Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 35 (2008) 1386–1399
Expert Systems with Applications
www.elsevier.com/locate/eswa

Fuzzy-genetic approach to recommender systems based on a novel hybrid user model

Mohammad Yahya H. Al-Shamri , Kamal K. Bharadwaj

School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi 110 067, India

Abstract

The main strengths of collaborative filtering (CF), the most successful and widely used filtering technique for recommender systems, are its cross-genre or 'outside the box' recommendation ability and that it is completely independent of any machine-readable representation of the items being recommended. However, CF suffers from sparsity, scalability, and loss of neighbor transitivity. CF techniques are either memory-based or model-based. While the former is more accurate, its scalability compared to model-based is poor. An important contribution of this paper is a hybrid fuzzy-genetic approach to recommender systems that retains the accuracy of memory-based CF and the scalability of model-based CF. Using hybrid features, a novel user model is built that helped in achieving significant reduction in system complexity, sparsity, and made the neighbor transitivity relationship hold. The user model is employed to find a set of like-minded users within which a memory-based search is carried out. This set is much smaller than the entire set, thus improving system's scalability. Besides our proposed approaches are scalable and compact in size, computational results reveal that they outperform the classical approach.

Keywords: Recommender systems; Collaborative filtering; Web personalization; User model; Fuzzy sets

1. Introduction

The popular use of web as a global information system has flooded us with a tremendous amount of data and information. The end users are overloaded with a huge amount of information from myriad resources. This explosive growth of data has generated an urgent need for powerful automated web personalization tools that can intelligently assist us in transforming the vast amount of data into useful information and knowledge. In other words, these tools ensure that the right information is delivered to the right people at the right time (Adomavicius & Tuzhilin, 2005; Eirinaki & Vazirgiannis, 2003). Web recommender systems (RS), the most successful example of Web personalization tools, tailor information access, trim down the information overload, and efficiently guide the user in a personalized manner to interesting items within a very large space of possible options (Burke, 2002). Typically RS recommend information (URLs, netnews articles), entertainment (books, movies, restaurants), or individuals (experts). Amazon.com (Linden, Smith, & York, 2003) and MovieLens.org (Miller, Albert, Lam, Konstan, & Riedl, 2003) are two well-known examples of RS on the web.

Recommender systems employ four information filtering techniques, demographic filtering (DMF) (Krulwich, 1997), content-based filtering (CBF) (Lang, 1995), collaborative filtering (Resnick, Iakovou, Sushak, Bergstrom, & Riedl, 1994; Shardanand & Maes, 1995), and hybrid filtering techniques (Balabanovic & Shoham, 1997; Pazzani, 1999; Shahabi, Banaei-Kashani, Chen, & McLeod, 2001). DMF categorizes the user based on the user personal attributes and makes recommendations based on demographic classes while the CBF suggests items similar to the ones the user preferred in the past.

* Corresponding author. Mobile: +91 (11) 9810196636; fax: +91 (11) 26717528.
E-mail addresses: mohamad.alshamri@gmail.com (M.Y.H. Al-Shamri), kbharadwaj@gmail.com (K.K. Bharadwaj).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.08.016

The most familiar and most widely implemented filtering is the collaborative filtering. The initial CF was introduced by Tapestry (Goldberg, Nichols, Oki, & Terry, 1992), and was automated by GroupLens (Resnick et al., 1994) and Ringo (Shardanand & Maes, 1995). Typically, CF explores similar users (neighbors), recognizes commonalities between the user and his neighbors on the basis of their ratings, and then accordingly generates new recommendations based on inter-user comparisons (Adomavicius & Tuzhilin, 2005; Eirinaki & Vazirgiannis, 2003). Further, CF and DMF have the unique capacity to identify cross-genre niches and can entice users to jump outside of the familiar 'outside the box' (Burke, 2002).

Breese, Heckerman, and Kadie (1998) classified CF algorithms as either memory-based (Resnick et al., 1994; Shardanand & Maes, 1995), the entire data is used for recommendations, or model-based (Shahabi et al., 2001) in which a model is derived offline from the data to be used for online recommendations. While the former is more accurate, its scalability compared to model-based is poor. Practically, CF faces two fundamental challenges, namely accuracy and scalability. Although memory-based algorithms are simple, provide high accuracy recommendations, and admit easy addition of new data, but they are computationally expensive as the size of the input dataset increases. Eventually, the user will leave the Web site before the processing completes. On the other hand, applying the model-based algorithm alone on such sparse data, though reduces the online processing cost, often comes at the cost of recommendation accuracy. One common threat in current RS research is the need to combine recommendation techniques to achieve peak performance because each technique has its own pros and cons.

Essentially, RS keep a profile for each user. Without any information about the user, the RS are not able to assist the user. The user profile contains raw information about the user. The terms user profile and user model are often used as synonyms. However, recent researchers (Froschl, 2005; Koch, 2000) differentiate between them according to the level of sophistication. The user profile is a collection of raw personal information represents preferences, background, personal details, and interactions with the system. Depending on the user profile, a user can be modeled. Thus, the user profile is used to retrieve the needed information to build up a model for the user. Koch (2000) describes the user model as the representation of the system's beliefs about the user and the user profile as a simple user model.

Some efforts have been made towards introducing fuzziness in RS. Nasraoui and Petenes (2003) used fuzzy approximate reasoning to develop a general framework for the recommendation process while Suryavanshi, Shiri, and Mudur (2005) used relational fuzzy subtractive clustering. Shahabi et al. (2001) proposed a Yoda RS that softly classifying active user based on typical patterns of users and then generating soft recommendations for him. The user profile includes many features that can be described as fuzzy. But at the item level it is difficult to fuzzify the profile because it would require prohibitively large space and long processing time.

Our work in this paper is an attempt towards introducing hybridization at four different levels, namely feature-level, model-level, CF algorithm-level, and approach-level. Firstly, we proposed hybrid features that exploit both user ratings for high rated items and some content descriptions of the items. At the model-level we built a user model from the set of hybrid features and DMF profile. Hybridization between model-based and memory-based algorithms of CF is done at the CF algorithm-level. The user model is used to find a set of like-minded users within which a memory-based search is carried out. This set is much smaller in size than the original set, thus making the technique scalable. Finally, we developed a hybrid fuzzy-genetic RS by employing GA to evolve appropriate weights for each feature of the user model and proposing a novel fuzzy distance metric to match users.

The contributions of this paper are three-fold:

- A novel user model is built that enables hybrid filtering, reduces the system complexity and the computational time by roughly a factor of six.
- A novel fuzzy distance function is proposed for users matching.
- A hybrid fuzzy-genetic approach to recommender systems is developed.

The rest of this paper is organized as follows: some background on the RS is given in the next section. In Section 3, the proposed user model is presented, while the fuzzy and hybrid fuzzy-genetic approaches are given in Section 4. The experimental results of the proposed approaches and classical approach are discussed in Section 5. Finally, in the last section we conclude our work with a review of our contributions along with some future research directions.

2. Background

2.1. Recommender systems

Recommender systems have gained an increasing importance since the early work on CF in the mid-1990s when researchers started focusing on RS that explicitly rely on the ratings structure (Adomavicius & Tuzhilin, 2005). Normally, explicit ratings from users are binary ratings (like/dislike) or follow a specified numerical scale indicating the degree of preference (e.g., 1 – bad to MAX – excellent, where MAX is the maximum possible rating for a given system). Usually, one of the following information filtering techniques is employed to generate recommendations:

- *Demographic filtering (DMF)*: The user will be recommended items similar to the ones other people with same

demographic preferred. Lifestyle Finder (Krulwich, 1997) uses demographic groups from marketing research to suggest a range of products and services.

- *Content-based filtering (CBF)*: The user will be recommended items similar to the ones he preferred in the past. Example of such systems is NewsWeeder (Lang, 1995).
- *Collaborative filtering (CF)*: The user will be recommended items people with similar tastes and preferences liked in the past. GroupLens (Resnick et al., 1994), MovieLens (Miller et al., 2003), and Ringo (Shardanand & Maes, 1995) are some examples of such systems.
- *Hybrid filtering*: These techniques combine more than one filtering technique to enhance the performance like Fab (Balabanovic & Shoham, 1997) and Amazon.com (Linden et al., 2003).

The main shortcomings of CBF are *content limitations*; only a very shallow analysis of certain kinds of content can be supplied, and *over-specialization*; the user is restricted to see items similar to those already rated by him only (Balabanovic & Shoham, 1997). These systems are not suitable for dynamic and very large environments, where items are millions and inserted in the system frequently (Adomavicius & Tuzhilin, 2005).

The great power of CF relative to CBF is its cross-genre or 'outside the box' recommendation ability. Moreover CF is completely independent of any machine-readable representation of the items being recommended (Adomavicius & Tuzhilin, 2005; Burke, 2002). However, CF suffers some weaknesses: *new user problem* (cold start problem), *sparsity*, *scalability*, and *loss of neighbor transitivity* (Breese et al., 1998; Vozalis & Margaritis, 2003). Usually each user rates only a very limited percentage of items, when compared to the available total. This leads to sparse user-item matrices, the set of users cross the set of items, therefore weak recommendations could be produced because the successful neighbors cannot be found.

On the other hand, the computational cost of RS grows fast with both the number of users and items giving rise to a scalability problem. Further, relying directly on individual items ratings result in a loss of neighbor transitivity. Assume that we have three users u_i, u_j, and u_k. Users u_i and u_j correlate highly, also u_j and u_k correlate highly. Because of transitivity there is a possibility that users u_i and u_k correlate highly too. Such a transitive relationship is not captured in pure CF, unless users u_i and u_k have rated many common items (Vozalis & Margaritis, 2003). To remedy the aforementioned weaknesses of CBF and CF, trust-aware RS (Massa & Avesani, 2004) – recommendations are based only on ratings given by users trusted directly or indirectly by the active user – and hybrid filtering recommenders (Balabanovic & Shoham, 1997; Linden et al., 2003; Pazzani, 1999; Shahabi et al., 2001) are proposed. A recent comprehensive survey of the state of the art in RS, various limitations of the current generation, the various ways to extend their capabilities, and various hybridization methods are described by Adomavicius and Tuzhilin (2005) and Burke (2002).

Basically to build a RS using any filtering technique, a model for each user is required which must reflect the user preferences and taste. The key success of any filtering technique comes from the way it learns user models. The CBF learns the model from the features describing the items the user has rated in the past, whereas CF learns the model from ratings of the items themselves (Breese et al., 1998; Resnick et al., 1994; Vozalis & Margaritis, 2003). In real life, users put some priorities for each feature and these priorities need learning also. Ujjin and Bentley (2004) used the evolutionary search to learn these priorities tailoring the recommendation process to the preferences of each individual user.

Formally, in CF recommenders, we have a set of users $U = \{u_1, u_2, \ldots, u_m\}$ rating a set of items $S = \{s_1, s_2, \ldots, s_n\}$, such as books, movies, or CDs. The spaces S and U are large and can be very large in some applications. Each user u_i, $i = 1, 2, \ldots, m$ has rated a subset of items S_i. Specifically, the rating of user u_c for item s_j, $j = 1, 2, \ldots, n$ is denoted by $r_{c,j}$. All the available ratings are collected in a $m \times n$ user-item matrix denoted by R. Four phases are required to perform the recommendation task in CF recommenders:

(a) Data collection
(b) User model formation
(c) Neighborhood set selection
(d) Making recommendations

2.1.1. Data collection

Generally, three types of data can be collected from users, demographical data through the registration process, explicit ratings for a subset of the available items, and implicit data from the user's online behavior. In order to conduct our experiments we used the original MovieLens dataset (http://www.movielens.umn.edu). The dataset consists of 100,000 ratings, assigned by 943 users on 1682 movies. All ratings follow the 1 – bad, 2 – average, 3 – good, 4 – very good, and 5 – excellent numerical scale. Each user has rated at least 20 movies. Simple demographical data such as age, gender, occupation and zip code are included for all users, which are collected when a new user registers on the system. The movie title, release date, video release date, and genre data are given for each movie. The genre feature specifies if the movie is an action, adventure, animation, children's, comedy, crime, documentary, drama, fantasy, film-noir, horror, musical, mystery, romance, sci-fi, thriller, war, or western. A single movie can belong to more than one genre.

2.1.2. User model formation

The difference between a user profile and a user model lies in the different levels of sophistication. The user profile is simply a collection of personal information about the user, which can be described as a simple user model.

Depending on the content and the amount of information about the user, which is stored in the user profile, a user can be modeled (Froschl, 2005; Koch, 2000). To be precise, a user model is a representation of the knowledge and personal characteristics, which the system believes that a user possesses.

The profiling information can be elicited from demographical data (e.g., user's age, gender, occupation, etc.), user preferences about features of the items (e.g., movie title, genre, director, year of release, leading actors, etc.), and user ratings on experienced items (e.g., previously seen movies) (Massa & Avesani, 2004). To build a user model two questions have to be taken into account, what information has to be represented in the model and how this information is effectively represented? The first question is general for all applications while the second is application-dependent (Koch, 2000). An effective, tailored recommendation depends on the underlying user model, which is in turn used for similarity computations. A representative user model would appropriately reflect user's tastes, preferences, and needs.

Most previous work on user model construction relies only on explicit ratings (Breese et al., 1998; Goldberg et al., 1992; Resnick et al., 1994; Shardanand & Maes, 1995). However, in real life, the way in which two people are said to be similar is not based solely on whether they have close opinions on a specific subject, e.g., movie ratings, but also on other factors, such as their background and personal details. Therefore, issues such as age, gender, and preferences of movie genres should also be taken into account (Ujjin & Bentley, 2004).

2.1.3. Neighborhood set selection

Once user models have been established, the system can match the active user to the available database and a set of neighbors for him needs to be formed and ranked according to a suitable distance function. The size of the neighborhood set could be fixed by selecting the top K users or could be variable by selecting the users whose similarity value is above a certain threshold (Breese et al., 1998; Vozalis & Margaritis, 2003). Various functions have been used to compute the distance, $d(a,c)$, between users u_a and u_c in CF recommenders. The most popular function for memory-based CF is the Pearson correlation coefficient (Resnick et al., 1994), where the distance between two users is based only on the ratings both users have declared. The Pearson correlation coefficient is given by

$$\mathrm{corr}(x,y)=\frac{\sum_{s\in S_{xy}}(r_{x,s}-m_x)(r_{y,s}-m_y)}{\sqrt{\sum_{s\in S_{xy}}(r_{x,s}-m_x)^2\sum_{s\in S_{xy}}(r_{y,s}-m_y)^2}}, \tag{1}$$

where S_{xy} is the set of items rated by both users u_x and u_y. Let us call the RS, which uses formula (1) for similarity computations as Pearson RS (PRS).

Clearly, formula (1) is not appropriate if other features are included in the model because it considers only the common items for both users. The modified Euclidean distance function gives another way to compute similarity, which takes into account multiple features (Ujjin & Bentley, 2004)

$$d(\mathbf{x},\mathbf{y})=\frac{1}{z}\sum_{i=1}^{z}\sqrt{\sum_{j=1}^{N}(x_{i,j}-y_{i,j})^2}. \tag{2}$$

Here $x_{i,j}$ is the jth feature for the common item s_i, N is the number of features, and $z=|S_{xy}|$, the cardinality of S_{xy}. Note that a vector of features represents each user therefore it is written bold in formula (2).

2.1.4. Making recommendations

In this phase, RS assign a predicted rating to all the items seen by the neighborhood set and not by the active user. The predicted rating, $\mathrm{pr}_{a,j}$, indicates the expected interestingness of the item s_j to the user u_a, is usually computed as an aggregate of the ratings of user's (u_a) neighborhood set for the same item s_j

$$\mathrm{pr}_{a,j}=\mathrm{aggr}_{u_c\in C}\ r_{c,j}, \tag{3}$$

where C denotes the set of neighbors who have rated item s_j. The most widely used aggregation function is the weighted sum (Adomavicius & Tuzhilin, 2005; Breese et al., 1998; Vozalis & Margaritis, 2003), which is called also Resnick's prediction formula (Resnick et al., 1994)

$$\mathrm{pr}_{a,j}=m_a+k\sum_{u_c\in C}d(\mathbf{a},\mathbf{c})\times(r_{c,j}-m_c). \tag{4}$$

The multiplier k serves as a normalizing factor and is usually selected as $k=1/\sum_{u_c\in C}|d(\mathbf{a},\mathbf{c})|$ and m_c is the average rating of user u_c

$$m_c=\frac{1}{|S_c|}\sum_{s\in S_c}r_{c,s}. \tag{5}$$

3. A novel hybrid user model

The construction of user models is a key task – the system's success will depend to a large extent on the models to represent the users' actual interests. A pure CF user profile (Goldberg et al., 1992; Resnick et al., 1994) consists of a vector of items with their ratings continuously augmented as the user interacts with the system over time. This huge amount of data needs a very large space and a long processing time. At query time, searching the entire database to find the best set of neighbors is computationally very expensive. Eventually, the user will leave the Web site before the processing completes.

On the other hand, the actual user preferences cannot always be captured by explicit ratings, some content descriptions of items are required. This problem gets solved by hybrid filtering. However, most current hybridization methods (Burke, 2002) construct two separate profiles and implement the online process for each filtering technique separately. Finally, some merger to give the final result is used. What if we construct a model according to

some filtering technique and then apply on the constructed model another filtering technique? In our approaches only one online filtering process (CF) is needed, while the other filtering techniques (CBF and DMF) are used to dense the data by building a compact user model, which is an offline process.

Further, a sparse user-item matrix causes a scalability problem for CF. However, we can overcome this problem to some extent by combining or integrating many information sources. In our work, we develop a set of *hybrid* features that combines some of the users' and items' properties. An example of a hybrid feature (Basu, Hirsh, & Cohen, 1998) would be the set of *Movies of the Comedy Genre User X Highly Rated*. We derive the hybrid feature based on the user's ratings for a set of high rated movies and the content descriptions of the genres corresponding to this set of movies. The hybrid features are utilized as the basis for formulating a *genre interestingness measure* (GIM). Once we have an appropriate formula for GIM, a user model can be constructed from DMF user profile and GIMs.

Block diagram of the proposed work is given in Fig. 1. First of all, a hybrid user model is built using the hybrid features and DMF user profile, thereafter CF recommender generates a neighborhood set according to model-based CF. Finally the entire database of this set is retrieved to recommend items according to memory-based CF. Before going to details, let us present necessary formulae for the MovieLens dataset and illustrate GIM computations through an example.

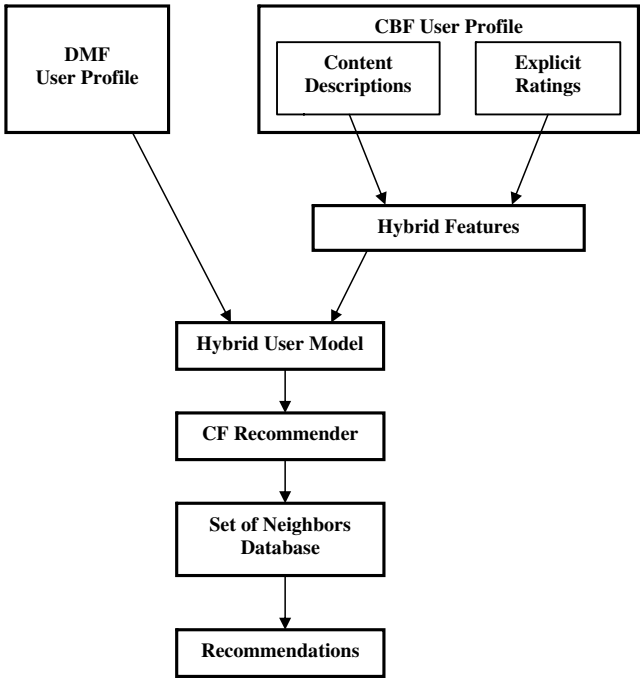

Fig. 1. Block diagram of the proposed work.

3.1. Necessary formulae

The total ratings (TR) of user u_i is

$$\mathrm{TR}(i) = \sum_{s \in S_i} r_{i,s}. \tag{6}$$

Here S_i is the set of movies rated by user u_i. The genre rating (GR) (resp. genre frequency (GF)) for high rated items of genre G_j corresponding to user u_i is computed using formula (7) (resp. formula (8))

$$\mathrm{GR}(i,j) = \sum_{s \in G_j \subset S_i, r \geqslant 3} r_{i,s}, \tag{7}$$

$$\mathrm{GF}(i,j) = \sum_{s \in G_j \subset S_i} \delta_k(r_{is}), \quad k \in \{3,4,5\}, \tag{8}$$

where,

$$\delta_k(r_{i,s}) = \begin{cases} 1, & k = r_{i,s}, \\ 0, & k \neq r_{i,s}. \end{cases} \tag{9}$$

It is to be noted that only the items having rating $r \geqslant 3$ are considered, i.e. the items rated as good (3), very good (4), or excellent (5). Such items would be called items with *high ratings*. Finally, relative genre rating (RGR) (resp. relative genre frequency (RGF)), the ratio of u_i's ratings (resp. frequency) for high rated items of G_j to his total ratings (resp. frequency), is computed as

$$\mathrm{RGR}(i,j) = \frac{\mathrm{GR}(i,j)}{\mathrm{TR}(i)}, \tag{10}$$

$$\mathrm{RGF}(i,j) = \frac{\mathrm{GF}(i,j)}{\mathrm{TF}(i)}, \tag{11}$$

where $\mathrm{TF}(i) = |S_i|$, the cardinality of S_i.

3.2. Example 1

For the sake of simplicity, we consider a table (Table 1) of only three users who have rated movies belonging to four genres. In columns G_i, $i = 1,2,3,4$, one (1) indicates the belongingness of a given movie to G_i, and 0 otherwise. Also a non-zero value in the user ratings columns indicates

Table 1
Data of Example 1

Movie	Corresponding genres				User ratings		
	G_1	G_2	G_3	G_4	User-1	User-2	User-3
1	1	0	1	0	1	5	0
2	1	0	1	1	3	0	0
3	1	1	0	0	1	5	3
4	0	1	1	0	5	1	0
5	0	1	1	**1**	4	0	**4**
6	0	1	0	0	0	2	0
7	1	1	0	1	0	0	0
8	0	0	1	**1**	0	0	**5**
9	0	1	1	0	0	2	0
10	1	1	1	0	0	4	1
11	1	1	0	**1**	0	3	**3**
12	1	0	0	0	0	0	3

that the movie has been rated, and zero indicates that the movie has not been rated yet. Table 2 lists the ratings associated with various genres. It is to be noted that the user rating for a specific movie gets associated with all the genres to which the movie belongs. For example, the entry of user-3 for G_4 is 4, 5, and 3 (highlighted) where they are corresponding to movie-5, movie-8, and movie-11 (Table 1), respectively, and so on. As discussed earlier, the hybrid features are used as the basis for the genre interestingness measure, therefore a user is interested more in G_i if it has more high ratings, i.e. good, very good, or excellent. The number of hybrid features depends on the number of genres. On this basis, one can infer that user-1, user-2, and user-3 are interested more in G_3, G_1, and G_4, respectively (Table 2).

Although some low ratings ($r < 3$) occur for genres but these can be filtered out at the collaborative stage. Our aim here is to find a compact user model to generate close neighbors for that user. This set of neighbors is used in a collaborative manner to recommend some items, which might be liked by the user. Therefore any item with a predicted rating less than good (3) cannot be recommended to the active user.

Based on the hybrid features, one can evolve GIMs according to a suitable formula. To explore the possibility of choosing a formula for GIM, we examined formulae (10) and (11). The first formula, RGR, works well for user-1 and user-2 but for user-3 it gives G_1 and G_3 the same value (Table 3), whereas they are quite different. G_1 has three good (3) ratings while G_3 has very good (4) and excellent (5) ratings. This is because the number of movies for each genre is ignored in this formula. The second formula, RGF, works well for user-2 but for user-3 it gives G_1, G_2 and G_4 the same value, whereas they are quite different. Moreover, G_3 value is lower than that of G_1, and G_2, whereas the ratings for G_3 are very good and excellent. This is because only the number of high rated items is considered and all high ratings contributed equally in this formula.

Table 2
List of ratings associated with various genres

User	TF	TR	G_1	G_2	G_3	G_4
1	5	14	1,3,1	1,5,4	1,3,5,4	3,4
2	7	22	5,5,4,3	5,1,2,2,4,3	5,1,2,4	3
3	6	19	3,1,3,3	3,4,1,3	4,5,1	**4,5,3**

Table 3
Genre interestingness measure according to various formulae

GIM	User	G_1	G_2	G_3	G_4
RGR	1	0.214	0.643	0.857	0.500
	2	0.773	0.545	0.409	0.136
	3	0.474	0.526	0.474	0.632
RGF	1	0.200	0.400	0.600	0.400
	2	0.571	0.429	0.286	0.143
	3	0.500	0.500	0.333	0.500
MRGF	1	0.067	0.333	0.400	0.200
	2	0.429	0.286	0.238	0.048
	3	0.167	0.222	0.278	0.333

Table 4
Comments on genre interestingness measure according to various formulae

GIM	User	Comments
RGR	1	Values reflect the actual order of interests
	2	Values reflect the actual order of interests
	3	G_4 has the highest value as expected. But G_3 and G_1 have the same value while there is a difference between them
RGF	1	G_3 has the highest value as expected. But G_4 and G_2 have the same value while there is a difference between them
	2	Values reflect the order of interests
	3	G_4 has the highest value as expected. But G_1 and G_2 have the same value also. G_3 has smaller value than G_1 and G_2 while user's interest is more for G_3

Comments on all the GIM formulae for the three users are given in Table 4. Obviously, RGR and RGF are appropriate GIM formulae with some weaknesses. To remedy these weaknesses we introduce in the next subsection a formula that combines both RGR and a modified version of RGF.

3.3. The proposed hybrid user model

User preference for a certain item is declared by the rating for that item like good, very good, or excellent. The exact degree of preference is not captured by RGF, because it gives all high ratings the same weight. The following definition introduces a modified version of RGF formula, which tries to reflect the exact preference for items with high ratings.

Definition 1. For a rating-based movie recommender system, the modified relative genre frequency (MRGF) of genre G_j for user u_i is defined as

$$\mathrm{MRGF}(i,j) = \frac{\sum_{s \in G_j \subset S_i} \delta_3(r_{i,s}) + 2 \times \delta_4(r_{i,s}) + 3 \times \delta_5(r_{i,s})}{3 \times TF(i)}. \tag{12}$$

As expected, MRGF (Table 3) overcomes the drawbacks of RGF. To develop more accurate GIM, the relative genre rating needs to be considered also. Indeed, RGR identifies the preferences for genres with some drawbacks as discussed before. However, a combined formula of RGR and MRGF will bring out the best in both formulae. The following definition gives one possible form of such a formula.

Definition 2. For a rating-based movie recommender system, the genre interestingness measure (GIM) of genre G_j for user u_i is defined as

$$\mathrm{GIM}(i,j) = \frac{2 \times \mathrm{nf} \times \mathrm{RGR}(i,j) \times \mathrm{MRGF}(i,j)}{\mathrm{RGR}(i,j) + \mathrm{MRGF}(i,j)}. \tag{13}$$

Here nf is the normalization factor for a given system.

Table 5
Genre interestingness measure according to formula (13)

User	GIM(i,j)			
	G_1	G_2	G_3	G_4
1	0.510	2.194	2.727	1.429
2	2.759	1.876	1.505	0.355
3	1.235	1.561	1.752	2.181

Formula (13) gives the harmonic mean of RGR(i,j), and MRGF(i,j) multiplied by nf. The range of GIM(i,j) is $[0, \text{MAX}]$ that agrees with the system's rating structure, i.e. 1 – bad…MAX – excellent. Here MAX is the maximum possible rating for a given system. The normalization factor nf could take the value of MAX or the global average rating of a given user (TR(i)/TF(i)).

The results based on formula (13) with nf = MAX are shown in Table 5. The results reflect exactly the degree of interestingness for each genre; user-1 is interested more in G_3, user-2 in G_1, and user-3 in G_4. Three main advantages are gained by using formula (13) for GIM. Firstly, the user preferences in genres are captured easily. Secondly, a compact user model is obtained. Finally, GIM ensures the user neighbor transitivity therefore users having a close GIMs are correlated.

GIM represents user's X interestingness for genre Y. However, as noted earlier that two people are said to be similar is not based solely on whether they have close opinions on a specific subject, but also on other factors, such as their background and personal details. Therefore, we construct the user model from DMF user profile and GIMs. The proposed user model consists of age, gender, and occupation as demographical data and GIMs for 18 genres.

4. Fuzzy and hybrid fuzzy-genetic approaches

4.1. Fuzzy approach to recommender systems

Fuzzy sets were introduced by Zadeh (1965) as a generalization of the classical crisp sets in order to deal with vague concepts, like "young", "rich", "tall", and so on. Instead of the hard belongingness of the items to the crisp set (1 if the item belongs to the set, and 0 otherwise), fuzzy set allows items to have partial degree of belongingness, i.e. any value in the interval $[0,1]$. To develop a fuzzy RS, the user model needs to be fuzzified and an appropriate distance function needs to be proposed to match different users with fuzzy models. The following subsections discuss how to fuzzify the model and introduce a fuzzy distance function.

4.1.1. User model fuzzification

The crisp description of the age and genre interestingness measure does not reflect the actual case for human decisions. The distance, for example, between two users of age 15 and 19 is 4 while both users belong to the same age group, i.e. teenager. These fuzzy features need to be taken into account when doing the inter-user comparisons. In this subsection, our aim is to fuzzify the proposed model to get as close as possible a set of neighbors for the active user. First of all, age is fuzzified into three fuzzy sets (Klir & Yuan, 1995), young, middle-aged and old as shown in Fig. 2, with the following membership functions:

$$A_{\text{Young}}(x) = \begin{cases} 1, & x \leqslant 20, \\ (35-x)/15, & 20 < x \leqslant 35, \\ 0, & x > 35, \end{cases} \tag{14a}$$

$$A_{\text{Middle}}(x) = \begin{cases} 0, & x \leqslant 20, \quad x > 60, \\ (x-20)/15, & 20 < x \leqslant 35, \\ 1, & 35 < x \leqslant 45, \\ (60-x)/15, & 45 < x \leqslant 60, \end{cases} \tag{14b}$$

$$A_{\text{Old}}(x) = \begin{cases} 0, & x \leqslant 45, \\ (x-45)/15, & 45 < x \leqslant 60, \\ 1, & x > 60. \end{cases} \tag{14c}$$

The gender and occupation values are considered as fuzzy points with membership value of one. Finally, the genre interestingness measure is fuzzified into six fuzzy sets, very bad (VB), bad (B), average (AV), good (G), very good (VG), and excellent (E) with the following membership functions (Fig. 3):

$$B_{\text{VB}}(x) = \begin{cases} 1-x, & x \leqslant 1, \\ 0, & x > 1, \end{cases} \tag{15a}$$

$$B_{a(i)}(x) = \begin{cases} 0, & x \leqslant i-2, \quad x > i, \\ x-i+2, & i-2 < x \leqslant i-1, \quad i = 2,3,4,5. \\ i-x, & i-1 < x \leqslant i, \end{cases} \tag{15b}$$

Here $a(i) = B, \text{AV}, G,$ and VG for $i = 2,3,4,$ and 5, respectively.

$$B_{\text{E}}(x) = \begin{cases} 0, & x \leqslant 4, \\ x-4, & 4 < x \leqslant 5. \end{cases} \tag{15c}$$

4.1.2. Fuzzy distance function

A great advantage can be gained by fuzzifying the user model, but how can we compare two models having many features? In turn, each feature has many fuzzy sets. Actually, the choice of a distance function is an important issue for the system and depends heavily on the problem itself.

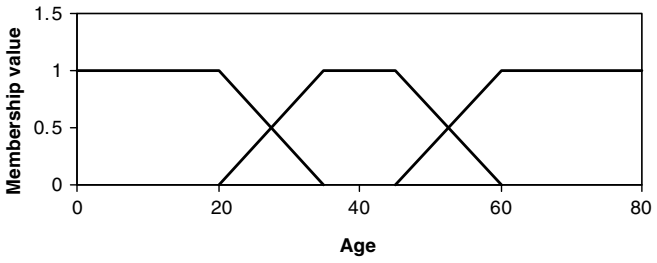

Fig. 2. Membership functions for age.

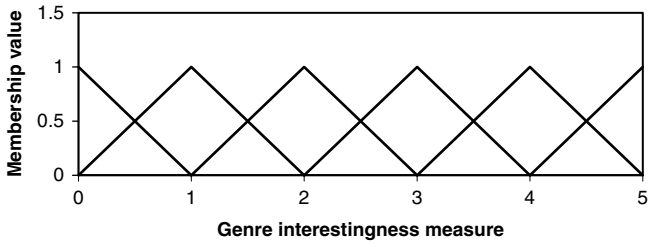

Fig. 3. Membership functions for genre interestingness measure.

The distance between two fuzzy sets or points is discussed extensively in the literature (Dumitrescu, Lazzerini, & Jain, 2000; Klir & Yuan, 1995). For our model, a vector of 21 features represents the user. A local fuzzy distance should be found for each feature. Hence, for each pair of users, we have 21 local fuzzy distances. The global fuzzy distance is obtained by two methods (Gadi, Daoudi, & Matusiak, 1999). The first method uses a fuzzy IF-THEN rule of the form:

IF (x_1 is close to y_1) and (x_2 is close to y_2)... and (x_{21} is close to y_{21}) THEN ($\mathbf{x}$ is similar to $\mathbf{y}$).

In this case, the global fuzzy distance is given by

$$\mathrm{fd}(\mathbf{x},\mathbf{y}) = \min\{\mathrm{fd}(x_1,y_1), \mathrm{fd}(x_2,y_2), \ldots, \mathrm{fd}(x_{21},y_{21})\}. \quad (16)$$

The second method considers each fuzzy distance between two features as an opinion. The global fuzzy distance is a global opinion from all. In fuzzy logic, this requires an aggregation operator. The aggregation operator may be the average of the 21 local fuzzy distances:

$$\mathrm{fd}(\mathbf{x},\mathbf{y}) = \frac{1}{21}\sum_{i=1}^{21}\mathrm{fd}(x_i,y_i). \quad (17)$$

For our model, formula (16) does not work well because it considers only the feature with the minimum distance and ignores the remaining features. According to fuzzy sets and distance concepts, we need a local fuzzy distance metric, $\mathrm{fd}(x_i,y_i)$, that fulfils the following conditions:

(A) A zero value for the same feature values.
(B) A zero value for different feature values in the same fuzzy set having the same membership values.
(C) Minimize the distance between any two feature values belonging to the same fuzzy set having near membership values.
(D) Maximize the distance between any two feature values belonging to two different fuzzy sets.

The condition (A) is the basic requirement for any distance function. To clarify the condition (B), assume that we have two users with age 40 and 35. Both users are middle-aged with membership values of one (Fig. 2). The crisp distance between them is 5, whereas they are similar users from the fuzzy sets point of view. To make the distance between the two users zero, we need another term gives zero value for this and similar cases. What makes these two users similar is their equal membership values in the same fuzzy set. In the following, we define an appropriate fuzzy distance function that satisfies all the above four conditions.

Definition 3. Let $\mathbf{a}$ and $\mathbf{b}$ be the membership vectors correspond to two crisp values a, and b for a given feature with l fuzzy sets. The fuzzy distance between a and b is defined as

$$\mathrm{fd}(a,b) = d(\mathbf{a},\mathbf{b}) \times d(a,b), \quad (18)$$

where $d(a,b)$ is simply the difference operator, $\mathbf{a}$ and $\mathbf{b}$ are vectors of size l, and $d(\mathbf{a},\mathbf{b})$ is any vector distance metric.

In our experiments, the Euclidean distance function is used for $d(\mathbf{a},\mathbf{b})$:

$$d(\mathbf{a},\mathbf{b}) = \sqrt{\sum_{j=1}^{l}(a_j - b_j)^2}, \quad (19)$$

where a_j is the membership value of the feature a in its jth fuzzy set. To illustrate the effectiveness of the proposed fuzzy distance function, let us consider the following example.

Example 2. Suppose we have to compute fuzzy distance between two users having the age:

Case i: 35 and 40,
Case ii: 45 and 60,
Case iii: 18 and 23.

According to Fig. 2,

Case i: $\mathbf{a} = \langle 0,1,0\rangle$, $\mathbf{b} = \langle 0,1,0\rangle$. Therefore

$$d(\mathbf{a},\mathbf{b}) = \sqrt{2\times(0-0)^2 + (1-1)^2} = 0,$$

$$d(35,40) = 40 - 35 = 5,$$

$$\mathrm{fd}(35,40) = 0 \times 5 = 0 \quad \text{(Similar users)}.$$

Case ii: $\mathbf{a} = \langle 0,1,0\rangle$, $\mathbf{b} = \langle 0,0,1\rangle$. Therefore

$$d(\mathbf{a},\mathbf{b}) = \sqrt{(1-0)^2 + (0-1)^2} = \sqrt{2},$$

$$d(60,45) = 60 - 45 = 15,$$

$$\mathrm{fd}(60,45) = \sqrt{2} \times 15 \quad \text{(Far users)}.$$

Case iii: $\mathbf{a} = \langle 1,0,0\rangle$, $\mathbf{b} = \langle 0.8,0.2,0\rangle$. Therefore

$$d(\mathbf{a},\mathbf{b}) = \sqrt{(0.8-1)^2 + (0.2-0)^2} = 0.283,$$

$$d(23,18) = 23 - 18 = 5,$$

$$\mathrm{fd}(23,18) = 0.283 \times 5 \quad \text{(Near users)}.$$

Results in Example 2 show that formula (18) satisfies all the four conditions of the required local fuzzy distance metric. Accordingly, for the fuzzy approach, the fuzzy distance function between two users can be aggregated using

formula (17) or the Euclidean distance function as given below:

$$\mathrm{fd}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{21}(\mathrm{fd}(x_i,y_i))^2}, \tag{20}$$

where $\mathrm{fd}(x_i,y_i) = d(\mathbf{x_i},\mathbf{y_i}) \times d(x_i,y_i)$, $d(\mathbf{x_i},\mathbf{y_i}) = \sqrt{\sum_{j=1}^{l}(x_{i,j}-y_{i,j})^2}$, l is the total number of fuzzy sets for the ith feature, and $x_{i,j}$ is the membership value of the ith feature in the jth fuzzy set. Let us call the RS uses formula (20) for similarity computations as Fuzzy RS (FRS).

4.2. Hybrid fuzzy-genetic approach to recommender systems

The proposed user model contains 21 features contributing equally during similarity computations in FRS. This does not capture the real life situation where users put different weights to different features. These weights are subject to change with time and evolving preferences of each user. An efficient learning mechanism to capture these weights is required. By imposing features' weights to formula (20), genetic algorithm (GA) can be used to find these weights leading to a hybrid fuzzy-genetic RS (FGRS). For this approach formula (20) takes the following form:

$$\mathrm{fd}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{j=1}^{21} w_j \times (\mathrm{fd}(x_j,y_j))^2}, \tag{21}$$

where w_j is the weight for the jth feature. The following subsections give a brief introduction to genetic algorithms and the fitness function.

4.2.1. Genetic algorithms

A genetic algorithm processes a population of competing candidate solutions. The chromosome decodes a set of tuned parameters mapped into a potential solution to an optimization problem (Goldberg, 1989). An objective function evaluates the quality of a solution, which is called a *fitness function*. Genetic operators such as crossover and mutation are applied to the parents in order to generate new offspring. Individuals that achieve a high fitness are more likely to be selected as parents and generate offspring by means of crossover and mutation. The good newly generated individuals (if exist) replace the current bad individuals to form the new population for the next generation. GA terminates either if a maximum number of generations elapses or a desired level of fitness is reached (Goldberg, 1989). Every user puts a different priority on each feature, which can be referred to as feature weight. In order to implement truly personalized RS, these weights need to be captured and fine-tuned to reflect each user's preference. In our experiment, GA adapts the feature weights to capture the user priorities for different features (Ujjin & Bentley, 2004).

The feature weights of user u_a are represented as a set of weights, $\mathrm{weight}(u_a) = [w_i]_{i=1,\ldots,n}$, where n is the number of features. The genotype of w_i is a string of binary values. When the weight for any feature is zero, that feature is ignored. This enables feature selection to be adaptive to each user's preference.

4.2.2. The fitness function

Finding an appropriate fitness function is a challenging problem for GA applications (Goldberg, 1989). For this application it is not a trivial task, every set of weights in the GA population must be employed by the users matching process within the RS so the RS needs to be re-run on the entire database for each new set of weights in order to find the fitness (Ujjin & Bentley, 2004). A poor (good) set of weights might (should) result in a poor (good) neighborhood set of users for the active user, and hence poor (good) recommendations. One way to find the fitness function is by reformulating the problem as a *supervised learning task*. For that purpose the actual ratings of the active user are randomly divided into two disjoint sets, test ratings set (66%) and training rating set (34%). To find the fitness score for the evolved set of weights, the RS must be operated and the predicted ratings for each movie in the training ratings set must be computed. The average of the differences (Ujjin & Bentley, 2004) between the actual and predicted ratings of all movies in the training ratings set is used as the fitness score for that set of weights

$$\mathrm{fitness} = \frac{1}{n_R}\sum_{j=0}^{n_R}|r_j - \mathrm{pr}_j|, \tag{22}$$

where n_R is the training ratings set cardinality of a given active user.

5. Experiments

Based on MovieLens dataset we considered only users who have rated at least 60 movies, 20 to build a user model and 40 for testing. Out of 943 users only 497 users satisfied this condition and contributed 84,596 ratings out of 100,000. This dataset is used as the basis to generate five random splits into training and active users. For each random split, 50 users were chosen randomly as active users, and the remaining 447 users as training users – treated as historical data for the RS. Such a random separation was intended for the execution of five-fold cross-validation, where all the experiments are repeated five times, once with each split. These splits will be referred to as split-1, split-2, ..., split-5. The set of training users (447 users) is used to find a set of neighbors for the active user while the set of active users (50 users) is used to test the performance of the system. During the testing phase, each active user's ratings are divided randomly into two disjoint sets, training ratings (34%) and test ratings (66%). The training ratings are used to model the user

and to supervise the learning process of FGRS, whereas the test ratings are treated as unseen ratings that the system would attempt to predict.

Two evaluation metrics are used to evaluate the effectiveness of different RS, the mean absolute error (MAE), and the total coverage of the system. The MAE measures the deviation of predictions generated by the RS from the true ratings specified by the user. The MAE(i) for active user u_i (Breese et al., 1998; Vozalis & Margaritis, 2003) is given by the following formula:

$$\mathrm{MAE}(i) = \frac{1}{n_i}\sum_{j=1}^{n_i} |\mathrm{pr}_{i,j} - r_{i,j}|, \tag{23}$$

where n_i is the cardinality of the test ratings set of user u_i. The total MAE over all the active users, N_T (in our experiments $N_\mathrm{T} = 50$) can be calculated as

$$\mathrm{MAE} = \frac{1}{N_\mathrm{T}}\sum_{i=1}^{N_\mathrm{T}} \mathrm{MAE}(i). \tag{24}$$

Lower MAE corresponds to more accurate predictions of a given RS. On the other hand, coverage is the measure of the percentage of items for which a RS can provide predictions. The RS may not be able to make predictions for every item. Low coverage value indicates that the RS will not be able to assist the user with many of the items he has not rated (Breese et al., 1998; Massa & Avesani, 2004). We compute coverage as the percentage of items over all users for which a prediction was requested and the system was able to produce a prediction (Vozalis & Margaritis, 2003)

$$\mathrm{Coverage} = \frac{\sum_{i=1}^{N_\mathrm{T}} p_i}{\sum_{i=1}^{N_\mathrm{T}} n_i}. \tag{25}$$

Here, p_i is the total number of predicted items for active user u_i.

In the following two experiments, all the five splits of data are used to show the effectiveness of the proposed user model, fuzzy and hybrid fuzzy-genetic approaches to RS. All experiments are conducted on a PC with 2.66 GHz Intel (Pentium 4) CPU, 256MB RAM.

5.1. Experiment 1

In this experiment we run the proposed FRS and compare its results with classical PRS. For PRS experiment, if the correlation coefficient is negative, then users u_x and u_y are negatively correlated. This means that each user discourages the other. In our experiments, we follow the most used strategy of keeping only positive correlation values because users with a negative correlation are dissimilar to the active user and hence it is better not to consider their ratings (Massa & Avesani, 2004). The neighborhood set size is kept 30 for all the experiments.

Usually, authors run PRS for a selected set of users only but in this experiment we run PRS over entire training users' database, even if it is time consuming. The difference between gender (occupation) values is either 0, if the two users have the same gender (occupation) or 1 otherwise. This agrees with our reasoning for making opposite values as far as possible. Moreover, some sort of normalization (Han & Kamber, 2001) is used for age values to ensure that the values fall within the same range of GIM range, i.e. $[0, 5]$. Each age value is multiplied by (5/80), where age 80 is assumed to be the maximum possible age.

The system picks the movies, from the test ratings set of the active user, one by one. Thereafter predicts ratings for them using formula (4) over the set of all neighbors who have rated the same movie. After getting the predicted ratings, the system compares them with the actual ratings given by the active user. Figs. 4 and 5 show the correct prediction percentage obtained from PRS and FRS for the fifty active users of split-1 (best split) and split-3 (worst split), respectively. Each graph shows the percentage of the number of ratings that the system predicted correctly out of the total number of available test ratings by the active user.

5.1.1. Analysis of the results

The implementation of PRS for 50 active users took around 13.16 min, while it was around 2.13 min only for FRS for the same set of active users. The computational time complexity is reduced by approximately a factor of six with the proposed user model. Results summarized in

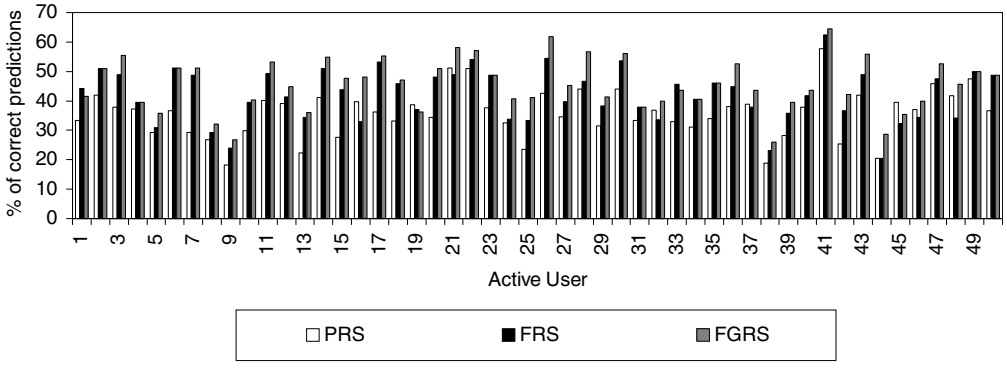

Fig. 4. Correct predictions percentage for active users of split-1.

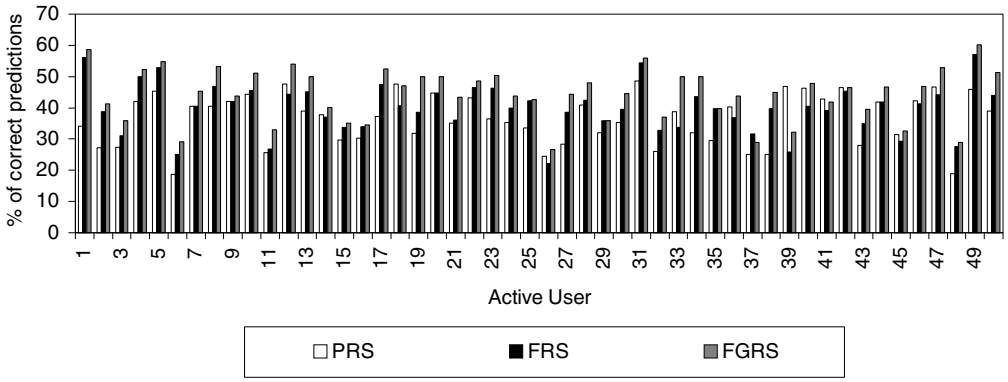

Fig. 5. Correct predictions percentage for active users of split-3.

Table 6 show that FRS outperforms PRS for all the five splits. Results of FRS are better than PRS values on 66% on the worst split (split-3) and 82% on the best split (split-1). The total MAE for FRS is always smaller than the corresponding value for PRS as shown in Table 7, whereas the coverage is greater for all the splits except split-3 where it is smaller. The higher prediction values for FRS obviously illustrate that better set of matching users is found and therefore the accuracy of the RS gets enhanced. On the other hand, the lack of access to the content of the movies and the non-existence of neighborhood transitivity relationship in a classical PRS, prevent similar users from being matched unless they have rated exactly the same movies.

5.2. Experiment 2

An elitist genetic algorithm is used in this experiment for evolving features' weights using simple parameter values as shown in Table 8. A simple unsigned binary encoding is

Table 6
Comparison between FSR with PRS, FGRS with PRS, and FGRS with FRS

Split	FRS with PRS			FGRS with PRS			FGRS with FRS		
	Greater	Same	Smaller	Greater	Same	Smaller	Greater	Same	Smaller
1	41	1	8	48	0	2	38	9	3
2	37	2	11	42	4	4	44	4	2
3	33	4	13	46	1	3	47	2	1
4	35	4	11	45	2	3	39	6	5
5	29	10	11	45	2	3	48	2	0

Table 7
Total MAE and coverage for PRS, FRS, and FGRS

Split	MAE			Coverage		
	PRS	FRS	FGRS	PRS	FRS	FGRS
1	0.805011	0.707672	0.663464	95.27965	96.82935	97.04311
2	0.844795	0.768797	0.716272	95.54466	95.88573	96.22681
3	0.791428	0.719163	0.675140	95.90455	95.77991	96.38533
4	0.776416	0.713459	0.663480	94.67433	95.02690	95.62071
5	0.770952	0.720282	0.660131	95.70990	96.97650	97.56895

Table 8
GA parameter values used in Experiment 2

Parameter name	Parameter value	Description
Population size	10	The size of individuals in the population at each generation
Termination threshold	0.025	When the fitness score of the best individual is below the threshold, a good solution is found and this set of weights is used as the final result for the current run
Maximum generation (per run)	30	If the number of generations reaches this value and the solution has not been found, the best individuals for that generation are used as the final result
Number of runs	10	The number of times the system was run for each active user

used in the implementation, using 8-bits for each of the 21 genes. The GA begins with random genotypes. A decimal value for each weight is obtained by converting the alleles of the binary genes to decimal. The total decimal value of phenotype is then calculated by summing up the decimal values for all the 21 weights. Finally, by dividing the decimal value of each weight by the total value, the weighting value for each feature can be found.

A supervised learning is used to learn weights. The GA learns weights using guidance of the actual ratings in the training ratings of the active user. The evolved set of weights is employed to find the predicted ratings for each movie in the training ratings set. The average of the differences between the actual and predicted ratings of all movies in the training set is used as the fitness score for that set of weights. The fitness function used is the same as given in Section 4.2.2.

The population of size 10 is kept fixed for all the experiments. Ten more individuals are generated from generation to generation. The best ones from these new 10 individuals replace the bad ones (if exist) of the previous fixed size population individuals. Parents are selected randomly from the top 8 individuals to generate 8 new individuals using a single point crossover. The mutation is applied to generate the remaining two individuals.

The predicted rating for a given movie in the test ratings set is computed using formula (4) over the set of all neighbors who have rated that movie. The predicted ratings, produced by the FRS and the hybrid FGRS, are compared against those produced by the PRS as shown in Figs. 4 and 5.

5.2.1. Analysis of the results

For FGRS, out of 10 runs for each active user, the run with the best weights was chosen and plotted against the results from the PRS and FRS as shown in Figs. 4 and 5 for split-1 and split-3, respectively. Results summarized in Table 6 show that FGRS outperforms PRS and FRS for all the five splits, where it is greater than PRS (FRS) values on 92% (94%) for split-3 and 96% (76%) for split-1. The MAE for FGRS is always smaller than the corresponding values for PRS and FRS, while the coverage value is always greater for all the splits.

The main features that reflect users' preferences are different for different users. For example, some users rely mainly on their explicit ratings, others on the similar age and gender groups, whereas some others rely on all features but with a little difference in their priorities. Looking at the final weights obtained for each active user, many interesting observations are made. Some users give more emphasis on specified features more, while others do not show any interest for some features. Let us focus our discussion only on two splits, split-1 (Fig. 4) and split-3 (Fig. 5), and consider the following observations:

- The correct predictions of the active user 15 (split-1) are higher (43.81%) for FRS as compared to PRS (27.62%). By degrading the effect of the gender feature this value further increased to 47.6% using FGRS.
- When all features contributing equally, the correct predictions for the active user 18 (split-3) are smaller (40.64%) than that of PRS (47.59%). However, by ignoring the gender feature and emphasizing more on the occupation, the correct predictions increased to 47.10%.
- For the active users 8 (split-1) and 10 (split-3), the correct predictions of FRS are higher than that of PRS. Further by ignoring the age these values increased to 32.06% and 51.14%, respectively.
- FRS performed better than PRS for the active user 11 (split-3). However, by emphasizing more on the demographical data, the correct predictions increased to 32.93%.
- By incorporating the demographical data for the active user 37 (split-3), the correct predictions are higher (31.58%) for FRS as compared to PRS (25%). However, by ignoring the demographical data and relying only on GIMs this value deceased to 28.95% using FGRS.
- By ignoring the age and occupation features of the active user 43 (split-3), the correct predictions of FGRS exceed that of FRS and PRS.
- For the active user 10 (split-1), the correct predictions are higher (39.52%) for FRS as compared to PRS (29.84%). Further, by emphasizing more on the age this value increased to 40.32%.
- The feature weights of the active user 28 (split-1) are interesting. They show that when all features are contributing equally using FRS, the number of correct predicted movies increases only by 4. As soon as the gender feature is ignored and the age takes high weight, FGRS outperforms the PRS by 19 movies.
- The correct predictions of the active user 32 (split-1) are smaller (33.6%) under FRS than that of PRS (36.8%). However, when the weight of the gender feature gets higher, this value increased to 40% using FGRS.

Table 9
A summary of the results corresponding to the 7th active user of split-1

RS	Correct predictions (%)	MAE (7)	Coverage (7) (%)	Neighbors		Age range
				Male	Female	
Pearson	29.268	0.900	97.56	23	7	19–70
Fuzzy	48.780	0.675	97.56	1	29	22–51
Fuzzy-genetic	51.219	0.675	97.56	0	30	22–51

- FGRS performed worse than PRS but better than FRS for the active users 45 (split-1), and 39, 41 (split-3). This is because the evolved weights obtained till now in our experiments did not describe those users realistically.

From the above observations, we can say that demographical data is more than or at least as important as ratings for some active users. The logic for FRS agrees with the real life everyday experience where most people listen to the recommendations made by their friends who are most likely to be in their own group. Table 9 illustrates one such case where the active user is a female aged 32. Out of 30 recommenders for this female active user, 23 are males with age ranging from 19 to 70 using the classical PRS, whereas for FRS, 29 recommenders are females in the age group 22–51.

6. Conclusion

This work has shown a considerable reduction in the complexity of recommender system (RS). Building concise and representative user model reduces the effect of user item matrix sparsity, and the computational time complexity that is reduced by approximately a factor of six for our approaches as compared to Pearson recommender system (PRS). Human decisions employ a wide range of fuzzy terms therefore a substantial improvement is gained by fuzzifying the user model. Amongst the various distance functions, the proposed fuzzy distance function reflects more appropriately the fuzziness of each fuzzy feature. Experimental results show that the proposed fuzzy and hybrid fuzzy-genetic approaches considerably improved the accuracy of RS as compared to simple RS like PRS.

By employing a GA, each user priority for each feature is captured. The computational time complexity is the major problem for this approach, however this difficulty gets overcome by performing fine-tuning offline and the best set of feature weights is stored on the user's local machine or in a separate weight matrix along with the user item matrix. This set of weights that we have evolved for each user can be used as the initial weights during future access to the system.

Because of the computational time cost of GA, weights for FGRS are found based on small parameter values (Table 8). By increasing the parameter values of GA, better and more representative weights can be evolved. However, one of the most promising research directions for evolving appropriate weights would be the use of a learnable evolution model (LEM) (Michalski, 2000), which has shown speed-up of two or more orders of magnitude in terms of the number of evolutionary steps in its early experiments.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transaction on Knowledge and Data Engineering, 17*(6), 734–749.

Balabanovic, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM, 40*(3), 66–72.

Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 15th national conference on artificial intelligence, Madison, Wisconsin*.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on uncertainty in artificial intelligence* (pp. 43–52). Madison, WI/San Francisco, CA: Morgan Kaufmann.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*, 331–370.

Dumitrescu, D., Lazzerini, B., & Jain, L. C. (2000). *Fuzzy sets and their application to clustering and training*. CRC Press LLC.

Eirinaki, M., & Vazirgiannis, M. (2003). Web mining for web personalization. *ACM Transactions on Internet Technology, 3*(1), 1–27.

Froschl, C. (2005). User Modeling and User Profiling in Adaptive E-learning Systems. Master Thesis, Graz University of Technology, Graz, Austria.

Gadi, T., Daoudi, R. B. M., & Matusiak, S. (1999). Fuzzy similarity measure for shape retrieval. In *Vision Interface '99, Trois-Rivières, Canada* (pp. 19–21).

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Pearson Education.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*(12), 61–70.

Han, J., & Kamber, M. (2001). *Data mining, concepts and techniques*. Academic Press.

Klir, G., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic, theory and applications*. Prentice-Hall.

Koch, N. (2000). Software Engineering for Adaptive Hypermedia Systems. PhD thesis, Ludwig-Maximilians-University Munich/Germany.

Krulwich, B. (1997). Lifestyle Finder: Intelligent user profiling using large-scale demographic data. *Artificial Intelligence Magazine, 18*(2), 37–45.

Lang, K. (1995). NewsWeeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning, Tahoe City, CA*.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing, 7*(1), 76–80.

Massa, P., & Avesani, P. (2004). Trust-aware collaborative filtering for recommender systems. *CoopIS/DOA/ODBASE (1)*, 492–508.

Michalski, R. (2000). Learnable evolution model: Evolutionary processes guided by machine learning. *Machine Learning, 38*, 9–40.

Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., & Riedl, J. (2003). MovieLens unplugged: Experiences with an occasionally connected recommender system. In *Proceedings of the international conference on intelligent user interfaces*.

Nasraoui, O., & Petenes, C. (2003). An intelligent web recommendation engine based on fuzzy approximate reasoning. In *Proceedings of the IEEE international conference on fuzzy systems* (pp. 1116–1121).

Pazzani, M. (1999). A framework for collaborative, content-based, and demographic filtering. *Artificial Intelligence Review*, 393–408.

Resnick, P., Iakovou, N., Sushak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of Netnews. In *Proceedings of ACM conference on computer supported cooperative work* (pp. 175–186). NC: Chapel Hill.

Shahabi, C., Banaei-Kashani, F., Chen, Y., & McLeod, D. (2001). Yoda: An accurate and scalable web-based recommendation systems. In *Proceedings of the sixth international conference on cooperative information systems (CoopIS 2001), Trento, Italy*.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating 'Word of Mouth'. In *Proceedings of the*

conference on human factors in computing systems (CHI'95), Denver (pp. 210–217).

Suryavanshi, B. S., Shiri, N., & Mudur, S. P. (2005). A fuzzy hybrid collaborative filtering technique for web personalization. In *Proceedings of the third workshop on intelligent techniques for web personalization (ITWP'05), Edinburgh, Scotland, UK*.

Ujjin, S., & Bentley, P. (2004). Using evolutionary to learn user preferences. In K. Tan, M. Lim, X. Yao, & L. Wang (Eds.), *Recent advances in simulated evolution and learning* (pp. 20–40). World Scientific Publishing.

Vozalis, E., & Margaritis, K. G. (2003). Analysis of recommender systems' algorithms. In *Proceedings of the sixth Hellenic-European conference on computer mathematics and its applications (HERCMA). Athens, Greece*.

Zadeh, L. A. (1965). Fuzzy sets. *Information Control, 8*, 338–353.