User Model User-Adap Inter (2006) 16: 63–82
DOI 10.1007/s11257-006-9005-6

ORIGINAL PAPER

TV program recommendation for multiple viewers based on user profile merging

Zhiwen Yu · Xingshe Zhou · Yanbin Hao · Jianhua Gu

Received: 1 June 2005 / Accepted in revised form: 16 April 2006 / Published online: 10 June 2006

Abstract Since today's television can receive more and more programs, and televisions are often viewed by groups of people, such as a family or a student dormitory, this paper proposes a TV program recommendation strategy for multiple viewers based on user profile merging. This paper first introduces three alternative strategies to achieve program recommendation for multiple television viewers, discusses, and analyzes their advantages and disadvantages respectively, and then chooses the strategy based on user profile merging as our solution. The selected strategy first merges all user profiles to construct a common user profile, and then uses a recommendation approach to generate a common program recommendation list for the group according to the merged user profile. This paper then describes in detail the user profile merging scheme, the key technology of the strategy, which is based on total distance minimization. The evaluation results proved that the merging result can appropriately reflect the preferences of the majority of members within the group, and the proposed recommendation strategy is effective for multiple viewers watching TV together.

Keywords Digital television · Television program recommendation · Multiple viewers · User profile merging · Total distance minimization

Z. Yu (✉) · X. Zhou · J. Gu
School of Computer Science, Northwestern Polytechnical University, P.R. China
e-mail: zhiwenyu@nwpu.edu.cn

Y. Hao
Management School, Northwestern Polytechnical University, P.R. China
e-mail: haoyanbin-000@sohu.com

X. Zhou
e-mail: zhouxs@nwpu.edu.cn

J. Gu
e-mail: gujh@nwpu.edu.cn

1 Introduction

The rapid growth of communication technologies and the invention of the set-top-box (STB) and personal digital recorder (PDR) have enabled today's digital television to receive and store tremendous numbers of television (TV) programs. The abundance of TV programs precipitates a need for smart "recommenders" to help people obtain programs that they really want to watch. To fulfill this requirement, many personalized TV systems have been built to assist individual users in filtering and selecting programs based on their respective personal preferences.

However, for social custom and economic reasons, television is often a common equipment in both private (e.g. living room) and public areas (e.g. student dormitory). The TV is usually viewed by multiple members of the groups sitting together. So the TV recommendation system should not only provide personalized programs for individuals, but also be able to recommend programs to multiple viewers taking care of the preferences of the majority of viewers, in the case where the viewers are watching TV at the same time, and in the same spot. Existing TV recommendation systems do not propose solutions to address this issue. This paper proposes a TV program recommendation strategy for multiple viewers using profile merging. The user profile merging is based on total distance minimization.

The rest of this paper is organized as follows. Section 2 discusses previous work relevant to this paper. Section 3 provides three alternative strategies to achieve program recommendation for multiple television viewers, discusses, and analyzes their advantages and disadvantages, and then chooses the strategy based on profile merging as our solution. Section 4 describes the user profile merging scheme, which is based on total distance minimization. The prototype implementation and evaluation are presented in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Related work

There has been much research done in the area of TV program personalization. Several personalized TV systems have been built in recent years to help users deal with the overabundant TV programs, such as PTV (Smyth and Cotter 2004), Interactive Personalized TV (O'Sullivan et al. 2004), TV3P (Yu and Zhou, 2004), TV recommender (Gutta et al. 2000), TV-Advisor (Das and Horst 1998), and P-EPG (Ehrmantraut et al. 1996). But none of them have proposed solutions to recommend TV programs for multiple viewers. A possible solution for recommending TV programs to multiple viewers is to model a group by merging individual user models. This is referred to as *group recommendation* or *group modeling*. There have been several group recommender systems presented during the past few years.

Group Modeling (Masthoff 2004) discusses different strategies for combining individual user models to select TV items to suit groups of viewers. Many issues related with group recommendation are discussed, such as normalization, linearity, misery, order, solidarity, and fairness. Through experiments, the author explores how viewers select programs for a group to watch based on ratings for each of them, investigates how satisfied the group believe they would be with programs chosen by different strategies, and analyzes three proposed algorithms for presenting a sequence of items that take order and ratings into account.

TRAVEL DECISION FORUM (Jameson 2004) proposes a user interest aggregation method for group recommender by allowing the current member optionally to view (and perhaps copy) the preferences already specified by other members. This method has several advantages, such as saving of effort, learning from other members, and encouraging assimilation to facilitate the reaching of agreement.

INTRIGUE (Ardissono et al. 2003) recommends tourist attractions for heterogeneous groups of tourists that include relatively homogeneous subgroups (e.g. children and disabled).

PolyLens (O'Conner et al. 2001) recommends movies to groups of users. It is extended from MovieLens system, which is based on an individual's taste as inferred from ratings and collaborative filtering. PolyLens allows users to create groups and ask for a recommendation for that group. It also explores several design issues of a group recommender, such as the nature of a group, how groups are formed, the privacy issues in showing recommendations to groups, etc.

A hypertext system's network structure is used to store the aggregation of users' collective knowledge on a given domain, which the author refers to as Group User Model (Bollen 2000). The group user model is combined with individual user interests to generate personalized hyperlink recommendations.

Let's Browse (Lieberman et al. 1999) recommends web pages to a group of two or more users who are browsing the web together. It is assisted by an intelligent agent to keep track of the user's interests, and then uses a simple linear combination of the profiles of each user.

MusicFX (McCarthy and Anagnost 1998) selects background music to suit a group of people in a fitness center. The system generates a group preference from the individual preferences that have been previously specified by the members who are currently working out.

Our work differs from previous work in several aspects. First, we provide TV program recommendation to multiple viewers through merging user profiles. INTRIGUE (Ardissono et al. 2003) recommends tourist attractions to heterogeneous groups of tourists by combining the subgroup-related satisfaction scores in a weighted way. By its very nature, it merges recommendations rather than user models. PolyLens (O'Conner et al. 2001) obtains movie recommendations for group users by employing traditional collaborative filtering techniques. It does not address the problem from the perspective of user model merging. Second, we merge individual user preferences on features (e.g. genre, actor, and keyword about a program) not individual ratings on programs. Group Modeling (Masthoff 2004) arrives at a recommendation decision to a group of users through combining individual user ratings on whole programs rather than features. Furthermore, Group Modeling recommends a sequence of programs and tries to make nobody in the group really unhappy (avoiding misery), while our approach recommends one TV program at a time and makes the majority of the group happy. Third, as for profile merging, our merging algorithm is based on total distance minimization. We are the first to introduce distance concepts to measure the difference between user profiles. The total distance minimization guarantees that the merged result should be close to most users' preferences. Let's Browse (Lieberman et al. 1999) and MusicFX (McCarthy and Anagnost 1998) use very rough methods for user preference merging, such as simple linear combination and sum of square. To aggregate a group of users' interests, TRAVEL DECISION FORUM (Jameson 2004) provides an interface for explicit preferences input and allows each member to see the other member's preferences. Neither user profile merging nor recommendation merging is used. Group User Model (Bollen 2000) is the collective knowledge of a group of users on a given domain transformed from hyperlink structure. It is used to improve and recommend hyperlinks to individual users rather than a group of users.

3 Alternative strategies

In this section, we first present three recommendation strategies toward groups of users, and then briefly analyze them.

3.1 Strategy 1—Group agent

In this recommendation strategy, the users register a common account for them, and input their original preferences to generate a common profile. When the user group intends to watch TV together, they log in with the common account. As a result, the group agent learns preferences for the group, and recommends programs to them.

3.2 Strategy 2—Merging recommendations

In Strategy 2, the system first uses a recommendation approach to generate a program recommendation list for each user according to their respective profiles. Then it merges these recommendation lists to generate a common program recommendation list for the group. The schematic of this recommendation strategy is shown in Fig. 1.

3.3 Strategy 3—Merging user profiles

In Strategy 3, the system first merges all user profiles to generate a common user profile. Then it uses a recommendation approach to generate a common program recommendation list for the group according to the common user profile. The schematic of this recommendation strategy is shown in Fig. 2.

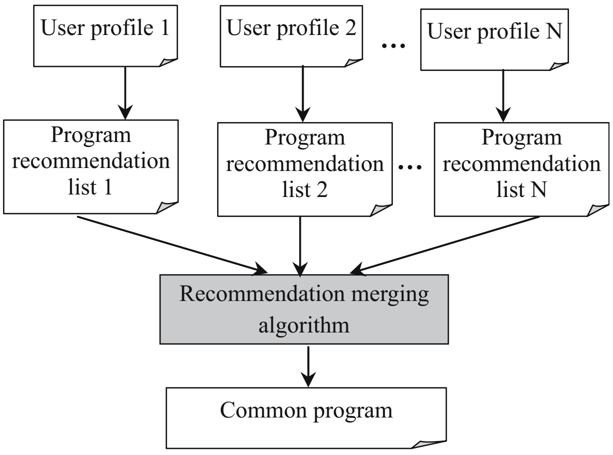
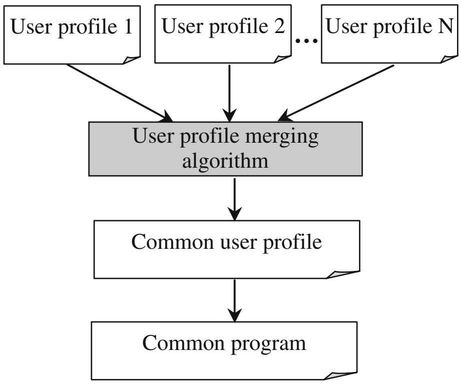

User profile 1
User profile 2
...
User profile N
Program recommendation list 1
Program recommendation list 2
...
Program recommendation list N
Recommendation merging algorithm
Common program

Fig. 1 Strategy 2: merging recommendations

Fig. 2 Strategy 3: merging user profiles



Compared to the last two strategies that use a merging algorithm, the strategy using the group agent is simple and direct, and sometimes more accurate. But, it is not adaptive and flexible. First, it requires the group of viewers to watch TV together for a long time so as to learn their common preferences. Second, the group agent cannot work when some members of the group rather than the whole group is watching TV. In contrast, whenever a subgroup of users wants to enjoy TV shows together, the last two strategies can generate the common program recommendation list for them. Several approaches, such as “hidden eye” technology, remote control functionality or smart card technology (Bozios et al. 2001), can be adopted to identify the members of the user group who are currently watching TV. For its convenience and intelligence, in this paper we choose Strategy 3 to achieve TV program recommendation for multiple viewers. The superiority of Strategies 3 to 2 is verified by experiment in Sect. 5. The individual user profiling and the approach to generate a recommendation list from a user profile have been presented in our early work (Yu and Zhou 2004). The user profile merging, which is the key technology to implement this strategy, is described in the following section.

4 User profile merging based on total distance minimization

The user profile merging algorithm merges individual profiles so as to form a common user profile that reflects most and consistent preferences of the group (Yu et al. 2004, 2005). In this section, we first describe the merging scheme, then give an example to demonstrate the merging process.

4.1 Universal vector representation

In each user profile, there are many features (e.g. genre, actor, and keyword about TV programs) as well as weights indicating the relative importance of features. We gather all the features from user profiles alphabetically as a lexicon, and use a thesaurus to reduce the feature list. The lexicon is represented as a vector.

$$\text{Lexicon} = (\text{feature}_1, \text{feature}_2, \ldots, \text{feature}_n). \tag{1}$$

With the lexicon vector, we can define each user profile universally as a vector V:

$$V = (f_1, f_2, \ldots, f_n), \tag{2}$$

where V is a vector with n (total number of terms in above lexicon) items, where f_i is the value assigned to corresponding feature$_i$ $(1 \leq i \leq n)$ in the lexicon vector. Supposing user's feature weight belongs to $[-1, 1]$, of which '-1' means maximum disliking (aversion), '1' means maximum liking (desire), the value f_i is assigned complying with the following rules:

- Rule (1): if feature$_i$ is included in the user profile and its weight is positive, then f_i=1;
- Rule (2): if feature$_i$ is included in the user profile and its weight is negative, then f_i=-1;
- Rule (3): if feature$_i$ is included in the user profile and its weight is zero, or *feature$_i$* is not included in the user profile, then f_i=0.

4.2 Definition of merging result

Dalal (1988) proposes a distance concept for belief revision. We adopt Dalal's distance concept to measure the inconsistency between two user preferences, and define the merging result.

Definition 1 Distance between two bits (x and y), d(x, y), $x \in \{1, 0, -1\}$, $y \in \{1, 0, -1\}$,
d(1, 0) = d(0, 1) = d(0, −1) = d(−1, 0) = 1,
d(1, 1) = d(0, 0) = d(−1, −1) = 0,
d(1, −1) = d(−1, 1)=2.

Definition 2 Distance between two user profiles (U_i and U_j), $D_u(U_i, U_j)$,
$D_u(U_i, U_j) = D(V_i, V_j) = \sum_{k=1}^{n} d(a_k, b_k)$, $a_k \in V_i$, $b_k \in V_j$, V_i and V_j are corresponding vectors extracted from U_i and U_j.

Definition 3 ψ_n is the set of all vectors composed of n elements, each of which belongs to $\{1, 0, -1\}$.

Definition 4 Suppose that $V_1, V_2, \ldots, V_N$, and $V \in \psi_n$, the merging result of $V_1, V_2, \ldots, V_N$ is V. It satisfies $\forall B \; \{B \in \psi_n\}$, $\sum_{i=1}^{N} D(V_i, V) \leq \sum_{i=1}^{N} D(V_i, B)$, which means the total distance is minimum.

4.3 Merging procedure

The merging procedure consists of two steps. The first step is feature selection, whose task is to determine whether a feature should be included in the target common user profile. As we know, each user profile may contain a large number of features. The total number of features that occur in all user profiles may be very large. So we should select part of them to represent common interest. The second step is weight assignment, which means how much weight should be assigned to a selected feature. The schematic structure of profile merging is shown as Fig. 3.

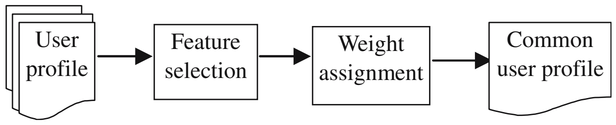


Fig. 3 Schematic structure of profile merging

4.3.1 Feature selection

Feature selection is based on total distance minimization. For each element of the target vector (merging result), choosing which one from $\{1, 0, -1\}$ as its value depends on whether it makes the total distance minimum.

Theorem 1 *Assume N user profile vectors, where each vector has only one element,* $V_i = \{a_i\}$, $(i = 1, 2, \ldots, N, a_i \in \{1, 0, -1\})$. *Define* c_1 *as the total number of times a* '1' *occurs in the vectors;* c_0 *as the total number of times a '0' occurs in the vectors;* c_{-1} *as the total number of times a* '-1' *occurs in the vectors. Then, under Definitions* 1–4, *the merging result of the N vectors is:*

$$\textit{Merging result (V)=}\begin{cases} 0, & \text{if } c_1 + c_{-1} < c_0 + 2c_1 \text{ and } c_1 + c_{-1} < c_0 + 2c_{-1}, \\ \pm 1 \text{ or } 0, & \text{if } c_1 + c_{-1} = c_0 + 2c_1 = c_0 + 2c_{-1}, \\ 0 \text{ or } -1, & \text{if } c_1 + c_{-1} = c_0 + 2c_1 \text{ and } c_1 + c_{-1} < c_0 + 2c_{-1}, \\ 0 \text{ or } 1, & \text{if } c_1 + c_{-1} < c_0 + 2c_1 \text{ and } c_1 + c_{-1} = c_0 + 2c_{-1}, \\ \pm 1\,, & \text{else.} \end{cases} \quad (3)$$

Proof According to Definition 1, the total distance of '1' to the N vectors is $c_0 + 2c_{-1}$; the total distance of '0' to the N vectors is $c_1 + c_{-1}$; and the total distance of '-1' to the N vectors is $c_0 + 2c_1$. If $c_1 + c_{-1} < c_0 + 2c_1$ and $c_1 + c_{-1} < c_0 + 2c_{-1}$, according to Definition 4, the merging result whose total distance should minimize, is 0. Similarly, if $c_1 + c_{-1} = c_0 + 2c_1 = c_0 + 2c_{-1}$, the merging result is ± 1 *or* 0; *if* $c_1 + c_{-1} = c_0 + 2c_1$ and $c_1 + c_{-1} < c_0 + 2c_{-1}$, the merging result is 0 or -1; if $c_1 + c_{-1} < c_0 + 2c_1$ and $c_1 + c_{-1} = c_0 + 2c_{-1}$, the merging result is 0 or 1; otherwise the merging result is ± 1. □

In the cases of $c_1 + c_{-1} = c_0 + 2c_1 = c_0 + 2c_{-1}$, $c_1 + c_{-1} = c_0 + 2c_1$ and $c_1 + c_{-1} < c_0 + 2c_{-1}$, $c_1 + c_{-1} < c_0 + 2c_1$, and $c_1 + c_{-1} = c_0 + 2c_{-1}$, the result is not certain. For these cases, we set the merging result as ± 1. So Theorem 1 can be revised as:

Theorem R1 *Assume N user profile vectors, where each vector has only one element,* $V_i = \{\mathrm{a}_i\}$, $(i = 1, 2, \ldots, N, a_i \in \{1, 0, -1\})$. *Define* c_1 *as the total number of times a* '1' *occurs in the vectors;* c_0 *as the total number of times a '0' occurs in the vectors;* c_{-1} *as the total number of times a* '-1' *occurs in the vectors. Then, under Definitions* 1–4, *the merging result of the N vectors is*:

$$\textit{Merging result}(V) = \begin{cases} 0, & \text{if } c_1 + c_{-1} < c_0 + 2c_1 \text{ and } c_1 + c_{-1} < c_0 + 2c_{-1}, \\ \pm 1, & \text{else} \end{cases} \quad (4)$$

(Proof omitted).

Theorem 2 *Assume N user profile vectors, where each vector has K elements,* $V_i = \{a_1, a_2, \ldots, a_j, \ldots, a_K\}$, $(i = 1, 2, \ldots, N, a_j \in \{1, 0, -1\})$. *Define V as the merging result of these N vectors. Then, each element of V is the merging result of the corresponding element in* V_1, V_2, …, V_N.

Proof The K-element vectors $V_1, V_2, \ldots, V_N$ are represented as follows:

$$\begin{aligned}
&V_1: \{a_{11}, a_{12}, \ldots, a_{1j}, \ldots, a_{1K}\},\\
&V_2: \{a_{21}, a_{22}, \ldots, a_{2j}, \ldots, a_{2K}\},\\
&\ldots\\
&V_N: \{a_{N1}, a_{N2}, \ldots, a_{Nj}, \ldots, a_{NK}\},\\
&V: \{a_1, a_2, \ldots, a_j, \ldots, a_K\},
\end{aligned}$$

a_{ij} denotes the jth element of the ith vector. V is the merging result.

Now, calculate the total distance (TD) of vector V to $V_1, V_2, \ldots, V_N$:

$$\begin{aligned}
\mathrm{TD} = &\sum D(V_i, V) = D(V_1, V) + D(V_2, V) + \cdots + D(V_i, V) + \cdots + D(V_N, V)\\
&(i = 1, 2, \ldots, N)\\
= &\sum d(a_{1j}, a_j) + \sum d(a_{2j}, a_j) + \cdots + \sum d(a_{Nj}, a_j)(j = 1, 2, \ldots, K)\\
= &\sum d(a_{i1}, a_1) + \sum d(a_{i2}, a_2) + \cdots + \sum d(a_{iK}, a_K)(i = 1, 2, \ldots, N).
\end{aligned}$$

Obviously, if we make $\sum d(a_{ij}, a_j)(i = 1, 2, \ldots, N, j = 1, 2, \ldots, K)$ minimum, TD can be minimum. $\sum d(a_{ij}, a_j)$ is the total distance of the jth element in V to corresponding element (that is, the jth element) in $V_1, V_2, \ldots, V_N$. So Theorem 2 is proved. □

We can select features to form the common user profile according to Theorems R1 and 2. If the value of the jth element of V is ± 1, this means we should select the corresponding feature (the jth feature in the lexicon vector). If the value of the jth element of V is 0, this means we should discard the corresponding feature. In Sect. 4.4, we will give an example to illustrate the selection process.

4.3.2 Weight assignment

The individual user profile acquisition and update by integrating two main techniques—explicit input/modification and learning from feedback—was presented in our early work (Yu and Zhou, 2004). Explicit input/modification means inputting features and weights during registration or modifying them after log in through a Graphical User Interface (GUI). Learning from feedback utilizes a relevance feedback mechanism to modify the preference features and their weights, respectively, by observing the user behavior. Through the learning approach, the weights are consistent in all the user profiles. However, the same weight value obtained through explicit input/modification in different user's profile often signals a different importance. The reason is that different users have different rating criteria and customs. For instance, 0.8 input by Tom may mean the feature is the favorite feature, but for Jack, it may simply imply moderate interest.

Through this profiling scheme, weights in different user profiles are not obtained with universal measurement. So, we should not assign a weight to a selected feature by simply summing all the weights in the individual user profiles, but instead should first normalize them.

(1) *Weight normalization*

For each user profile, we normalize the original weight of each feature according to the following general normalization formula:

$$w_i' = \frac{w_i - w_{\min}}{w_{\max} - w_{\min}} \times (U_{\max} - U_{\min}) + U_{\min}, \tag{5}$$

where w_i is feature$_i$'s original weight, w'_i is the normalized value of w_i, $w_{\max}$ and $w_{\min}$ are the maximum and minimum weight in the user profile, respectively $[U_{\max}, U_{\min}]$ is the value universe of feature weight. Here $U_{\max} = 1$ and $U_{\min} = -1$, so formula (5) can be simplified to the following one:

$$w'_i = \frac{2w_i - w_{\max} - w_{\min}}{w_{\max} - w_{\min}}. \tag{6}$$

Through formula (6), the normalized value is always on a standard universe, say $[-1, 1]$.

(2) *Weight calculation*

For each selected feature,

$$\text{Weight}_i = \frac{\sum_{j=1}^{m} w'_{ij}}{n}, \tag{7}$$

where Weight$_i$ is the weight assigned to feature$_i$ in the merged user profile, w'_{ij} is the normalized weight value of feature$_i$ in the jth user profile. In the group, which is composed of n members, there are totally m user profiles containing feature$_i$. Formula (7) also results in the weight values of the selected features falling into the range of $[-1, 1]$.

In weight calculation, we used n-mean value as the weight of a selected feature in order to take all members' preferences into consideration. If $m = n$, it is clearly a typical average method. If $m < n$, we consider the other members whose profiles do not include the feature through dividing the sum by n not by m.

4.4 Example

Suppose there are five user profiles, namely U_1, U_2, U_3, U_4, and U_5, as shown in Table 1.

Step 1 Represent user profiles as (feature, weight) vectors

$$U_1 = ((A, -0.3), (C, 0.8), (D, -0.7), (E, -0.6), (H, 0.9), (I, -0.2), (J, 0.6), (N, 0.4), (P, 0.3)),$$
$$U_2 = ((A, 0.7), (B, -0.9), (D, 0.3), (E, 0.5), (F, -0.8), (K, -0.2), (N, 0.5), (O, -0.3)),$$
$$U_3 = ((A, 0.3), (D, -0.4), (E, 0.2), (H, 0.5), (J, 0.7), (K, -0.1), (L, -0.4), (M, 0.8)),$$

Table 1 User profile examples

U_1		U_2		U_3		U_4		U_5	
Feature	Weight	Feature	Weight	Feature	Weight	Feature	Weight	Feature	Weight
A	−0.3	*A*	0.7	*A*	0.3	*A*	0.2	*C*	0.7
C	0.8	*B*	−0.9	*D*	−0.4	*D*	−0.3	*E*	−0.8
D	−0.7	*D*	0.3	*E*	0.2	*E*	0.6	*F*	0.4
E	−0.6	*E*	0.5	*H*	0.5	*H*	0.8	*G*	0.8
H	0.9	*F*	−0.8	*J*	0.7	*I*	−0.3	*J*	−0.2
I	−0.2	*K*	−0.2	*K*	−0.1	*J*	0.5	*K*	−0.2
J	0.6	*N*	0.5	*L*	−0.4	*K*	0.7	*L*	−0.4
N	0.4	*O*	−0.3	*M*	0.8	*M*	−0.3	*N*	0.7
P	0.3							*O*	0.6

Table 2 Universal vector

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
V_1	−1	0	1	−1	−1	0	0	1	−1	1	0	0	0	1	0	1
V_2	1	−1	0	1	1	−1	0	0	0	0	−1	0	0	1	−1	0
V_3	1	0	0	−1	1	0	0	1	0	1	−1	−1	1	0	0	0
V_4	1	0	0	−1	1	0	0	1	−1	1	1	0	−1	0	0	0
V_5	0	0	1	0	−1	1	1	0	0	−1	−1	−1	0	1	1	0

$$U_4 = ((A, 0.2), (D, -0.3), (E, 0.6), (H, 0.8), (I, -0.3), (J, 0.5), (K, 0.7), (M, -0.3)),$$
$$U_5 = ((C, 0.7), (E, -0.8), (F, 0.4), (G, 0.8), (J, -0.2), (K, -0.2), (L, -0.4), (N, 0.7), (O, 0.6)).$$

Step 2 Construct the lexicon

Lexicon= $\{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P\}$.

Step 3 Construct the universal vector for each user profile, shown in Table 2

Step 4 Select feature

According to Theorems R1 and 2, for feature A, $c_0 = 1$, $c_1 = 3$, $c_{-1} = 1$, since $c_1 + c_{-1} > c_0 + 2c_{-1}$, merging result of feature $A, V(A) = \pm 1$. For feature B, $c_0 = 4, c_1 = 0$, $c_{-1} = 1$, since $c_1 + c_{-1} < c_0 + 2c_1$ and $c_1 + c_{-1} < c_0 + 2c_{-1}$, merging result of feature $B, V(B) = 0$. Similarly, we can get $V(C) = 0$, $V(D) = \pm 1$, $V(E) = \pm 1$, $V(F) = 0$, $V(G) = 0$, $V(H) = \pm 1$, $V(I) = 0$, $V(J) = \pm 1$, $V(K) = \pm 1$, $V(L) = 0$, $V(M) = 0$, $V(N) = \pm 1$, $V(O) = 0$, $V(P) = 0$. That is, the merging result $V = (\pm 1, 0, 0, \pm 1, \pm 1, 0, 0, \pm 1, 0, \pm 1, \pm 1, 0, 0, \pm 1, 0, 0)$. So the target common user profile contains the following features: A, D, E, H, J, K, and N.

Step 5 Normalize the weight

For U_1, since feature A, D, E, H, J, and N are included in the merged user profile, we should normalize the weights of these features. In U_1, the maximum weight is 0.9 and the minimum weight is −0.7, so

$$w'(A) = \frac{2 \times (-0.3) - 0.9 - (-0.7)}{0.9 - (-0.7)} = -0.5,$$

$$w'(D) = \frac{2 \times (-0.7) - 0.9 - (-0.7)}{0.9 - (-0.7)} = -1,$$

$$w'(E) = \frac{2 \times (-0.6) - 0.9 - (-0.7)}{0.9 - (-0.7)} = -0.875,$$

$$w'(H) = \frac{2 \times 0.9 - 0.9 - (-0.7)}{0.9 - (-0.7)} = 1,$$

$$w'(J) = \frac{2 \times 0.6 - 0.9 - (-0.7)}{0.9 - (-0.7)} = 0.625,$$

$$w'(N) = \frac{2 \times 0.4 - 0.9 - (-0.7)}{0.9 - (-0.7)} = 0.375.$$

Similarly, we can perform normalization for U_2, U_3, U_4, and U_5.

Step 6 Calculate the target weight

For feature "A" in the merged user profile, it occurs in U_1, U_2, U_3, and U_4.

$$\text{Weight}(A) = \frac{-0.5 + 1 + 0.1667 - 0.0909}{5} = 0.1152.$$

Similarly, we can obtain $\text{Weight}(D) = -0.5$, $\text{Weight}(E) = -0.0977$, $\text{Weight}(H) = 0.5$, $\text{Weight}(J) = 0.3326$, $\text{Weight}(K) = -0.0114$, and $\text{Weight}(N) = 0.4$.

Step 7 Generate the target merged user profile

So the merged user profile, $U_{\text{merged}} = ((A, 0.1152), (D, -0.5), (E, -0.0977), (H, 0.5), (J, 0.3326), (K, -0.0114), (N, 0.4))$.

5 Implementation and evaluation

5.1 Implementation

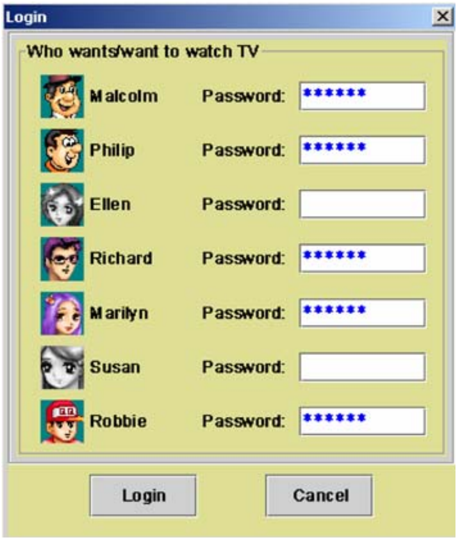
With the proposed user profile merging algorithm, we built a TV recommender system for multiple viewers called TV4M. The TV4M system leverages our earlier developed personalized TV system, namely TV3P (Yu and Zhou 2004) for individual user profiling and programs ranking. TV3P accomplishes the individual user profile acquisition and learning by employing an implicit and explicit profiling scheme. It adopts Vector Space Model (VSM) (Salton 1989) as its object information representation method, and the cosine of the angle between the program vector and the user profile vector as similarity measure. Programs are ranked according to similarities. The TV4M system is developed using the JAVA programming language (JDK1.4.1).

To identify who (i.e. which members) are watching television programs together, we provided a GUI for multiple viewers to log in to the system, which is shown in Fig. 4. In this scenario, five of seven members of the family intend to log in and watch TV together.

Figure 5 is the main interface of TV4M system after the group users logged in. It mainly consists of three parts. In the left column, there are some buttons to trigger system administration functions (e.g. login, logout, and exit the system) and profile related functions (e.g. modify and display the user profile). The middle column contains the program recommendation list and a description for a selected item. A Java Media Framework (JMF) based media player is integrated in the right column. In this scenario, the recommendation list contains four local programs ("I Guess, Guess, Guess", "Nike Shoes", "Sea Fish Sales", and "Philips Flat TV") ordered according to their scores (i.e. similarities). When the program "I Guess, Guess, Guess" is selected, its description can be browsed below the recommendation list. After the users click "Display", the media player is switched to show the program of "I Guess, Guess, Guess".

5.2 Evaluation

There were totally 25 users invited to test the system. The experiments involved a total of 200 distinct video contents (TV programs, movies, advertisements, and videos


Login

Who wants/want to watch TV

Malcolm	Password:	******
Philip	Password:	******
Ellen	Password:	
Richard	Password:	******
Marilyn	Password:	******
Susan	Password:	
Robbie	Password:	******

Fig. 4 Interface for multi-user login

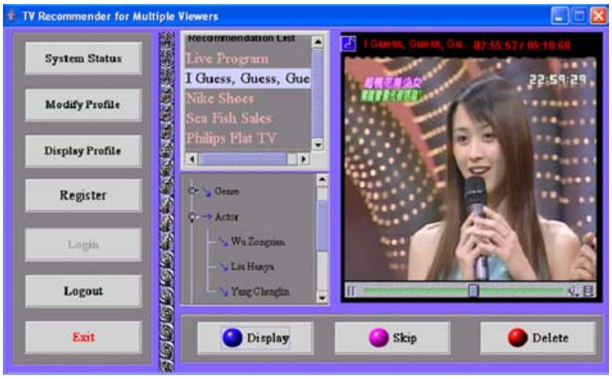


Fig. 5 Main interface

made by DIY) lasting from 38 s to several hours. A wide variety of genres were incorporated, including soccer match, romance movie, soap TV program, documentary, etc. We performed six experiments to test the efficacy and overhead of the proposed user profile merging algorithm.

5.2.1 *Experiment 1*

(1) Evaluation method

In this experiment, we aimed to evaluate the Precision and Recall (Rijsbergen 1979) of the profile merging in recommending the appropriate TV programs. It measures satisfaction of a group of users to the programs recommended to them based on the merged user profile. The evaluation method includes the following steps:

(1) Get a group of viewers;
(2) Merge their profiles to generate a common user profile;
(3) Show the title, genre, actor, and brief introduction of each program to the group, and ask the group to specify their choices, in which they are interested and in

which they are not interested. The goal is to get the number of interesting programs in the collection, e.g. the group is interested in 20 programs in the testing set. To deal with group dynamics when the group specified their choices, i.e. preventing the members from expressing their opinions influenced by some dominant or popular persons in the group, we allowed each member to separately indicate whether a program is interesting, and then decided the group's choice by voting;

(4) Use a recommendation approach to choose programs for the group according to the merged user profile;

(5) Count how many programs are recorded, and compare the recorded programs with the set of interesting programs indicated in Step (3). The goal is to get the number of programs totally recorded, and the number of interesting programs already recorded;

(6) Use Precision and Recall defined as follows to calculate their satisfaction percentage, and draw the Recall-Precision graph.

$$\text{Precision} = \frac{\text{number of interesting programs recorded}}{\text{total number of programs recorded}}, \tag{8}$$

$$\text{Recall} = \frac{\text{number of interesting programs recorded}}{\text{number of interesting programs in collection}}. \tag{9}$$

(2) Results

Seven users (5 males and 2 females) constituting a group participated in the experiment. The experimental results are shown in Fig. 6. There are five sessions in the experiment. We get a Recall-Precision pair in each session. Every Recall-Precision pair determines a node in the graph. According to F1, the harmonic mean of Recall and Precision (Rijsbergen 1979), the best performance happens during the session where the values of Recall and Precision are 0.63 and 0.57, respectively.

5.2.2 Experiment 2

(1) Evaluation method

In this experiment, we aimed to evaluate the efficacy of the profile merging algorithm by comparing scores explicitly assigned by the group and similarities measured by a recommendation approach according to the merged user profile. It goes as follows:

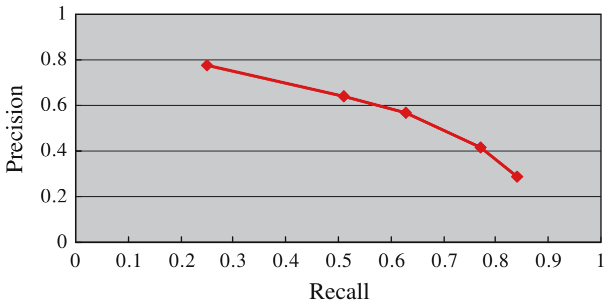


Fig. 6 Recall-Precision graph of experiment 1

(1) Get a group of viewers;
(2) Merge their profiles to generate a common user profile;
(3) Show them the program choices for several clips, and ask them to mark the programs ranging from 0 to 1 with step 0.1, e.g. the group assigns 0.7 to a specific program. To handle group dynamics, we asked each member to rate each program independently, and then averaged the scores to obtain the group rating for the program;
(4) Use a recommendation approach to evaluate the programs (similarity measure) according to the merged user profile;
(5) Compare the results (scores assigned and similarities measured) by using standard deviation (SD1) indicated as formula (10). The smaller SD1 is, the better performance the strategy has.

$$\mathrm{SD1} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathrm{score}_i - \mathrm{similarity}_i)^2}. \qquad (10)$$

(2) Results

Another five users (1 male and 4 females) constituting a group participated in the experiment. We asked the group to rank some programs and assign scores to them. To test the effect of program variety, i.e., liked or disliked, we selected ten representative programs that the group rated differently with values from 0 to 0.9 with 0.1 steps (i.e., 0, 0.1, 0.2, ..., 0.9). The results are shown in Table 3. The value of SD1 is 0.0784, which is very small.

5.2.3 Experiment 3

(1) Evaluation method

In this evaluation, we aimed to evaluate the efficacy of the profile merging by comparing feature weights merged and feature weights learned through the same learning approach for individual user profile acquisition and update. The evaluation method includes four steps:

(1) Get a group of viewers;
(2) Merge their profiles to generate a common user profile;

Table 3 Results of experiment 2

Program name	Score assigned	Similarity measured
Titanic	0.9	0.93
Pretty girls	0.8	0.75
Tokyo love story	0.7	0.82
Forrest gump	0.6	0.61
I, Robot	0.5	0.64
Star wars	0.4	0.37
Psycho	0.3	0.25
Philips flat TV	0.2	0.12
God’s hand of Maradona	0.1	0.00
Chanel perfume	0.0	0.09

(3) Show programs to the group, and use a learning approach to capture the group's preferences in terms of feature and its weight;
(4) Compare the results (feature weights merged and feature weights learned) by using standard deviation (SD2) indicated as formula (11). Also, the smaller SD2 is, the better performance the strategy has.

$$\mathrm{SD2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathrm{weightMerged}_i - \mathrm{weightLearned}_i.)^2} \tag{11}$$

(2) Results

Another five male students constituting a group participated in the experiment. We ranked the features in the merged common user profile according to their weights. Then we took the top ten features and compared their weights with corresponding weights in the learned profile. The results are shown in Table 4. The value of SD2 is 0.1998. We can observe that the weights of the top seven features in the merged common user profile are very close to those in the learned user profile, while the weights of the last three features vary largely especially the feature "Satellite" and "James Bond". The reason is that in the learning process, there are not many programs involving the feature "Satellite" and "James Bond". We believe that if the period of learning process is sufficient, and the number of testing programs is as large as possible, the performance could be fine.

5.2.4 Experiment 4

(1) Evaluation method

All the above three experiments aim at testing the efficacy of TV4M. In this experiment, we tested the overhead of the user profile merging algorithm in terms of time and storage cost. We compared our proposed approach (eliminating features using Dalal's distance (EFD)) versus simple approach (keeping all features (KAF)). We measured the time costs of EFD and KAF, storage costs of the profile via EFD and KAF, and time costs of recommendation using the profile via EFD and KAF. We performed the experiment on a PC with 2.66 GHz Pentium 4 CPU and 512 MB memory running Windows XP.

Table 4 Results of experiment 3

Feature name	Weight merged	Weight learned
Soccer	1.0000	0.9064
Diego Maradona	0.9329	0.9330
Stephen Chow	0.8893	0.9127
Zinedine Zidane	0.8832	0.8571
Comedy	0.8800	0.7069
Discovery	0.8796	0.8271
Drama	0.8665	0.6712
Satellite	0.8663	0.4200
Ge You	0.6534	0.5098
James Bond	0.6440	0.3299

Table 5 Results of Experiment 4

Overhead	EFD	KAF
Time cost of merging	578 ms	547 ms
Storage cost of the merged user profile	48 KB	97 KB
Time cost of recommendation	328 ms	391 ms

(2) Results

Table 5 shows the results of this experiment. The time cost of merging is very close. Our merging approach spends 578 ms, which is more than the simple approach by mere 31 ms. The merged user profiles via EFD and KAF take up storage space of 48 KB (102 features) and 97 KB (301 features), respectively. Obviously, the storage cost of the merged user profile via KAF is two times that of EFD. Then we used the two merged user profiles to recommend programs. The average recommendation time by using the profile via EFD is 328 ms, which is less than that of KAF (391 ms) by 63 ms. For a group, the merging executes only once, while the recommendation process performs many times. Therefore, the recommendation time influences the system performance more than the merging time. Furthermore, the storage cost of the user profile using KAF is much more than using EFD.

5.2.5 *Experiment 5*

(1) Evaluation method

In this experiment, we aimed to compare the efficacy of our proposed user profile merging algorithm (EFD) to KAF, random recommendation, and TV program merging in terms of the Precision and Recall. The measure of Precision and Recall for EFD and KAF is the same as experiment 1, but using different merged profiles. For random recommendation, the system does not choose TV programs according the profile merged with our proposed approach, but randomly selects TV programs to be recorded. The measure of Precision and Recall for the strategy of TV program merging is somewhat different from experiment 1. It includes the following steps:

(1) Get a group of viewers;
(2) Ask the group to specify the interesting programs in a collection. The group dynamics handling is the same as experiment 1;
(3) Use a recommendation approach to choose programs for each member based on the corresponding individual user profile;
(4) Combine all of the programs recorded for the individual users, and compare the recorded programs with the set of interesting programs indicated in Step (2) so as to get the number of programs totally recorded, and the number of interesting programs already recorded
(5) Calculate the Precision and Recall, and draw the Recall-Precision graph.

(2) Results

A group including five users (3 males and 2 females) participated in the experiment. The users got program recommendations through EFD, KAF, random recommendation, and TV program merging respectively. There were five sessions for each approach in the experiment. We put the results of these four approaches on the same Recall-Precision graph, as shown in Fig. 7, to determine which run is superior. Curves

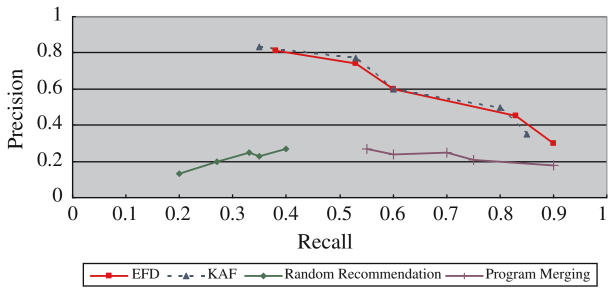


Fig. 7 Recall-Precision graph of experiment 5

close to the upper right-hand corner of the graph (where Recall and Precision are maximized) indicate better performance.

It can be observed that the performance of random recommendation is the worst. The curve of it is very close to the lower left-hand corner of the graph with both of the Recall and Precision very small. The strategy of TV program merging is better than random recommendation. The Recall is acceptable, but the Precision is small. The reason is that the total programs collected from all of the members includes most of the contents that the group were commonly interested, but also includes many programs merely interesting to individual users not to the whole group. The curves of EFD and KAF are very near and close to the upper right-hand corner of the graph. So the efficacies of EFD and KAF are almost the same and better than random recommendation and TV program merging. Since the overhead of EFD is lower than KAF as verified in experiment 4, we think the overall performance of EFD is better than KAF.

(3) Statistical testing

Since different recommendation methods are being compared in this experiment, it is necessary to evaluate the statistical significance of the results. The statistical significance tests ascertain whether the differences in evaluation results are really meaningful or just by chance (Hull 1993). We adopted the test called paired *t*-test (Hull 1993) for this purpose, which is widely used in IR community. The test is depicted briefly as follows:

Let X_i and Y_i be the scores (e.g., the average precision) of recommendation methods X and Y for a group profile i where $i = 1 \ldots n$, and define $D_i = X_i - Y_i$. The test assumes that the model is additive, i.e. $D_i = \mu + \varepsilon_i$, where μ is the mean value and the errors ε_i are normally distributed. The null hypothesis H_0 is $\mu = 0$ (X performs as well as Y), and the alternative hypothesis H_1 is $\mu > 0$ (X performs better than Y in terms of average precision).

Paired *t*-test

$$t = \frac{\bar{D}}{\sqrt{s^2/n}}, \quad \text{where } \bar{D} = \frac{1}{n}\sum_{i=1}^{n} D_i \quad \text{and } s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(D_i - \bar{D})^2 \tag{12}$$

follows the *t*-distribution with $n - 1$ degrees of freedom, where n is the number of samples, $\bar{D}$ and s^2 are the sample mean and the variance.

We tested each recommendation method against each of the other methods. The degree of freedom in the test was 4. Table 6 shows the results, in which the value of t and corresponding p-value are presented. Regarding significance levels, we used $\alpha = 0.05$

Table 6 Results of statistical testing

Pairs (X versus. Y)	t	p-value
KAF versus EFD	3.162	$0.01 < p$-value<0.05
EFD versus random recommendation	5.213	p-value < 0.005
EFD versus program merging	4.503	p-value < 0.01
KAF versus random recommendation	6.112	p-value < 0.005
KAF versus program merging	5.255	p-value < 0.005
Program merging versus random recommendation	1.510	p-value > 0.05

and $\alpha = 0.01$. From Table 6, we can see that the null hypothesis for testing KAF against EFD must be rejected at $\alpha = 0.05$ $(0.01 < p < 0.05)$. Therefore, we can conclude that KAF is likely to be better than EDF in terms of average precision, but only at the 0.05 level. For all of the other tests except program merging versus random recommendation, the null hypothesis is rejected at the 0.01 level. It means both EFD and KAF perform better than random recommendation and program merging. For program merging versus random recommendation, the null hypothesis cannot be rejected even at $\alpha = 0.05$, hence we must regard their average precisions as equivalent.

In general, the paired *t*-test for statistical significance testing proved that our results presented in Fig. 7 were really meaningful and not just due to chance.

5.2.6 Experiment 6

(1) Evaluation method

In this experiment, we aimed to evaluate the impact of homogeneity and heterogeneity of the user groups, in other words, how different the users are, would be on our proposed group recommendation approach. Two different groups were set up for this test. One group included five male undergraduates who were classmates and in the same dormitory. So the group could be homogeneous. The other group was set up to be heterogeneous, which comprized two male master students who preferred sports programs, one little boy who liked cartoon, two female undergraduates who were interested in romantic stories and one female librarian who liked drama. The system merged the two group user profiles and recommended programs to them, respectively. Then we asked the two groups whether they were satisfied with the programs selected for them.

(2) Results

We found that quite a few programs were chosen for the homogeneous group. All of the members were satisfied with the recommendation result. While there were few programs for the heterogeneous group, and the members were not satisfied with the result. It could be observed that for the homogeneous group, many features were included in the target profile, and the merged result could reflect the members' common interests. But since the members in the heterogeneous group differed a lot in gender, age, profession, or education, the merged result could not be convergent. So we get the conclusion that the homogeneity and heterogeneity really have some impact on the group recommendation system. It works better when the members in a group have a close relation or something in common.

6 Conclusion

In this paper, we present a program recommendation strategy for multiple television viewers using profile merging. The user profile merging is based on total distance minimization, which guarantees that the merged result is close to most users' preferences. Two theorems are proposed and proved for feature selection, which simplifies the process of finding the target vector, and makes the system feasible and efficient. The evaluation results proved that the recommendation strategy is effective for multiple viewers watching TV together. Furthermore, the recommendation strategy is simple and feasible. It can be implemented in PDRs, as well as low cost STBs.

For future work, we plan to deploy the group recommendation method in different application areas, such as web and music recommendation.

Acknowledgements We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was partially supported by the National Natural Science Foundation of China (No. 60573161), and the Doctorate Foundation of Northwestern Polytechnical University of China (No. CX200417). We thank Phillip McKerrow (University of Wollongong, Australia) for his helpful discussion about performance evaluation.

References

Ardissono, L., Goy, A., Petrone, G., Segnan, M., Torasso, P.: INTRIGUE: personalized recommendation of tourist attractions for desktop and handset devices. Appl. Artif. Int. **17**(8–9), 687–714 (2003)

Bollen, J.: Group user models for personalized hyperlink recommendations. In: Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, LNCS 1892, pp. 38–50 (2000)

Bozios, T., Lekakos, G., Skoularidou, V., Chorianopoulos, K.: Advanced techniques for personalized advertising in a digital TV environment: the iMEDIA system. In: Proceedings of the eBusiness and eWork Conference, pp. 107–113. Venice, Italy (2000)

Dalal, M.: 1988, Updates in Propositional Databases. Technical Report DCS-TR-222, Department of Computer Science, Rutgers University

Das, D., ter Horst, H.: Recommender Systems for TV. In: Recommender Systems, Papers from the 1998 Workshop, Technical Report WS-98–08, Madison, WI, Menlo, Park, pp. 35–36. AAAI Press, CA (1998)

Ehrmantraut, M., Härder, T., Wittig, H., Steinmetz, R.: The personal electronic program guide – towards the pre-selection of individual TV programs. In: Proceedings of the 5th International Conference on Information and Knowledge Management (CIKM'96), pp. 243–250. Rockville, MD, USA (1996)

Gutta, S., Kurapati, K., Lee, K. P., Martino, J., Milanski, J., Schaffer, J. D., Zimmerman, J.: TV content recommender system. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 1121–1122. Austin, TX, USA, (2000)

Hull, D.: Using statistical testing in the evaluation of retrieval experiments. In: Proceedings of the 16th International ACM SIGIR Conference, pp. 329–338. New York, USA (1993)

Jameson, A.: More than the sum of its members: challenges for group recommender systems. In: Proceedings of the International Working Conference on Advanced Visual Interfaces, pp. 48–54. Gallipoli, Italy (2004)

Lieberman, H., Dyke, N.W.V., Vivacqua A.S.: Let's Browse: a collaborative web browsing agent. In: Proceedings of the International Conference on Intelligent User Interfaces (IUI99), pp. 65–68. ACM Press, New York (1999)

Masthoff, J.: Group modeling: selecting a sequence of television items to suit a group of viewers. User Model. User-Adapt. Interact. J. Personalization Res. **14**(1), 37–85 (2004)

McCarthy, J.F., Anagnost, T.D.: MusicFX: an arbiter of group preferences for computer supported collaborative workouts. In: Proceedings of the 1998 Conference on Computer-Supported Cooperative Work, pp. 363–372. Seattle, WA, USA (1998)

O'Connor, M., Cosley, D., Konstan, J., Riedl, J.: PolyLens: a recommender system for groups of users. In: Proceedings of the European Conference on Computer-Supported Cooperative Work, pp. 199–218. Bonn, Germany (2001)

O'Sullivan, D., Smyth, B., Wilson, D. C., McDonald K., Smeaton, A.: Improving the quality of the personalized electronic program guide. User Model. and User-Adapt. Interact. J. Personalization Res. **14**(1), 5–36 (2004)

Rijsbergen, C.J.: Information Retrieval. Butterworths, 2nd edn. London, UK (1979)

Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley Longman Publishing, Boston, MA, USA (1989)

Smyth, B. P., Cotter, P.: Case-studies on the evolution of the personalized electronic program guide. In: Ardissono, L., Kobsa, A., Maybury, M.T. (eds.) Personalized Digital Television: Targeting Programs to Individual Viewers. Kluwer Academic Publishers, Dordrecht, Netherlands (2004)

Yu, Z.W., Zhou X.S.: TV3P: an adaptive assistant for personalized TV. IEEE Transactions Consum. Electron. **50**(1), 393–399 (2004)

Yu, Z.W., Zhou, X.S., Hao, Y.B., Gu, J.H.: User profile merging based on total distance minimization. In: Proceedings of the 2nd International Conference On Smart homes and health Telematics (ICOST 2004), pp. 25–32. IOS Press, Singapore (2004)

Yu, Z.W., Zhou, X.S., Zhang D.Q.: An adaptive in-vehicle multimedia recommender for group users. In: Proceedings of IEEE 61st Vehicular Technology Conference (VTC 2005-Spring), pp. 2800–2804. Stockholm, Sweden (2005)

Authors' Vitae

Zhiwen Yu
Zhiwen Yu is a Post-doctoral Researcher at the Information Technology Center, Nagoya University, Japan. He received his Ph.D. in Computer Science from the Northwestern Polytechnical University, P. R. China in 2005. This work was done when he was a Ph.D. candidate at the School of Computer Science, Northwestern Polytechnical University. His research interests include multimedia intelligent services, adaptive systems, personalization, and pervasive computing. From September 2004 to May 2005, he was a visiting researcher at the Institute for Infocomm Research, Singapore.

Xingshe Zhou
Xingshe Zhou is Professor and Dean of the School of Computer Science, Northwestern Polytechnical University, P. R. China. He received his M.S. degree in Computer Science from Northwestern Polytechnical University. His research interests lie in adaptive systems, distributed computing, pervasive computing, and sensor networks.

Yanbin Hao
Yanbin Hao is a Lecturer at the Management School, Northwestern Polytechnical University, P.R.China. He received his B.S. degree in Radio Electronics from Lan Zhou University in 1998, and his M. S. degree in Computer Science from Northwestern Polytechnical University in 2004. His research interests include information merging, distributed computing, and embedded computing.

Jianhua Gu
Jianhua Gu is a Professor at the School of Computer Science, Northwestern Polytechnical University, P.R.China. He received his Ph.D. in Computer Science from Northwestern Polytechnical University. His research interests include distributed computing, software engineering, computer operating systems, and embedded computing.