Regular Paper

An Advanced Movie Recommender System Based on High-Quality Neighbors

Saranya Maneeroj,† Yuka Kato† and Katsuya Hakozaki†

This paper proposes an advanced movie recommender system. The system is primarily based on the content-based collaborative filtering or hybrid filtering technique. The distinctive point of this system lies in an improved neighborhood formation method in order to get high-quality neighbors. Current hybrid systems only use user's opinions as user's rating values in selecting neighbors. That causes loss of user preference features, and tends to provide poor quality neighbors. The proposed system uses the user's opinions on various features of user preferences in selecting neighbors. That results in high-quality neighbors and high-quality recommendations will be obtained accordingly. An experimental movie recommender system, called Advanced Yawara system has been developed to prove the effectiveness of the method. The evaluation results show that the Advanced Yawara system provides higher quality of recommendations than current hybrid systems.

1. Introduction

Recommender systems [1] are widely used in the Internet, especially, in E-commerce sites [2] to help users to get interesting information easily. Typical examples are Amazon.com [3], a recommender on books, P-Tango [4], an online newspaper recommender, and Ringo [5], a music recommender. Many recommenders are based on the Collaborative Filtering (CF) technique, which uses the collaborative users' opinions in recommending items to a user. The CF based systems do not use any information regarding the actual content. Current systems or hybrid systems try to integrate Information Filtering (IF) or Content-Based Filtering (CBF) into CF systems in order to improve recommendation quality.

The recommendation quality of these hybrid systems has improved significantly compared with the former CF based systems. Particularly, if the system succeeds in selecting suitable people as collaborative users (or neighbors) having similar tastes with the target user to form recommendations. However, there are many occasions where current recommenders do not provide satisfactory recommendations because of forming improper or poor neighbors. There are two cases of forming poor neighbors. One is opinion representation and the other is improper neighborhood formation.

The current systems use the rating values on the items for evaluating users' preference opinions. The rating value represents the overall preference of the user. A user might express his/her opinion based on some specific features of the item. For more accurate recommendations, the users' interests in more detailed features should be taken into account. We call this problem the "rating value alone" problem.

As for the neighborhood formation, current systems try to find neighbors who have similar tastes with a target user. A subset of appropriate users is chosen based on their similarity to the user. Then, their opinions are used to generate recommendations for the target user.

Many neighborhood formation methods have been proposed. One well-known method is the similarity between co-rated items method. Rating values on the same rated items are compared to form neighborhood. Similarity between content-based user profiles is another method to find neighbors. Each pair of user profiles, which contain correlations between the content of items and the user preference are compared to form the neighborhood. However, the co-rated items are difficult to discover in the co-rated items method. Likewise, in the content-based method, the content-based user profile does not cover the features of the user interest.

The major purpose of this paper is to propose an advanced content/collaborative hybrid system based on a new neighborhood formation method to cope with the forming poor neighbor problem in recommending movies. Instead of using rating values alone, the method uses "user's opinions on features". In forming high-quality neighbors, two filtering processes are

† Graduate School of Information Systems, University of Electro-Communications

used to apply "user's opinions on features" and to eliminate the neighborhood formation problems stated above.

In the next sections, the details of poor neighbor problem are discussed. We then describe the proposed method for getting higher quality neighbors in Section 3. In Section 4, we describe the implementation of the prototype system called Advanced Yawara, and then the results from its evaluation is presented in Section 5. In Section 6, we discuss about the derived evaluation results. Finally, we give some conclusions and future works in the last section.

2. Poor Neighbor Problem

Various recommender systems have been developed and utilized. The current recommenders combine two or more recommendation techniques to obtain better recommendations. Recent recommender systems commonly use content/collaborative hybrid filtering [6], combination of CBF and CF.

In content/collaborative hybrid systems, the quality of neighbors mostly affects the recommendation quality. This is because the opinions of all the neighbors are formed to generate recommendations. The major problem is how to choose "good" neighbors.

The causes of poor neighbors can be classified into two categories. One is the opinion representation problem and the other is the improper neighborhood formation problem.

2.1 Opinion Representation

The opinion representation problem comes from disregarding users' opinions in detail.

Some hybrid systems treat the rating value as representing a user's opinion, which is not able to represent all the features of user preferences. The current systems do not have the capability of recognizing the two distinct interests represented in the same rating value. Therefore, comparing the interests between people by rating value alone may not be accurate. For example, if UserA and UserB rate the same score 1 for the movie Titanic as shown in **Table 1**, but UserA likes its actor and UserB likes its genre. However, current systems conclude that they have the same tastes. Therefore, the neighbors from their systems tend to be of low quality. We call this problem the "rating value alone" problem.

Another case is missing weight of features that affects user preference. For example if two users (UserA and UserB) like the same movie features; same actor (Tom Hanks), same actress (Meg Ryan) and same genre (Fantasy), current systems would usually conclude that both of them are good neighbors for each other. However, this conclusion may not be true. If UserA usually selects movies based on the genre, UserA may select "Lord of the Rings" (Fantasy) but not select "You've Got Mail" (Tom Hanks and Meg Ryan) though UserA likes Tom Hanks and Meg Ryan. The weight of genre feature has higher priority than actor and actress features in UserA's opinion. On the other hand, if the weights of actor and actress features are higher than genre feature in UserB's opinion, UserB will select "You've Got Mail" and not "Lord of the Rings", though UserB also likes Fantasy movies. It can be concluded that, although each couple of users likes the same movie features, they may select different movies. We call this problem the "missing weight feature" problem.

Table 1 Example of user's ratings.

Movie	User	UserA	UserB	UserC
Finding Nemo		−1		
Titanic		1	1	
Pretty Woman			0	1

2.2 Improper Neighborhood Formation

The neighborhood formation is carried out by choosing a subset of appropriate users who have similar tastes.

Numerous combination methods in content/collaborative hybrid systems have been proposed in order to increase recommendation quality. Burke [6], classified combination methods into seven categories: weighted, switching, mixed, feature combination, cascade, feature augmentation and meta-level. However, none of those methods focus on the combination of neighbors resulted from CBF and CF, even if the neighbor set significantly affects the quality of recommendations. Current systems employ either pure CF method or pure CBF method to form neighbors.

An example of a content/collaborative hybrid system that finds neighbors based on pure CF technique is MovieLens system [7]. It finds similarity between users based on the rating values on the co-rated items (the same rated items). Users, who have the same or similar history data associated with rating values on co-rated items, are recognized as neighbors. The system copes with the cold start problem by introduc-

ing "filterbot" (an IF agent), but other problems, such as sparsity rating, synonymy and scalability problems still remain. We call this neighborhood formation method the "co-rated items method".

The *cold start problem* occurs when a first user comes to a system. He rates items without receiving any recommendation. *Sparsity rating* occurs when each user has rated a small part of whole items. It causes a set of co-rated items is small. Accordingly, quality of neighbors tend to be poor. Different item names may be used for the same objects, but the co-rated items method cannot find this latent association and treats these items differently. This is called *synonymy* problem. The co-rated items method requires computation that grows with both the numbers of users and items. It may cause *scalability* problem.

On the other hand, a content/collaborative hybrid system that finds neighbors based on pure CBF technique, such as the e-Yawara system [8], which uses similarity between user vector profiles (content-based profiles) to find neighbors in order to cope with the sparsity problem. In that type of systems, it is difficult to extract and implement all the features to form a user profile. Usually, a few features are used to form a user profile. Thus, the quality of neighbors tend to be poor. We call this neighborhood formation method the "content-based neighborhood formation method".

3. Proposed Method

We focus to overcome the *poor neighbor* problem. In any case, in selecting neighbors, factors that well represent user's opinion should be taken into consideration. We therefore propose to represent the user's opinion as "user's opinion on features" to provide more features of preference. Unfortunately, if the "user's opinion on features" is directly applied to the current neighborhood formation methods, limitations still remain.

We have studied which movie features mostly influence individual users in selecting a movie. From the results of 40 questionnaires collected from 40 people, the most popular movie features are Style or Genre of Film, Actor, Actress, Director, Story Line, Awards won, Freshness or Year of Film, Popularity of Film (Top ranking), Visual or Animation Effects, and Film Studio.

In case of content-based neighborhood formation, it is difficult to create a user profile based on those movie features. For instance, a feature such as "actor" may have more than 100 actors for 100 different movies. Therefore, it is not realistic to directly apply those movie features to the content-based method.

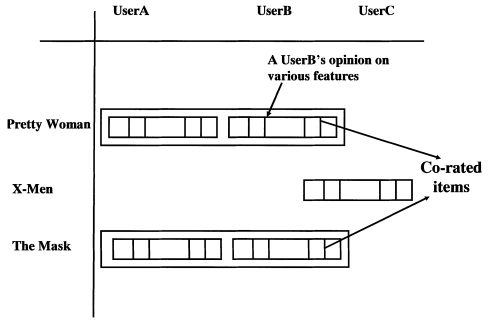

Fig. 1 The "user's opinion on various features" in the CF table.

On the other hand, if we adopt those features in the co-rated items method by adjusting the "user's opinion on features" in the CF table as shown in **Fig. 1**, it would not reduce the number of co-rated items. That is, the opinion represented by many features does not make sparsity rating and synonymy problems more severe. However, the original sparsity rating and synonymy problems still remain.

In order to eliminate those limitations, we propose a cascade model of combining neighbors obtained from both CBF and CF. That is, we first employ content-based neighborhood formation with small number of features to find a first set of neighbors. We then refine the first neighbor set through the use of the co-rated items method with the "user's opinions on various features". The content-based neighborhood formation method refers to "the first filtering" and the co-rated items method with the "user's opinions on various features" refers to "the second filtering".

3.1 The Basic Elements of the Method

There are four basic vectors in the proposed method.

3.1.1 Movie Feature Vector (mfv)

Target items, movie data in this case, are stored in a database with characteristic data for each item. The movie characteristic data are represented by a form of a movie feature vector which contains 20 elements about movie genre feature extracted from the Internet Movie Database (IMDB), such as comedy, drama, etc. The mfv is constructed when a new item is introduced into the system. Its characteristic

is $mfv(i) = (w_{i1}, w_{i2}, \ldots, w_{im})$; where w_{ij} is the weight that a movie(i) has towards a keyword(j); and m is the number of keywords. The weight ranges from -1 to 1. The keyword list is 20 movie genres.

3.1.2 User Feature Vector (UFV)

It contains the correlations between the genre of rated movies and the user preference. In other words, it contains the 20 elements about movie genre relating with mfv of all rated movies. Its characteristic is $UFV = (w_1, w_2, \ldots, w_m)$; where w_i is the weight that a user has towards a keyword (i); and m is the number of keywords. The weight ranges from -1 to 1. The keyword list is same 20 movie genres as in the mfv.

As mention about entering user's opinion in Section 4.1, there are three levels of user's opinion towards each movie: "Want to see" (score = 1), "Neutral" (score = 0) and "Don't want to see" (score = -1). When a user gives opinion for the first movie, his UFV will be automatically created and stored in the database. The UFV towards the first movie has the characteristic corresponding to the mfv of such movie multiplied with his opinion score towards that movie. For example, if UserA gives opinion that "don't want to see" the first movie, which is "Finding Nemo", the mfv of "Finding Nemo" will be multiplied with score (-1) to be the first version of UserA's UFV.

After the user gives opinions for more movies, his UFV will be updated to get closer to the mfv of the movies that the user needs, according to the successive change of user preference and user action on the system. For the updating process of UFV, we have adapted it from our e-Yawara system [8]. We assume that when a user clicks and views on some movie items frequently and he is very interested in those movies, his feature can be considered to become closer to the feature of those movies. Accordingly, the change of preference data and action history data of the user are mapped to the movie features, then these mapped properties will be used to update the UFV.

$$UFV_{updated} = (aH_{change} + bI_{change})MFV + UFV_{previous} \quad (1)$$

where UFV_{update} is the updated user feature vector. H_{change} is a vector which represents the history data. In proposed method, the history data refers to only the number of clicks on the movie. I_{change} is a vector which represents the preference value towards the movie: "Want to see" (score = 1), "Don't want to see" (score = -1), and "Neutral" (score = 0). MFV is matrix represented all movie feature vectors; and a and b are coefficients.

3.1.3 User Preference Vector (UPV)

It represents a "user's opinion on features" or shows how much each user feels towards what features affecting in selecting each movie. The UPV will be automatically created for each movie every time each user gives opinion for that movie. That is, if UserA gives opinions for ten movies, then ten $UPVs$ of UserA for such ten movies will be automatically created and stored in the database. Its characteristic is $UPV(j) = (f_{1j}, f_{2j}, \ldots, f_{nj})$; where f_{ij} is a user preference (opinion) value for the movie feature(i) of a movie(j). If a user wants to see the movie(j) because of the movie feature(i), f_{ij} will be set to be 1. If a user doesn't want to see the movie(j) because of the movie feature(i), f_{ij} will be set to be -1. Otherwise, f_{ij} will be set at 0. $n = 10$ refers to the number of necessary movie features collected from the questionnaire.

3.1.4 Feature Dependency Vector (FDV)

It represents user's dependency on features. From each user behavior, if he usually selects a movie based on some specific features, the values corresponding to the elements in his FDV are set higher. The FDV of each user will be automatically created, when he gives opinion for the first movie. It then will be automatically updated, when he gives opinions for more movies. Its characteristic is $FDV = (g_1, g_2, \ldots, g_n)$. If a user wants to see a movie because of the movie feature(i), g_i will be added by 1. Otherwise, g_i is not changed. $n = 10$ which is the same number of features as in the UPV.

3.2 Neighborhood Formation Process

Our neighborhood formation method has two filtering processes (see **Fig. 2**). The first filtering employs content-based neighborhood formation method in selecting the first rough neighbor set. The second filtering then refines the neighbors derived from the first filtering through the use of co-rated items method with the "user's opinions on features", in order to find the final neighbors. The following details the basic idea of these two filtering processes.

In the first filtering, the similar content-based user profiles could not be concluded that these similar users have ever rated the same

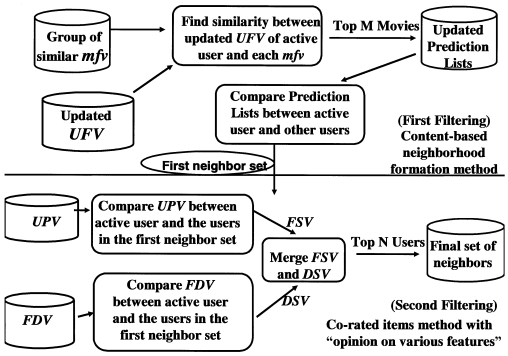

Fig. 2 Our neighborhood formation process.

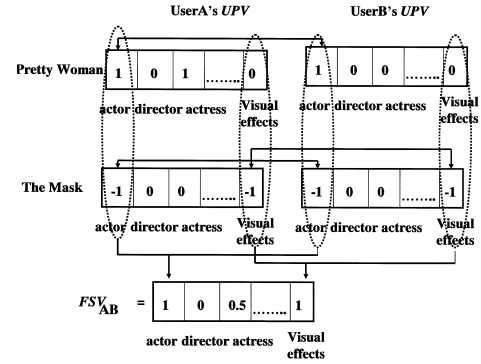

Fig. 3 Users'opinions on features (UPV) of two users on the co-rated items and the example of FSV.

movies. Accordingly, the neighbor set, which is the results from direct comparison between the content-based user profiles, may not be able to increase the number of co-rated items in the second filtering. It does not reduce the sparsity rating and synonymy problems. A new solution to increase the number of co-rated items is introduced.

The new solution is to create a "Prediction List" by calculating similarity between user feature (UFV) and each movie feature (mfv) of all movies in the database and then the movies that highly relate to the user feature will be selected to form the list. This list will be presented to a user in order for him to entering opinions for interesting movies.

The users who have similar profiles will have similar Prediction Lists. The rating for the same movies by these users will increase. This enhances the co-rated items in the second filtering much more likely to be discovered. Accordingly, it reduces sparsity rating problem in the second filtering.

For the synonymy problem, usually it is unlikely to find similarity between two users in the following case. One rates for "The X-Men" and the other rates for "The Matrix" in the second filtering. However, these two movies have the very similar genres. There is high possibility that these 2 movies appear in their Prediction Lists. Therefore, it is possible that these two users may rate for both movies from their lists. As a result, the co-rated items could be found in the second filtering and then the similarity can be found.

In the worst case, if no co-rated item is found in the second filtering, no recommendation is done. However, the Prediction List is a set of basic recommendations. These basic recommendations can be an elegant way to overcome the cold start problem.

Contrarily, if two users have some numbers of the same movies in their Prediction Lists, it implies that they have similar profiles. The way to find neighbors in the first filtering is to create a Prediction List to be presented to the user. If any two users have some certain number of the same movies in their lists, these two users are considered to be neighbors to each other.

In the second filtering, the similarity between users is found how much each pair of users likes or dislikes the same movie features. In other words, it is to find whether they are likely to have the same history data associated with the specific features. In order to calculate this similarity, first of all, we adjust the "opinions on various features" ($UPVs$) to the CF table as presented in Fig. 1. After that, their opinions on the co-rated items are compared.

For example, Fig. 1 shows that the co-rated movies between UserA and UserB are the movies "Pretty Woman" and "The Mask". Suppose that UserA wants to see "Pretty Woman" because of actor and actress, and doesn't want to see "The Mask" because of actor and visual effects. UserB wants to see "Pretty Woman" because of actor, and doesn't want to see "The Mask" because of actor and visual effects. Since there are three levels of opinion: "Want to see" (score = 1), "Don't want to see" (score = -1), and "Neutral" (score = 0), their opinions on features can be presented in the **Fig. 3**, where the non-specified features are all set at 0.

In Fig. 3, the movie "Pretty Woman", the scores for actor feature of both users are the same, which is equal to 1. It implies that they like the same specific actor "Richard Gere". For the movie "The Mask", the scores for actor and

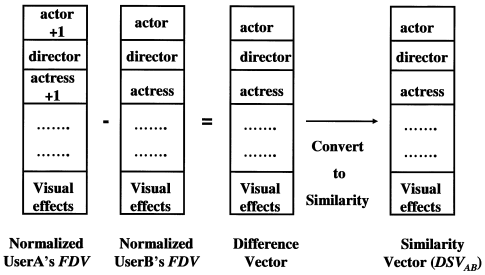

Fig. 4 Example of FDV and the way to get DSV.

visual effects features are the same, which is equal to -1. It implies that they dislike the same specific actor "Jim Carry" and dislike the same specific style of visual effects.

Each pair of users are likely to have the similar history data associated with specific features, if they have the similar $UPVs$ on the list of co-rated movies. This similarity is represented by a form of a feature similarity vector (FSV). From this example, when comparing each feature between their $UPVs$ on these two co-rated movies for all features, the FSV will be formed as presented at the bottom part of Fig. 3. These two users are likely to have the same history data associated with the specific actors, because they always like and dislike the same specific actors. Accordingly, the value of actor element in FSV_{AB} will be high.

As mentioned about the missing weight feature problem, the weight of each feature should be clarified in finding neighbors. That is, a feature dependency vector (FDV) mentioned in the Section 3.1 should be considered to express behavior of each user what features are usually used in selecting movies. For example, if UserA wants to see a movie because of its actor and actress, the actor and actress elements in UserA's FDV will be increased by 1 (see UserA's FDV in **Fig. 4**). Each pair of users are likely to have the similar behavior in selecting movies, if they have the similar $FDVs$. This similarity is represented by a form of a feature dependency similarity vector (DSV). Figure 4 shows the way to get DSV between UserA and UserB, but the details is described in the following section.

The two users will be the neighbors to each other, if they are likely to have the similar history data associated with the specific features and similar behavior in selecting a movie. That is the results of similarity of their $UPVs$ on the list of co-rated movies, which is FSV, and similarity of their $FDVs$, which is DSV, should be merged. After that, the Top N users, who have highly merged score towards the active user (the user who is interacting with the system), will be selected to be the final neighbors.

The proposed method for refining neighbors are likely to reduce sparsity rating, synonymy and cold start problems as mentioned above. The scalability problem is likely to be reduced also, because the users compared with the active user in the second filtering will be the ones who are in the neighbor set obtained from the first filtering, not from the entire user database as made in the current systems.

4. Advanced Yawara System

A prototype recommender system called Advanced Yawara system is implemented to evaluate the proposed method. It is an online movie recommender system accessible through the Web. In Advanced Yawara, Tomcat4 on a Linux PC acts as the WWW server. It was implemented by JSP in order to access mysql database in server from user clients. The process of the system is classified into 4 parts: entering user's opinions, forming the first neighbor set, finding the final neighbors, and generating recommendations

4.1 Entering User's Opinions

Each user starts with entering a user name and password. After that, the search page for entering any desired queries will emerge for each user to search for the required movies. The user is allowed to search for specified movies through queries about Title, Year of film, Rank of film, Genre, Actor, Actress, Director and Awards won. The search results page then displays the movie results using handbills.

After a user clicks on the movie item in the search results page, the movie detail page for that movie will emerge (see **Fig. 5**). The top part of the movie detail page contains movie trailer which a user can watch as a movie preview. It also contains movie information or details of those 10 movie features referred in the questionnaire.

From the movie detail page, the user can watch the movie trailer and all movie information. He/she then gives an opinion about that movie. In the system, there are three opinion levels. They are "Want to see", "Neutral" and "Don't want to see". When a user specifies "Want to see" or "Don't want to see". The system will ask more about what reasons (which movie features presented at the top part of page) make he/she "Want" or "Don't want"

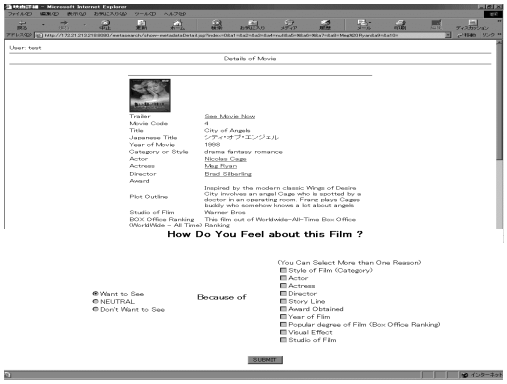

Fig. 5 Movie detail page.

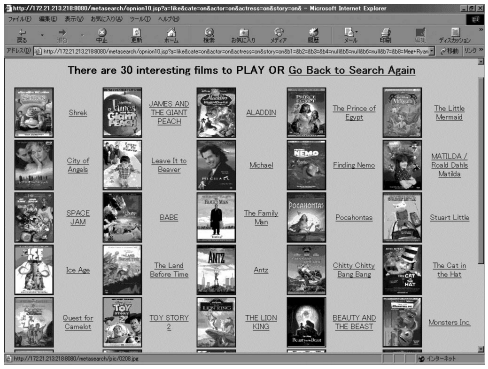

Fig. 6 Prediction List.

to see that movie. The user can select one or more movie features by checking the boxes at the bottom part of the movie detail page (Fig. 5) to make his/her answer more concrete.

4.2 Forming the First Neighbor Set

The first filtering employs content-based neighborhood formation method in selecting the first rough neighbor set. According to the basic idea mentioned in Section 3.2 and the process in Fig. 2, the first filtering is carried out by three steps.

(1) For all users, the distance between each user and each movie is calculated using the UFV and the mfv.
(2) For each user, the Top M movies that are close to his/her interest are selected in order to create Prediction List.
(3) The first set of neighbors is produced.

Step(1): The prediction score towards each movie is produced by calculating the distance between UFV and mfv. A short distance movie will have a high prediction score or it is close to his/her preference. The distance between vector $A\ (w_{a1}, w_{a2}, \ldots, w_{am})$ and vector $B\ (w_{b1}, w_{b2}, \ldots, w_{bm})$ is defined as follows:

$$d = \sum_{i=1}^{m} |w_{ai} - w_{bi}| \qquad (2)$$

where, m is the number of weight elements; $0 \leq d \leq 2m$; and the size of each weight w is $-1 \leq w \leq 1$.

The metric about similarity between two vectors is defined as the difference between the value of full distance $(2m)$ and distance (d), $(2m - d)$. Then normalize the similarity.

$$Similarity = 1 - \frac{d}{2m} \qquad (3)$$

where, $0 \leq Similarity \leq 1$.

In this step, the *scalability problem* can be slightly reduced by grouping similar movies in the offline process. Instead of calculating the distance towards all movies, it calculates the distance towards only the groups of movies.

Step(2): The Top M movies that have high prediction scores are selected to create a Prediction List; a list of interesting movies. It will then be displayed to the user (see **Fig. 6**).

As mentioned in Section 3.1, the UFV will be updated to get closer to the mfv of the movie that the user needs, according to the successive change of user preference. It implies that the Prediction List will be dynamically updated to be better every time, according to updated UFV. This dynamic Prediction List relates to user preference, so it can be the interesting movie list presented to a user for entering opinions [9),10)].

At this point, a user has two choices in selecting movies to enter opinions: selecting movies in a Prediction List or going back to search the specified movies in the search page.

Step(3): As mentioned about basic idea of the first filtering in Section 3.2, the users who have similar profiles (or similar $UFVs$) will have similar Prediction Lists. Contrarily, if two users have some numbers of the same movies in their Prediction Lists, it implies that they have similar profiles. That is neighbors in the first neighbor set will be the ones whose updated Prediction Lists have some certain numbers of the same movies with the updated Prediction List of the active user. In this time of implementing Advanced Yawara, the users, whose updated Prediction Lists have at least 10 percentage of the same movies with the active user's, are selected to be neighbors in the first neighbor set.

4.3 Finding the Final Neighbors

In order to find the final neighbors, the second filtering refines the neighbors derived from the first filtering through the use of co-rated items method with the "user's opinions on features" (or $UPVs$).

According to the basic idea mentioned in Section 3.2 and the process in Fig. 2, the second filtering is carried out by the following four steps.

(1) For each pair of users, the similarity between their $UPVs$ on the list of co-rated items is calculated.
(2) For each pair of users, the similarity between their $FDVs$ is calculated.
(3) The similarity values from (1) and (2) are merged to produce the final similarity value between two users.
(4) The final neighbors are produced.

Step(1): When all $UPVs$ are put into the CF table, the $UPVs$ of two users on the list of co-rated movies, which are called $co-UPVs$ will emerge. If considering $co-UPVs$ of two users, each user profile can be represented as a matrix called $co-UPVmatrix$ as shown in **Fig. 7**; where each line of these two $co-UPVmatrices$ refers to the $UPVs$ of two users on each co-rated movie. The $co-UPVmatrix$ for each user is expressed by Eq.(4).

$$co-UPVmatrix = \begin{pmatrix} f_{11} & f_{21} & \cdots & f_{n1} \\ f_{12} & f_{22} & \cdots & f_{n2} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ f_{1c} & f_{2c} & \cdots & f_{nc} \end{pmatrix} \tag{4}$$

where, f_{ij} is a user preference value for feature(i) on the co-rated movie(j). n is the number of features in each UPV. c is the number of co-rated movies. According to Fig. 7, the element(f_{12}) in UserA's $co-UPVmatrix$ is the UserA's preference value for the feature (actor) on the co-rated movie ("E.T.").

To find how each pair of users likes or dislikes the same movie features, similarity between each common column in their *co-UPV matrices* is calculated using similarity metric in Eq.(3).

After similarity between their common columns of all features are calculated, the feature similarity vector (FSV) of these two users will be produced as shown in Fig. 7. Its characteristic is $FSV = (y_1, y_2, \ldots, y_n)$; where y_i refers to the similarity value between two users about how much they like or dislike the same specific movie feature(i). $n = 10$ which is the same number of features as in UPV.

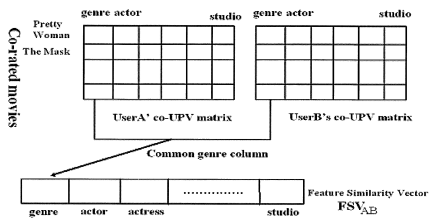

Fig. 7 co-UPV matrices and the way to get FSV.

Step(2): In order to find how each pair of users has similar behavior in selecting movies, the similarity between their $FDVs$ is calculated as shown in Fig. 4. The system first normalizes the FDV of each user and then calculates the difference between their vectors. After that, the system converts this difference vector to a similarity vector by calculating the difference between full difference and difference (full difference - difference) for each element of the vector. The similarity result is in a form of a feature dependency similarity vector (DSV). Its characteristic is $DSV = (v_1, v_2, \ldots, v_n)$; where v_i refers to the similarity value between the behavior of two users in selecting movies on the movie feature(i). $n = 10$ which is the same number of features as in FSV.

Step (3): The system generates the FSV and the DSV of all other users that have towards an active user. The system then merges FSV and DSV by weighted average using the DSV as weight in order to get the final similarity values between the active user and other users as follows:

$$s_{aB}(i) = \frac{v_{aB}(i) \times y_{aB}(i)}{\sum_{i=1}^{n} v_{aB}(i)} \tag{5}$$

$$S_{aB} = \frac{\sum_{i=1}^{n} (v_{aB}(i) \times y_{aB}(i))}{\sum_{i=1}^{n} v_{aB}(i)} \tag{6}$$

where, S_{aB} is a final similarity value between an active user(a) and UserB. s_{aB}(i) is a final similarity value between the active user(a) and UserB on the feature(i). v_{aB}(i) is a value on feature(i) in the vector DSV_{aB}. y_{aB}(i) is a value on feature(i) in the vector FSV_{aB}. n is the number of features in both DSV_{aB} and FSV_{aB}.

Step (4): The final neighbors are the users (in the first neighbor set) who have the Top N high final similarity values calculated from Eq.(6).

Table 2 Contingency table.

	Relevant by user(A)	Non-Relevant(A)
Accepted by system(B)	$A \cap B$	$A \cap B$
Rejected by system($\bar{B}$)	$A \cap \bar{B}$	$\bar{A} \cap \bar{B}$

4.4 Generating Recommendations

After the final neighbors are discovered, a recommendation value for each movie will be calculated according to Eq.(7). This calculates the weighted average on UPVs of all final neighbors by using the final similarity value between the active user and those final neighbors on each feature (from Eq.(5)) as a weight. Then, the system presents the recommendation results to the user with the ranked list of recommended movies.

$$R_{aj} = \frac{\sum_{K \in Neighbors} \left(\frac{\sum_{i=1}^{n} s_{aK}(i) \times f_{ij}(K)}{\sum_{i=1}^{n} s_{aK}(i)} \right)}{N} \tag{7}$$

where, R_{aj} is a recommendation value for the active user(a) on the movie(j). K is a neighbor in the final neighbor set. s_{aK}(i) is a final similarity value between active user(a) and UserK on the feature(i). f_{ij}(K) is a preference value for feature(i) on the movie(j) in UserK's UPV. n is the number of features in UPV. N is the number of neighbors in the final neighbor set.

5. Experimental Evaluation

The objective of experimental evaluation is to answer the following question. In the area of content/collaborative hybrid systems, could Advanced Yawara, which is based on neighbors produced from the cascade model of content-based method and co-rated items method with the "user's opinions on various features", provide more accurate recommendations and higher capability in retrieving information than the systems based on neighbors produced from pure content-based method or pure co-rated items method?

We therefore compared Advanced Yawara with two other content/collaborative hybrid systems. One is e-Yawara [8], our first hybrid system that uses the similarity between content-based user profiles in finding neighbors. The other is MovieLens system [7], a well-known hybrid system that uses the similarity between co-rated items in finding neighbors.

5.1 Data

In the experimental evaluation, the data of 1020 movies was provided in the movie database and 50 users used the system. Total opinions collected from the experiment sum up to 975 ratings (*training set*). The minimum number of movies rated by each user is 8. The maximum is 36. The average is 17.03.

Accuracy of the recommendations generated by the system will be revealed when the users say that they want to see the like most movies predicted and don't want to see the dislike most movies predicted.

We simulated the methods of e-Yawara and MovieLens on the same data set of Advanced Yawara, then the 10 movies each user likes most and the 10 movies each user dislike most are predicted by each system using ratings in the *training set*. Then these 20 movies were displayed to a user. After that, each system will ask each user in return how he/she feels toward these 20 movies; "Want to see", "Don't want to see" or "Neutral". Since there are 50 users and each user has to answer in return to each system for 20 predicted movies, there are 1000 movies in our *test set*.

5.2 Evaluation Criteria

Five criteria were used for determining the accuracy and quality of the recommendations.

MAE (Mean Absolute Error) [5] is the average absolute deviation between the system's recommendation value and the user's actual preference value. The MAE is represented as Eq.(8). The lower the MAE, the more accurate the results.

$$|\bar{E}| = \frac{\sum_{i=1}^{T} |R_i - p_i|}{T} \tag{8}$$

where, R_i is a recommendation value for each movie in the *test set* (the like most movies predicted (score = 1) and the dislike most movies predicted (score = -1)). p_i is the user's actual preference value for each movie in the *test set* (gotten when the system asks each user in return how he feels towards each predicted movie: "Want to see" (score = 1), "Don't want to see" (score = -1), and "Neutral" (score = 0)). T is the number of movies in the *test set*.

To detail the four criteria below, the famous contingency table is introduced (**Table 2**). A is a set of relevant items by user, and $\bar{A}$ is a set of non-relevant items by the user. Relevant items

Table 3 Evaluation results between Advanced Yawara and two other hybrid systems.

System	MAE	Recall	Precision	Specificity	Negative Predictive Value
Advanced Yawara	0.493	73.38%	81.6%	96.23%	51.2%
e-Yawara	0.775	62.39%	70%	70.66%	36.6%
MovieLens	0.783	61.92%	67%	65.08%	38.4%

refer to the movies (in the *test set*) that user answers for "Want to see". Conversely, non-relevant items are the movies (in the *test set*) that user answers for "Don't want to see". B is a set of items accepted by the system (the like most movies predicted), and $\bar{B}$ is a set of items rejected by the system (the dislike most movies predicted).

Recall (or Sensitivity) [11] is the probability that the relevant items will be accepted by the system.

$$Recall = \frac{|A \cap B|}{|A|} \quad (9)$$

Precision (or Positive Predictive Value) [11] is the probability that the accepted items are relevant.

$$Precision = \frac{|A \cap B|}{|B|} \quad (10)$$

Specificity [11] is the probability that non-relevant items will be rejected by the system.

$$Specificity = \frac{|\bar{A} \cap \bar{B}|}{|\bar{A}|} \quad (11)$$

Negative Predictive Value [11] is the probability that the rejected items are non-relevant.

$$NegativePredictiveValue = \frac{|\bar{A} \cap \bar{B}|}{|\bar{B}|} \quad (12)$$

If either one or more values of the Recall, Precision, Specificity or Negative Predictive Value are high, the high-quality recommendations and the high retrieval capability will be obtained.

5.3 Evaluation Results

We employed all criteria in Section 5.2 to compare Advanced Yawara with the two other content/collaborative hybrid systems; e-Yawara and MovieLens. As presented in **Table 3**, the **MAE** of Advanced Yawara is lower than e-Yawara and MovieLens. It can be concluded that Advanced Yawara provides more accurate recommendations than these two systems. Table 3 also shows that the capability of Advanced Yawara in retrieving relevant movies is higher than e-Yawara and MovieLens, because the **Recall** and **Precision** values from Advanced Yawara are higher than both of these systems. In addition, the values of **Specificity** and **Negative Predictive Value** from Advanced Yawara are also higher than these two systems, so it can be concluded that Advanced Yawara is more capable than both e-Yawara and MovieLens in rejecting non-relevant movies.

6. Discussion

As mentioned in the previous section, Advanced Yawara provides higher quality recommendations than both e-Yawara and MovieLens. One reason is e-Yawara only employs the similarity between content-based user profiles and MovieLens only employs the similarity between co-rated items in finding neighbors. Therefore, the limitations still remain in their respective neighborhood formation methods. In contrast, Advanced Yawara employs two filtering processes on both of these methods with the aim of refining neighbors. The Prediction List is generated in the first filtering in order to enhance the co-rated items in the second filtering much more likely to be discovered. Therefore, it can reduce sparsity rating and synonymy problems. The Prediction List contains the list of interesting movies, so it can be the rough recommendation list for the user when co-rated items could not be discovered in the second filtering. That is, it can overcome the cold start problem also.

Another reason is Advanced Yawara can prevent both the *rating value alone* and *missing weight feature* problems, because it represents the user's opinion using both the user preference vector, a vector that expresses many features of user preference, and the feature dependency vector, a vector that represents user's dependency on features. In contrast, MovieLens employs rating value alone in representing the user's opinion. Likewise, the content-based user profiles in e-Yawara can only be learned from a genre feature of movie, not from all the necessary movie features. That is the user profiles in both MovieLens and e-Yawara cannot cover the necessary features of user interest. Therefore, Advanced Yawara provides higher

quality neighbors than these two systems. After higher quality neighbors are found in Advanced Yawara, higher quality recommendations and higher capability in retrieving the relevant movies and rejecting non-relevant movies will be produced accordingly.

When we concentrate on the results between e-Yawara and MovieLens, e-Yawara provides a little better retrieval efficiency than MovieLens. This is because, in the experiment, the average number of movies rated by each user is only 17.03 movies. It is a very small number when compared with the number of movies in the entire database (1020 movies). This means that the ratings of each user is very sparse. Since MovieLens uses only co-rated items in finding neighbors, this sparsity tends to cause its recommendations to be of low quality.

Although Advanced Yawara uses co-rated items in the second filtering, it still provides high-quality results, because the process used to find neighbors in the first filtering leads co-rated items to be found easily in the second filtering.

7. Conclusions and Future Works

In this paper, an advanced content/collaborative hybrid system has been proposed to recommend movies. It is based on a new neighborhood formation method, which is likely to produce high-quality neighbors and high-quality recommendations accordingly. Instead of rating value alone, the proposed system represents a user's opinion using both a user preference vector, a vector that expresses many features of user preference, and a feature dependency vector, a vector that represents user's dependency on features, in order to prevent both *rating value alone* and *missing weight feature* problems. It also employs two filtering processes to apply both of these vectors and to overcome limitations of current neighborhood formation methods. That is, it can overcome *sparsity rating* and *synonymy* problems in the co-rated items method and problem about the quality of neighbors is restricted to a small number of item features in the content-based neighborhood formation method. Therefore, it can provide higher quality neighbors than neighborhood formation methods of the current content/collaborative hybrid systems.

Although, it seems that, the two filtering processes may cause more computation, it can still reduce the *scalability problem*. Because, in finding the distance between each user feature and the features of all the movies, the first filtering groups similar movies in the offline process, so it calculates the distance between each user and the groups of movies, instead of all movies in the entire movie database. Furthermore, the users compared with the active user in the second filtering will be the ones who are in the neighbor set obtained from the first filtering, not from the entire user database as made in the current systems.

Moreover, the proposed system can provide an elegant solution to the *cold start problem*, since initial recommendations are the predictions in the Prediction List obtained from the first filtering. This dynamic Prediction List relates to the user preference, so it can be the interesting movie list presented to a user for entering opinions.

An experimental movie recommender system called Advanced Yawara has been created to evaluate the proposed method. As presented in the experimental evaluation, Advanced Yawara can provide higher quality recommendations and be more capable in retrieving interesting information and rejecting those which are less interesting than current content/collaborative hybrid systems.

From the experiment, the average number of movies rated by each user is only 17.03 movies. It is a very small number when compared with the number of movies in the entire database (1020 movies). It illustrates that Advanced Yawara can provide good quality recommendations, in the case that the number of ratings is small and sparse. This is good, because users will be bored if they have to rate a large number of items before getting the required information. However, Advanced Yawara will meet the computation problem when the number of user's opinions (ratings) increase. For future work, the scalability problem would be more concentrated to increase the performance of our recommender system.

Acknowledgments The authors wish to express our special thanks to our laboratory members for useful comments.

References

1) Resnick, P. and Varian, H.R.: Recommender Systems, *Comm. ACM*, Vol.40, No.3, pp.56–58 (1997).
2) Ben, J., Konstan, J.A. and Riedl, J.: E-Commerce Recommendation Applications,

Proc. Data Mining and Knowledge Discovery, pp.115–153, Kluwer Academy Publishers, (2001).
3) Linden, G., Smith, B. and York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering, *Proc. February 2003 issue of IEEE Internet Computing* (2003). http://dsonlin.computer.org/0301/d/w1lind.htm
4) Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M.: Combining Content-Based and Collaborative Filters on an Online Newspaper, *Proc. Recommender System Workshop at ACM SIGIR 1999* (1999).
5) Upendra, S. and Patti, M.: Social Information Filtering: Algorithms for Automating "Word of Mouth", *Proc. ACM CHI'95 Conference on Human Factors in Computing Systems*, pp.210–217 (1995).
6) Burke, R.: Hybrid Recommender Systems: Survey and Experiments, *User Modeling and User-Adapted Interaction*, Vol.12, No.4, pp.331–370 (2002).
7) Good, N., Shafer, J.B., Herlocker, J. and Riedl, J.: Combining Collaborative Filtering with Personal Agents for Better Recommendations, *Proc. Conference of the American Association of Artificial Intelligence* (*AAAI-99*), pp.438–446 (1999).
8) Maneeroj, S., Kanai, H. and Hakozaki, K.: An Improved Recommendation Method for Better Filtering Information out of Database, *IPSJ Transactions on Databases*, Vol.43, No.SIG 5(TOD 14), pp.66–73 (June 2002).
9) Rashid, A.M., Albert I., Cosley, D., Lam, S.K., McNee, S.M., Konstant, A.J. and Riedl, J.: Getting to Know You: Leaning New User Preferences in Recommender Systems, *Proc. IUI 2002*, pp.127–134 (2002).
10) McNee, S.M., Lam, S.K., Konstant, J.A. and Rield, J.: Interfaces for Eliciting New User Preferences in Recommender Systems, *Proc. UM 2003*, pp.178–188 (2003).
11) MedCalc Organizations: ROC curve analysis: introduction. http://www.medcalc.be/manual/mpage06-13a.php

(Received September 20, 2004)
(Accepted December 30, 2004)

(Editor in charge: *Chieko Nakabasami*)

Saranya Maneeroj is a Ph.D. student at Graduate School of Information Systems, the University of Electro-Communications, Japan. She received a M.S. in computer engineering from Graduate School of Information Systems, the University of Electro-Communications, Japan in 2001. Her current research focuses on Recommender System based on Collaborative Filtering and Information Filtering techniques.

Yuka Kato received her B.Sc. degree from The University of Tokyo, Japan in 1989, and her M.E. and Ph.D. degrees from the University of Electro-Communications, Japan in 1999 and 2002 respectively. From 1989 to 1998, She was with NTT and engaged in research on traffic control in ATM networks and interactive multimedia systems. Since 2002, she has been with Graduate School of Information Systems at The University of Electro-Communications, where she is a research associate. Her research interests include QoS (Quality of Service) control for distributed multimedia systems and user-oriented QoS. She is a member of IEEE and the Institute of Electronics, Information and Communication Engineers (IEICE).

Katsuya Hakozaki received B.S. degree in electronic engineering in 1963 and D. Eng. degree in 1981 both from Kyushu University. He joined NEC Corporation in 1963 and was engaged in research and development of computer architecture, performance evaluation, and computer networks. He is a professor of the University of Electro-Communications, Graduate School of Information Systems since 1994. His recent interests are digital libraries, mobile communication aids, multimedia internet applications. He was awarded IPSJ Best Paper Award in 1982. He is a member of Information Processing Society in Japan, Institute of Electronics, Information and Communication Engineering, IEEE and ACM.