and products the user has searched for.

All the search queries that are made in the system for similar items are found and products that have similar attributes like same music band (Fields et al., 2010) or author or director are recommended to the user. One major challenge this approach faces is that the recommendations made are too generic or too narrowed down and predictable. As there are millions of queries made, it is almost impossible to look at all the queries. In order to remedy that only one or more subsets of all the queries made are looked at. Even then, the overall recommendation quality is relatively poor.

All of the above mentioned approaches have one common challenge, namely noise. Noise can be classified as natural and malicious. Natural noise is user-generated stemming from inconsistency in user rating patterns. Malicious noise is the noise that is introduced in the recommender system deliberately in order to dilute quality of recommendations.

Collaborative filtering is the most effective approach for movie RSs (Wang et al., 2014; Ansari et al., 2000; Göker and Thompson, 2000; Wu et al., 2015). Based on candidate users' purchase history, the ratings of items for a target user are predicted. A collaborative filtering RS adopts the nearest neighbour approach. Neighbours are chosen based on the similarity of item ratings using collaborative filtering. It is safe to assume that the 'tastes' of neighbours are similar to that of the target user's. Recommendations are provided based only on the neighbours (Herlocker et al., 2000; Herlocker et al., 2002; Anand and Bharadwaj, 2014). One of the most popular applications available online that uses collaborative filtering is LensKit. LensKit basically tries to identify neighbours based on the difference in the item ratings of each rating pair. The rating pair consists of the target user and one of the candidates. In LensKit, a target user and a predetermined number of candidates are provided as input, and the similarity based on the ratings between the rating pair is computed. Candidate users that have similar tastes are chosen as neighbours (Herlocker et al., 2000). The predictions provided to the target user are then computed based on the neighbours' rating.

Input can be captured from the graphical user interface in case of stand-alone applications or a web interface for internet applications. User preference can be captured in two ways, implicit and explicit (Breese et al., 1998). Implicit information collection is inferred and therefore not reliable. It is captured by observing user consumption pattern. Implicit information capture is based on the assumption that the amount of time spent on a product is directly proportional to the user interest in it. Explicit feedback is obtained using some sort of feedback questionnaire. It is an explicit expression of preference or dissatisfaction (Breese et al., 1998). It might be scale of 1–5 values or an upvote/downvote system.

3 Architecture and implementation

The architecture comprises three major parts namely, data acquisition and repository, RS and user interface. As illustrated in Figure 1, the three constituent units work collaboratively. The registration details including user demographics and user ratings of movies are stored in corresponding data structures. The MovieLens datasets are also obtained. In this implementation, the data repository is housed locally. The RS sub-unit is responsible for performing collaborative filtering, user taste generation and computation of Euclidean scores with which movie recommendations are made. Movie preferences and ratings from the end-user are obtained and recommendations made are outputted through the graphical user interface.

Figure 1 RS architecture

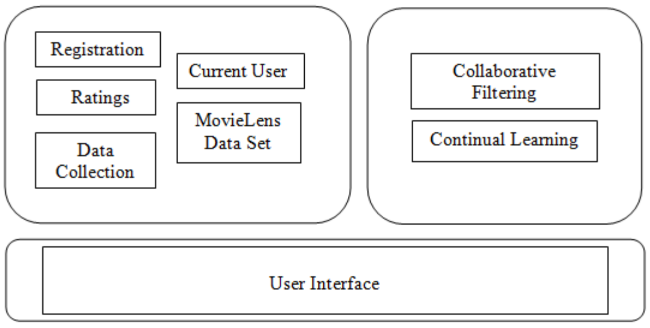
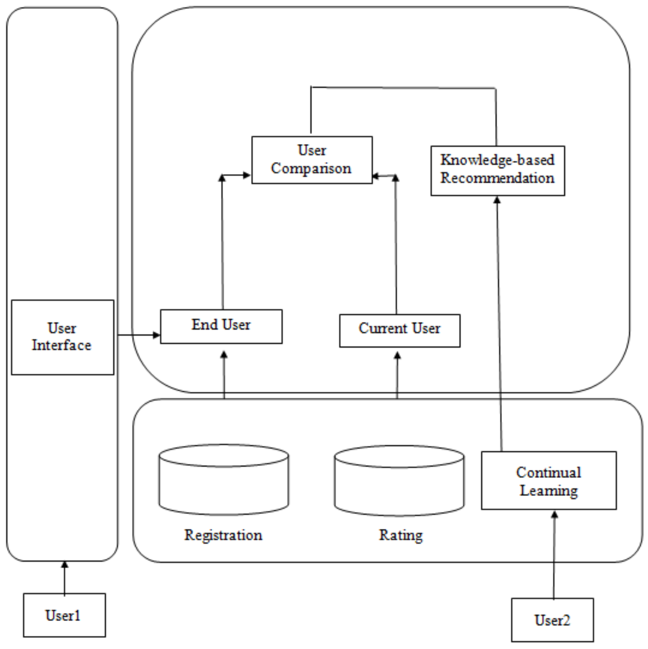


Figure 2 Functionality and implementation of recommender system



3.1 *Data acquisition*

Data from users is obtained through this component. In a recommender system data plays a very important role. There are three main files from the MovieLens dataset. They are the users information file, the movie list file and the ratings file.

The user information file contains the personal background of the user. It contains a unique ID which is assigned to each user, the user's gender, occupation, age and their zip-code. The next file is the movie list file which contains the movie ID which is unique for each movie. It also contains the movie name and the genre of the movie. The last file is the ratings file which contains the user ID, the movie ID and the timestamp when the user has rated the movie. Each of these features can be leveraged to improve the recommender system.

A user interface has been created so that the user can interact with the recommendation engine. The interface is also enables the user to enter their details in the registration screen through which the user details emerge. The user also has to rate the movies through which the rating file gets updated on a regular basis.

3.2 Collaborative filtering

A user-based collaborative filtering method is used. In this method, users with similar tastes in movies are identified. Intuitively each user is a point in space where the axes are the movies. So users with similar tastes will be close to each other. The Euclidean distance method of collaborative filtering is used. The distance score is calculated using equation (1).

$$\begin{aligned} d(\mathbf{p},\mathbf{q}) &= d(\mathbf{p},\mathbf{q}) \\ &= \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}. \end{aligned} \tag{1}$$

Here, p and q are two users. p_i and q_i are the ratings given by users p and q for the movie with id $i.d(p, q)$ is the Euclidean distance score.

This is done for each and every user every time a new movie is rated. The list of movies rated by the end user is taken and written into a separate file. Then, for each of the other users their ratings are written into another file, one at a time. Now the movies seen by the current user and the end user is compared. Then the entries are written accordingly to a third file. The Euclidean distance is calculated from this file.

3.3 Continual learning

In this part, the changing tastes of the user are taken into account. This is done by modifying the Euclidean distance formula. A weighted form of the formula is used where the square of the distance is multiplied by a factor based on the timestamp. Recently, rated movies are given a higher weight. This is used to understand the changing preferences.

The users with least weighted Euclidean distances are ranked. The top three users are taken. Then, the highly rated movies from the users are ranked. From this list, the top five rated movies are taken. This is shown in the recommendation screen.

4 Algorithm – collaborative filtering

```
Initialisation:
    Parameters Initialisation:
        CurrentUserID = 6041
    Data Population:
        endUser.csv
        currentUser.csv
        compareUser.csv
While ($u < Umax$)
{
    For $i$ = 1 to MaxUserID
    {
        If iUserID = EndUserID
        {
            Copy movieID, rating to endUser.csv
        }
    }
    CreateendUserList
    For $i$ = 1 to MaxUserID
    {
        If iUserID = uUserID
        {
            Copy movieID, rating to currentUser.csv
        }
    }
    Create currentUserList
    Compare endUserList and currentUserList and create
    compareUserList
    Compute EuclideanDistanceScore
                $d(w) = \sqrt{w}$
                $w = w_1^2 + w_2^2 + \dots + w_n^2$
        For $i$ = 0 to numberCommonMovies
            If (currentTime – ratingTime < 157788000)
            {
                    $w_i = 2 * (p_i - q_i)$
            }
            Else
            {
                    $w_i = (p_i - q_i)$
            }
    WriteuUserID and EuclideanDistanceScore into
    EuclideanDistanceComparison.csv
    $u = u + 1$
}
Find similarTaste1, similarTaste2 and similarTaste3 from
EigenValueComparison.csv
Compute recommendedMovies and display
End
```

5 MovieLens datasets

The datasets have been collected from MovieLens Research Project website. It has been curated by University of Minnesota. The dataset consists of one million anonymous movie ratings. It also has user demographic information and a list of about three thousand movies from which the users can rate.

Table 1 Movie list with movie ID and genre

Movie ID	*Movie name*	*Genre*
1	Toy Story (1995)	Animation|Children's|Comedy
2	Jumanji (1995)	Adventure|Children's|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children's
9	Sudden Death (1995)	Action
10	Golden Eye (1995)	Action|Adventure|Thriller

Table 2 List of movie ratings along with timestamps

User ID	*Movie*	*Rating*	*Timestamp*
1	1,193	5	978300760
1	661	3	978302109
1	914	3	978301968
1	3,408	4	978300275
1	2,355	5	978824291
1	1,197	3	978302268
1	1,287	5	978302039
1	2,804	5	978300719
1	594	4	978302268

Table 1 shows the movie list. It contains over 3,900 titles. The list contains the movie identity, movie name and the genre.

Table 2 contains the list of movie ratings given by all users. There are over 1 million movie ratings in MovieLens.

Table 3 contains user demographic information. It has the user identity number, gender of the user, age, occupation code and zip code. It contains over 6000 user details.

Table 3 User demographic information

User ID	*Gender*	*Age*	*Occupation*	*Zip code*
1	F	1	10	48067
2	M	56	16	70072
3	M	25	15	55117
4	M	45	7	2460
5	M	25	20	55455
6	F	50	9	55117
7	M	35	1	6810
8	M	25	12	11413
9	M	25	17	61614
10	F	35	1	95370

5.1 Movie rating and recommendation

Figure 3 shows the ratings users give for particular movies. This data is stored in the ratings file. It is in CSV file format. The file contains *user ID*, *movie ID*, *rating* and *timestamp*.

The file is ordered by ascending user ID values. This helps improving run-time as number of file lookups is reduced. The page also allows to rate other movies.

Figure 3 Movies rating by end user (see online version for colours)

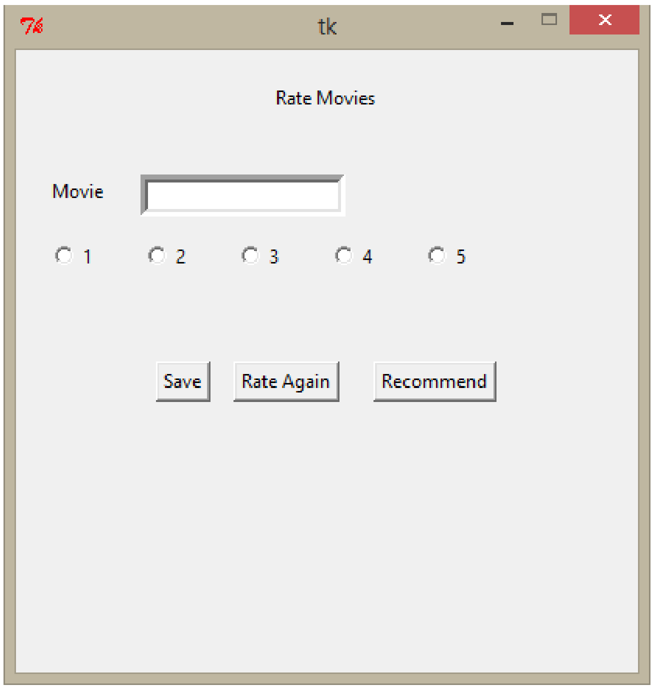
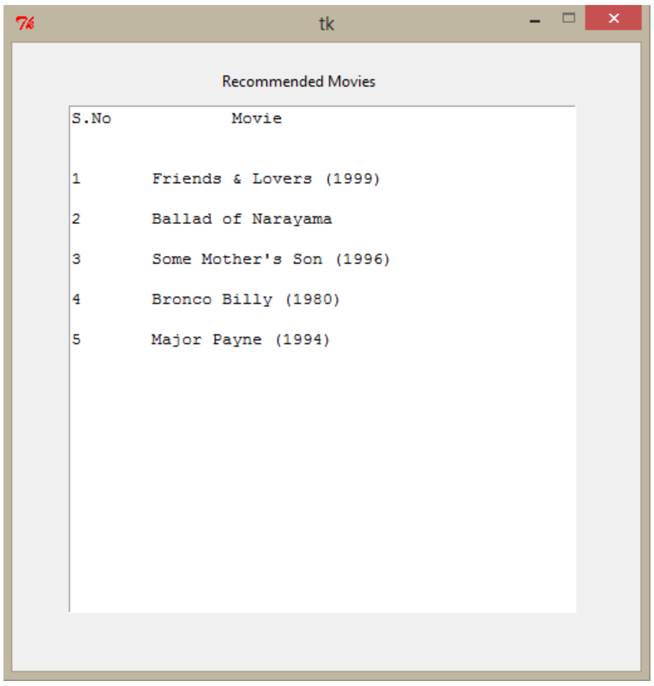


Table 4 Results of Precision, Recall and F-measure

Users	*Execution*	*Relevant user*	*Relevant system*	*Non-relevant user*	*Non-relevant system*	*Recommended*	*Precision*	*Recall*	*F-measure*
A	1	390	352	3,510	3,412	374	0.941	0.902	0.921
	2	390	368	3,510	3,415	380	0.968	0.943	0.955
B	1	387	349	3,513	3,398	377	0.926	0.902	0.913
	2	387	365	3,513	3,410	382	0.955	0.943	0.948
C	1	394	357	3,506	3,411	385	0.927	0.906	0.916
	2	394	371	3,506	3,420	390	0.915	0.941	0.946
D	1	384	349	3,516	3,414	375	0.931	0.901	0.919
	2	384	365	3,516	3,422	376	0.971	0.951	0.961
E	1	385	351	3,515	3,415	382	0.919	0.911	0.914
	2	385	364	3,515	3,421	384	0.948	0.945	0.946
F	1	387	353	3,517	3,420	385	0.917	0.912	0.914
	2	387	367	3,517	3,425	385	0.953	0.948	0.950
G	1	375	349	3,525	3,430	365	0.956	0.931	0.943
	2	375	360	3,525	3,439	370	0.973	0.960	0.966
H	1	377	351	3,523	3,435	374	0.938	0.936	0.937
	2	377	363	3,523	3,441	375	0.968	0.963	0.966
Avg.	1	384.875	351.375	3,516	3,416.875	377.725	0.930 (93%)	0.912 (91.2%)	0.921 (92.1%)
	2	384.875	365.375	3,516	3,424.125	380.25	0.961 (96.1%)	0.951 (95.1%)	0.956 (95.6%)

Figure 4 Movie recommendations (see online version for colours)



```
S.No            Movie

1        Friends & Lovers (1999)
2        Ballad of Narayama
3        Some Mother's Son (1996)
4        Bronco Billy (1980)
5        Major Payne (1994)
```

Figure 4 presents the recommended movie list. This is done by computing the movies which have been rated similar by user with similar movie interests.

6 Performance evaluation

To evaluate the movie recommendation engine three different metrics are used. These metrics are precision, recall and F-measure (Billsus and Pazzani, 1998; Avery and Zeckhauser, 1997; Karypis, 2001). The three metrics are popular in information retrieval. They were initially used by Salton and Gill in 1986. These measures will help quantify the effectiveness of the recommendation engine. The most important factor which it will quantify is the quality of the output.

The above three metrics have always been the preferred measure for quantifying the quality of output from information retrieval systems. In recent times they have been used frequently in evaluating movie recommender systems. In 2013, Ruotsalo et al. used these measures in an experiment to find the real life usefulness of their proposed method. They measured their RS's performance by using precision and recall. Using precision, the number of relevant retrievals that were made is found. Recall helps in measuring the number of irrelevant retrievals that are made. The users decide whether the output is of interest to them and assigns relevant and irrelevant. F-measure is the harmonic mean between precision and recall. These metrics are then contextualised to make sense for a movie RS.

6.1 Metrics

In this paper, precision is calculated as the number of relevant recommendations that the user gets. Equation (2) presents the formula to calculate precision measure:

$$Precision = \frac{Correctly\ recommended\ items}{Total\ recommended\ items} \quad (2)$$

where *correctly recommended* items is the number of movies classified as relevant by the user that are recommended by the system. *Total recommended items* is the total number of movies recommended by the system.

Recall is the ability of the system to recommend as few non-relevant movies as possible. The recall value is computed using equation (3).

$$Recall = \frac{Correctly\ recommended\ items}{Total\ relevant\ items} \quad (3)$$

It can be inferred that the irrelevant recommendations were not enjoyed by the user.

The formula for calculating F-measure is as presented in equation (4):

$$F - measure = \frac{(2 * precision * recall)}{(precision + recall)} \quad (4)$$

6.2 Use case

The classification of recommendations as being relevant or non-relevant is subjective. So a real user is asked to use the system to evaluate the recommendations. Based on their responses, precision, recall and F-measure are computed.

Two use cases are run. In the first, all ratings are considered equal, i.e., the recommendations are not time-sensitive.

In the second scenario, time is taken into consideration to reflect the changing tastes of the user. The two use cases are run on the same group of people and find the results.

6.3 Solution

Initially, the users were asked to register. They filled out the registration form and are given uniquely generated user identity numbers. Then, the users are asked to rate the movies that they had already watched. Based on this, the recommendation engine was able to understand each individual users interests.

6.4 Comparison results

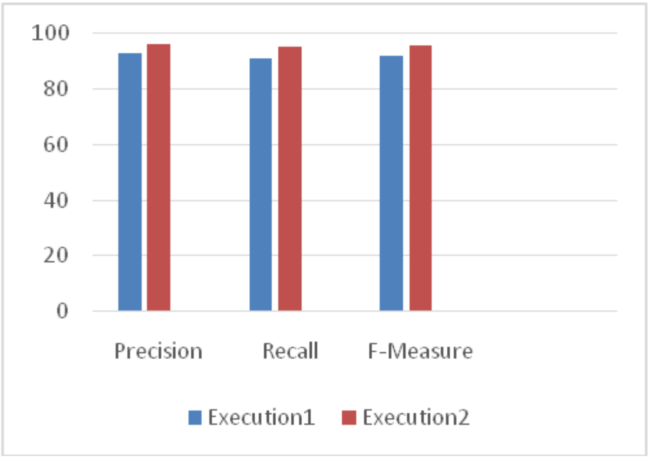
Table 4 gives the results obtained after applying the above formulae for precision, recall and F-measure.

The column 'relevant user' means those are the movies which the user considers as relevant and 'non-relevant user' means those movies which are considered irrelevant by the user. Similarly, the system generates 'relevant system' and 'non-relevant system'. The column 'recommended' means the total number of recommend movies.

It can be understood from the results that in the first execution, the best precision is 96% (user 'G') and the lowest is 92% (user 'F'). The average precision is 93% and in all cases it has crossed 90%. The recall rates also are high, the lowest recall is 90% (user 'D') and the highest recall 94% (user 'H'). The average recall is 91% and all the recall values are above 90%. The recall values are all slightly lesser than the precision values. Similarly, the F-measure values range from 91% (user 'B') to the highest 94% (user 'G'). The F-measure values are the harmonic mean of the recall and precision. All the F-measure values are above 90% also.

The second execution took into account the changing movie preferences with time. Consider an example: A nine-year-old is more likely to love and be recommended an animation movie. As she grows up, the animation recommendations might not be particularly relevant and be of interest to her. For the change in taste to be accommodated, recently rated movies are given higher weightage. The average value of precision has increased by 3% (93% to 96%) in this execution. Similarly, the recall value has increased by 4% (91% to 95%). As both precision and recall increase, so would the F-measure value. Hence, it is understood that the performance of the second execution is better than the first as it can clearly be observed as in Figure 5.

Figure 5 Comparison of precision, recall and F-measure measurements from two executions (see online version for colours)



7 Conclusions and future work

In this paper, a recommendation engine is developed using the concept of collaborative filtering. In the proposed approach, standard user demographics such as gender, age, occupation are used. User ratings are used to map out other users with similar tastes and recommendations are made by excluding common entities. Neighbours are identified by computing Euclidean distance score. The user with the least Euclidean distance score is found. Recommendations are made based on the nearest neighbour's best rated movies. The recommendations are also made time-sensitive to keep up with the changing tastes of user. The results are evaluated using three metrics – precision, recall and F-measure.

As businesses attempt to make more use of their data and as this field is relatively a new phenomenon, a lot of improvements can be made over the existing model to suit the problem at hand more precisely and generate more accurate and more personalised recommendations. More demographic information like nationality, race, location, mother-tongue, and languages spoken can be used to make the recommendations better. The application can be made available on cross platforms so as to extend its reach and functionality. Instead of just recommending movies to the end user, the functionality of the application can be further expanded to finding and providing theatres nearby that screen the recommended movies as the location can be obtained by reading the user's GPS coordinates. Input from consolidated internet databases and authentic review sites like IMDb and Rotten Tomatoes can be factored into the recommendation process.

Recommendations can be made more personalised with integration of social media profiles with the application. Recently, social media has proven to be a goldmine for data-analytics-driven predictive applications. Social media content can be mined to better identify user taste and interest. A whole lot of other mobile apps can also be used to improve recommendations. Location-based apps like Foursquare can be used to map not just user location but also other real-life user interests, based on which suggestions can be made.

Acknowledgements

The authors are grateful to Science and Engineering Research Board (SERB), Department of Science & Technology, New Delhi, for the financial support (No. YSS/2014/000718/ES).

References

Adomavicius, G. and Tuzhilin, A. (2005) 'Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions', *IEEE Transactions on Knowledge and Data Engineering*, June, Vol. 17, No. 6, pp.734–749.

Amatriain, X., Pujol, J. and Oliver, N. (2009a) 'I like it. . . I like it not: evaluating user ratings noise in recommender systems', *Lecture Notes in Computer Science, UMAP 2009*, Vol. 553, pp.247–258, Springer.

Amatriain, X., Pujol, J.M., Tintarev, N. and Oliver, N. (2009b) 'Rate it again: increasing recommendation accuracy by user re-rating', *ACM RecSys '09*, ACM, pp.173–180.

Anand, D. and Bharadwaj, K.K. (2014) 'Exploring graph-based global similarity estimates for quality recommendations', *International Journal of Computational Science and Engineering*, Vol. 9, No. 3, pp.188–197.

Ansari, A., Essegaier, S. and Kohli, R. (2000) 'Internet recommendation systems', *Journal of Marketing Research*, August, Vol. 37, No. 3, pp.363–375.

Avery, C. and Zeckhauser, R. (1997) 'Recommender systems for evaluating computer messages', *Communications of the ACM*, March, Vol. 40, No. 3, pp.88–89.

Balabanovic, M. and Shoham, Y. (1997) 'Fab: content-based, collaborative recommendation', *Communications of the ACM*, Vol. 40, No. 3, pp.66–72.

Bell, R.M. and Koren, Y. (2007) 'Scalable collaborative filtering with jointly derived neighborhood interpolation weights', *IEEE ICDM 2007*, pp.43–52.

Bennett, J. and Lanning, S. (2007) 'The netflix prize', *KDD Cup and Workshop '07*.

Billsus, D. and Pazzani, M.J. (1998) 'Learning collaborative information filters', *AAAI 2008 Workshop on Recommender Systems*.

Brand, M. (2003) 'Fast online SVD revisions for lightweight recommender systems', *SIAM International Conference on Data Mining*, pp.37–46.

Breese, J.S., Heckerman, D. and Kadie, C. (1998) 'Empirical analysis of predictive algorithms for collaborative filtering', *UAI 1998*, AAAI, pp.43–52.

Colombo-Mendoza, L.O., Garcia, R.V., Rodriguez-Gonzalez, A., Alor-Hernandez, G. and Samper-Zapater, J.J. (2015) 'RecomMetz: a context-aware knowledge-based mobile recommender system for movie show times', *Expert Systems with Applications*, Vol. 42, No. 3, pp.1202–1222.

Deshpande, M. and Karypis, G. (2004) Item-based top-N recommendation algorithms', *ACM Transactions on Information Systems*, Vol. 22, No. 1, pp.143–177.

Ekstrand, M.D., Kannan, P., Stemper, J.A., Butler, J.T., Konstan, J.A. and Riedl, J.T. (2010b) 'Automatically building research reading lists', *ACM RecSys '10*, ACM, pp.159–166.

Ekstrand, M.D., Riedl, J.T. and Kontan, J.A. (2010b) 'Collaborative filtering recommender systems', *Foundations and Trends in Human-Computer Interaction*, Vol. 4, No. 2, pp.81–173.

Fields, B., Rhodes, C. and d'Inverno, M. (2010) 'Using song social tags and topic models to describe and compare playlists', *Workshop on Music Recommendation and Discovery 2010*, CEUR, September, p.633.

Göker, M. and Thompson, C. (2000) 'Personalized conversational case-based recommendation', *Advances in Case-Based Reasoning, Lecture Notes in Computer Science*, Vol. 1898, pp.29–82, Springer.

Goldberg, D., Nichols, D., Oki, B.M. and Terry, D. (1992) 'Using collaborative filtering to weave an information tapestry', *Communications of the ACM*, Vol. 35, No. 12, pp.61–70.

Guy, I., Zwerdling, N., Carmel, D., Ronen, I., Uziel, E., Yogev, S. and OfekKoifman, S. (2009) 'Personalized recommendation of social software items based on social relations', *ACM RecSys '09*, ACM, pp.53–60.

Herlocker, J., Konstan, J.A. and Riedl, J. (2002) 'An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms', *Information Retrieval*, Vol. 5, No. 4, pp.287–310.

Herlocker, J.L., Konstan, J.A. and Riedl, J. (2000) 'Explaining collaborative filtering recommendations', *ACM CSCW '00*, ACM, pp.241–250.

Karypis, G. (2001) 'Evaluation of item-based top-N recommendation algorithms', *ACM CIKM '01*, ACM, pp.247–254.

Linden, G., Smith, B. and York, J. (2003) 'Amazon.com recommendations: item to item collaborative filtering', *IEEE Internet Computing*, Vol. 7, No. 1, pp.76–80.

Martin, F.J. (2009) 'RecSys '09 industrial keynote: top 10 lessons learned deploying and operating real-world recommender systems developing', *ACM RecSys '09*, ACM, pp.1–2.

McNee, S.M., Riedl, J. and Konstan, J.A. (2006) 'Being accurate is not enough: how accuracy metrics have hurt recommender systems', *ACM CHI '06 Extended Abstracts*, ACM, pp.1103–1108.

Ruotsalo, T., Haav, K., Stoyanov, A., Roche, S., Fani, E., Deliai, R., Mäkelä, E., Kauppinen, T. and Hyvönen, E. (2013) 'SMARTMUSEUM: a mobile recommender system for the web of data', *Journal of Web Semantics*, Vol. 20, No. 1, pp.50–67.

Shen, H. and Zhou, S. (2008) 'Reconciliation of compound actions in internet-based distributed collaborative systems', *International Journal of High Performance Computing and Networking*, Vol. 5, Nos. 5–6, pp.309–322.

Subramaniyaswamy, V. and Pandian, S.C. (2014) 'Topic ontology-based efficient tag recommendation approach for blogs', *International Journal of Computational Science and Engineering*, Vol. 9, No. 3, pp.177–187.

Wang, Z., Yub, X., Feng, N. and Wang, Z. (2014) 'An improved collaborative movie recommendation system using computational intelligence', *Journal of Visual Languages and Computing*, Vol. 25, No. 6, pp.667–675.

Wu, H., Chou, W.K., Hao, N., Wang, D. and Li, J. (2015) 'Collaborative filtering recommendation based on conditional probability and weight adjusting', *International Journal of Computational Science and Engineering*, Vol. 10, No. 1, pp.164–170.

Xiao, Q. and Xie, H. (2015) 'A social tag recommendation method alleviating cold start based on probabilistic graphical model', *International Journal of Embedded Systems*, Vol. 7, No. 2, pp.162–169.