A Rule based Approach to Group Recommender Systems

A Project Report Submitted in the partial fulfillment of the requirements for the award of degree of

Master of Technology

in

Artificial Intelligence

By

Siva Krishna Seemala

Department of Computer and Information Sciences
University of Hyderabad
Hyderabad

30th June, 2009

A Rule based Approach to Group Recommender Systems

Siva Krishna Seemala

A Project Report Submitted in the partial fulfillment of the requirements for the award of degree of
Masters in Technology,Artificial Intelligence.
30-june-2009

Abstract

Recommender systems have become valuable resources for users seeking intelligent ways to search through the enormous volume of information available to them. The problem of building Recommender Systems has attracted considerable attention in recent years, but most recommender systems are designed for recommending items for individuals. In this paper, we develop a content based group recommender system that can recommend TV shows to a group of users using machine learning techniques. We propose a rule based approach (DLRL) which is based on the well known Ripper rule learner to learn the rule base by taking into consideration the users viewing history and a method called RTL strategy which is based on social choice theory strategies to generate group ratings. We compare our learning algorithm with the existing c4.5 rule learner and the experimental results show that the performance of our rule learner is better in terms of literals learned (size of the rule set) and our rule learner takes time that is linear to the number of training examples. We then propose two new methods to generate group ratings that is better than the existing social choice theory strategies.

Certificate

This is to certify that the project work entitled "**A Rule based Approach to Group Recommender Systems**" being submitted to University of Hyderabad by **Siva Krishna Seemala** (Reg. No. 07MCMI18), in partial fulfillment for the award of the degree of Master of Technology in Artificial Intelligence, is a bonafide work carried out by him under my supervision.

Dr.Vineet P Nair
Project Supervisor,
Department of CIS,
University of Hyderabad.

Head of Department,
Department of CIS,
University of Hyderabad.

Dean,
School of MCIS,
University of Hyderabad.

Dedicated to,

My Family and Friends

Acknowledgements

I would like to express my sincere gratitude to Dr.Vineet P Nair, my project supervisor, for valuable suggestions and keen personal interest throughout the progress of my course of research and for advising to make this work to be communicate in "the 22nd Australasian Joint Conference on Artificial Intelligence (AI'09)". He encouraged, supported, corrected and guided me during the project. The project has been a learning and growing experience for me. It is a good piece of work done in his guidance and a great opportunity to learn many things.

I would like to thank Prof.Bapi Raju for his valuable suggestions about my project.

I am thankful to my lab mates for their cooperation while doing my project in that lab.

I am extremely grateful to our Head of the Department, Prof. Arun Agarwal, for providing excellent computing facilities and a nice atmosphere for doing my project. I convey my heartfelt thanks to AI Lab staff for their help in completing the project work successfully.

I would like to take this opportunity to thank my friends who have been morale boosters for me, encouraged me to take up this course and supported me throughout this course with their love and affection. My special thanks to my wonderful parents who have always supported me in all my decisions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There is lot of data available in the internet in the form of books, articles, movies, music, web sites etc. Therefore, selecting a particular item that is of our own interest is very difficult task. So we need systems for recommending items (e.g. books, movies, web sites) that are likely of our interest. These systems are called as Recommender Systems. Recommender systems have become valuable resources for users seeking intelligent ways to search through the enormous volume of information available to them. The problem of building Recommender Systems has attracted considerable attention in recent years, but most recommender systems are designed for recommending items for individuals. The aim of our work is to develop a group recommender system that is able to recommend the TV shows to a group of users. The proposed model uses rule learning algorithm based on RIPPER [11] rule learner to learn rule base from user viewing history and a method called LMTH strategy based on social choice theory strategies [15] to generate group ratings.

Recommender Systems are of two types: collaborative recommender systems and content based ecommender systems. In Collaborative Recommender Systems, a database consisting of many users ratings of a variety items is maintained. Each user has his own user profile. Initially these user profiles can be build in two diffrent ways: 1.By explicit data collection 2.By implicit data collection. Examples of explicit data collection include the following:

- Asking a user to rate an item on a sliding scale.
- Asking a user to rank a collection of items from favorite to least favorite.
- Presenting two items to a user and asking him/her to choose the best one.

- Asking a user to create a list of items that he/she likes.

Examples of implicit data collection include the following:

- Observing the items that a user views in an online store.
- Analyzing item/user viewing times.
- Keeping a record of the items that a user purchases online.
- Obtaining a list of items that a user has listened to or watched on his/her computer.
- Analyzing the user's social network and discovering similar likes and dislikes.

After collecting user profiles (explicitely or implicitly), the colloborative method recommend items for an active user(The user who wants recommendations is called an active user) as follows, it simply searches the database to find other similar users for the active user (similar users are found based on their user profiles). Based on those similar users, it will recommend the items. But there are cold start, first rater, popularity bias problems with collaborative recommender systems.Coldstart occurs when there is no enough similar users in the database. First rater problem occurs when the item is new or it hasn't been rated earlier by any user. Popularity bias occurs when an item is recommended based on the opinion of other similar users.

In recent years, a few group recommnder systems have been designed. All of these systems have been developed based on collaborative filtering techinique [10, 12, 20]. Hence, they suffer with the problem of collaborative filtering as stated above. Our approah uses content based recommendation.

In Content based approch, Recommendations are based on information on the content of items rather than on other users opinions. It uses a machine learning algorithm to induce the profile of the user preferences from training examples based on a featural description of content. To learn user profile, many machine learnig algorithms have already been used. Among those, popular algorithms are (1) decision tree rule learners and (2) Naive Bayes classifier. But Tv content is a kind of structured data, so we tried to use rule learners. Decision tree rule learners have already been used to develop recommenender systems. But there are some problems with decision trees:

- decision trees are often quite complex and hard to understand. Even a pruned decision trees may be too cumbersome, complex, and inscrutable to provide insight into the domain at hand [13].
- There is a replicated subtree problem.it often happens that identical subtrees have to be learned at various places in a decision tree [13].
- Sequential Covering algorithms (Decision List Learners) are better than Simultanious covering algorithms(Decision Tree rule learners)when plenty of data available [16].
- we can prove that decision list (Ordered set of rules) with at most k-conditions per rule strictly more expressive than decision tree of depth k [19].

Therefore, we use Decision List rule learner based on RIPPER to learn rulebase and LMTH strategy based on social choice theory strategies for group recommendation.

This thesis is organized as follows: In Chapter 2 ,we briefly discuss about collaborative and content based approaches. Chapter 3 describes recent approaches to group recommender system. Chapter-4 describes about proposed methods to group recommender system. Chapter-5 provdes the experimental results. Finally conclusion, bibilography and screenshots of implemented program follows.

Chapter 2

Recommender Systems

Recommender systems are software applications that aim to support users in their decision-making while interacting with large information spaces. They recommend items of interest to users based on preferences they have expressed, either explicitly or implicitly. The ever-expanding volume and increasing complexity of information on the web has therefore made such systems essential tools for users in a variety of information seeking or e-commerce activities. Recommender systems help overcome the information overload problem by exposing users to the most interesting items, and by offering novelty, surprise, and relevance.

Recommender Systems are of two types:

- Collaborative Recommender Systems
- Content based Recommender Systems

2.1 Collaborative Recommender Systems

In Collaborative Recommender Systems, there is a database consisting of users and their ratings for the variety of items which have been seen by them. The user who wants recommendations is called active user. If we want to recommend items for an active user, it simply searches the database to find other similar users for the active user. Based on those similar users, it will recommend the items. For example, in Figure 2.1, there are six users and their profiles consiting of A, B, C ....Z items and ratings of these items are shown. Here, for the given active user similar usres are found based on the correlation match. In figure, there are two similar users. Among these two users , one person has a reating of 8 for item C. Therefore, collaborative method recommends the item C for the active user.

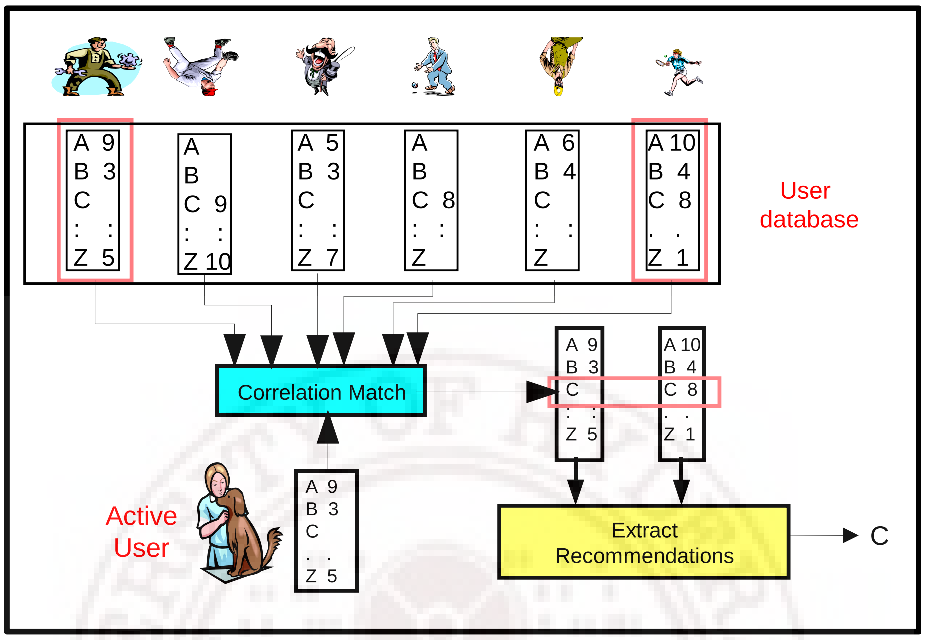


Figure 2.1: Collaborative Recommender Systems

2.1.1 Methods used in Collaborative Recommnder Systems

1. **Nearest neighbourhood classification :** This method is very simple. There is user database which contains user ratings for different items. The user who need recommendations is called an active user. To recommend items to active user by this method, it finds similar user (nearest neighbour) or k-similar users (in case of k-nearest neighbourhood algorithm) in the database. To measure similarity, Euclidian distance can be used. Finally, the items that are recommended by these similar users will be recommended to the active user.

   For example, Let us consider a database of three users namely U_1, U_2, U_3 and four movies namely Billa, Kick, Tagore, Mallanna. The ratings for these four movies can be shown as a vector consisting of four entries:
   Rating Vector $= < R_{Billa}, R_{Kick}, R_{Tagore}, R_{Mallanna} >$.
   where R_{Billa} is the rating for movie Billa, similarly $R_{Mallanna}$, R_{Tagore}, R_{Kick} are the rating for movies Mallanna, Tagore and Kick respectively. Assume the

rating vectors for :
$U_1 =< 10, 3, 9, - >$
$U_2 =< -, 9, 6, 2 >$
$U_3 =< 1, 7, -, 3 >$.
Consider an active user whose rating vector as $U4 =< 9, -, -, - >$.
If we consider the Euclidian distace between the active user and other three users, the user U_1 is similar to the active user. Since user U_1 has a rating of 9 for the movie Tagore, it will be recommended to the active user.

2. **Pearson correlation:** In statistics, correlation (often measured as a correlation coefficient, ρ) indicates the strength and direction of a linear relationship between two random variables. The most common measure of correlation is the Pearson Product Moment Correlation (called Pearson's correlation for short). The formula used to calculate pearson correlation as follows:

$$Pearson\ correlation\ between\ user\ a\ and\ user\ u = \frac{Covariance(r_a, r_u)}{\sigma_{ra}\sigma_{ru}} \tag{2.1}$$

here σ is the standard deviation.
Covariance and standard deviation are calculated as follows:

$$Covariance(r_a, r_u) = \frac{\sum_{i=1}^{m}(r_{a,i} - \overline{r_a})(r_{u,i} - \overline{r_u})}{m} \tag{2.2}$$

here,

$$\overline{r_a} = \frac{\sum_{i=1}^{m} r_{a,i}}{m} \tag{2.3}$$

And standard deviation is calculated as

$$\sigma_{r_x} = \sqrt{\frac{\sum_{i=1}^{m}(r_{x,i} - \overline{r_x})^2}{m}} \tag{2.4}$$

We can use this person correlation to find the similar users to the active user. Now we can recommend the items that are liked by that similar users.

2.1.2 Problems with Collaborative Filtering:

- **Cold Start:** We need enough other users already in the user database to find a match. Otherwise it cannot recommend the items.

- **First Rater:** It cannot recommend an item that has not been previously rated. That means if the item is new or that hasn't been rated earlier by any other user ,than it cannot recommend that item.

- **Popularity Bias:** it cannot recommend items to someone with unique tastes. it recommnend the items based on other user profiles which tends to recommend popular items.

2.2 Content Based Recommender Systems

In content based approach [17], Recommendations are based on information on the content of items rather than on other users opinions. It uses a machine learning algorithm to induce a profile of the users preferences from training examples based on a featural description of content. Based on that learned user profile, the items that are matched to the profile will be recommend to the active user.

For example, consider the LIBRA system that is used by Amazon to recommend books, shown in Figure 2.2. There are Rated examples collected from Amazon pages. There is machine learning algorithm that learns the user profile for the collected training examples. To recommend the books for an active user , the learned user profile used. The book that matches the user profile will be recommneded.

2.2.1 Methods Used in Content based Recommender Systems

1. **Naive bayes classifier :** One highly practical bayesian learning method is the naive bayes classifier. The naive bayes classifier applies to learning tasks where each instance x is described by a conjuction of attribute values and where the target function f(x) can take on any value from some finite set V. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values $< a_1, a_2, a_3.., a_n >$. The learner is asked to predict the target value, or classification for this new instance. The naive bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. In other words, the assumption is that given the target value of instance, the

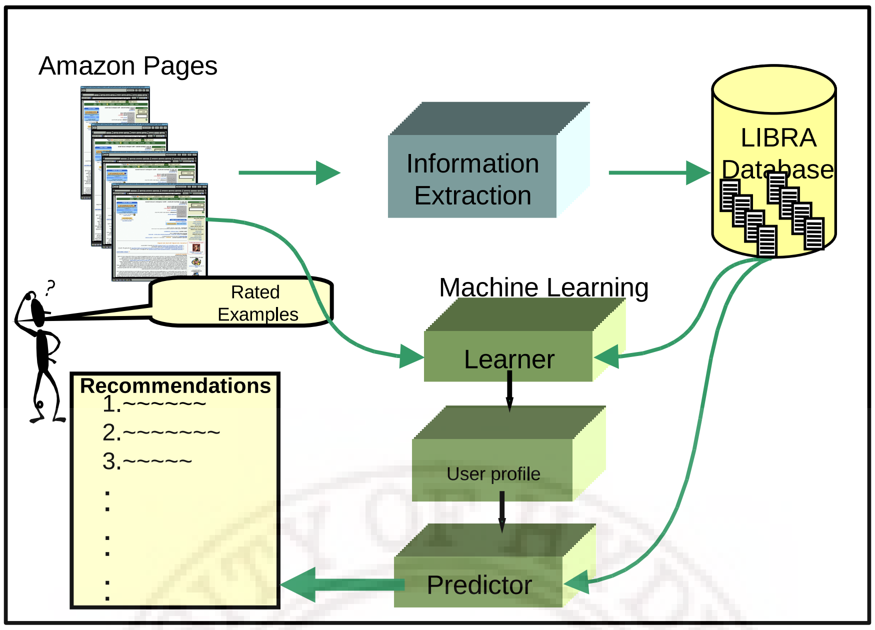


Figure 2.2: Content based Recommender Systems

probability of observing the conjunction $a_1, a_2, a_3.., a_n$is just the product of the probabilities for the idividual attributes: $P(a_1, a_2, a_3.., a_n|v_j) = \Pi_j P(a_i|v_j)$.

$$V_{NB} = argmax_{v_j \epsilon V} P(v_j) \Pi P(a_i|v_j) \tag{2.1}$$

For example, Let us consider three documents.
$D_1 = \{cat, dog, fly, cow\} \rightarrow yes$
$D_2 = \{crow, strow, fly, zebra\} \rightarrow no$
$D_3 = \{cat, dog, z00m, flex\} \rightarrow yes$
Let us take one instance $D_4 = \{cat, zoom, fly, dog\}$. Now we find the classification for this instance by naive bayes classifier. Total number of words are nine. Number of unique words in "yes class" are six. Number of unique words

in "no class" are four.

Therefore,

$P(cat|yes) = 2/6$

$P(cat|no) = \epsilon$

$P(zoom|yes) = 1/6$

$P(zoom|no) = \epsilon$

$P(fly|yes) = 1/6$

$P(fly|no) = 1/4$

$P(dog|yes) = 2/6$

$P(dog|no) = \epsilon.$

And, $P(yes|D_4) = P(cat|yes) * P(zoom|yes) * P(fly|yes) * P(dog|yes)$

$= 2/6 * 1/6 * 1/6 * 2/6$

$= 0.003$

$P(no|D_4) = P(cat|no) * P(zoom|no) * P(fly|no) * P(dog|no)$

$= \epsilon * \epsilon * 1/4 * \epsilon$

$= 0.003$

$where\ \epsilon = 0.1$

Here $P(yes|D_4) > P(no|D_4)$, Therefore classification for the given instance is "yes".

2. **Decision tree rule learner:** The decision tree can be build using Quinlan's C4.5 rule learner [18] that uses an information-theoretic approach based on entropy. C4.5 builds the decision tree using a top-down, divide-and- conquer approach: it first selects an attribute, then divides the training set into subsets characterized by the possible values of the attribute, and follows the same procedure recursively with each subset until no subset contains objects from more than one class.

   When the training data is structured, rule based recommandation is better. For example, consider a movie recommender system. Here the movie data is structured becuase the information about the movie like director, hero etc are related. One of the popular rule learner is decision tree rule learner. It learns the decision tree that can be further used to get the set of rules to classify the given training examples. By using these rules, we can classify the items. For example, consider the structerd data,

Director = "Ragavender rao" and Hero="Chiranjeevi" → like

Title="police academy" → not like

Decision tree for this example is shown in figure 2.3. By using this decision tree, we can classify the movies and based on that classification we can say that a particular movie is recommendable or not.

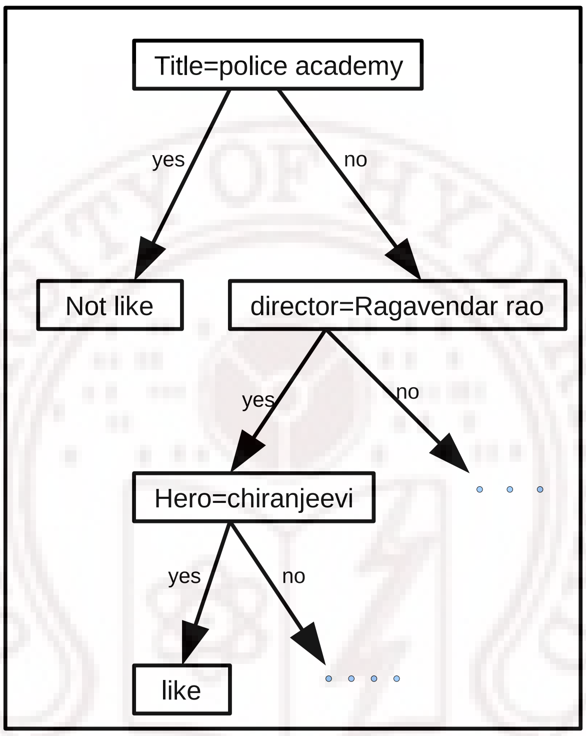


Figure 2.3: Decision tree rule learner

2.3 Popular Recommender Systems

- **MovieLens [1]:** MovieLens is a recommender system that uses collaborative filtering to provide the user with movie recommendations based on his/her personal references. Their preferences are matched with the preferences of other users with similar movie preferences. MovieLens is a project of GroupLens Research.

- **WhatShouldIReadNext.com [2]:** This site helps one to determine the next book he/she should read. Add in the title of a recent book he/she enjoyed and the site will return recommended books from a database of over 20,000 titles. The user can also check out what other readers have enjoyed. The site is adding film and music recommendations.

- **Last.fm** [3]**:** Last.fm is a free Internet radio station that uses a recommender system based on tracking what users listen to and makes suggestions based on the users tastes. This is not an on-demand service and one can not request that a piece of music be played at a specific time.

- **StumbleUpon** [4]**:** StumbleUpon is a recommender system for web surfers. Using ratings gathered with a collaborative opinion rating system it can match users with interesting websites based on their preferences. Users can also rate interesting sites for others to check out.

- **MyStrands** [5]**:** MyStrands is a free personal Internet radio station. A behavior-based recommender system is used to track the music that the users have selected on iTunes, Windows Media player or have stored on their iPod. The system can recommend music tailored to user tastes, help organize play lists, auto fill mobile devices with music and help find more music.

- **ChoiceStream** [?]**:** ChoiceStream makes recommendations based on user preferences. They develop marketing strategies for online shopping, television (based on the TV show being watched), and for mobile phones (using information such as ring tones and music downloads). The company also develops direct marketing and email marketing campaigns that can be customized to reach specific users.

- **CleverSet** [6]**:**

  CleverSet helps Internet retailers get sales by using information gathered about Internet shoppers to make product recommendations. CleverSet analyzes a customer's past and current buying and searching trends including what was looked at last, what is currently being looked at, amount of time spent on a web page, time of day and the day of the week a user was on a site, time spent looking at a product or similar product(s), and the region of the country the user is from. They also gather information about the product including description, manufacturer, price and product image.

- **Whattorent.com [7]:**

  Whattorent.com will recommend movies to the users based on the answers that he/she give to their survey. The initial survey is 20 questions including favorite movie genre (just fun, real life, harsh real life, super harsh), how long it takes them to fall asleep, and more. Based on their answers they will receive a suggested movie.

- **Netflix [8]:**

  Netflix is a DVD movie rental site. Netflix will recommend a movie or movies based on member reviews, critic reviews, popular rental lists and how user rate movies. When user signup for this service he/she submit a list of DVDs that would like to rent from their extensive list of movies. User can receive from one to four DVDs at a time; when the user return a DVD to Netflix they will send another DVD from him/her list. Several rental plans are available with options as to the number of DVDs that can be ordered per month (depending on the monthly price of the plan). Netflix offers a free trial subscription.

- **Pandora [9]:**

  Music recommendation system. Pandora emerged from the Music Genome Project: a comprehensive analysis of music in which a team of musician-analysts listen to and study each song and note nearly 400 attributes of each. Pandora makes this information available to the public. Users enter their favorite songs and artists into Pandora, then Pandora recommends music similar to their taste. Then one can listen to the new music through the site.

Chapter 3

Different Approaches to GRS

Group recommneder system(GRS) is the recommneder system that recommend items for group of users. Group recommender problem can be described as: Let I=$i_1, i_2, i_3, i_4, ...i_n$ set of all items and $U = u_1, u_2, u_3...u_m$ set of all users. $I_G$is the set of items that are not rated by G (subset of U). Goal is to find ratings for items in I_G or to find whether the items in I_G are recommendable to the given group of people(G).

3.1 Strategies for Group Recommender Systems

There are three strategies to develop a Group recommender systems [21].

1. **Group Agent :** In this approach, there is a common account for the group. Group agent learns the preferences for the group, based on that it will recommend the items. But the drawback here is that it works well for the whole group, but if some members in the group are not present then it will not recommend.

2. **Merging recommendations :** In this approach, each user profile is learned initially. From each user profile, separate recommendation list is generated. Finally, all user recommendation lists will be meged to get a group recommendation list. Figure 3.1 shows this method. There are N-users and their profile are shown. Form these user profiles N-Program recommendation lists generated. By the nelp of some recommendation merging algorithm, Recommended program list for the group will be generated.

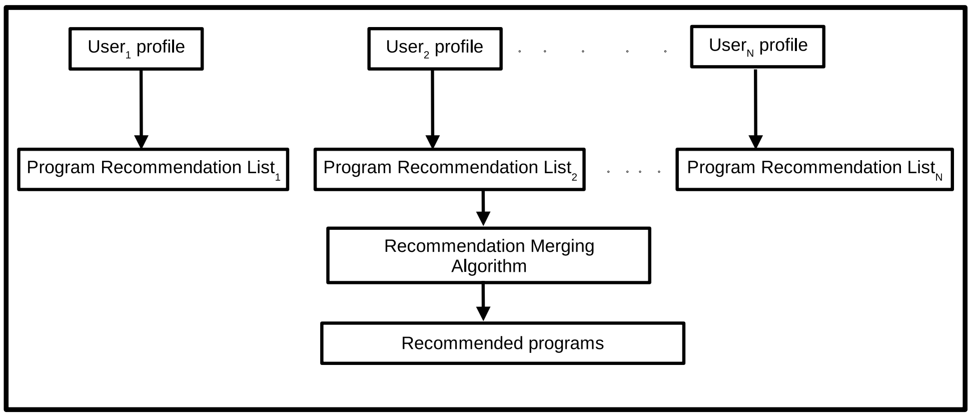

User1 profile
User2 profile
UserN profile
Program Recommendation List1
Program Recommendation List2
Program Recommendation ListN
Recommendation Merging Algorithm
Recommended programs

Figure 3.1: Merging Recommendations

3. **Merging User profiles :** In this approach, each user profile learned initially. All these user profiles are merged to get a common user profile. Finally, recommandations are generated using common user profile. Figure 3.2 shows this method. There are N-users and their user profiles are merged by using some profile merging algorithm, we get a common user profile. By the help of this common user profile, the items will be recommended for groups.

3.2 Previous Methods for GRS

3.2.1 GRS with Considiration of Interaction among Group Members

This method [10] is a novel group recommendation approach based on collaborative filtering and genetic algorithm to predict the possible interactions among group members so that it can correctly estimate the rating that a group of members might give to an item. The major steps in this method are as follows:
G is a group of members which is a subset of U. I is the item set that have group ratings. $\overline{I}_G$ is the item set in which the items haven't been rated earlier.

- Step 1 (Create a possible neighbourhood $\overline{I}_Z$): For item $i_p \epsilon \overline{I}_G$, find similar items by using pearson correlation. Let the set of users who rated both i_p *and* i_z be

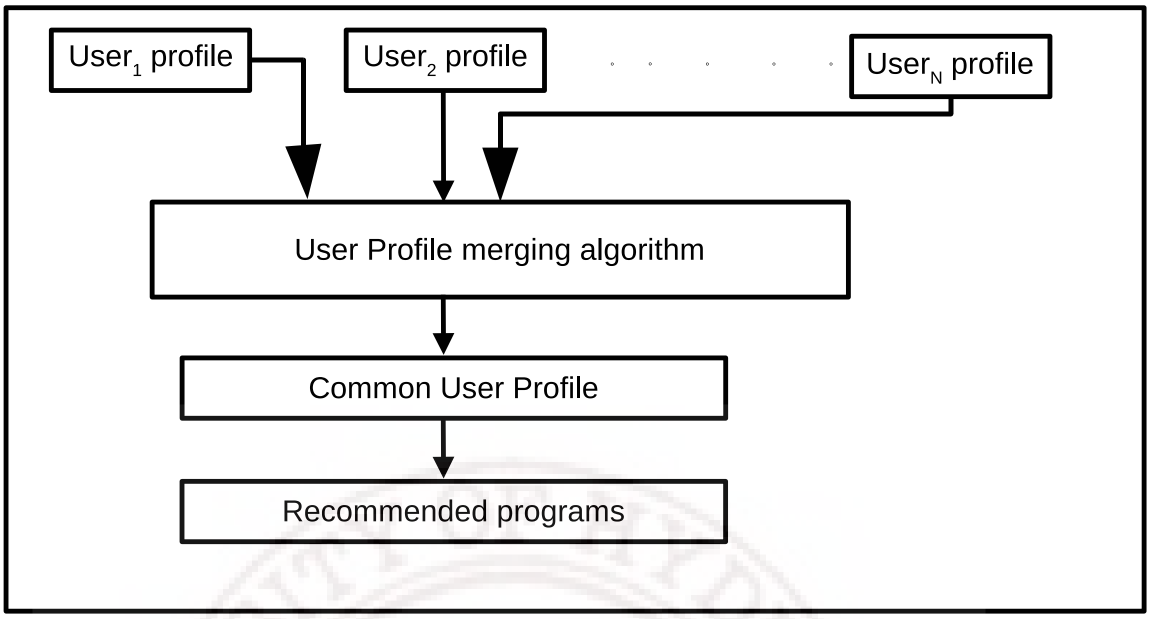


Figure 3.2: Merging User Profiles

denoted by $\overline{U}$.Then, the correlation similarity is given by

$$w_{i_p,i_z} = \frac{\sum_{u_j \epsilon \overline{U}} (R_{u_j,i_p} - \overline{R}_{i_p})(R_{u_j,i_z} - \overline{R}_{i_z})}{\sqrt{\sum_{u_j \epsilon \overline{U}} (R_{u_j,i_p} - \overline{R}_{i_p})^2 \sum_{u_j \epsilon \overline{U}} (R_{u_j,i_z} - \overline{R}_{i_z})^2}} \tag{3.1}$$

where w_{i_p,i_z} is the correlation coefficient between i_p and i_z. R_{u_j,i_p}, R_{u_j,i_z} are the ratings given by user u_j for i_p and i_z respectively. Collect all items that are similar to i_p, denoted by$\overline{I}_Z$

- Step 2 (Filter neighbors $\overline{I}_Z$ and Form neighbors I_Z): It is worth nothing that not every item i_z in the possible neighborhood $\overline{I}_Z$ can be used to predict the rating R_{G,i_p}. An item i_z in $\overline{I}_Z$ is suitable for prediction only if either group G has already rated item i_z or if there is a enough information for G to predict the potential rating of i_z. In the later case, we can predict what rating G would give to i_z by using GA to learn the interactions among group members from the known ratings. If not enough information is available to make a prediction, we remove item i_z from $\overline{I}_Z$.

- Step 3 (Generate group rating for item i_p): It is difficult for a recommendation system to recommend items when there is no prior experience. The neighborhood-based method is probably the most popular method used to resolve the serendipitous items problem. According to this method, we can predict unknown ratings by using similar items in the neighborhood. The formula used to generate group rating is as follows:

$$R_{G,i_p} = \overline{R}_{i_p} + \frac{\sum_{i_z \epsilon I_Z} (R_{G,i_z} - \overline{R}_{i_z})(\overline{w}_{i_p,i_z})}{\sum_{i_z \epsilon I_Z} \overline{w}_{i_p,i_z}} \tag{3.2}$$

3.2.2 TV program Recommendation for Multiple Viewers based on User Profile Merging

In this method [21], First all user profiles are merged to construct a common user profile, and then uses a recommendation approach to generate a common program recommnedation list for the group according to the merged user profile. Merging of user profile is done based on total distance minimization technique. There are seven major steps in this approach :

- Step 1 **(Representing user profile as (feature,weight) vector):** Each user profile is represented as (feature,weight) vectors. Here the features are genre, actor etc. weight is the relative importance given to that feature.
- Step 2 **(Construct the lexicon):** In each user profile, there are many features (e.g.,genre, actor) as well as weights indicating the relative importance of feature. From all these features , we can gather some imporatant features alphabetcally as lexicon.

$$Lexicon = (feature_1, feature_2, ..., feature_n) \tag{3.3}$$

- Step 3 **(Construct the universal vector for each user profile):** With above lexicon vector , we can define each user profile universally as vector V:

$$V = (f_1, f_2, ..., f_n) \tag{3.4}$$

where V is the vector with n items, f_i is the value assigned to corresponding $feature_i$. Supposing user's feature weight belongs to [-1,1] , of which '-1' means maximum disliking, '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.

- Step 4 **(Selecting the features):** Feature selection is based on total distance minimization. For each element of the target vector (merging result), choosing which from{-1,0,1} as its value depends on whether it makes the total distance minimum. Merging result is computed by following equation:

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

- Step 5 **(Normalize the weight):** For each user profile, we can normalize the original weight of each feature according to the following general normalization formula:

$$\overline{w_i} = \frac{w_i - w_{min}}{w_{max} - w_{min}} \times (U_{max} - U_{min}) + U_{min} \tag{3.5}$$

where w_i is $feature_i$ original weight, $\overline{w_i}$ is the normalized values of w_i,and w_{max}, 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 can be simplified as the following:

$$\overline{w_i} = \frac{2w_i - w_{min} - w_{max}}{w_{max} - w_{min}} \tag{3.6}$$

- Step 6 **(Calculate the target weight):** For each selected feature, Weight is calculated as follows:

$$Weight_i = \frac{\sum_{j=1}^{m} \overline{w_{i,j}}}{n} \tag{3.7}$$

where $Weight_i$ is the weight assigned to $feature_i$ in the merged profile, $\overline{w_{i,j}}$ is the normalized weight value of $feature_i$ in the j^{th} user profile.

- Step 7: Generate the target merged user profile

Let us take an example to demoistrate the above steps. Consider five user profiles namely U_1, U_2, U_3, U_4 and U_5.

- **Step 1:** Representing user profile as (feature, weight) vector.
  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))
  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:** Constructing the universal vector for each user profile, shown in Table 3.1.

	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

Table 3.1: Universal vector

- **Step 4:** Select feature.
  According to the formulas given in step4(above),
  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) = ± 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 frature B,V(B) = 0.
  Similarly,
  V(C) = 0, V(D) = ± 1, V(E) = ± 1, V(F) = 0, V(G) = 0, V(H) = ± 1, V(I) = 0, V(J) = ± 1, V(K) = ± 1, V(L) = 0, V(M) = 0, V(N) = ± 1, V(O) = 0, V(p)

= 0. That is, merging result V = (±1,0,0,±1,±1,0,0,±1,0,±1,±1,0,0,±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 -0.7,so
  $\overline{w}(A) = \frac{2\times(-0.3)-0.9-(-0.7)}{0.9-(-0.7)} = -0.5$
  $\overline{w}(D) = \frac{2\times(-0.7)-0.9-(-0.7)}{0.9-(-0.7)} = -1$
  $\overline{w}(E) = \frac{2\times(-0.6)-0.9-(-0.7)}{0.9-(-0.7)} = -0.875$
  $\overline{w}(H) = \frac{2\times(0.9)-0.9-(-0.7)}{0.9-(-0.7)} = 1$
  $\overline{w}(D) = \frac{2\times(0.6)-0.9-(-0.7)}{0.9-(-0.7)} = 0.625$
  $\overline{w}(D) = \frac{2\times(0.4)-0.9-(-0.7)}{0.9-(-0.7)} = 0.375$
  similarly we can perform normalization for U_2, U_3, U_4, 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,Therefore
  $Weight(A) = \frac{-0.5+1+0.1667-0.0909}{5} = 0.1152.$
  similarly, we can obtain Weight(D) = -0.5, Weight(E) = -0.0977, Weight(H) = 0.5, Weight(J) = 0.3326, Weight(K) = -0.0114, and Weight(N) = 0.4.

- **Step 7:** Generate the target merged user profile
  So the merged user profile, U_{merged} = ((A,0.1152), (D,-0.5), (E,-0.0977), (H,0.5), (J,0.3326), (K,-0.0114), (N,0.4)).

3.2.3 A TV Anytime Metadata Approach to GRS

In this method [20], in order to elaborate personilized recommendations for a group of users watching TV together, they start from an OWL ontology that (1) describes TV programs, (2) classifies them in a content hierarchy, and (3) relates them to other programs through their semantic characteristics (attributes such as cast members, location, dates....). Actually, taking the most of the TV-Anytime capabilities, they have devised an enhanced ontological model that reflects the multidimensional content classification scheme of TV-Anytime into their heirarchy structure (there is an ontology for each and every TV-Anytime classification scheme). The descriptions of the programs to be broadcast are recieved in TV-Anytime format incuding their classifications in these schemes (as created by content producers). The instances of the sheduled programs are added to the system database, linking and classifying them in one or several of our ontologies, enabling coparisions in multiple dimensions in order to compute similarity. Regarding these dimensions, their implementation works with four ontologies extracted from the classification schemes Intension, Format, Content, IntendedAudience defined in the TV-A standards.

They defined a semantic similarity metrics to measure the adequacy of a candidate program for a given user by comparing that program to those stored in his/her profile. Given a candidate program P and a program B_i defined in the profile of a user U, their metric has two components:

- The first component is the heirarchical similarity denoted by Sim_{Hie}. It measures the closeness in the content heirarchy of that programs(i.e.,P and B_i). This closeness is computed as:

  $$Sim_{Hie}(P, B_i) = \frac{depth(LCA(P, B_i))}{Max(depth(P), depth(B_i))} \tag{3.8}$$

  where depth of a program is its level in the content heirarchy, and LCA of two programs identifies their Lowest Ancestor in that heirarchy.

- The second component is the inferential similarity denoted by Sim_{Inf}. It discovers associations between programs that share characterstics semantically related to each other, by using reasoning techniques employed in the Semantic web. The inferential similarity can be measured as :

  $$Sim_{Inf}(P, B_i) = \frac{1}{\#SC} \sum_{j=1}^{\#SC} DOI(SC_j) \tag{3.9}$$

The semantic similarity can be measured by combining the above two components and it is calculated as follows:

$$SemSim(P, U) = \alpha.Sim_{Inf}(P, U) + (1 - \alpha).Sim_{Hie}(P, U) \tag{3.10}$$

where α is constant. To apply this single user technique to group(G) of n-users, we can use the following formula:

$$SemSim(P, G) = \Pi_{i=1}^{n} K_i.SemSim(P, U_i) \tag{3.11}$$

where K_i endow each user with a different importance if necessary.

3.2.4 TV Content Recommender System

In this method [14],they mention about an application prototype. It aims to be a smart electronic program guide (smart EPG) that enables a user to search and browse through a TV programs database. It is smart because it maintains an adaptive user profile and makes recommendations of TV programs, computed according to the profile. The application can be divided into 3 parts as described below.

In the current version of prototype, user profiles can be used as search criteria to generate system recommendations of TV content. We are generating TV program recommendations are based on implicit profiles of TV viewers. An implicit profile is built from the viewing history of a TV viewer. The viewing history is a list of shows that a viewer has watched (positive examples) and not watched (negative examples). The implicit nature of our profiling method stems from the fact that the process does not involve any explicit interaction with TV viewers, regarding their likes and dislikes, other than collecting information about what shows have been watched.

The first of our TV program recommender systems uses the Bayesian classifier approach to compute the likelihood that the viewer will like or dislike a particular TV program. We approach the problem with a 2-class Bayesian decision model, where a show belongs to either the class, watched, or the class, not watched. The user profile, in the Bayesian context, is a collection of attributes (or features) together with a count of how many times an attribute occurs in positive and negative examples. From this profile, they first compute the prior probability that a show belongs to a particular class and then the conditional probability that a given feature will be present if a show is in either of the two classes. Using these probabilities we finally compute the a posteriori probability for a new show, given its feature set, that it belongs to a particular class.

The second approach construct rules for classifying shows given a training set of positive and negative shows that are part of the TV viewing history. We begin by deriving a decision tree (DT) which is then decomposed into rules for classifying the shows. The decision tree employed is Quinlan's C4.5 [18] that uses an information-theoretic approach based on entropy. C4.5 builds the decision tree using a top-down, divide-and- conquer approach: it first selects an attribute, then divides the training set into subsets characterized by the possible values of the attribute, and follows the same procedure recursively with each subset until no subset contains objects from more than one class. The single-class subsets correspond to the leaves of the decision tree, while a node indicates that a further test needs to be performed on that show to determine which class the show belongs to. When a new show, which is not part of the training set, is encountered, the DT is parsed to obtain a probabilistic class distribution for the show and the class with the highest probability is the predicted class.

3.2.5 Social Choice Theory Strategies

Social choice (also called as group decision making)-deciding what is best for a group given the opinions of individuals. Television viewing is largely a family or social activity(Barwise and Ehrenberg, 1988;Kasari and Nurmi,1992). Children most often watch with their siblings(Van Evra,1998), Young people would like to watch telivision with friends(Livingstone and Bovill, 1999). For these reasons,we believe that adaptive television should be able to adapt to groups of people watching together.The strategies [15] are as follows:

1. **Plurality Voting:** In this Strategy, each voter votes for his or her most preferred alternative. The alternative with the most votes wins. For example, A,B,C..J are 10 TV progarms and corresponding rating to each tv program for three different users(John, Adam, Mary) can be shown in Table 4.1. From this table we can say that Siva's first preference are the TV programs: A, E, I. For Krishna's first preferance of the TV programs are B, D, F ,H and for Rama's first preferance is the TV program A. Similarly second, third ..8th preferences are shown in Table 4.2. From these alternative preferences , group preferences will be generated, based on the maximum occurance of the TV program in user preferances. These group preferances can be seen in Table 4.2.

Table 3.2: Example to demonstrate social choice theory strategies

TV Programs	A	B	C	D	E	F	G	H	I	J
Siva	10	4	3	6	10	9	6	8	10	8
Krishna	1	9	8	9	7	9	6	9	3	8
Rama	10	5	2	7	9	8	5	6	7	6

Table 3.3: Plurality voting

	1	2	3	4	5	6	7	8
Siva	A,E,I	E,I	I	I	H,J	J	G	C
Krishna	B,D,F,H	B,D,F,H	B,D,F,H	B,D,H	B,H	B	B	C
Rama	A	E	F	D,I	H,J	J	B,G	C
Group	A	E	F	D	H	J	B	C

2. **Utilitarian Strategy:** In this strategy, instead of using ranking information, utility values are used. These utility values can be of two different types :additive or multiplicative. For example, Consider the users and items with corresponding ratings as shown in Table 4.3. The additive utility values for group has shown in tha same Table 4.3 and the multiplicative utility values are shown in Table 4.4. According to the additive utilitarian strategy E and F are the recommended programs because both having same highest utility value and according to the multiplicative ulilitarian strategy TV program F is the recommneded program because it has highest utility value.

Table 3.4: Additive Utilitarian Strategy

TV Programs	A	B	C	D	E	F	G	H	I	J
Siva	10	4	3	6	10	9	6	8	10	8
Krishna	1	9	8	9	7	9	6	9	3	8
Rama	10	5	2	7	9	8	5	6	7	6
Group	21	18	13	22	26	26	17	23	20	22

Table 3.5: Multiplicative Utilitarian Strategy

TV Programs	A	B	C	D	E	F	G	H	I	J
Siva	10	4	3	6	10	9	6	8	10	8
Krishna	1	9	8	9	7	9	6	9	3	8
Rama	10	5	2	7	9	8	5	6	7	6
Group	100	180	48	378	630	648	180	432	210	384

3. **Least Misery Strategy:** In this strategy, we make a group ratings based on

the minimum of the given individual ratings.Then the item with large minimum individual rating will be recommended. The idea behind this strategy is that, a group is as happy as its least happy member. For example, consider the users and items with corresponding ratings as shown in Table 4.5. Then, By least miserfy strategy, we will get the group ratings for each and every item as shown in that table.Here F has the maximum group rating. So TV program F is recommended by Least misery strategy.

Table 3.6: Least Misery Strategy

TV Programs	A	B	C	D	E	F	G	H	I	J
Siva	10	4	3	6	10	9	6	8	10	8
Krishna	1	9	8	9	7	9	6	9	3	8
Rama	10	5	2	7	9	8	5	6	7	6
Group	1	4	2	6	7	8	5	6	3	6

4. **Most Pleasure Strategy:** In this strategy, we make the group ratings list based on the maximum of individual ratings. For Example,Consider the users and items with corresponding ratings as shown in Table 4.6.Then, By most pleasure strategy, we will get the group ratings list as shown in that table. From that group ratings,A, E, I are having the highest group rating values. So either A, E and I will be the recommended TV programs.

Table 3.7: Most Pleasure Strategy

TV Programs	A	B	C	D	E	F	G	H	I	J
Siva	10	4	3	6	10	9	6	8	10	8
Krishna	1	9	8	9	7	9	6	9	3	8
Rama	10	5	2	7	9	8	5	6	7	6
Group	10	9	8	9	10	9	6	9	10	8

5. **Average without Misery strategy:** In this strategy,we make group ratigs list based on the average of the individual ratings,but without considering the items that score below a certain threshold for individuals. The item with maximum group rating value will be recommended. For example,Consider the

users and items with corresponding ratings as shown in Table 4.7 and take a threshold of 4. Then the items A, B, I are ignored because of having rating less than the thresold value and the TV program E and F are having highest utility values. So E and F are the recommended programs.

Table 3.8: Average without Misery Strategy

TV Programs	A	B	C	D	E	F	G	H	I	J
Siva	10	4	3	6	10	9	6	8	10	8
Krishna	1	9	8	9	7	9	6	9	3	8
Rama	10	5	2	7	9	8	5	6	7	6
Group	-	-	13	22	26	26	17	23	-	22

Chapter 4

Proposed Method to Group Recommender Systems

As mentioned earlier, our method for TV group recommender is developed based on decision list rule learner and social choice theory strategies. In our approach, each training example is a collection of 12 attribute-value pair. Those attributes are :Date, Day, Time, Channal, Program name, Category, Genre, Classification, Starring, Language, Duration, Rating.Here ,the rating is 5-scaled one 0,1,2,3,4. 0 indicates bad program, 1 indicates average program, 2 indicates above average program, 3 indicates a good program, 4 indicates an excellent program.

Initially, the system has no idea to recommend any TV programs except if we add some extenal rules. For few weeks, It collects ratings for each and every program the user has watched. These are the training examples to the learing algorithm. From these traing examples the learning algorithm learns the set of rules that cover all training examples. This process will be done for every user i.e.,for each user, we get a separate rule base (User profile). We know that TV guide contains the upcoming TV program information i.e., day ,date, time, channal etc. This TV guide i.e. the collection of TV programs will be classified by the each user's Rule base, Hence we get a predicted ratings for each TV progarm in the TV guide. These programs with predicted ratings are nothing but program recommnedation list for each individual user. Finally, social choice theory strategies are used to get a combined (group) recommnedation list. The complete process of our method is shown in Figure 4.1, There are N-users ,we will get feedback that is rating for the programs They have watched. These collected TV program with corresponding rating are shown as a

N-user viewing histories. DLRL (Decision List Rule learner) uses the collected user viewing history and learns the ordered rules called as RuleBase, for each user. Now the TV guide contains the instances (upcoming TV programs) that will be classified by the N-Rule bases. We will get N-Tv program recommendation lists. These program recommendation lists are merged by using social choice theory strategies to get a common group recommendation list.

In this method, we are collecting the ratings for each and every program the user watched. But some members are unlike to give ratings, therefore we need a method to generate the ratings for the programs in viewing history. I suggest one method is that for each and every program, calculate probabilty of the program in viewing history as the occurance of the similar programs in the veiwing history and based on that we can give ratings. If a similar kind of programs occur many times than we can give high rating for those programs and so on.

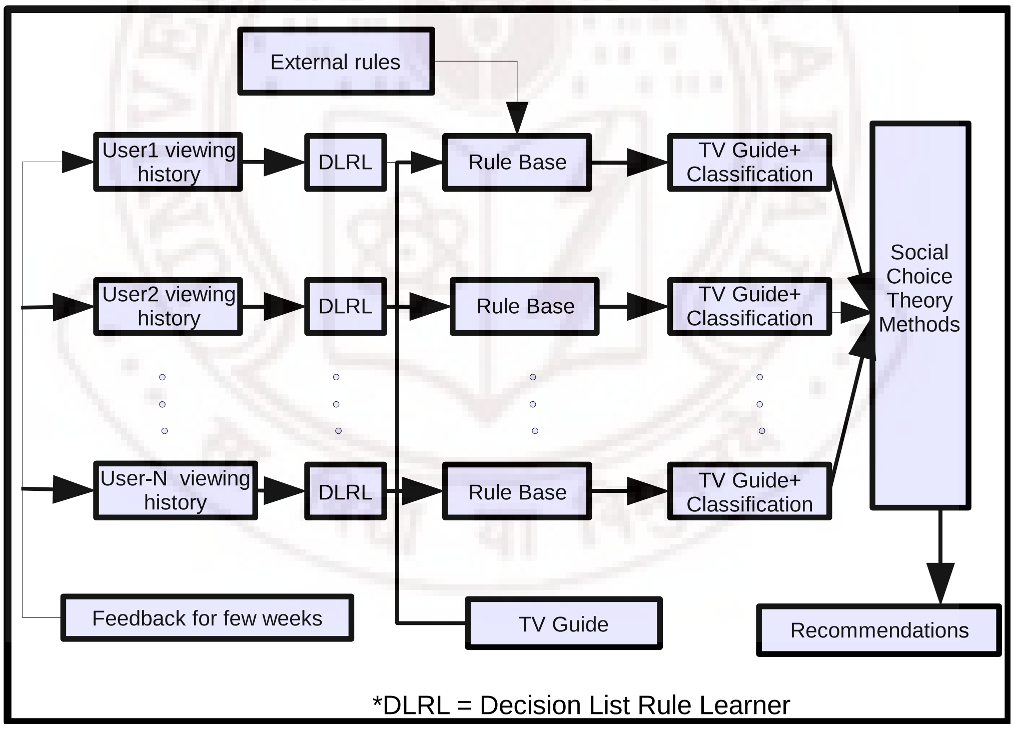


Figure 4.1: Proposed method to group recommender system

4.1 Learning Algorithm

Learning algorithm plays major role in content based recommendation approach. It is used to learn user profiles. Our learning algorithm is decision list rule learner based on RIPPER and FOIL rule learners.It is a multi-class rule learner.In our case, there are five classes : bad, average, above average, good, excellent. . Initially, All training examples are devided into two sets: train data and prune data. Train data is used to learn the set of rules. Prune data is used to prune the rules to avoid overfitting.FOIL Information gain is as follows:

$$FOIL\ Gain(L, R) = t(\log_2(\frac{p_1}{p_1 + n_1}) - \log_2(\frac{p_0}{p_0 + n_0})) \tag{4.1}$$

Where L is the candidate literal to add to rule R, p_0 is the number of positive bindings of R, n_0 is the number of negative bindings of R, p_1 is the number of positive bindings of R + L, n_1 is the number of negative bindings of R + L, t is the number of positive bindings of R also covered by R + L.

The pruning criteria(v) used here can be defined as

$$v = \frac{(p - n)}{(p + n)} \tag{4.2}$$

Where p is the number of positive examples covered by the rule in prune data set and n is the number of negative examples covered by the rule in the prune data set. Pruning criteria is deleting the final sequence of conditions that maximizes v. The steps in our algorithm as shown in **Algorithm-1**.

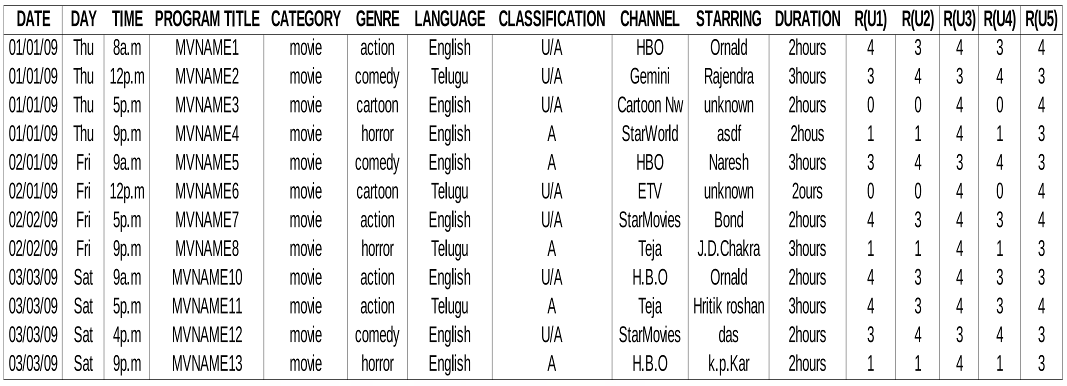
4.2 Example to Demonstrate the Proposed Method

To understand the working of the proposed method, Let us consider 12 TV proagrams (training examples) as shown in Figure 4.2 and five users namely User 1, User 2, User 3, User 4, User 5. User ratings of these TV programs are shown in Figure 4.2 under the colums R(U1) to R(U5). As we described earlier, each training example is a collection 11 attribute value pair. Here the ratings 0, 1, 2, 3, 4 correspond to bad, average, above average, good, excellent classes respectively.

To show the working of external rules, let us assume that User 1 and User 3 wants to watch programs only in Telugu and English. Other three users wants to watch the programs in Hindi also. These conditions can be added to the rule base as

Algorithm 1: Learning Algorithm used in proposed method

```
Input: Train Data,Prune Data
Output: set of rules
Step 1:  foreach class do
    Find the number of training examples for that class;
end
Take the class with maximum number of examples, make that as Default
class;
Step 2: Take an empty RuleSet;
while No class has left do
    take the next smallest class;
    Consider training examples for that class as positive, remaining as
    negative;
    while All positive examples covered do
        Take empty Rule;
        Add conjuncts to rule as soon as it improves FOIL Information gain;
        prune the rule by deleting any final sequence of conditions;
        Mark covered positive examples by this rule;
        Add this rule to RuleSet;
    end
end
Step 3:Add Default Rule to RuleSet;
Return RuleSet;
```



DATE	DAY	TIME	PROGRAM TITLE	CATEGORY	GENRE	LANGUAGE	CLASSIFICATION	CHANNEL	STARRING	DURATION	R(U1)	R(U2)	R(U3)	R(U4)	R(U5)
01/01/09	Thu	8a.m	MVNAME1	movie	action	English	U/A	HBO	Ornald	2hours	4	3	4	3	4
01/01/09	Thu	12p.m	MVNAME2	movie	comedy	Telugu	U/A	Gemini	Rajendra	3hours	3	4	3	4	3
01/01/09	Thu	5p.m	MVNAME3	movie	cartoon	English	U/A	Cartoon Nw	unknown	2hours	0	0	4	0	4
01/01/09	Thu	9p.m	MVNAME4	movie	horror	English	A	StarWorld	asdf	2hous	1	1	4	1	3
02/01/09	Fri	9a.m	MVNAME5	movie	comedy	English	A	HBO	Naresh	3hours	3	4	3	4	3
02/01/09	Fri	12p.m	MVNAME6	movie	cartoon	Telugu	U/A	ETV	unknown	2ours	0	0	4	0	4
02/02/09	Fri	5p.m	MVNAME7	movie	action	English	U/A	StarMovies	Bond	2hours	4	3	4	3	4
02/02/09	Fri	9p.m	MVNAME8	movie	horror	Telugu	A	Teja	J.D.Chakra	3hours	1	1	4	1	3
03/03/09	Sat	9a.m	MVNAME10	movie	action	English	U/A	H.B.O	Ornald	2hours	4	3	4	3	3
03/03/09	Sat	5p.m	MVNAME11	movie	action	Telugu	A	Teja	Hritik roshan	3hours	4	3	4	3	4
03/03/09	Sat	4p.m	MVNAME12	movie	comedy	English	U/A	StarMovies	das	2hours	3	4	3	4	3
03/03/09	Sat	9p.m	MVNAME13	movie	horror	English	A	H.B.O	k.p.Kar	2hours	1	1	4	1	3

Figure 4.2: Example to Demonstrate the Proposed Method

1. **bad**:-Language=¬telugu OR ¬english for user1 and user3.
2. **bad**:-Language=¬telugu OR ¬english OR ¬hindi for user2, user4 and user5.

DLRL (Decision List Rule Learner) orders the classes according to the class prelavance, that is the number of traing examples for that class. For each user the number of examples for different classes are shown in Table 4.1.

Table 4.1: DLRL Class ordering

	Bad	Average	Above Average	Good	Excellent
User 1	2	3	0	3	4
User 2	2	3	0	3	4
User 3	0	9	0	0	3
User 4	2	3	0	3	4
User 5	0	5	0	0	7

DLRL learns the rules from the traing examples for each user. Learnig strats from the least prelavance class to highest prelevance class. For example, in Table 4.1, user1 has two bad class examples which is having least class prelavance as compared to other classes, so rules for that class will be learned first. After that good, average class rules will be learned. The class with highest prelavance is taken as default class and it is add to the rule base as default rule meaning that if all above rules fails to classify a particular TV program then it belongs to the default class. Each rule looks like :

Class :- List of conditions.

meaning that

If (list of conditions) then Class.

The learned rules for all users as follows:

- **RULE BASE for User1:**

  External rules:

  Bad :- Language=¬Telugu OR ¬English

  Learned Rules:

  Bad :-CATEGORY=movie AND GENRE=cartoon AND CLASSIFICATION=U/A

  Good:-CATEGORY=movie AND GENRE=comedy

  Average:-TIME=9p.m AND CATEGORY=movie AND GENRE=horror

  Default Rule:Excellent:-()

- **RULE BASE for User2 and User 4:**

  External rules:

  Bad :- Language=¬Telugu OR ¬English OR ¬Hindi

  Learned Rules:

  Bad :- CATEGORY=movie AND GENRE=cartoon AND CLASSIFICATION=U/A

  Excellent: CATEGORY=movie AND GENRE=comedy

  Average:- TIME=9p.m AND CATEGORY=movie AND GENRE=horror

  Default Rule: good:-()

- **RULE BASE for User3:**

  External rules:

  Bad :- Language=¬Telugu OR ¬English

  Learned rules:

  Good :- CATEGORY=movie AND GENRE=comedy

  Default: Excellent:-()

- **RULE BASE for User5:**

External rules:

Bad :- Language=¬Telugu OR ¬English OR ¬Hindi

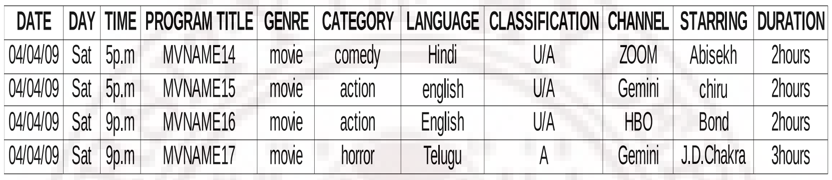
Learned Rules:

Excellent :- CATEGORY=movie AND GENRE=horror

Excellent :- TIME=5p.m AND CATEGORY=movie AND GENRE=cartoon

Default Rule: good:-()

Let us consider four upcoming TV programs shown in Figure 4.3 as new instances.



DATE	DAY	TIME	PROGRAM TITLE	GENRE	CATEGORY	LANGUAGE	CLASSIFICATION	CHANNEL	STARRING	DURATION
04/04/09	Sat	5p.m	MVNAME14	movie	comedy	Hindi	U/A	ZOOM	Abisekh	2hours
04/04/09	Sat	5p.m	MVNAME15	movie	action	english	U/A	Gemini	chiru	2hours
04/04/09	Sat	9p.m	MVNAME16	movie	action	English	U/A	HBO	Bond	2hours
04/04/09	Sat	9p.m	MVNAME17	movie	horror	Telugu	A	Gemini	J.D.Chakra	3hours

Figure 4.3: Example Instaces

From the above learned rules we get the classifications for these instances as shown in Table 4.2.

Table 4.2: Classification of Instances

	Instance 1	Instance 2	Instance 3	Instance 4
User 1	3	4	0	1
User 2	4	3	4	1
User 3	3	4	0	4
User 4	4	3	4	1
User 5	3	4	3	4

The working of social choice theory strategies to generate the group ratings are as follows:

- **Utilitarian strategy:** As we described in the previous chapter, by this method, group ratings are generated based on the utility values. These utility values can be taken as sum or multiplication of all the ratings for the particular instance. These utility values are shown in Table 4.3. From these values, we can say that instance 2 is the recommended TV program by this utilitarian strategy because it has highest utility value.

Table 4.3: Utilitarian strategy

	Instance 1	Instance 2	Instance 3	Instance 4
User 1	3	4	0	1
User 2	4	3	4	1
User 3	3	4	0	4
User 4	4	3	4	1
User 5	3	4	3	4
Utility(Addition)	17	18	5	11
Utility(Multiplication)	432	576	0	16

- **Least misery strategy:** As we described in the previous chapter, by this method, group ratings are generated based on least rating of the particular item. These least rating values are shown in Table 4.4. From these values, we can say that instace 1 and instance 2 are equally recommended TV programs by this strategy because these instances have 3 as the maximum value in the least rating row in Table 4.4.

Table 4.4: Least Misery Strategy

	Instance 1	Instance 2	Instance 3	Instance 4
User 1	3	4	0	1
User 2	4	3	4	1
User 3	3	4	0	4
User 4	4	3	4	1
User 5	3	4	3	4
Least Rating	3	3	0	1

- **Most pleasure strategy:** As we described in the previous chapter, by this method, group ratings are generated based on most rating of the particular item. These most rating values are shown in Table 4.5. From these values, we can say that all instances are equally recommended TV programs by this strategy because these instances are having 4 as the maximum value in that most rating row in Table 4.5.

Table 4.5: Most Pleasure Strategy

	Instance 1	Instance 2	Instance 3	Instance 4
User 1	3	4	0	1
User 2	4	3	4	1
User 3	3	4	0	4
User 4	4	3	4	1
User 5	3	4	3	4
Least Rating	4	4	4	4

- **Average without misery strategy:** As we described in the previous chapter, by this method, group ratings are generated based on the Average of all the ratings for a particular item and the items are considered if all the ratings shoul be more than the given thresold. Let us take thresold = 1. These average values are shown in Table 4.6 and the instance 3 and four are ignored because of having the ratings less than the assumed thresold. From these values, we can say that instance 2 is the recommended TV program by this strategy because it has highest average value.

Table 4.6: Average without Misery Strategy

	Instance 1	Instance 2	Instance 3	Instance 4
User 1	3	4	0	1
User 2	4	3	4	1
User 3	3	4	0	4
User 4	4	3	4	1
User 5	3	4	3	4
Average Rating	3.4	3.6	-	-

But, we observed that no method alone is sufficient to recommend the best TV progarm. Because In case of Utilitarian strategy, suppose let us take five users and two TV programs with ratings{4,3,2,4,3},{4,3,3,3,3}. By utilitarian strategy these two prorams are equally recommended becuse the sum of all ratings in both lists is equal, but in these two TV programs TV program2 is best, becuase for TV program2 group member's happy is atleat 3, where as for TV program1 it is 2. In case of least misery strategy, suppose take two users and five TV programs with ratings{4,3,3,4,3},{4,3,3,3,3}. These two TV programs are equally recommended by least misery strategy because in both cases least rating is 3. But we can say that first program is best if we consider whole group happiness. In case of most pleasure strategy, we consider only the single user's happiness which is not a suuggestable method. In case of Average without misery strategy, we ignore some TV programs based on thresold, in some situations we may have the TV programs which are average programs for all users.

4.3 RTL Strategy to Generate Group Ratings

As we pointed out above a single startegy alone wouldn't be sufficient to get the most accurate result as far as ruled based group recommenders are concerned. Therefore, we need a combined strategy that considers three factors: 1) Least group member happy (like least misery strategy) 2) Most group member happy (like most pleasure strategy) 3) Total group happy (like Utilitarian strategy). Hence we suggest a method based on these strategies. We name this strategy as RTL strategy (Repeat Total plus Least group happiness strategy). Let G be a group consists of N users. Here the method is to calculate the sum of least happiness and total happiness for each instances and calculate the maximum of this summed values and then recommened that instance. If we have maximum value for multiple instances than, take only that instaces, remove the minimum values (least happiness) from these instances and apply the same process for the new set of instances repeatedly. Hence the name Repeat Total plus Least group happiness strategy). The working proces of RTL strategy shown in Algorithm 2.

Let us take five users and two TV programs with ratings,
I_1={1,2,1,1,4}, I_2 = {1,2,2,2,2}.
C_1=9+1=10. C_2= 9+1=10, countNoOfRepeat=0, max =10.
Therefore, I_1 will be {2,4} and I_2 will be {2,2,2,2}

Algorithm 2: RTL strategy used in proposed method

```
Input: InstanceRatingVector(I), Number of users (N)
Output: Recommended Instances
Step 1: if all Instances in I has an user rating "0" then
    goto step 2;
end
else
    Remove the instances in I which has an user rating 0 ;
    goto step 2;
end
Step 2: countNoOfRepeat=0;
foreach instance i in I do
    $C_i$ = the sum of all user rating + minimum user rating for instance "i";
end
max = maximum value in $C_i$;
if countNoofRepeat = N then
    foreach Instance i in I do
        if max=$C_i$ then
            return i;
        end
    end
end
if max appears multiple times in $C_i$ and countNoofRepeat < N then
    foreach instance i in I do
        Remove the least rating entries for instance "i"
    end
    countNoOfRepeat++;
    repeat step2;
end
else
    return i
end
```

here, $C_1 = 6 + 2 = 8$ and $C_2 = 8 + 2 = 10$. Therefore ,Instance 2 will be recommended.

4.3.1 Weightage of a Group Member

Another intuition we had is to assign weights to group memebrs. We thought that in real life, weightage of group members needs to be considered in certain situations. For instance 1) In a family, channels are chosen based on the decision of the father or grandfather or some other senior member of that family 2) There might be an occasion like Birthday, Wedding day of members in a group. These members should be having more weightage on that day. To apply weightage to RTL strategy, just modify the InstanceRatingVector by two things:)For user who having the minimum rating for that instance devide his rating by his weightage 2)For other users, multiply user ratings with their weightage and apply the same Algorithm 2.

4.4 Clustering Based Approach to Group TV Recommender System

In this approach, All collected training examples are arranged into a number of clusters, each belonging to different category (movies, news, sports, drama, business etc,.). Each different cluster can be further devided into five number of sub-clusters. Each sub-cluster belonging to different kind of class (bad, average, above average, good, excellent). **Figure-4** shows the way of arrangement of the training examples into clusters. Cluster contains similar kind of examples. To measure similarity, euclidian distance has used. To get the classification for the instances(upcoming TV programs), find appropriate cluster (movie, sports, news, drama etc..)that the example belongs to and then claculate the similarity between the examples in sub-cluster of that cluster and the instance. The most smilar example will give the classification for the instance.

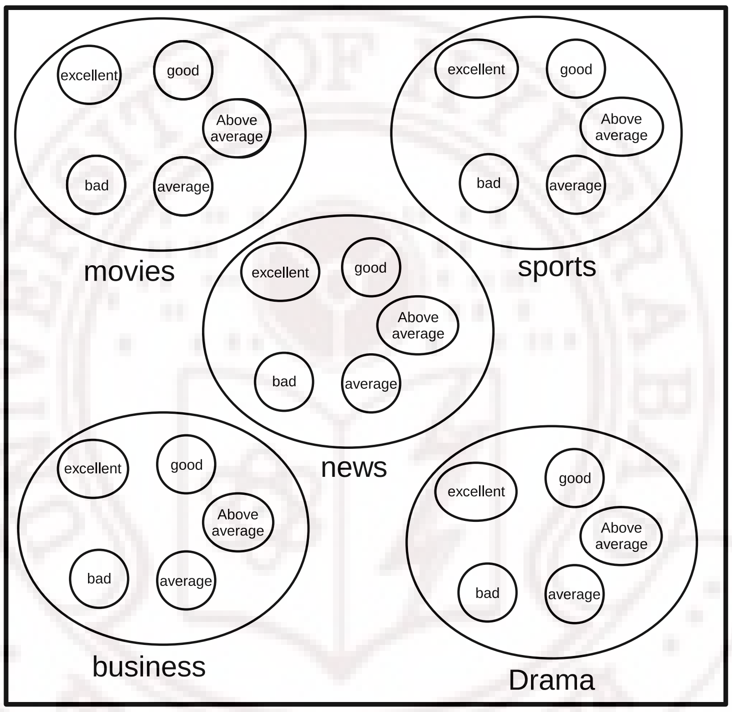


Figure 4.4: Cluster Based Approach to group TV recommender system

Chapter 5

Experimental Results

Unfortunately, we were not able to get any existing datasets for the group rec- ommender systems. Hence we created our own dierent datasets containing 46, 74, 93, 106 and 119 training examples for dierent TV programs. The number of literals learned for these datasets for Decision tree rule learner (C4.5), Clus- ter based approach, Decision list rule learner(DLRL: Our method) are shown in Figure 5.1. From this graph we can say that DLRL learns less number of literals (conditions) than the decision tree rule learner(C4.5) and cluster based approach that are sufficiant to classify the trianing data. Anyway, we know that cluster based approach requires to store all the examples in order to recommend the items, Hence the graph of this method is having high values . We noted the time required to learn the different datasets and this is shown in Figure 5.2 (No.of training examples and Time in seconds). From these experimental results, we can say that the performance of DLRL is almost linear to the training examples.

We run the C4.5 and our rule learner with the dataset containg 74 training examples. The screenshots of these two approaches are as shown in Figure 5.3 and Figure 5.4 respectively. Figure 5.3 shows that decision tree learned by the C4.5 rule learner and tree size is 22 letarals(conditions) with error 1.4% of the training dataset. Figure 5.4 shows that the decision list(set of ordered rules) learned by our approach. Here the size is 14 literas only and the error on training dataset is zero.

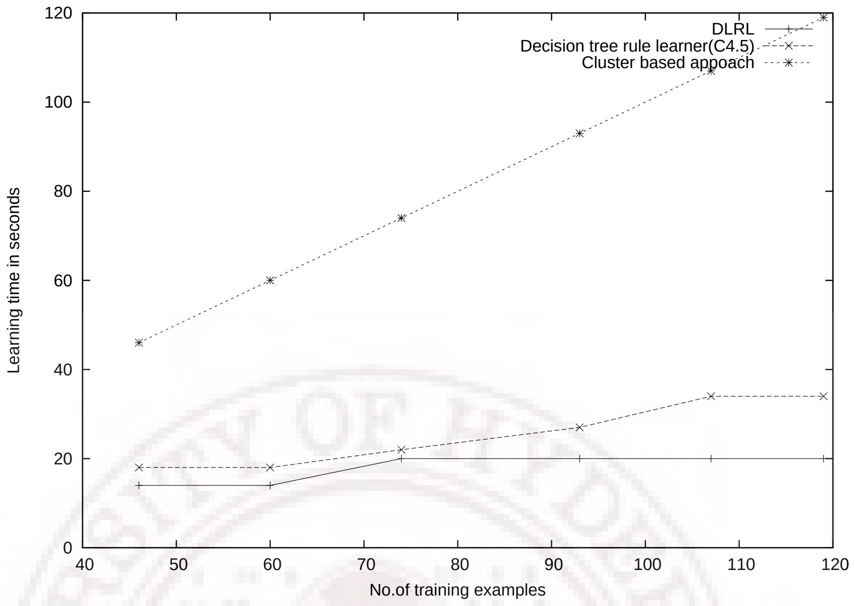


Figure 5.1: comparision between cluster based C4.5 and DLRL

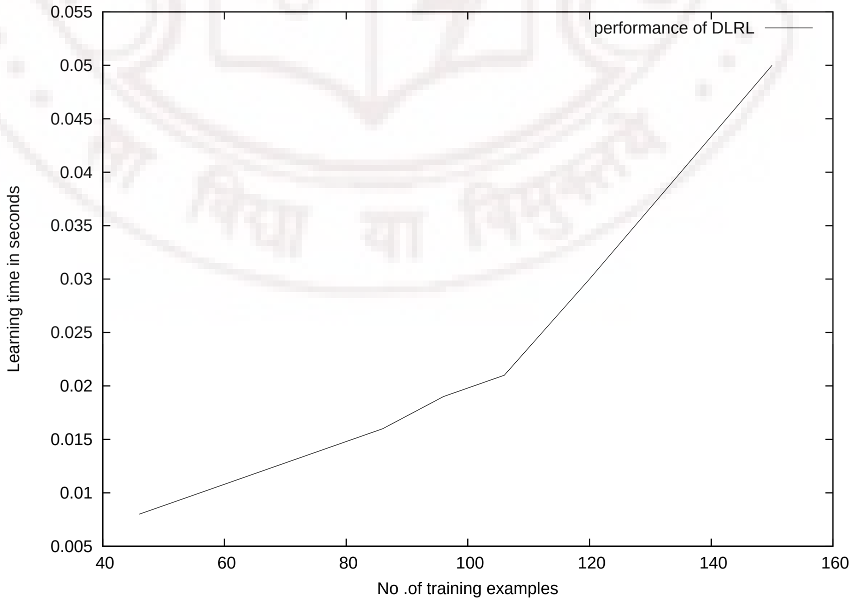
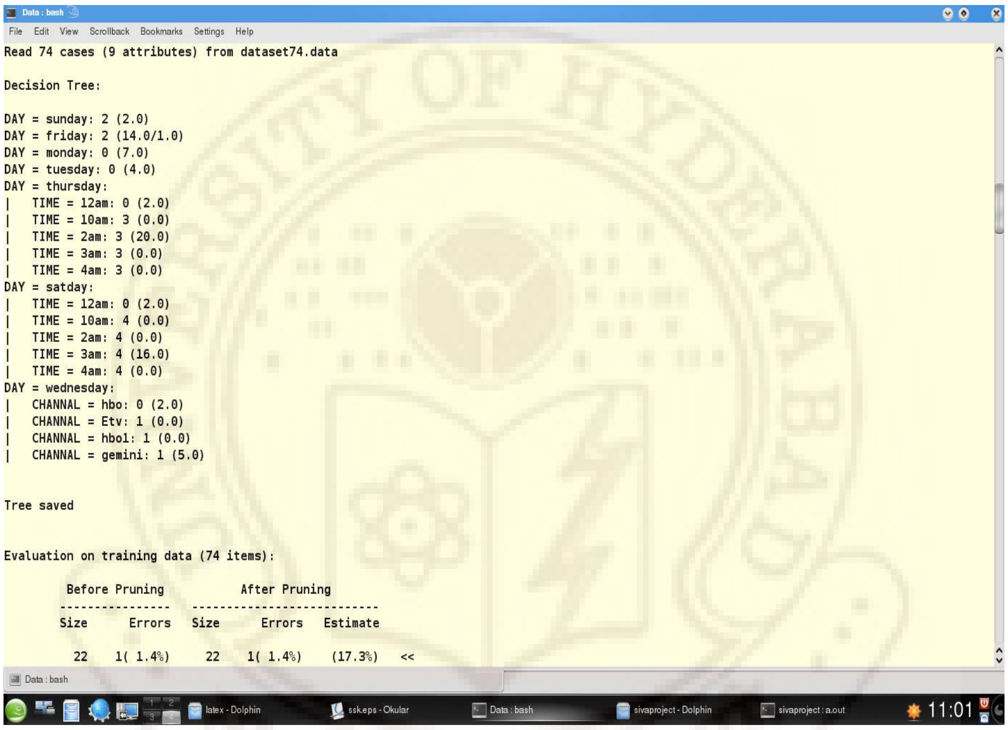
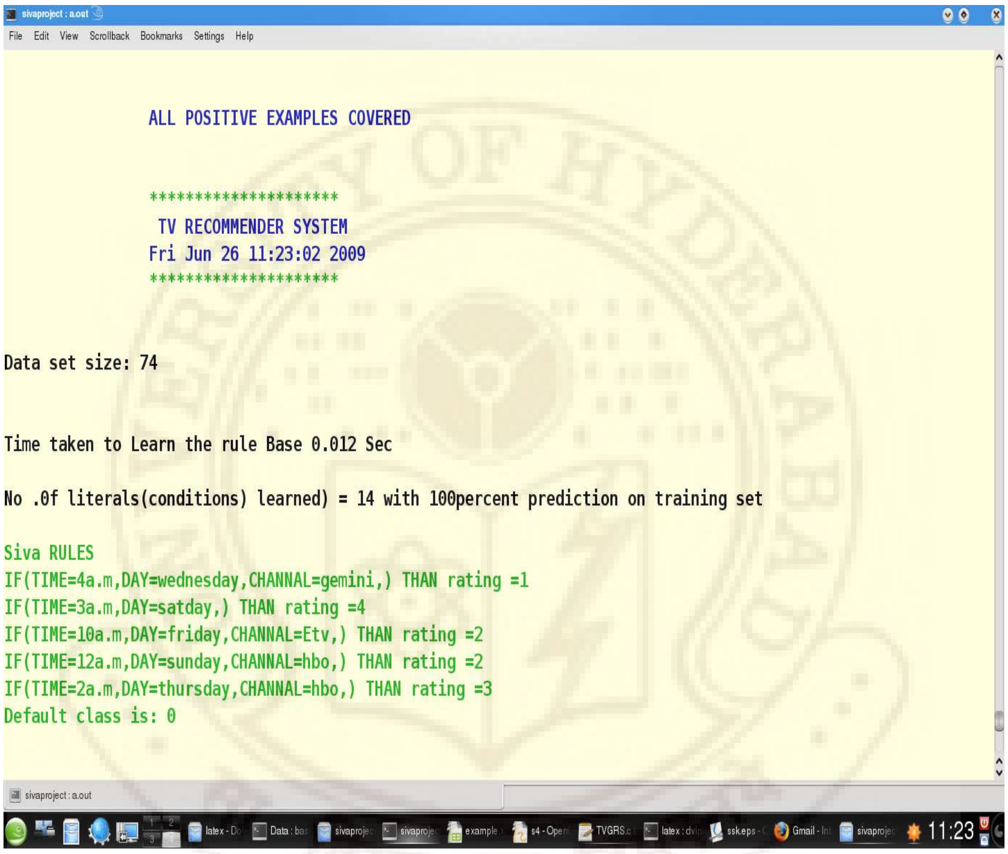


Figure 5.2: performance of Decision List Rule Learner



```
Read 74 cases (9 attributes) from dataset74.data

Decision Tree:

DAY = sunday: 2 (2.0)
DAY = friday: 2 (14.0/1.0)
DAY = monday: 0 (7.0)
DAY = tuesday: 0 (4.0)
DAY = thursday:
|   TIME = 12am: 0 (2.0)
|   TIME = 10am: 3 (0.0)
|   TIME = 2am: 3 (20.0)
|   TIME = 3am: 3 (0.0)
|   TIME = 4am: 3 (0.0)
DAY = satday:
|   TIME = 12am: 0 (2.0)
|   TIME = 10am: 4 (0.0)
|   TIME = 2am: 4 (0.0)
|   TIME = 3am: 4 (16.0)
|   TIME = 4am: 4 (0.0)
DAY = wednesday:
|   CHANNAL = hbo: 0 (2.0)
|   CHANNAL = Etv: 1 (0.0)
|   CHANNAL = hbol: 1 (0.0)
|   CHANNAL = gemini: 1 (5.0)


Tree saved


Evaluation on training data (74 items):

         Before Pruning           After Pruning
        ----------------   ---------------------------
        Size      Errors   Size      Errors   Estimate

          22    1( 1.4%)     22    1( 1.4%)    (17.3%)   <<
```

Figure 5.3: Screenshot of C4.5 Rule learner



```
                ALL POSITIVE EXAMPLES COVERED


                *********************
                 TV RECOMMENDER SYSTEM
                Fri Jun 26 11:23:02 2009
                *********************


Data set size: 74


Time taken to Learn the rule Base 0.012 Sec

No .0f literals(conditions) learned) = 14 with 100percent prediction on training set

Siva RULES
IF(TIME=4a.m,DAY=wednesday,CHANNAL=gemini,) THAN rating =1
IF(TIME=3a.m,DAY=satday,) THAN rating =4
IF(TIME=10a.m,DAY=friday,CHANNAL=Etv,) THAN rating =2
IF(TIME=12a.m,DAY=sunday,CHANNAL=hbo,) THAN rating =2
IF(TIME=2a.m,DAY=thursday,CHANNAL=hbo,) THAN rating =3
Default class is: 0
```

Figure 5.4: Screenshot of DLRL Rule learner

Conclusions

In this work, we developed a content based group recommender system for TV shows based on decision list rule learner (DLRL) and a method called RTL strategy to generate group ratings based on social choice theory strategies. We compare our learning algorithm and the existing c4.5 rule learner. Results shows that our rule learner performnace is better in terms of literals learned(size of the rule set) and our rule learner takes time that is linear to the number of training examples. We propose a new method (RTL strategy) to generate group ratings that is better than the existing social choice theory strategy. We also concerned the weightage of a group member and it is used in our approach.

Bibliography

[1] http://movielens.umn.edu/login.

[2] http://www.whatshouldireadnext.com/.

[3] http://www.last.fm/.

[4] http://www.stumbleupon.com/.

[5] http://www.mystrands.com/download.

[6] http://www.cleverset.com/solution/.

[7] http://www.whattorent.com/userinfo.php.

[8] http://www.netflix.com/.

[9] http://www.pandora.com/.

[10] Yen-Liang Chen, Li-Chen Cheng, and Ching-Nan Chuang. A group recommendation system with consideration of interactions among group members. *Expert Syst. Appl.*, 34(3):2082–2090, 2008.

[11] William W. Cohen. Fast effective rule induction. In *ICML*, pages 115–123, 1995.

[12] Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, and Miguel A. Rueda-Morales. Group recommending: A methodological approach based on bayesian networks. In *ICDE Workshops*, pages 835–844, 2007.

[13] Johannes Fürnkranz. Separate-and-conquer rule learning. *Artif. Intell. Rev.*, 13(1):3–54, 1999.

[14] Srinivas Gutta, Kaushal Kurapati, K. P. Lee, Jacquelyn Martino, John Milanski, J. David Schaffer, and John Zimmerman. Tv content recommender system. In *AAAI/IAAI, July 30 - August 3, 2000, Austin, USA*, pages 1121–1122, 2000.

[15] Judith Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Model. User-Adapt. Interact.*, 14(1):37–85, 2004.

[16] Tom M. Mitchell. *Machine Learning.* McGraw Hill, USA, 1997.

[17] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2007.

[18] J. R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[19] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[20] R. Tubio, R. Sotelo, Y. Blanco, M. Lopez, A. Gil, J. Pazos, and M. Ramos. A tv-anytime metadata approach to tv program recommendation for groups. In *Consumer Electronics, 2008. ISCE 2008. IEEE International Symposium on*, pages 1–3, April 2008.

[21] Zhiwen Yu, Xingshe Zhou, Yanbin Hao, and Jianhua Gu. Tv program recommendation for multiple viewers based on user profile merging. *User Model. User-Adapt. Interact.*, 16(1):63–82, 2006.

Appendix A

Implemention and screenshots

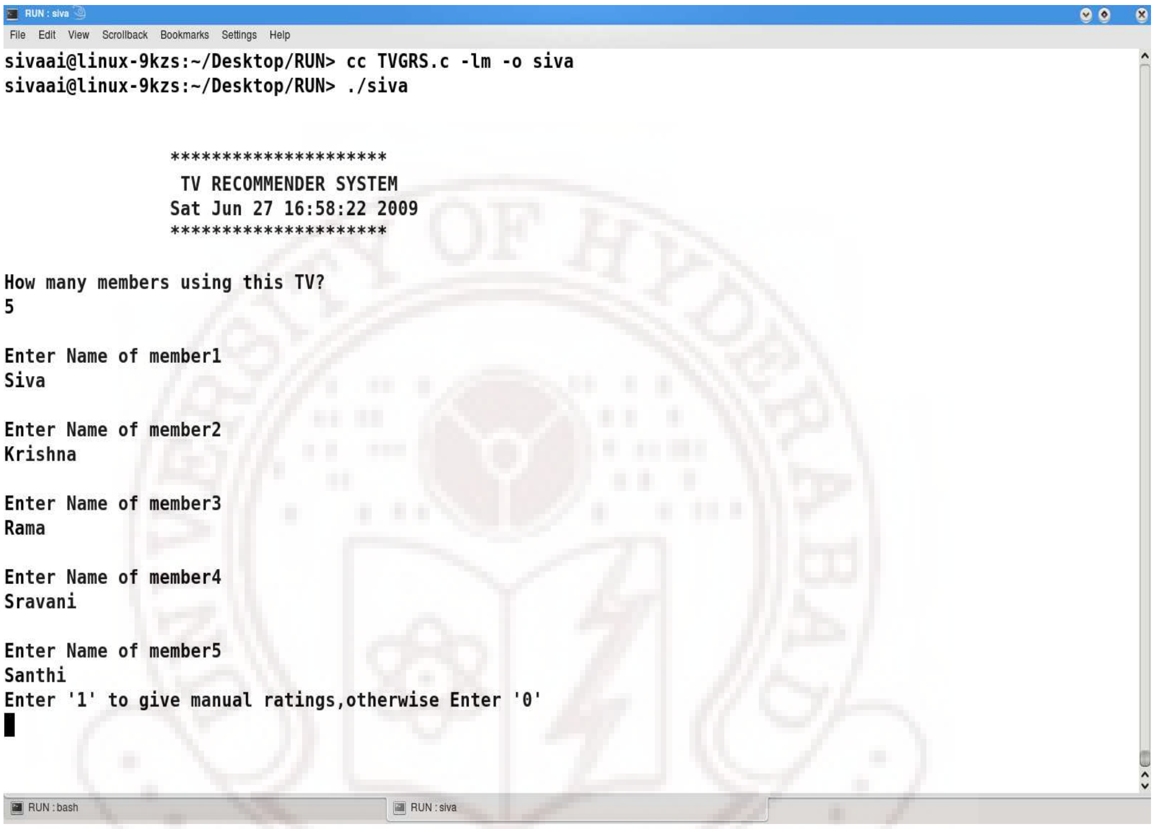
The project has been implemented in C language. It requires three input files:

1. **example.txt:** This file contains the training examples. Each training example is a collection of 11-attribute value pair. These attributes as follows: Date, Time, Day, Channal Name, Program Name, Category(movie, drama, sports...), Classification(A,U/A), Starring, Language, Genre, and Duration. These 11-attributed values should separated by space.

2. **ratings.txt:** This file contains the ratings of training examples for various users. First row in the file should contain the ratings of all users separated by space for the first example,similarly second row contains the ratings of all users separated by space for the second example and so on.

3. **instances.txt:** This file contains the upcoming TV programs or the TV programs that need to classify. These are in the same format as the training example(i.e,.11-attribute value pair).

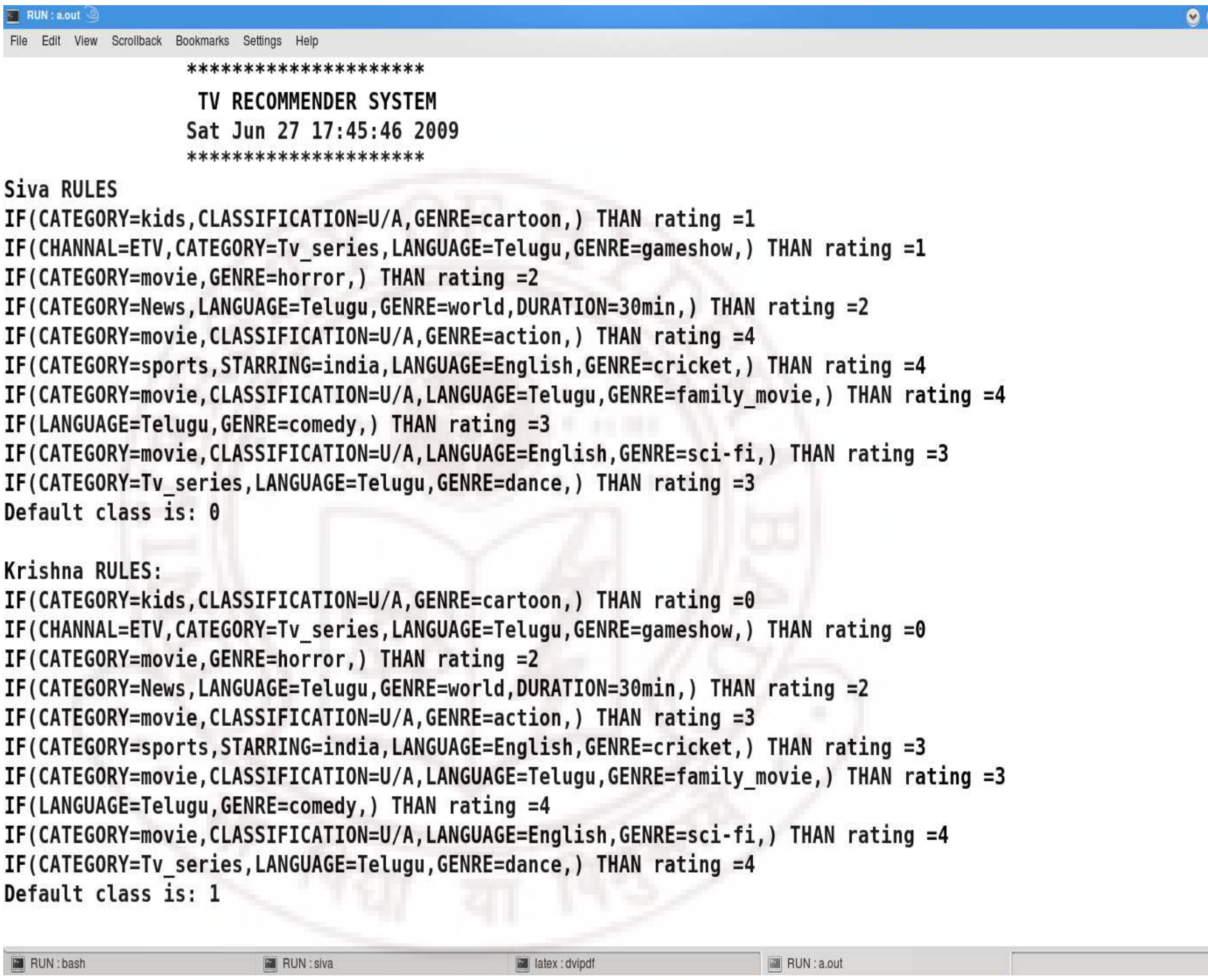
If we compile and run the program using command **cc TVGRS.c -lm and ./a.out**, then screen like Figure A.1. It asks the number of users and their names. After that the users can give manual ratings for each TV program in the example.txt file (if press option 1), otherwise (if press option 0) it will take the ratings from the file "rating.txt".



```
sivaai@linux-9kzs:~/Desktop/RUN> cc TVGRS.c -lm -o siva
sivaai@linux-9kzs:~/Desktop/RUN> ./siva


                ********************
                 TV RECOMMENDER SYSTEM
                Sat Jun 27 16:58:22 2009
                ********************

How many members using this TV?
5

Enter Name of member1
Siva

Enter Name of member2
Krishna

Enter Name of member3
Rama

Enter Name of member4
Sravani

Enter Name of member5
Santhi
Enter '1' to give manual ratings,otherwise Enter '0'
█
```

Figure A.1: Screen1

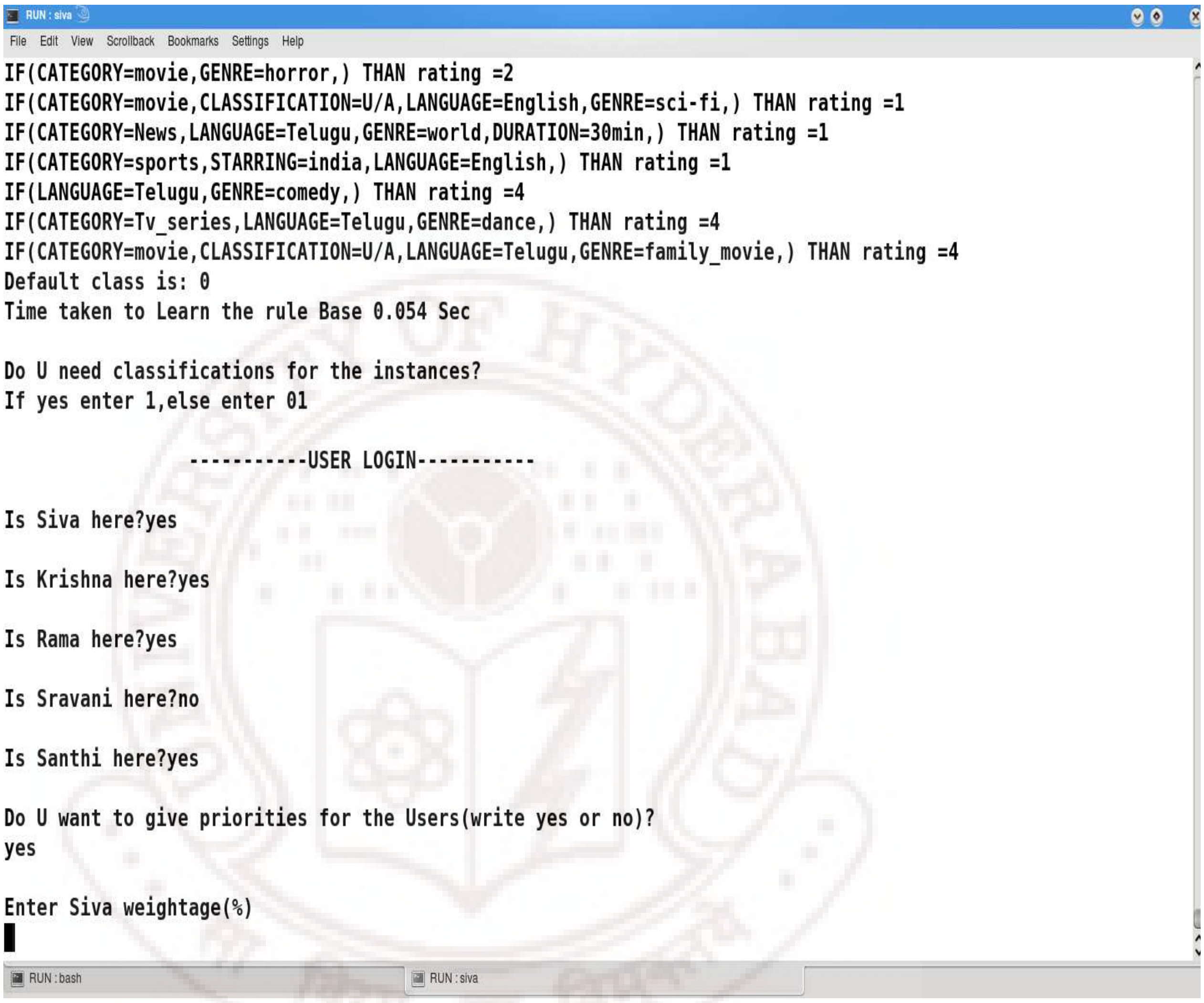
Figure A.2 shows the learned rules for the two users. Here 150 training examples used. From the siva rulebase in the Figure A.2 we can say that siva's excellent(rating 4) TV program are action movies, family movies in Telugu, and sport cricket. Similarly we can see the rules for rating 1, 2, 3 and the default classification is rating 0.



```
                    *********************
                     TV RECOMMENDER SYSTEM
                    Sat Jun 27 17:45:46 2009
                    *********************
Siva RULES
IF(CATEGORY=kids,CLASSIFICATION=U/A,GENRE=cartoon,) THAN rating =1
IF(CHANNAL=ETV,CATEGORY=Tv_series,LANGUAGE=Telugu,GENRE=gameshow,) THAN rating =1
IF(CATEGORY=movie,GENRE=horror,) THAN rating =2
IF(CATEGORY=News,LANGUAGE=Telugu,GENRE=world,DURATION=30min,) THAN rating =2
IF(CATEGORY=movie,CLASSIFICATION=U/A,GENRE=action,) THAN rating =4
IF(CATEGORY=sports,STARRING=india,LANGUAGE=English,GENRE=cricket,) THAN rating =4
IF(CATEGORY=movie,CLASSIFICATION=U/A,LANGUAGE=Telugu,GENRE=family_movie,) THAN rating =4
IF(LANGUAGE=Telugu,GENRE=comedy,) THAN rating =3
IF(CATEGORY=movie,CLASSIFICATION=U/A,LANGUAGE=English,GENRE=sci-fi,) THAN rating =3
IF(CATEGORY=Tv_series,LANGUAGE=Telugu,GENRE=dance,) THAN rating =3
Default class is: 0

Krishna RULES:
IF(CATEGORY=kids,CLASSIFICATION=U/A,GENRE=cartoon,) THAN rating =0
IF(CHANNAL=ETV,CATEGORY=Tv_series,LANGUAGE=Telugu,GENRE=gameshow,) THAN rating =0
IF(CATEGORY=movie,GENRE=horror,) THAN rating =2
IF(CATEGORY=News,LANGUAGE=Telugu,GENRE=world,DURATION=30min,) THAN rating =2
IF(CATEGORY=movie,CLASSIFICATION=U/A,GENRE=action,) THAN rating =3
IF(CATEGORY=sports,STARRING=india,LANGUAGE=English,GENRE=cricket,) THAN rating =3
IF(CATEGORY=movie,CLASSIFICATION=U/A,LANGUAGE=Telugu,GENRE=family_movie,) THAN rating =3
IF(LANGUAGE=Telugu,GENRE=comedy,) THAN rating =4
IF(CATEGORY=movie,CLASSIFICATION=U/A,LANGUAGE=English,GENRE=sci-fi,) THAN rating =4
IF(CATEGORY=Tv_series,LANGUAGE=Telugu,GENRE=dance,) THAN rating =4
Default class is: 1
```

Figure A.2: screen2

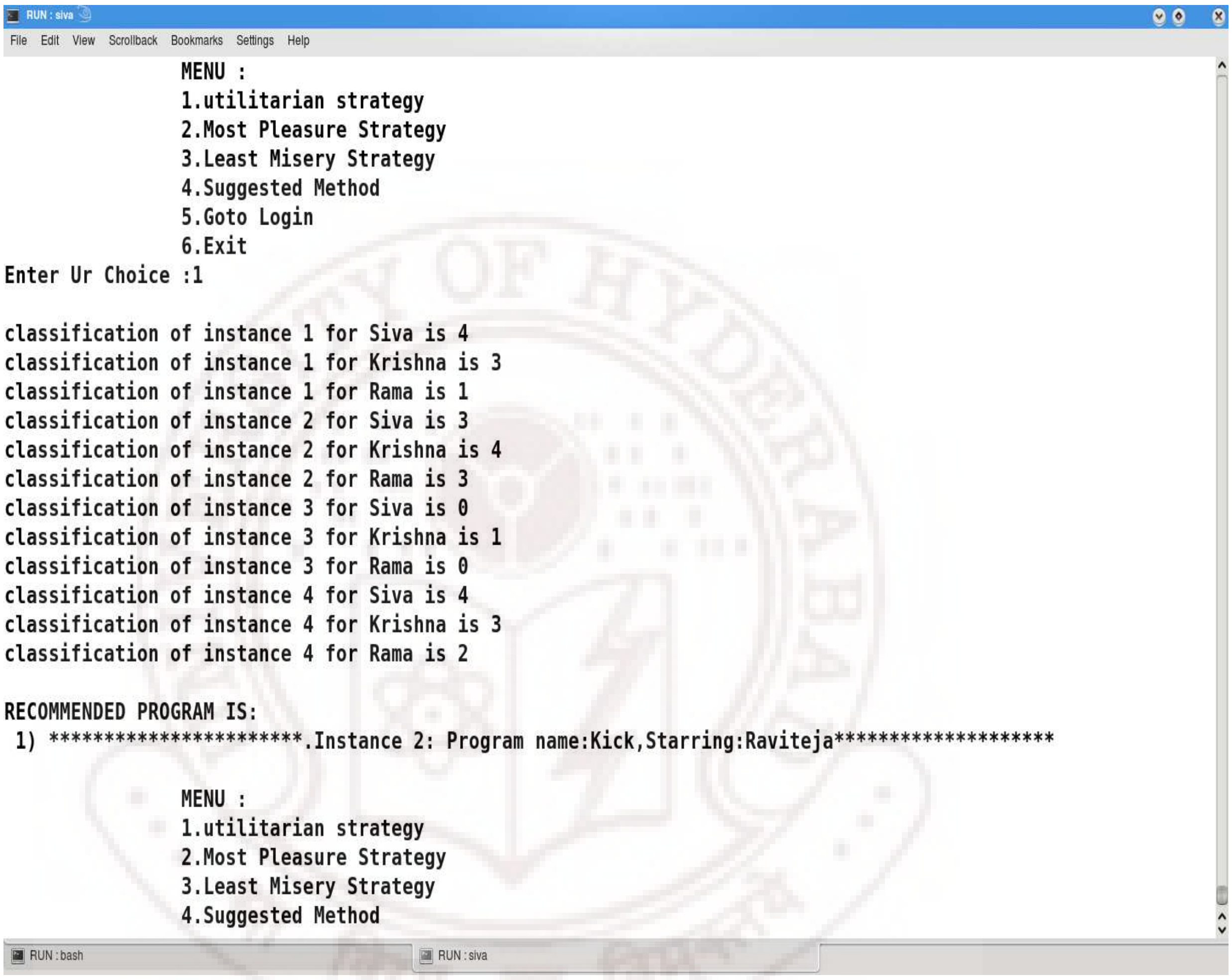
Figure A.3 shows the time taken to learn the rulebase 0.064sec and it also shows the login page. Here we give yes if the group member present, no if the group member absent.



```
IF(CATEGORY=movie,GENRE=horror,) THAN rating =2
IF(CATEGORY=movie,CLASSIFICATION=U/A,LANGUAGE=English,GENRE=sci-fi,) THAN rating =1
IF(CATEGORY=News,LANGUAGE=Telugu,GENRE=world,DURATION=30min,) THAN rating =1
IF(CATEGORY=sports,STARRING=india,LANGUAGE=English,) THAN rating =1
IF(LANGUAGE=Telugu,GENRE=comedy,) THAN rating =4
IF(CATEGORY=Tv_series,LANGUAGE=Telugu,GENRE=dance,) THAN rating =4
IF(CATEGORY=movie,CLASSIFICATION=U/A,LANGUAGE=Telugu,GENRE=family_movie,) THAN rating =4
Default class is: 0
Time taken to Learn the rule Base 0.054 Sec

Do U need classifications for the instances?
If yes enter 1,else enter 01

                 -----------USER LOGIN-----------

Is Siva here?yes

Is Krishna here?yes

Is Rama here?yes

Is Sravani here?no

Is Santhi here?yes

Do U want to give priorities for the Users(write yes or no)?
yes

Enter Siva weightage(%)
```

Figure A.3: screen3

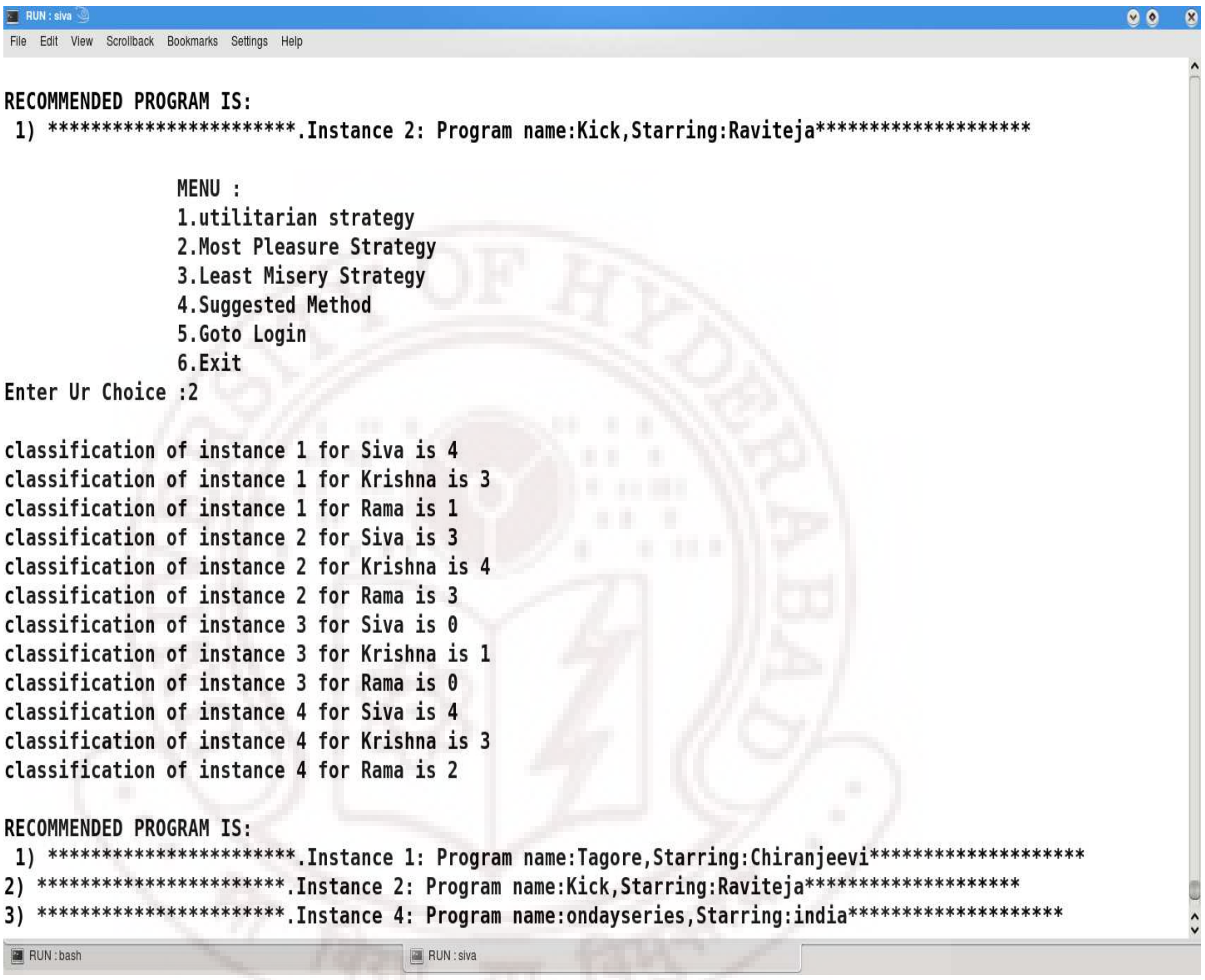
Figure A.4 shows the usage of utilitarian strategy to generate the group recommendation. Here three users and four instances are taken and the classification for these instances also can see in that figure. Here instance 2 is recommended because it has maximum utility value (10 = 4+3+3) than other instances.



```
                MENU :
                1.utilitarian strategy
                2.Most Pleasure Strategy
                3.Least Misery Strategy
                4.Suggested Method
                5.Goto Login
                6.Exit
Enter Ur Choice :1

classification of instance 1 for Siva is 4
classification of instance 1 for Krishna is 3
classification of instance 1 for Rama is 1
classification of instance 2 for Siva is 3
classification of instance 2 for Krishna is 4
classification of instance 2 for Rama is 3
classification of instance 3 for Siva is 0
classification of instance 3 for Krishna is 1
classification of instance 3 for Rama is 0
classification of instance 4 for Siva is 4
classification of instance 4 for Krishna is 3
classification of instance 4 for Rama is 2

RECOMMENDED PROGRAM IS:
 1) ***********************.Instance 2: Program name:Kick,Starring:Raviteja*******************

                MENU :
                1.utilitarian strategy
                2.Most Pleasure Strategy
                3.Least Misery Strategy
                4.Suggested Method
```

Figure A.4: screen4

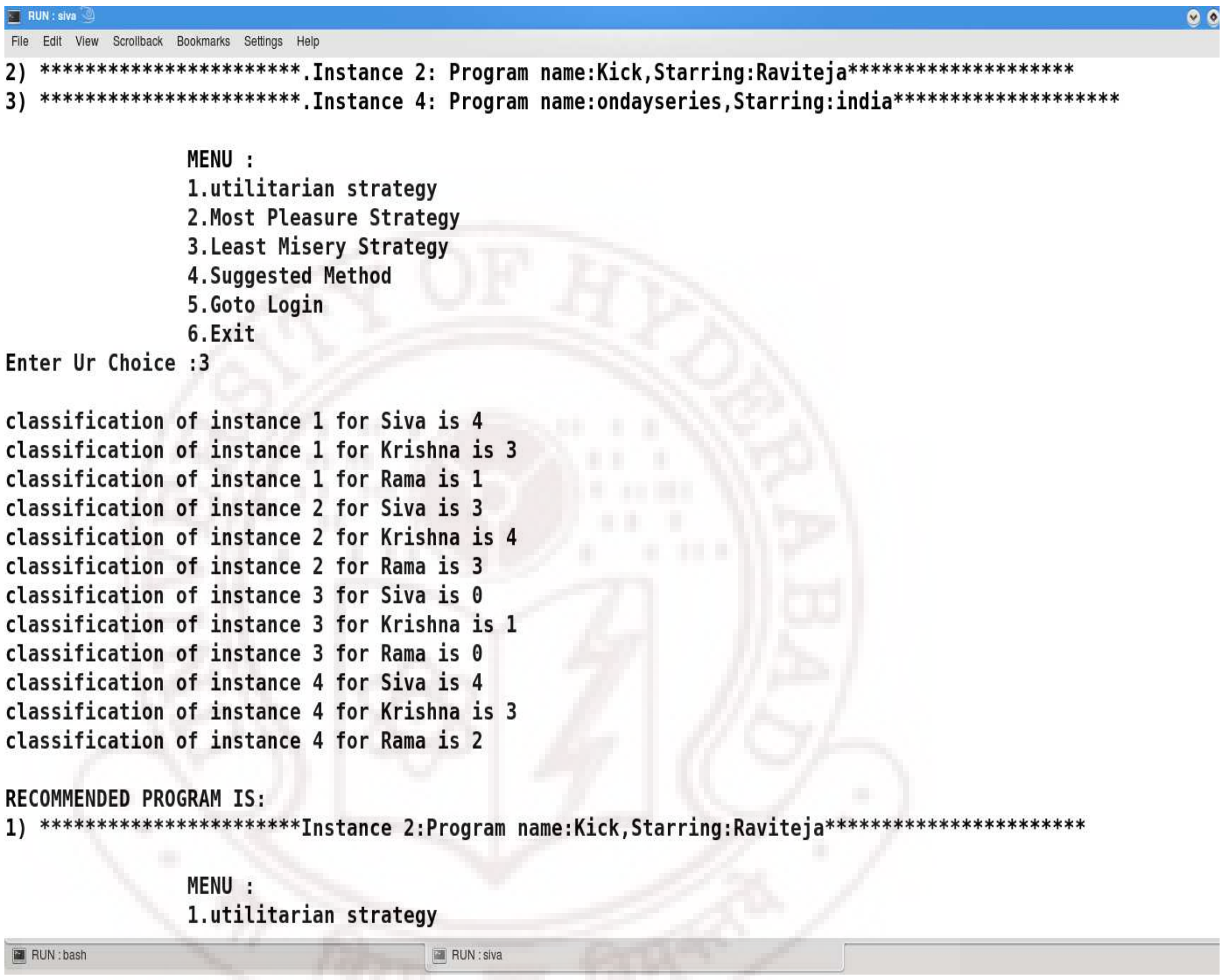
Figure A.5 shows the usage of most pleasure strategy to generate the group recommendation. Here three users and four instances are taken and the classification for these instances also can see in that figure. Here instance 1,2,4 are recommended because of having the maximum rating value (4) than instance 3.



```
RECOMMENDED PROGRAM IS:
 1) **********************.Instance 2: Program name:Kick,Starring:Raviteja*******************

                MENU :
                1.utilitarian strategy
                2.Most Pleasure Strategy
                3.Least Misery Strategy
                4.Suggested Method
                5.Goto Login
                6.Exit
Enter Ur Choice :2

classification of instance 1 for Siva is 4
classification of instance 1 for Krishna is 3
classification of instance 1 for Rama is 1
classification of instance 2 for Siva is 3
classification of instance 2 for Krishna is 4
classification of instance 2 for Rama is 3
classification of instance 3 for Siva is 0
classification of instance 3 for Krishna is 1
classification of instance 3 for Rama is 0
classification of instance 4 for Siva is 4
classification of instance 4 for Krishna is 3
classification of instance 4 for Rama is 2

RECOMMENDED PROGRAM IS:
 1) **********************.Instance 1: Program name:Tagore,Starring:Chiranjeevi*******************
2) **********************.Instance 2: Program name:Kick,Starring:Raviteja*******************
3) **********************.Instance 4: Program name:ondayseries,Starring:india*******************
```

Figure A.5: screen5

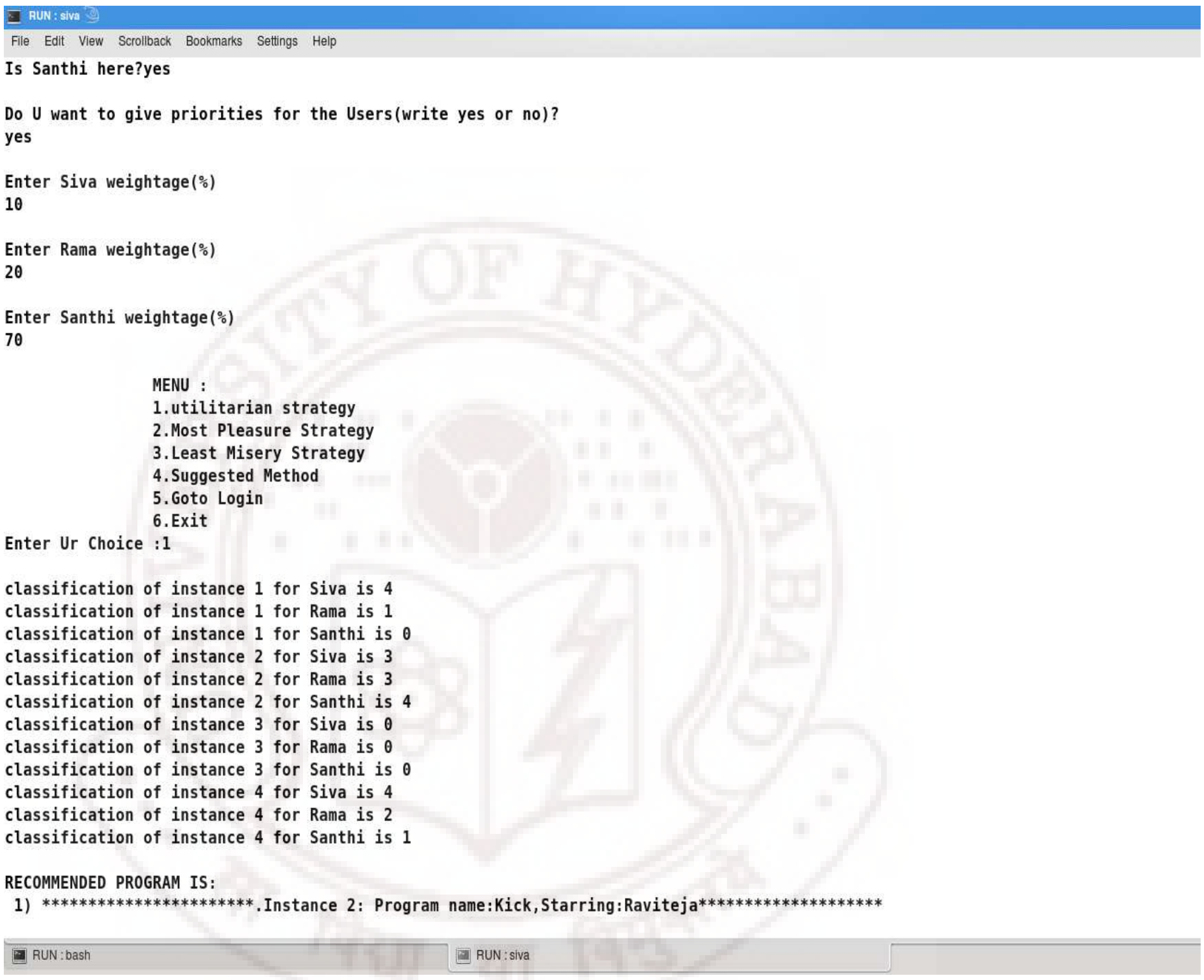
Figure A.6 shows the usage of least misery strategy to generate the group recommendation. Here three users and four instances are taken and the classification for these instances also can see in that figure. Here instance 2 is recommended because of having the maximum of least ratings (3 in{1,3,0,2}) than other instances.



```
2) ***********************.Instance 2: Program name:Kick,Starring:Raviteja*******************
3) ***********************.Instance 4: Program name:ondayseries,Starring:india*******************

                MENU :
                1.utilitarian strategy
                2.Most Pleasure Strategy
                3.Least Misery Strategy
                4.Suggested Method
                5.Goto Login
                6.Exit
Enter Ur Choice :3

classification of instance 1 for Siva is 4
classification of instance 1 for Krishna is 3
classification of instance 1 for Rama is 1
classification of instance 2 for Siva is 3
classification of instance 2 for Krishna is 4
classification of instance 2 for Rama is 3
classification of instance 3 for Siva is 0
classification of instance 3 for Krishna is 1
classification of instance 3 for Rama is 0
classification of instance 4 for Siva is 4
classification of instance 4 for Krishna is 3
classification of instance 4 for Rama is 2

RECOMMENDED PROGRAM IS:
1) **********************Instance 2:Program name:Kick,Starring:Raviteja**********************

                MENU :
                1.utilitarian strategy
```

Figure A.6: screen6

Figure A.7 shows the usage of weightage of group members. Here siva has 10%, rama has 20% and santhi has 70% weightage. Here santhi has high weightage, Hence the recommended program should be of her own interest. Therefore instance 2 is recommended.



```
Is Santhi here?yes

Do U want to give priorities for the Users(write yes or no)?
yes

Enter Siva weightage(%)
10

Enter Rama weightage(%)
20

Enter Santhi weightage(%)
70

                MENU :
                1.utilitarian strategy
                2.Most Pleasure Strategy
                3.Least Misery Strategy
                4.Suggested Method
                5.Goto Login
                6.Exit
Enter Ur Choice :1

classification of instance 1 for Siva is 4
classification of instance 1 for Rama is 1
classification of instance 1 for Santhi is 0
classification of instance 2 for Siva is 3
classification of instance 2 for Rama is 3
classification of instance 2 for Santhi is 4
classification of instance 3 for Siva is 0
classification of instance 3 for Rama is 0
classification of instance 3 for Santhi is 0
classification of instance 4 for Siva is 4
classification of instance 4 for Rama is 2
classification of instance 4 for Santhi is 1

RECOMMENDED PROGRAM IS:
 1) **********************.Instance 2: Program name:Kick,Starring:Raviteja*******************
```

Figure A.7: screen7

TV'03: Personalization in Future TV

Proceedings of the UM 2003 Workshop on Personalization in Future TV

Pittsburgh, PA, June 23rd, 2003

Selected papers

Liliana Ardissono and Mark Maybury (Eds.)

Volume Editors

Liliana Ardissono
Dipartimento di Informatica
Università di Torino
Corso Svizzera 185
I-10149 Torino
Italy

Mark Maybury
Information Technology Division
The MITRE Corporation
MS K312
202 Burlington Road
Bedford, MA, USA
maybury@mitre.org

Table of Contents

Preface

The 3rd workshop on Personalization in Future TV follows TV'01, held in association with the 8th International Conference on User Modeling (Sonthofen, 2001), and TV'02, organized in conjunction with the 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (Malaga, 2002). Both TV events have been organized as forums in which researchers from diverse areas such as machine learning, knowledge engineering, cognitive sciences, adaptive user interfaces, and business intelligence could share their experiences in the design, development and exploitation of user interfaces for future TV services. After the success of such workshops, which attracted the attention of academic and industrial researchers and provided an excellent overview of the current international work in the area of digital TV, we organized the third edition of such event. TV'03 will be held in Pittsburgh in conjunction with the 9th International Conference on User Modeling.

The present volume includes a selection of the papers submitted to TV'03, which will be presented during the workshop in Pittsburgh. The papers can be classified in four main areas:

- Electronic Program Guides (EPGs) and recommendation techniques
- Design aspects in EPG development and guidelines for the management of the interaction with the TV viewer
- Viewing behavior and content personalization in Interactive TV
- Social aspects of iTV

We hope that the workshop represents, as in the previous editions, a great opportunity for disseminating information about the most recent research in iTV and personalization. For instance, the most relevant outcome of the TV'01 and TV'02 events is the forthcoming *Special Issue on User Modeling and Personalization for Television of the International Journal on User Modeling and User-Adapted Interaction*, published by Kluwer Academic Publishers. We welcome your feedback on this glimpse into the future of television.

Liliana Ardissono and Mark Maybury

A Zooming, Electronic Programming Interface

Pete Tinker, Jason Fox, Mike Daily

HRL Laboratories, LLC
3011 Malibu Canyon Road
Malibu, CA 90265 USA
patinker@hrl.com, jrfox@hrl.com, mjdaily@hrl.com

Abstract. We describe an Electronic Programming Guide (EPG) that uses interactive panning and zooming, gestures, and spoken dialog to enable viewers to access relevant information in large hyperlinked databases. The guide is imbued with a "personality" that gives the impression to the viewer of interacting with an intelligent, opinionated agent.

Introduction

Most Electronic Programming Guides (EPGs) are based on a two-dimensional grid representation of textual program schedules, typically laid out chronologically [1,2]. With the rapid expansion of available content, this viewer interface approach is inadequate to find programming that a viewer desires to watch.

The Zooming, Electronic Programming Interface (ZEPI) is a device interface abstraction that allows a viewer to interact with devices on three principles: speech, gesture, and organization. Our embodiment of ZEPI is an EPG that uses a *zooming* visual interface distinct from typical EPG grid-based interfaces. It features smooth transitions from one view to another and compensates for low screen resolutions of NTSC broadcast television sets. The speech component of ZEPI makes the system flexible to use, and makes presentation of information personable. ZEPI can carry on a running context sensitive dialog with a viewer, augmenting opinionated responses with appropriate visual images. ZEPI's linked information structure enables rapid movement across contexts and highlights the hierarchical structure in the information.

ZEPI Architecture

The Zooming, Electronic Programming Interface is designed with a modular architecture. In general, the primary components consist of interface modules, a command processor, and a display processor. ZEPI abstracts the interface away from the hardware, making it possible for viewers to interact with a wide variety of electronic devices without learning specific interfaces for each.

In practice ZEPI could use a variety of hardware for input. In our implementation, the interface module uses a hand held wireless pointing device and a microphone. The pointing device allows the viewer to make direct selections. Typical grid-based programming guides force the viewer to follow a prescribed path through a menu by pushing arrow keys on a remote. The interface modules recognize viewer actions, track various gestures made with the pointing device and conversation, and formulate standardized representations. Fig. 1 shows a detailed view of the ZEPI components.

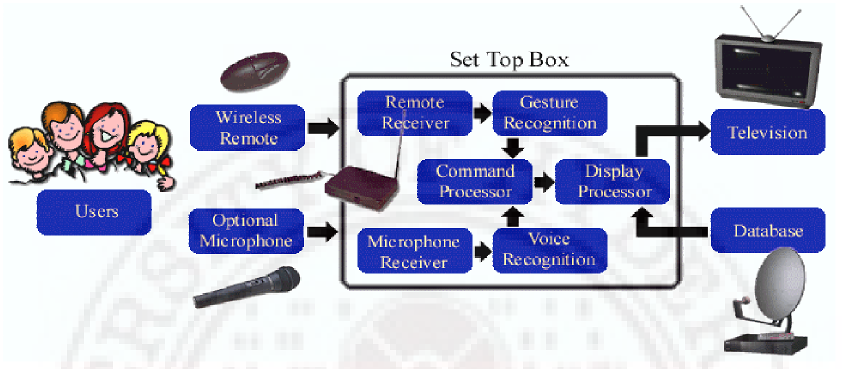


Fig. 1. ZEPI includes multimodal input processors and a link to an external database

The command processor performs semantic mapping, correlating the viewer's actions into device specific functions. The command processor is ZEPI's most hardware specific software, and typically resides on the device that it controls, or uses a network interface such as those available for appliances in "smart homes." After the device performs its function or the semantic mapping concludes that the standard speech or gesture is not supported, the display processor receives a notification.

The display module reports the results of viewer actions through visual and aural modalities via the television set itself or a set-top box. It communicates the results of queries, images, text, and audio. If necessary, it requests more information from the viewer after determining the best method of acquiring the information. At the viewer's discretion, the display manager may present content or request information in a manner that imparts the device with a personality.

The display processor was implemented in TCL/TK running on an SGI™ O2. The command processor was written in Java™ and the speech interface used the Java Speech API with IBM™ ViaVoice™ as the underlying recognizer and voice synthesizer. C++ was used to write the gesture recognition processor for a GyroPoint™ wireless pointer. For our tests the database was stored on a remote file system accessed over a network.

ZEPI Zooming Visual Interface

Viewers use ZEPI to find information with a *zoomable* display capability [3,4]. ZEPI provides a coherent view of information that can be highly complex, such as programming content of broadcast television, along with ancillary information about content or schedules. Thousands of shows, actors, schedules, images, reviews, and other information are available but the zoomable display allows a viewer to browse intelligently through the many resources at his disposal.

ZEPI gives the television *personality*. Since it is designed for natural interaction with viewers, it makes sense for it to have a "persona" to whom the viewer can relate. Based on the context of the interaction ZEPI can comment on presented information using the voice and opinions of fictional characters, movie stars, or critics, if available.

Visually, ZEPI consists of multilayered panels, each containing a view of a subset of the available categories of content, content providers, schedules, and information, as shown in Fig. 2. When a viewer selects a category or provider by speech or gesture, the interface dynamically zooms into the next hierarchy layer, displaying the contents of that layer in the panel. The panel may contain any type of textual or graphical information, including but not limited to icons for content or providers, graphical schedules, reviews, photos of actors and actresses, biographical data; and scripts.

ZEPI relies on a database that relates many different types of information and media. Currently, this information is provided by handcrafted linked databases, forming a linked set of related information nuggets. ZEPI is able to change the interactive context very quickly by recognizing the different types of information available organized into a graph structure. Selecting a type of program, for example, may change the context from "type of program" to "examples of action programs," which in turn may lead to "actors in Action Movie." Context is a combination of category selection and conversation tracking that is critical if ZEPI is to have a dialog with the viewer.

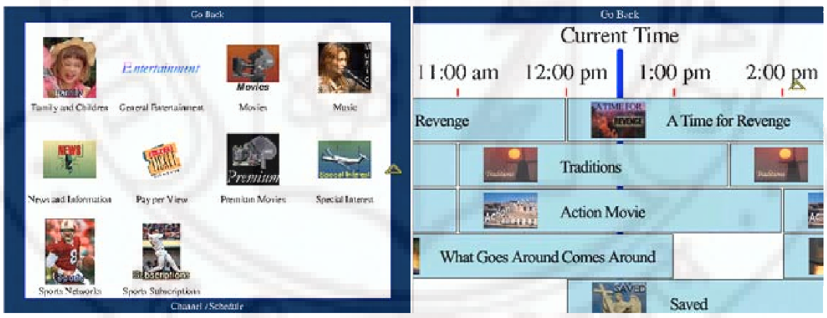


Fig. 2. Typical views provided by ZEPI

The viewer navigates through information by panning, zooming, and selecting. Panning allows viewing off screen information while zooming allows customization of the amount of content displayed and provides support for many screen resolutions, including NTSC and HDTV. Selection allows movement between multiple panels of information. Information navigation with ZEPI is distinct from grid-based program guides because ZEPI uses spoken language and gestures made by the viewer. Moving the pointer or saying "show it bigger" can result in zooming. These navigation modes are designed to give the viewer control over the viewing area, making it possible to see a large body of information with a small display surface, as in Fig. 3.

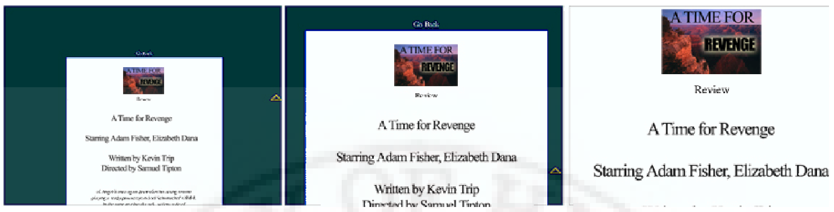


Fig. 3. Interactive zoom and pan make text easier to read on low-resolution displays

There is no single means of navigation control. A viewer may use various combinations of buttons and gestures, or spoken language, to signify a desired interaction. The interface modules supports various gesturing devices with many buttons, or none, including wireless or corded mice, wireless pointers, wireless remotes, and other devices that otherwise simulate two or three button mice.

ZEPI Context-Sensitive Speech Interaction

ZEPI makes it possible for a viewer to indicate desired actions with pointing motions and context specific phases such as "Give me that one." Instead of hunting for tiny buttons or digging out the instruction manual, a viewer can talk to the device and use hand motions to communicate needs. When necessary, the device can ask for more information, either verbally or using a visual display.

The speech interface module has a customizable language grammar. It responds to simple phrases like "go back," "pick," "zoom in," "pan screen," or complex queries like "Show me comedies starring Adam Fisher playing Saturday after six." Ambiguous queries elicit a request for more information, for instance asking "What films have Smith in them?" could result in this question from ZEPI: "Do you mean Joseph Smith or Tina Smith?" Context sensitive dialog is key to making ZEPI flexible to use.

A Short ZEPI Example

The scenario in Fig. 4 is typical of ZEPI capabilities. The dialog is taken from an actual ZEPI session, except that names have been replaced for legal reasons. As a viewer speaks aloud, ZEPI interprets his words in context, and display images relevant to the discussion. Note that the display and spoken dialog are not preprogrammed "command" responses, but are sensitive to the context of the interaction.

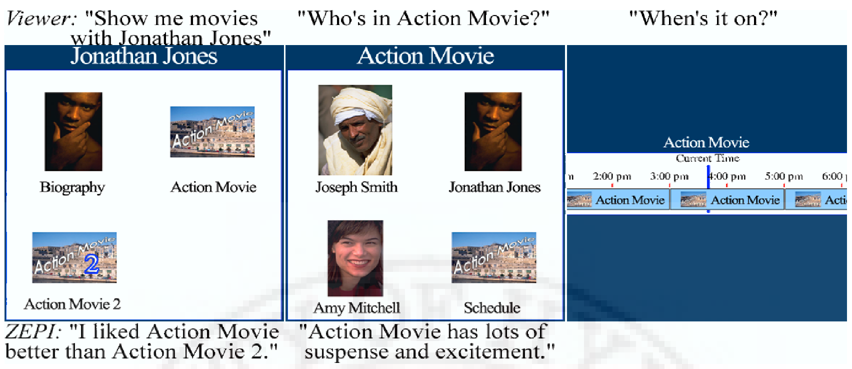


Fig. 4. ZEPI responds to queries with context sensitive commentary

Conclusions

The novelty of ZEPI is the combination of a panning, zooming display and flexible multimodal interface. With smooth interactive control of the display the viewer can read text that would be illegible otherwise. A viewer can get information by having a conversation with ZEPI rather than navigating a complex menu system. ZEPI can have a personality, making interaction more interesting and fun.

References

[1] Hee-Kyung Lee, Han-Kyu Lee, Jeho Nam, Beetnara Bae, Kom, Munchurl, Kyeongok Kang, and Jinwoong Kim, 2002, Personalized Contents Guide and Browsing based on Viewer Preference: Proceedings of the AH'2002 Workshop on Personalization in Future TV.

[2] Fabio Pittarello, 2002, Time-Pillars: a 3D Cooperative Paradigm for the TV Domain: Proceedings of the AH'2002 Workshop on Personalization in Future TV.

[3] Bederson, B. B., Hollan, J. D., Steward, J., Rogers, D., Druin, A., Vick, D., Ring, L., Grose, E, and Forsythe, C., 1997, A Zooming Web Browser, *in* Human Factors in Web Development. New Jersey, Lawrence Earlbaum.

[4] Hong Jiang Zhang, C. Y. Low, Stephen W. Smoliar, and D. Zhong, 1995, Video parsing, retrieval and browsing: An integrated and content-based solution: Proceedings of the Conference on Multimedia 15-24.

Dynamic Personalized TV Recommendation System

Weizhen Dai and Robin Cohen

School of Computer Science, University of Waterloo, Canada
wdai@math.uwaterloo.ca, rcohen@pythagoras.math.uwaterloo.ca

Abstract. Personalized TV models based on user modeling are the development direction of today's TV system. Recommendation Systems for TV programs are the key to implementing personalized TV models. Various system architectures have been proposed and developed in the past ten years. In this paper, we propose a dynamic recommendation system, taking advantage of explicit and implicit, content-based and collaborative filtering recommendation systems. Included in our model is a proposal for incorporating a pseudo-user profile, as the basis for the collaborative filtering process. This pseudo-user profile reflects groups of similar users, thus reducing the search time in matching a current user to other like-minded users, in determining a recommendation. We also propose a server-based architecture for delivering recommendations, in an effort to overcome problems with static information in the collaborative filtering process. We discuss how the proposed architecture for selecting shows based on viewers' personal interests provides some improvements to other models in the literature and suggest some directions for future research.

Introduction

With more and more programs provided from cable and satellite television services, people have trouble selecting their favourite programs. It may take hours for people to decide, using a traditional TV guide service. For this reason, TV recommendation systems have been introduced and developed by companies and research groups.

Since hybrid recommendation systems that combine content-based and collaborative filtering techniques have shown advantages over other (content-based or collaborative filtering) systems (e.g. [1]), we design a server-based multi-agent system which takes advantage of explicit and implicit, content-based and collaborative filtering recommendation systems. In particular, we propose a "dynamic" collaborative filtering method to adapt to changes of users' implicit or explicit profiles. Although we have not yet implemented the system, we present some reasonable details and explanations for why the architecture provides an improvement over other designs of TV recommendation systems.

One motivation for our work is to address some difficulties with current local recommendation systems. On a local machine, there are a fixed number of possibly outdated user profiles used for collaborative filtering. Our aim is to provide a continuously updated base of user profiles, to improve the collaborative filtering recommendations. We also introduce a "pseudo" user profile which will be used to improve the matching efficiency while applying the collaborative filtering technique. To handle the final recommendation, we propose a "Recommendation Manager" which applies a competitive mechanism to decide what program should finally be delivered to the active user.

System Model

In this section, we outline our proposed dynamic personalized TV recommendation system. We first present a design for delivering the recommendations to users, aimed at supporting a dynamic recommendation process. We then describe our proposed architecture for generating recommendations to users, based on profiles of the user's preferences.

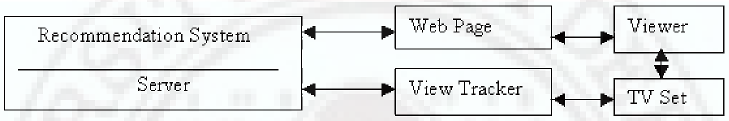


Fig. 1. Work Flow between User and System

As described in Figure 1, our proposed workflow between user and system allows users to retrieve the recommended program list through the TV set screen or alternatively from a web page. The recommendation system is located on a server. The server provides a centralized control of the system. It retrieves the information from each end user (viewer) and distributes the recommendations to active users. It also provides a necessary environment for our dynamic system, which cannot be implemented on a local machine, e.g. database support, online recommendation service, etc. In addition, there is one equipment between the TV set and the Server, called the View Tracker. Its purpose is to track the viewer's behaviour, e.g. programs, times, channels, etc. and to forward this information to the recommendation system. One assumption in our system design is that the system makes recommendations for only one viewer. In future sections, we will discuss further how this architecture is ideal for supporting a dynamic recommendation process.

Our hybrid recommendation system merges recommendation from different subsystems (see Figure 2). An overview of the system recommendation strategies is as follows. Details will be provided later.

Explicit RS: In explicit RS, programs specified by users are always in the recommendation list. We also include the programs selected by the content-based algorithm, using the explicit user profile. The recommendation list from explicit RS may be the longest one compared to the lists from CB and CR RS's.

CB RS: Like all existing content-based TV recommendation algorithms, CB RS computes the similarity between the available programs and the active user's favourite programs in the implicit user profile. The programs with highest similarities will be sent to the recommendation manager.

CF RS: By matching the personal profiles (implicit or explicit) with one of the pseudo user profiles, then the favourite programs of the "pseudo" user will be recommended to the active user. At the initial stage, the explicit user profile is used to match with one of the pseudo user profiles since there is no implicit profile available. While we have both explicit and implicit, the profile from the subsystem with higher credit will be used. The credit evaluation system will be explained later.

Recommendation Manager: It applies the algorithm we propose, merges the selected programs from each different list and finally presents the merged results to the active user.

We now proceed to explain the proposed architecture for our Recommendation System in more detail.

TV Shows means the available TV programs to the active user. The default is today's available programs. Each subsystem applies its algorithm to the available programs.

Profile in our system refers to the three kinds of profiles, explicit, implicit and "Pseudo" user profiles which we will discuss below. Although the architecture proposed above is sufficiently general to accommodate any chosen representation for user profiles, for the subsequent discussion we will consider user profiles with the same specified format as shown in Figure 3. The format will help the system recommend the general favourite programs for the active user.

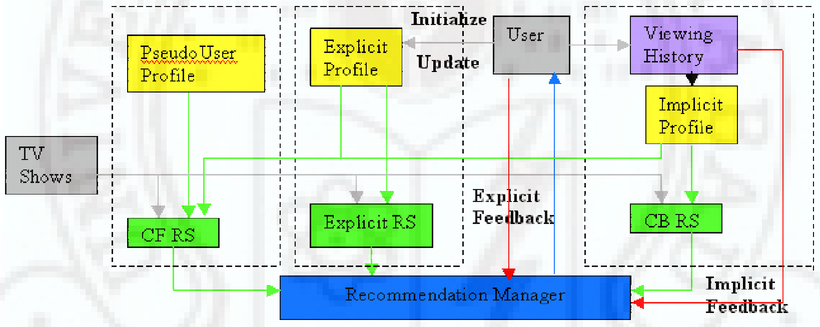
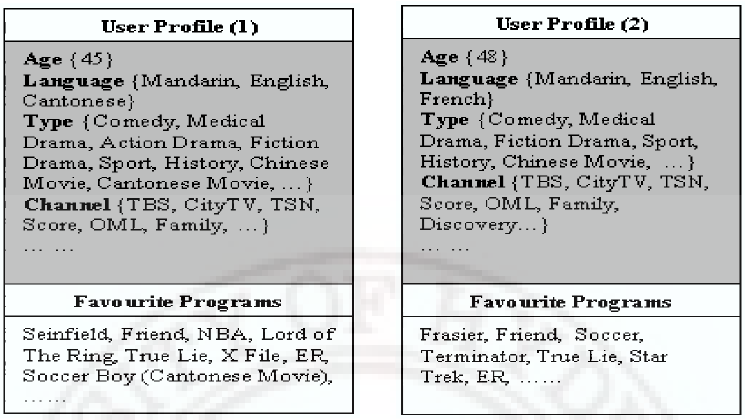


Fig. 2. Architecture of Recommendation System

Each profile consists of two sections. The first section (shaded section in Figure 3) contains a set of features about one user, the generalized information about the features of each user. Each feature has a set of values. The maximum number of distinct values for each feature is predefined, e.g. the number of feature "channel" values is 10, "language" is 3, etc. The size of each set of feature values is limited to help to narrow the set of possible recommendations, hopefully improving the accuracy of the recommendations as well. The second section (white section in Figure 3) contains the user specified favourite programs, or the most watched programs. There are some differences between how this is handled in the pseudo profile and explicit/implicit profile. This will be clarified later.

Pseudo User Profiles refers to a set of profiles which are stored in a database. Each pseudo-user profile has the same general format as explicit/implicit profiles. Since the pseudo user is the representative of a group of similar users, the second section in the format is the union of the specified programs of all users in the group.



User Profile (1)

Age {45}
Language {Mandarin, English, Cantonese}
Type {Comedy, Medical Drama, Action Drama, Fiction Drama, Sport, History, Chinese Movie, Cantonese Movie, …}
Channel {TBS, CityTV, TSN, Score, OML, Family, …}
… …

Favourite Programs

Seinfield, Friend, NBA, Lord of The Ring, True Lie, X File, ER, Soccer Boy (Cantonese Movie), ……

User Profile (2)

Age {48}
Language {Mandarin, English, French}
Type {Comedy, Medical Drama, Fiction Drama, Sport, History, Chinese Movie, …}
Channel {TBS, CityTV, TSN, Score, OML, Family, Discovery…}
… …

Favourite Programs

Frasier, Friend, Soccer, Terminator, True Lie, Star Trek, ER, ……

Fig. 3. Profiles for User (1) and (2)

This set of profiles is built up in the system when new users start to use this system. Users with similar features and preferences are placed in the same group. Then the most common features and preferences among users in a group are selected to form a "pseudo" user profile. One of the advantages of this system design is that these profiles are not "static". With the development of the system and more users starting to use it, more "pseudo" profiles can be created by reorganizing the existing profiles. In fact, if the profile of one member in one group is changed dramatically with time, we can treat this user as a new user. His/her latest profile will then be matched to a corresponding pseudo profile. The group and the pseudo profile for that group will also be updated since this "new" user is added. This process will help to refine the "pseudo" user profile. It means that the similarities among members of a group become more refined and the "pseudo" user profile becomes a better representation of the members of the group. In addition, having more refined profiles means that the active user has better chance to get good recommendations by the collaborative filtering technique.

A "similarity" value will be computed between an active user and every pseudo user profile. According to the value, the active user will fall into one of the groups. This process will be implemented by "similarity range". The similarity among group members should be greater than the similarity between members from different groups. If we want to have a better "pseudo" user profile, we can divide current members in one group into two or more groups by setting up a new "similarity range" and more profiles will be created correspondingly.

We can set the similarity range to be R (less than 1); the similarities among users in a group should be within range R. The smaller the range R, the better the refinement of the profiles and more pseudo profiles will be created. For example, we define the similarity between identical users to be 1. If we define the range $R > 0.5$, we may have 5 pseudo profiles; when we set the range $R > 0.7$, we maybe have 8 profiles to represent 8 groups. The same active user could fall into a new group whose pseudo profile matches more closely to the active user's profile. However, this process doesn't mean we should have unlimited templates for every user, i.e. $R = 1$.

For example, the similarity value of the user problem in Figure 3 should be computed as: Similarity = W_1(age) + W_2(language) + W_3(Preference) + W_4(Channel) + ……Wn (nth feature), where $W_1, W_2, \ldots\ldots Wn$ are the weights for each feature and $W_1+W_2+\ldots\ Wn = 1$, age, language, preference, etc. are the values reflecting the fraction of similarity between the user profiles. E.g. For feature Language, the value of language is 2/3 since there are only two matching values (Mandarin and English).

If the similarity is greater than the similarity range we have set up, then the two users will fall into one group. For the values of features in the pseudo profile, they are decided by the most commonly occurring values for each feature from these member profiles.

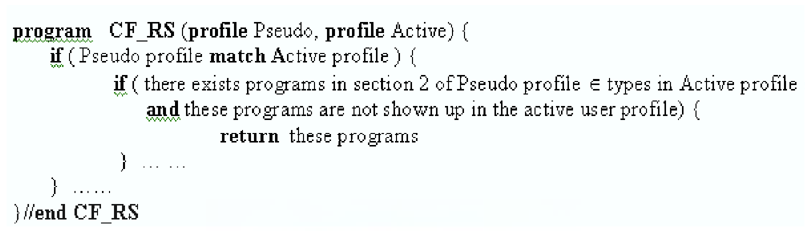
Explicit Profile is created during the first session between the system and a user through an interactive interface of TV or a web page access. This profile contains the general interests of a user for all genres of programs, the preferred time and date to watch TV and so on. After the first session, this profile can also be modified and updated by users, i.e. users can change their preferences. In an explicit profile, a user can specify which program s/he wants to watch, time, date, channel, etc. And these programs will always be recommended to the active user by the explicit recommendation system agent.

Implicit Profile is created on the basis of a user's viewing history. It contains the relevant information about the favourite programs the user has watched, e.g. type, time, date, programs, etc. For example, in an implicit profile, the types of the most often watched programs by a user are recorded as the favourite types in the profile. That is, if a user often watches "Friends" and "Frasier", it is very possible that the "Comedy" type will be recorded under "Type" feature. The system also adds the most often watched programs into the "Favourite Programs" section of the implicit profile. In this case, "Friends" and "Frasier" may be included in this user's "Favourite programs" in his/her implicit profile.

CF RS (Collaborative filtering recommendation system) works by matching the active user's profile (explicit or implicit profile) to one of the pseudo user profiles. The pseudo-user's favourite programs will be recommended to the active user. We give a simplified code (see Fig. 4) to explain the working process of CF RS.

We still use the example in Figure 3 to show how this works: now we assume that several users are close enough and form a group. We also suppose that the pseudo user profile for this group is User Profile (1). Suppose the active user is in fact user 2. Suppose that by computing the similarity between user 1 and user 2, we conclude that they are well matched. Now we check the programs in section 2 of profile (1). We can find "X File" is a program which doesn't appear in profile (2), but it is a "Fiction Drama" which is a type of show active user 2 likes. Then "X File" will be recommended by the CF RS to the active user.

CB RS (Content-based recommendation system) selects the programs from the available programs on the basis of the user's implicit profile. For example, a user likes "action, adventure drama; the preferred time and date is in the evening of weekend, … …", and now we have a movie "Indiana Jones" which is an "action, adventure drama" movie and will be available at 9:00 this Saturday, then there is a possibility that this movie will be recommended to the active user since this movie is very similar to the user's favourite programs.



```
program   CF_RS (profile Pseudo, profile Active) {
    if ( Pseudo profile match Active profile ) {
            if ( there exists programs in section 2 of Pseudo profile ∈ types in Active profile
                and these programs are not shown up in the active user profile) {
                        return  these programs
            } … …
    } ……
} //end CF_RS
```

Fig. 4. Algorithm for CF_RS

Explicit RS (Explicit recommendation system): According to the information in the user's explicit profile, this RS recommends programs to the user. The procedure is similar to CB RS, except the user profile is the explicit profile. For example, a user could have explicitly indicated that he likes "action, adventure drama; the preferred time and date is in the evening of weekend, … …", and he would have the same recommendation explained above.

Recommendation Manager: It analyzes the recommendation lists from CF, CB and Explicit RS and decides what the final programs will be recommended to the active user. One reasonable scheme for combining the recommendations from the different subsystems is to have the subsystems compete with each other and the manager merges their recommendations on the basis of their previous performances. More recommended programs will come from the subsystem with the better performance record, since it is more reliable.

However, how to decide which recommended programs are finally presented to the active user is a complex problem. First, we need to build an evaluation system used to evaluate the performance of each recommendation subsystem. Second, we need to create the metric to be used to evaluate the performance of a subsystem. Our proposition is: each subsystem is assigned a certain credit equally. If a recommended program receives a positive feedback (definition in next paragraph) from the active user, the credit of that subsystem will be increased; otherwise, that subsystem will lose its credit. More programs will be taken by the manager from the subsystem with a higher credit, and fewer programs from subsystems with a lower credit.

One rational choice of metric is to check the feedback from the user. Here the feedback refers to the "implicit" feedback, and "explicit" feedback from users. The "implicit" feedback means the feedback is tracked by observing the user's viewing history. Whether a feedback to a recommended program is positive or negative is decided by the "percentage of watching time". The "Percentage of watching time" is the percentage ratio of the watching time to the length of program, e.g. "Seinfeld" is a 30-minute program, I watched it for 15 minutes totally, maybe not continually, then the percentage of watching time = 50 %. Obviously, the higher percentage, the more positive the feedback. We can set up a "threshold", like 20% (This number would ideally be decided by testing results). If we are above this threshold, we think it's positive feedback; otherwise, it is negative. The explicit feedback is provided by the user via the TV set or webpage. The feedback is a direct response about the recommended programs. We can set several values, like Fail, Ok, Good, Perfect. There exists a corresponding relation between implicit and explicit feedback. For example, Threshold (20%) is equal to OK.

Both implicit and explicit feedback will be gathered and result in a final evaluation of each recommendation system. The aim is to credit subsystems with recommendations that are approved by the user. Since it will take some time build up the implicit user profile, it may be the case that initially a strong explicit user profile will lead to strong recommendations from the Explicit RS. But with time, the CB RS, relying on the implicit profile, may end up being preferable, since the data in the explicit user profile may become outdated. The recommender manager would end up forwarding the more reliable recommendations to the user.

Using our metric to evaluate the subsystems, we need to think about some kinds of situations. One is some recommended programs may not be watched by the user; the second is the watched programs are recommended by all three subsystems. In situation one, since the programs are not watched by the user and there is no feedback available, it won't affect the credit of each subsystem. (It is very possible that an unwatched program is a good recommendation, e.g. I like Soccer program and Basketball program, but at the same time, I can only watch one of these programs). In situation two, we can decide, if the feedback is negative, it won't affect the subsystem credit; if it is positive, we can increase the credit.

If the credit for CF RS is very bad, it may mean that the active user has been matched to the wrong group. And updating the status of the active user in the CF subsystem is necessary, i.e. computing the similarity of the active user with other users and deciding what current group he should be in. The reason is that the explicit/ implicit RS are content-based processes, and if the manager finds the explicit/implicit subsystem has better performance, it means the active user profile in the CF subsystem is out of date. We need to update the active user profile correspondingly by the implicit/explicit profile. In most cases, the choice should be implicit profile, unless the user often updates his/her explicit profile which will help the explicit subsystem catch up with the implicit subsystem.

Related Work

PTV[2] is an internet-based recommendation system. Users can register, login, and view their personalized TV guides as specially customized HTML or WML pages. It produces personalized TV guides by integrating user profiling, content-based (case-based) reasoning, and collaborative profiling techniques.

The disadvantage is the method of collecting user information and feedback. In PTV, no implicit technique is used to track and collect users' viewing history or behavior. Only the explicit technique is applied. This preliminary profile is built from the user registration time. After that, all the feedback from the users is accomplished by users explicitly grading the recommended programs on web pages. Although the explicit technique can provide better recommendation result, it increases the burden of users.

EPG: Personalized Electronic Program Guides [3,6], is a system based on a multi-agent architecture, where specialized agents collect data about the available events, monitor the user's behavior to retrieve information about the user's interests and select the events (programs) to be advertised in the personalized EPG, depending on the user's preferences at the time of day user wants to watch TV. The system is located on the user's Set Top Box, which can store structured information about TV events (programs) into the local database in the box. As we will discuss further, the stereotypes used for the collaborative filtering in this system are static and therefore may reflect outdated information.

Multi-Agent TV Recommender system [4,7] is designed for multiple agents to work together and collectively model the profile information on a particular user. It encapsulates three user information streams -- implicit view history, explicit TV viewing preferences, and feedback information on specific shows – into adaptive agents and builds a framework that allows for these multiple agents to collaborate and generate a combined program recommendation for a TV viewer. The advantage shown in this system is that the system gathers user feedback on specific TV shows and feeds that information to the implicit and explicit agents by a Feedback Agent.

Later, the authors proposed another kind of multi-agent system [7] which combines one explicit recommender agent with two implicit recommender agents (Bayesian statistics and Decision Trees). All three recommenders still are based on the content-based technique. However, if the integration of content-base and collaborative filtering is a good design for a recommendation system, this multi-agent system lacks the benefits of a collaborative filtering technique.

Discussion

In our system, we use a server-based "dynamic" collaborative filtering system instead of the "static" system located on a local machine. The number of pseudo user profiles in our system can be changed and each profile could be refined with more users using this system. We believe that the refined process of the profile will improve the probability of good recommendations for the active user. We call this an "evolution" system. This is why we design a centralized server-based recommendation system. For a local recommendation system, like EPG, the number of typologies cannot be changed after the user starts to use this system. The implicit or explicit profile must be classified into one set of typologies. Since these typologies are built into the local Set Top Box, there are no more choices. These typologies cannot evolve after the system is used. While the implicit profile is built better and better, the collaborative filtering system component will lose during the competition with the content-based system component which applies to the implicit profile. From our opinion, for a local system, it is not necessary to use a built-in collaborative filtering system, even though it may be beneficial for the cold-start problem.

The purpose of introducing "Pseudo User" is because of the feature complexity of TV viewer, program and feature value. If we use the active user profile (explicit or implicit) to compute the similarity with all other users, this will increase the time cost. By introducing "pseudo" user profile, the computation will be made between "pseudo" users and the active user, where the number of pseudo users is much less than the actual users. Although the method may sacrifice recommendation efficiency compared to other methods, it will improve the time efficiency. And if necessary, we can reset the "similarity range" value to improve the "pseudo user" profile.

In our system, we take the advantage of combining the "pseudo" user profile and the initialized Explicit Profile to deal with the "cold-start" problem (which means how to make high quality recommendations quickly after a user starts to use the system, e.g. [5]). A user needs to initialize an explicit profile, but it is unnecessary to be a complex and time-consuming profile. This profile will be used to provide recommendations by the Explicit RS after a user starts to use this system. On the other hand, the existing "pseudo" user profile will supplement the insufficient information in the initialized explicit profile by matching this explicit profile to one of the pseudo user profiles.

Regarding the "zero burden" problem (which means that the user doesn't need to do extra work, like building the explicit profile, to activate the recommendation system), we think the motivation is good. But for a recommendation system, it is an impossible mission. The explicit profile is a fundamental element in recommendation system. The problem here is how much work a new user must do to build the explicit profile, not a "do or don't" problem.

We think the future system should be a centralized server-based system. This design has many advantages, including commercial advantages. Users don't need to buy a "machine" box, just buy a service from TV program provider (cable company, satellite company, etc.). This mechanism is also beneficial for the developer of recommendation systems. They can modify, optimize or update the existing system at any time that they think it's necessary (new algorithms, building more "pseudo" profiles and so on). And the end-users don't feel the change. A local system is easily obsolete.

There is also one disadvantage in our centralized server-based system: the startup cost associated with gathering enough pseudo user profiles to make accurate user similarity matching.

Conclusion

Compared to PTV, EPG and Multi-Agent systems we mentioned, our system integrates the advantages of a collaborative filtering recommendation system which Multi-Agent system doesn't support, overcomes the shortcomings of "static" user typologies in EPG by "dynamic" pseudo user profiles and introduces the implicit user profile by tracking the user's viewing history which is lacking in PTV.

Since we haven't finished the whole system design, implementation and testing, there is no way to say our system should be better than others. What we are discussing here is our thoughts about the design of TV recommendation systems and its future trend.

One of the topics for future work in the Personalized TV model, we think, is to build a standardized feature representation for TV programs and viewers. For instance, current research on the MPEG-21 standard [8] is a good starting point. We also need to set up testing metrics to decide the performance of a recommendation system.

Other interesting avenues for future work on our proposed dynamic personalized TV recommendation system include: how to build up the base of pseudo-user profiles; how to set the feature values of the pseudo user profile; how best to evaluate similarity matching between the active user profile and pseudo-user profiles; and how best to combine the recommendations from different subsystems into one overall recommendation for the active user.

For the first issue, we need to be ensure that our algorithms perform sufficient merging of existing user profiles into pseudo-user profiles, in order to take advantage of the savings in searching for similar users that are advocating in our model. So, the worst case where each current user leads to one distinct pseudo-user profile should be prevented. For the second issue, consider the example in Figure 3. If user 1 and 2 form a group, the feature LANGUAGE in the pseudo profile will be {Mandarin, English, "unknown"}. What should the "unknown" be, Cantonese or French? In fact, it might be possible to decide from other features' values. One possible way is to check the TYPE, CHANNEL, ……, or maybe FAVOURITE PROGRAMS in order, and see which language has more relations to the members of this group. If we cannot tell by TYPE, we can check CHANNEL next, then other features one by one, finally by checking the favourite programs. For the third issue, it would be worthwhile to examine different similarity metrics to see the effect not only on recommendations but also on performance of the system in detecting the pseudo-user profile best suited to produce recommendations for the active user. For the last issue, we can consider whether it is more beneficial to present the programs recommended by the most reliable subsystem only or whether it is better to merge the recommendations being made by each of the subsystems.

In any case, the proposed architecture allows for the consideration of various recommendations and supports a dynamic recommendation process, allowing for user profile information to be updated, leading to more effective recommendations for users.

Reference

[1].Thomas Tran and Robin Cohen; Hybrid Recommender Systems for Electronic Commerce; Proceedings of AAAI00 workshop on Knowledge-Based Electronic Markets; 2000.

[2].Cotter, P, & Smyth, B. (2000); PTV: Intelligent Personalised TV Guides,Proceedings of the 12th Innovative Applications of Artificial Intelligence (IAAI-2000) Conference, AAAI Press.

[3].L. Ardissono, F. Portis, P. Torasso, R. Bellifemine, A. Chiarotto and Difino; Architecture of a System for the generation of personalized Electronic Program Guides,Workshop on Personalization in Future TV(2001), Othofen, Germany.

[4].Jurapati, K., Gutta, S., Schaffer, D., Martino, J., and Zimmerman, J.; A Multi-Agent TV Recommender, Workshop on Personalization in Future TV(2001), Sonthofen, Germany.

[5].Kaushal Kurapati and Srinivas Gutta; TV Personalization through Stereotypes, Workshop on Personalization in Future TV (2002), Malaga, Spain.

[6].Angelo Difino, Barbara Negro and Alessandro Chiarotto, Telecom Italia Lab; A Multi-Agent System for a Personalized Electronic Program Guide, Workshop on Personalization in Future TV (2002), Malaga, Spain.

[7].Anna L. Buczak, John Zimmerman and Kaushal Kurapati; Personalization: Improving Ease-of-Use, Trust and Accuracy of a TV Show Recommender, Workshop on Personalization in Future TV (2002), Malaga, Spain.

[8].Web site: http://mpeg.telecomitalialab.com/standards/mpeg-21/mpeg-21.htm, MPEG-21 Overview V.5

Involving Users in the Design of User Interfaces for TV Recommender Systems

Jeroen van Barneveld and Mark van Setten

Telematica Instituut,
P.O. Box 589, 7500 AN Enschede, The Netherlands
{Jeroen.vanBarneveld, Mark.vanSetten}@telin.nl

Abstract. In this paper we present guidelines for TV recommender interface design, which we obtained by conducting a design session with a diverse group of TV users. This design session, aiming to explore users' expectations, wishes and demands of user interfaces for TV recommender systems, was conducted as a first step in a user-centered iterative design process, and consisted of a brainstorm to generate ideas and an assignment to fabricate mock-ups for TV recommender interfaces. One of several conclusions is that users need an interface that supports planning a period of watching TV, but also supports a quick choice between programs that are broadcasted right now.

1 Introduction

With the introduction of more and more TV channels and programs, due to developments such as digital television, TV recommender systems [2] [6] have the potential to become important tools in aiding people to choose what they will watch on TV. In order for these systems to work properly and achieve desired goals, much attention should be paid to the user interface of these systems. If users do not feel comfortable in using TV recommender systems, they will be reluctant to use such systems.

In order to determine what an intuitive, easy-to-use interface for a TV recommender should contain, we initiated an iterative design process. Iterative design is a purposeful design process that tries to overcome the inherent problems of incomplete requirements specification by cycling through several designs, incrementally improving upon the final product with each pass [3]. As a first step in this user-centered design process [1], a design session was organized in order to involve potential users from the very start. This paper describes the process and results of this session.

2 Design Session

A design session was organized to explore users' basic expectations, wishes and demands of user interfaces for TV recommender systems. The main focus was on three important aspects of recommender systems: presentation of recommendations, presentation of explanations, and the way users can provide feedback. Presentation of recommendations is important, as people should be able to get a clear understanding of the predicted relevance. Explanations are necessary to inform the user about the reasons behind a prediction [4]. Providing feedback is necessary in order for the recommender to learn from the user and provide more accurate predictions in the future. [5]

2.1 Subjects

Potential users of a TV recommender system with no specific knowledge on recommender systems were invited. A total of 19 people participated in the design session. The group of subjects consisted of 9 males and 10 females between the age of 20 and 56 with various backgrounds (level of education, computer experience and occupation). Subjects participated on a voluntary basis.

To avoid intimidation of older people (with relatively little knowledge on ICT applications and new developments) by younger people (with more experience in this area), we split the design session in two separate days using the same approach. Participants younger than 45 were placed in group one, while older participants were placed in group two. This paper describes observations of both days combined.

2.2 Approach

The design session started with a brief introduction on the topic, followed by a brainstorm on the interface. Ideas on the main topics (i.e. presentation, feedback, and explanations of recommendations) were generated, written down, and posted visibly for every participant. These ideas were then clustered to get a better overview.

After a short discussion on the various ideas, while new ideas could still be added, groups of three to four participants were formed. Each group was asked to develop and present a mock-up TV recommender interface, based on the best ideas that were generated earlier. The interface had at least to be able to present a set of recommendations, give the user the chance to give feedback on recommendations and the chance to get an explanation on why a certain recommendation was made.

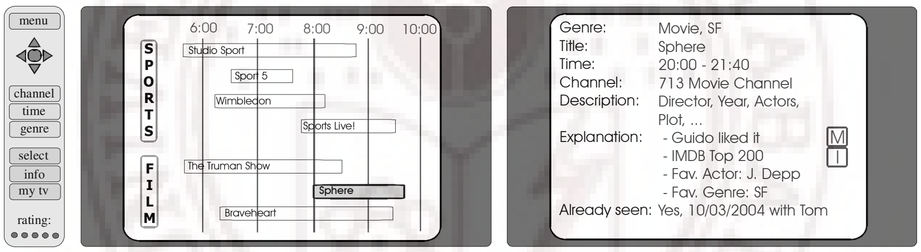
During the session, an effort was made to let people generate and work out their own ideas, and not lead them into any direction by giving hints or suggestions.

3 Results

The initial brainstorm resulted in a broad collection of ideas and propositions on TV recommender interfaces. Most of these can be summarized by the following points:

– The user should always be in full control. If desired, the user should be able to turn off recommendations. The user should have influence on a range of settings, such as the level of personalization, number and level of detail of recommendations;

– The TV recommender system should be available on a range of devices such as the personal computer, PDA/handheld, mobile phone and television. This way the recommender system can be consulted regardless of the whereabouts of the user;
– Watching TV is seen as a social activity. The possibility of multiple people watching TV and controlling the TV recommender system should be taken into account;
– Integration with a TV guide that offers information on all TV programs, not only on recommended programs, is desirable;
– Both explanations based on peer users' interests and explanations based on meta-data on TV programs (i.e. similarities of a recommended program and a user's favorite programs) are seen as interesting;
– Explanations should only be given when requested;
– Explanations should be easy to interpret. Textual explanations should be short, some users prefer visual explanations (for instance by means of charts);
– Providing feedback on recommended items should be as unobtrusive as possible. It should be easy, quick and should require only a small effort of the user. Implicitly generated feedback, for instance by processing viewing time of particular programs or uttered comments on programs, would be preferable.



Genre: Movie, SF
Title: Sphere
Time: 20:00 - 21:40
Channel: 713 Movie Channel
Description: Director, Year, Actors, Plot, ...
Explanation: - Guido liked it
- IMDB Top 200
- Fav. Actor: J. Depp
- Fav. Genre: SF
Already seen: Yes, 10/03/2004 with Tom

Fig. 1. Example of a mockup that uses a TV and adapted remote control. The main screen shows an overview of different TV programs with their titles on a channel/time grid, sorted on genre. More information is shown on the detailed TV program screen, which shows additional information such as exact time, plot summary, genre and recommendation explanation(s). This screen also allows users to directly send an e-mail to a peer user (on whom an explanation was based) or follow a link to related websites such as movie databases etcetera

The fabrication of mock-ups resulted in a wide variety of drawings and descriptions (two different resulting mockups are shown in figure 1 and figure 2). However, some important correspondences between different mockups can be noticed:
– Although participants state that a TV recommender system should be available on a range of different devices, almost all mockups are based on a TV with a remote control as operating device. One group proposed the use of a separate device (a hybrid of a PDA and a tablet PC) that facilitated the TV recommender interface and could simultaneously be used to operate the TV;
– As some participants remarked during the initial brainstorm, a TV recommender interface should ideally facilitate use by groups, since watching television is often a social event. A mockup that reflects this idea can be seen in figure 2;

– Every mockup sorts on genre by default, while some provide alternative sorting options on time, channel, etcetera;
– With nearly all mockups, the user always has the initiative involving recommendations. Recommendations are shown only when desired by the user. Only one group proposed the idea of recommendation messages (pop-ups in the bottom of the TV screen or via instant messaging mechanisms on PDA or mobile phone) to alert the user of a recommended TV program without a direct request from that user;
– Most of the mockups provide an easy way for users to give feedback on recommended items. Most common is a 5-point scale, operated by the remote control. Other options are a sliding continuous scale and voice recognition.

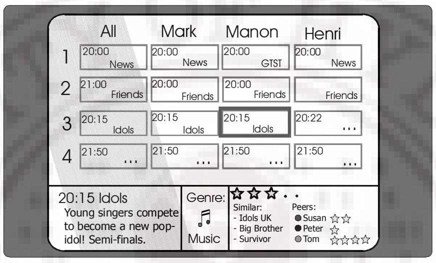


Fig. 2. Simple mockup described by participants as a helper for channel-switching, also using a television and adapted remote control. This mockup facilitates use by a group of users rather than one. A list of recommended TV programs is displayed for every user that is currently watching television in the group, as well as for the whole group of users watching television together. Feedback on recommended items is obtained through speech recognition. Other features such as detailed information are shown at the bottom of the screen

Among the various mockups, there are two main courses for interaction distinguishable. The first is based on the assumption that a user wishes to plan a couple of hours watching TV. Recommended programs can be selected and placed in a 'personal TV guide'. More detailed information on TV programs in this personal guide can be obtained, and these programs can be rated when watched. The mockup in figure 1 is an example of this kind of interface.

The second type of interaction is based on the idea that a user wants to watch a TV program at this moment that best suits his interests. These mockups provide a simpler type of interaction since there are fewer actions that have to be performed.

4 Conclusions and Further Research

The initial brainstorm session and following discussion as well as the mock-ups created by participants have provided some interesting information for TV recommender interface design. Although the design session focused on presentation of recommendations, explanations of recommendations and feedback on recommendations, more general conclusions for TV recommender interface can be drawn from the results.

A TV recommender interface should be very easy to use and should at all times give the user the feeling that he/she is in control. Essential is the fact that a TV recommender system should support the user's process of deciding what to watch, not replace it. Furthermore, the interface should facilitate an easy way to provide feedback on watched TV programs, preferably as unobtrusively as possible. Sorting on genre seems to be very important to users, and should be optional if not default. A TV recommender system should be accessible via TV, but also through various other devices such as PDA's and PC's. Integration of TV recommender systems with a TV Guide/EPG is desirable. Explanations on recommended items are seen as a very interesting feature, but should only be given when explicitly requested by the user.

Users seem to need both an interface that provides a way of planning television-watching for a number of hours as well as an interface that provides an easy way to discover what interesting TV programs are broadcasted right now. A well-designed complete interface should therefore facilitate both types of interaction between user and TV recommender system. Finally, since watching television is often a social activity, designers of TV recommender interfaces should develop systems that can be used by groups as well as single users.

The results of the described design session will be taken into account in the next phases of the iterative design cycle. Based on the results and an analysis of other recommender systems, mockups of TV recommender interfaces will be developed and usability tested to discover further issues, focus points and design guidelines.

Acknowledgements

The authors thank all participants of the design sessions and Andrew Tokmakoff, Ingrid Mulder, Betsy van Dijk and Mascha van der Voort for their helpful comments.

References

1. Barneveld, J.J.F. van: User Interfaces for Personalized Information Systems. Telematica Instituut, The Netherlands (2003). Online: https://doc.telin.nl/dscgi/ds.py/Get/File-28132
2. Baudisch, P., Brueckner, L.: TV Scout: Lowering the Entry Barrier to Personalized TV Program Recommendations. Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems, Malaga, Spain (2002) 58-68
3. Dix, A.J., Finlay, J.E., Abowd, G.D., Baele, R.:. Human-Computer Interaction, 2nd edn. Prentice Hall Europe, London (1998)
4. Herlocker, J.L., Konstan, J., and Riedl, J.: Explaining Collaborative Filtering Recommendations. Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work, Philadelphia, USA, (2000) 241-250
5. Setten, M. van, Veenstra, M., Nijholt, A.: Prediction Strategies: Combining Prediction Techniques to Optimize Personalization. Proc. of the workshop Personalization in Future TV'02, Malaga, Spain (2002), 23-32.
6. Smyth, B., Cotter, P.: A personalised TV listings service for the digital TV age. In: Knowledge-Based Systems, Vol. 13. (2000), 53-59

Natural Language Interaction in Personalized EPGs

Pontus Johansson

Dept. of Computer Science
Linkoping University, Sweden
ponjo@ida.liu.se

Abstract. In this paper, natural language (NL) dialogue is suggested as interaction technique for personalized EPGs to handle a variety of well-known problems. NL interaction addresses the new-user cold-start problem since the necessary user model can be gathered gracefully, without the high initial user effort prominent in traditional recommendation systems. It is argued that NL interaction enhances the user's freedom of expressing her preferences to increase recommendation accuracy. The conversational interaction style also enhances ease- and enjoyment-of-use. AdFilm, an adaptive movie recommender prototype, is presented to exemplify NL interaction in personalized EPGs.

1 Introduction

An important part of personalized electronic program guides (pEPGs) is recommendations. There are several interesting issues that demand attention in order for pEPGs to become successful. They include:

1. **Cold-Start Issues**: In order to give personalized recommendations, systems have to know about the user's interests. Collaborative filtering (CF) systems require the user to go through a tedious process of explicitly rating arbitrary items in the chosen domain. This delayed benefit is a well-known phenomenon known as the new-user cold-start problem. Users want to be able to efficiently start using the system right away, and get relevant information the minute they start using it.
2. **Recommendation Strategies**: While CF represents a major strategy for recommending items, it is not fool-proof. Sometimes users may want recommendations based not only on previous ratings, but rather on explicitly defined rules, and sometimes they may not. This calls for combining different prediction techniques, as well as an interaction technique that allows users to state such preferences in a natural and efficient way.
3. **Ease- and Enjoyment-of-Use**: pEPGs aim to serve a vast range of end-users, each carrying their own set of abilities, attitudes, and proficiencies. The usability requirement of a low learning threshold is thus very strong, as all potential users should be able to quickly start using the system. The system must therefore not require the user to learn and remember complex

commands or interaction techniques. The EPG context-of-use is typically a relaxed and casual home environment for entertainment purposes, which puts the requirement on the interaction to be focused on enjoyment as well as ease-of-use.

This paper aims to present a pEPG solution that provides movie information and recommendations. It is argued that the benefits of natural language (NL) dialogue interaction provide part of the solution to each of the stated problems. Section 2 presents motivations for using NL dialogue to address the new-user cold-start problem, section 3 discusses adaptive recommendation strategies and NL interaction, section 4 motivate NL interaction from the usability perspectives of ease- and enjoyment-of-use, and section 5 presents ongoing work with a NL movie recommeding system, AdFilm.

2 Cold-Starting Gracefully

The new-user cold-start problem is a well-known phenomenon (cf. [1]). If no information about a new user is present, the system cannot recommend anything specific. A traditional CF system needs several user ratings before it can recommend personalized items with accuracy. The web-based MOVIELENS project [2] relies on CF techniques and requires the user to explicitly rate movies from a given set. For a new user, recommendations are based on an average rating of all users. In effect, new users get a large number of recommendations that are of low quality. In general, users have seen several movies and know their way around in the information space compared to that of e.g. purchasing a new car [3]. Even if users generally can articulate what features they prefer in a movie (e.g. favorite genres or actors etc.), a pure CF based system has no way of capturing this explicitly. The user is thus left with the tedious process of browsing a set of movies, finding the ones she has seen, explicitly rating them, moving on to the next set of movies, and so on. Simply browsing an ordinary movie database could be just as rewarding for immediate benefit [4]. However, if the user were able to communicate her preferences freely and in a more direct way, the system could build a user model based on this input and provide better recommendations faster. NL is a suitable interaction technique for directly stating such preferences, and does not force users to learn and utilize a complex graphical user interface, or an artificial query language such as SQL [5]. NL utterances in human-human communication do not occur as isolated islands. They are part of an on-going dialogue, where participants gradually refine the quality of the conveyed information. By modeling NL dialogue in a recommendation system, we allow the system to ask clarifying and follow-up questions. The user's response is used to incrementally build the user model efficiently. Table 1 shows a human-human dialogue from the dialogue collection gathered in this project's initial user studies. It has been adapted to a user-system dialogue and represents a use-case for the AdFilm system (see section 5). It exemplifies how NL is used to gather preferences from a new user. During this first dialogue, the system tries to verify implicit ratings of movie titles, genres, and actors. Other underlying similarities are also assessed.

Table 1. Example dialogue where the system unobtrusively gathers preferences from the user, while simultaneously providing high-quality recommendations.

Dialogue	**System actions**
U: Can you recommend a good movie?	
S: Certainly. Do you want a certain kind of movie?	Suggests a list of genres.
U: No, just a good movie please	
S: Ok. Do you have a favorite movie?	
U: I liked Fight Club very much	
S: Do you generally like the drama/thriller combination?	Tries to verify genre bias based on Fight Club's genres
U: Sure, but in this case it is the actors that make the movie excellent	Verifies that the genre is somewhat important to the user; Retrieves actor information and notes it as important; Retrieves miscellaneous information, such as director.
S: I see. Have you seen Seven?	Searches the database with the actor and genre preferences. Among the results, the movie Seven has the same director as Fight Club and is thus suggested first.

By employing NL interaction, users can quickly explain what qualities they want in a movie, by stating favorite features, and exemplifying with favorite titles. In the example above, the system adapts to the user's reluctance to state a favorite genre, by instead letting him give a title example. By strategically asking follow-up questions the system can gracefully gather relevant information about user preferences in a natural and pleasant way. The dialogue also serves as a tool for determining preference confidence in several ways. The user's critique of a bad recommendation serves as an indication of what attributes are important to the user [6, 3]. Additionally, preference strength can be estimated depending on in what conversational circumstance a user states her preferences. For example, a preference given as a reason for rejecting a recommendation has higher preference strength than a preference given as answer to a system query [7].

3 Adaptive Recommendation Strategies

Every user has her own way of thinking about movie preferences. Some people are highly biased on genres, whereas others may want to watch all kinds of movies as long as they star their favorite actor, etc. This is highly individual, and providing a graphical or form-based interface to cater for all such preferences would require a very complex user interface. The Internet Movie Database provides a Power Search mode (http://us.imdb.com/list) where users can combine several details to get more precise movie title retrievals. For example, users can require certain persons to be included, and filter by genre, year, and other options. The search user interface consists of 17 text fields, 20 choice boxes, 6 check boxes, and requires vertical scrolling before a query can be submitted. Despite this complex interface, users are limited in several ways, e.g. to combining only two genres. Furthermore, there is limited support for negations. Negations and quantifications are generally hard to capture in graphical or form-based user interfaces, but easy to express in NL [8, 5]. Consider the example utterance "I usually like Bruce Willis but I can't stand his comedies". This is a valid user preference involving a negation that the system should be able to handle. By allowing users to state such preferences in NL the accuracy of the recommendations can be enhanced efficiently, and the interaction feels natural to the user. The graceful handling of the cold-start issue (see section 2) implies an adaptive recommendation strategy that is achieved by combining prediction techniques [9]. For a new user, a content-based approach based on what preferences the user has expressed in the dialogue is utilized. When enough ratings have been gathered CF techniques are gradually introduced to the content-based recommendation strategy. This means that we can modify the result of the CF recommender based on the preferences concerning genres or other features discovered throughout the dialogue.

4 Ease- and Enjoyment-of-Use

A pEPG has a wide range of users of varying age; background; and ability and interest in learning to use new software. Thus, the system should have as low threshold as possible before users can put it to effective use. One of the traditional arguments for using NL interaction in information-providing systems is that it is ideal for casual users since they do not need to spend time learning the system's communication language [5]. A language-understanding pEPG would thus not require users to learn a new way of interacting as for example a graphical user interface navigated with TV remote control buttons would. Previous studies on interaction with EPGs on digital TV set-top boxes show that navigating through TV program information requires several steps when the interaction is mediated through a remote control. Not surprisingly, a NL dialogue system was found to be easier to use than a traditional graphical user interface [10]. In another study it was found that NL input is a suitable strategy for efficient TV program information navigation [11]. In addition to ease-of-use and efficiency qualities, user studies of EPG usage in the home environment indicate that users find NL interaction in the TV domain to be enjoyable [10].

5 AdFilm - an Adaptive Movie Recommender

AdFilm is a combined information-providing and recommendation dialogue system prototype under development. The combination allows users to get quality results with minimal initial effort [4], while it unobtrusively gathers preference data. AdFilm accommodates the points raised in this paper, and is in its current version focused on the cold-start problem. It employs an adaptive dialogue strategy that allows for dialogues such as the one in Table 1. AdFilm gathers data about preferences to be used for content-based recommendations, and title preferences to be used for CF recommendations. The system assesses preference strengths depending on the conversational circumstance, as outlined by [7]. AdFilm also gathers preferences about the dialogue strategy. In the example dialogue in Table 1 the user is reluctant to provide a favorite genre, and prefer to volunteer a favorite movie. In subsequent sessions, AdFilm will ask this user for more favorite movies, instead of bothering him with genre preference questions. AdFilm also assesses that the user has an actor bias, since he explicitly said that the actors - not the genre - made Fight Club his favorite movie. This is utilized in subsequent dialogues when the system has the initiative and asks clarification questions. However, the user may at any time ask for information in whatever way he prefers.

References

1. S. Middleton, H. Alani, N. Shadbolt, and D. Roure, "Exploiting synergy between ontologies and recommender systems," in *The 11th International World Wide Web Conference (WWW2002)*, Hawaii, USA, 2002.
2. A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: learning new user preferences in recommender systems," in *Proceedings of the 7th international conference on Intelligent user interfaces.* ACM Press, 2002, pp. 127–134.
3. R. D. Burke, K. J. Hammond, and B. C. Young, "The findme approach to assisted browsing," *IEEE Expert*, vol. 12, no. 4, pp. 32–40, 1997.
4. P. Baudisch and L. Brueckner, "Tv scout: Guiding users from printed tv program guides to personalized tv recommendation." in *Proceedings of the 2nd Workshop on Personalization in Future TV*, Malaga, Spain, 2002.
5. I. Androutsopoulos, G. Ritchie, and P. Thanisch, "Natural language interfaces to databases–an introduction," *Journal of Language Engineering*, vol. 1, no. 1, pp. 29–81, 1995.
6. G. Linden, S. Hanks, and N. Lesh, "Interactive assessment of user preference models: The automated travel assistant," in *Proceedings of the 6th conference on User Modeling*, 1997, pp. 67–78.
7. S. Carberry, J. Chu-Carroll, and S. Elzer, "Constructing and utilizing a model of user preferences in collaborative consultation dialogues," *Computational Intelligence*, vol. 15, no. 3, pp. 185–217, 1999.
8. P. R. Cohen, "The role of natural language in a multimodal interface," in *Proceedings of the 5th symposium on user interface software and technology*, 1992, pp. 143–149.
9. M. Van Setten, M. Veenstra, and A. Nijholt, "Prediction strategies: Combining prediction techniques to optimize personalization," in *Proceedings of the 2nd workshop of Personalization in Future TV*, Malaga, Spain, 2002, pp. 23–32.
10. A. Ibrahim and P. Johansson, "Multimodal dialogue systems for interactive tv applications," in *Proceedings of 4th IEEE International Conference on Multimodal Interfaces*, Pittsburgh, USA, 2002, pp. 117–222.
11. A. Ibrahim, J. Lundberg, and J. Johansson, "Speech enhanced remote control for media terminal," in *Proceedings of Eurospeech 2001*, Aalborg, Denmark, 2001, pp. 2685–2688.

Readers and Skimmers Watching iTV: Evidences from Interactive Video-on-demand.

Alessandro Cappelletti, Marianna Nardon, Fabio Pianesi, Massimo Zancanaro

ITC-irst
Panté di Povo, Trento Italy
{cappelle,nardon,pianesi,zancana}@itc.it

Abstract

In this work we discuss the hypothesis that iTV viewers can be clustered, with respect to their viewing behavior, in two distinct groups that resembles the prototypical *readers* and *skimmers* found among hypertext users.

An experimental study with 55 subjects was conducted in a condition as close as possible the TV watching (use of a remote control, distant view, and so on). The viewers were asked to explicitly rate every additional scene accessed. Cluster analysis showed that two groups of watchers can be recognized: (a) medium/high–rating viewers who see about 4 additional scenes during the initial documentary's section and less than 2 additional scenes in the remaining sections, and (b) low-rating viewers who see less than 2 additional scenes during the whole documentary's vision. If confirmed by other experiments, these results demonstrate that TV watchers exhibit skimming and reading behaviors and that viewers' satisfaction can be influenced by their belonging to one class or the other.

Introduction

One of the most promising trends of iTV is the video-on-demand technology [iTVMarketer] that may, at least in principle, allow viewers to dynamically compose their own TV programs. In a previous work [Nardon et al., 2001], we experimentally evaluated three different interfaces for proposing online follow-ups in a interactive documentary. Our data showed that the best way of organizing information is to provide viewers with a default path through the documentary, this way reducing the number of possible choices at a given moment, while allowing follow-ups to be selected even when not contextually appropriate. Since our results are partially in contrast with other similar studies [Drucker et al. 2002] according to which viewers prefer interfaces requiring more time and more clicks, we decided to further investigate the behavior of viewers looking for evidence of different styles of interaction.

Golovchinsky [Golovchinsky, 1997] found that hypertext users can be divided in two clusters according to their browsing behavior: subjects who tend to spend much time reading texts (e.g., articles) to determine their relevance to the search topic, and subjects who tend to skim over the retrieved documents, making many quick relevance judgments. The two groups were characterized as *readers* and as *skimmers*, respectively. Our main goal in the present study was to investigate whether interactive viewers can be classified in a similar way. Moreover, we were interested in determining whether the satisfaction rate viewers assigns to optional scenes in the documentary were somehow correlated with the viewers behavior.

In future work, we will explore the possibility of using the information about the class the actual viewer belongs to, to automatically adapt the proposal of follow-up additional scenes.

Interactive Documentaries

In [Nardon et al. 2001], we defined a model for interactive documentaries as a sequence of *scenes*, where the content of each of them is in the form of a time-based media (such as video, audio, animation or a combination of those). An Interactive Documentary has two kinds of scenes: *main* and *additional*. Main scenes are collected in a sequence that forms a complete documentary, and each of them can have one or more additional scenes that provide further details.

Follow-up links are shown on the interface as selectable items using the scene title as anchor. The browsing capabilities have been implemented following the standards commonly used in DVD interfaces.

For the experimental study, we prepared two documentaries. The first describes the life and habits of the ibex. It consists of 3 main scenes for a total of 2 minutes of video content, and 3 additional scenes for a total of 2 minutes. This documentary was used in the training phase. The second documentary is about the bear. It contains 8 main scenes for a total of 6 minutes, and 9 additional scenes for a total of 5 minutes.

In order to simulate as closely as possibly the iTv scenario, we implemented a system where a user can watch a program in a passive way or interact choosing additional content. The documentary is projected on a screen (and not played on a computer display) and the viewer can only interact with the system through a remote control similar to the one commonly used in DVD. The actions that the viewer can perform are the common video controls (i.e. play, pause, stop, skip to next and previous scene), and the browsing and the selection of additional information. The system logged each action performed by the viewer.

Figure 1 shows the screenshot of the documentaries' index. From here the users can select the documentary they want to watch, choosing between the ibex's documentary and the brown bear's one.

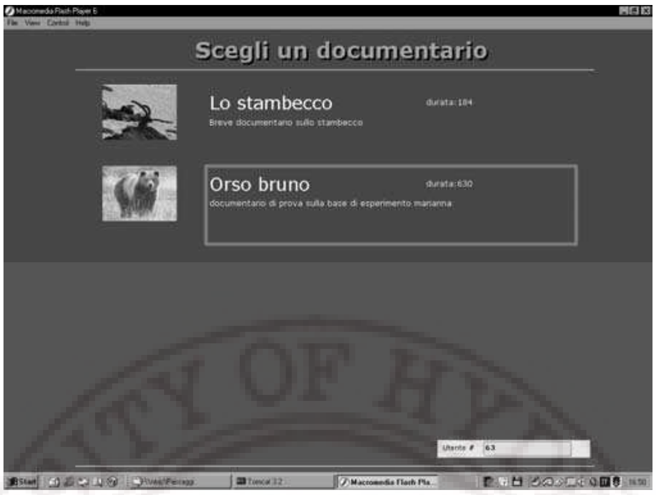

Macromedia Flash Player 6
File View Control Help
Scegli un documentario
Lo stambecco
durata:184
Breve documentario sullo stambecco
Orso bruno
durata:630
documentario di prova sulla base di esperimento marianna
Utente # 63

Figure 1: Documentaries' index

Figure 2 shows the screenshot that the users watch after the documentary's selection. It displays both the documentary (on the right side of the screen) and the additional scenes' index (on the left side of the screen). The user is free to interrupt the documentary at any moment and select an additional scene. The latter is displayed in the area were the documentary was playing. At the end of the additional scene, the documentary resumes from where it was interrupted.

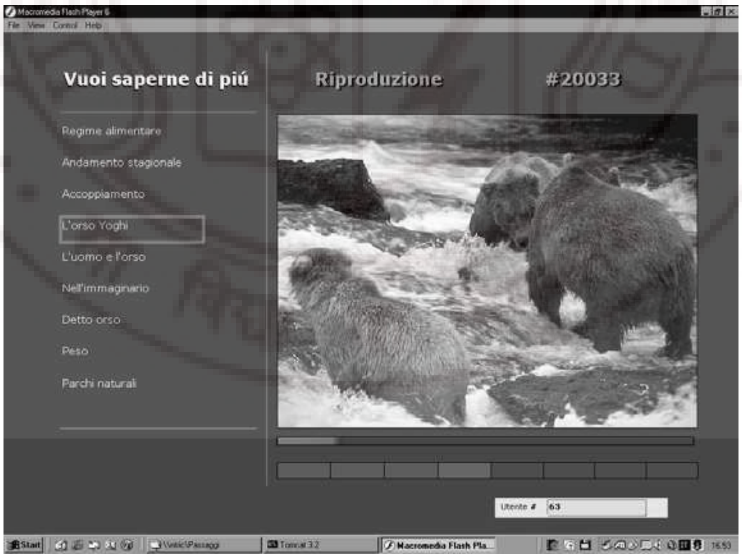


Figure 2: Interactive documentary and additional scenes' index

Besides being a suitable experimental setup, we believe that the system is a valid architecture for video-on-demand. The interface was realized on Flash MX which is planned to be ported on the major set-top boxes. The documentaries are MPEG movies stored in a relational database together with the documentary structure (i.e., the sequence of main scenes and the additional material). At present, the video material is delivered through an HTTP connection but in the next release it will be streamed through an RTMP channel.

The Experiment

The goal of the experiment was to determine whether there exist two clearly distinguishable groups of iTV viewers, similar to the skimmers/reader distinction of web users, and to find out whether viewers' satisfaction with respect to the documentary content, can be related to, and predicted from, their behavior. As behavioral indices we considered the following actions, as logged by the system: play, pause, stop, skip to next and previous scene, browsing, and selection of an additional scene. Following [Claypool et al., 2001] viewer satisfaction was measured by means of explicit ratings provided by subjects for each additional scene accessed.

Design

Fifty-five subjects participated in the experiment. Each subject was assigned to the same experimental condition consisting of the interface setting. The mean age was 18.56 years. All participants had enough computer and Internet surfing skills to successfully complete the task, as tested by means of a questionnaire. It was our assumption, in fact, that above an established (and very low) threshold computer literacy would not affect significantly task completion.

The subjects were recruited among participants in a summer camp, and participated in the experiment on a voluntary basis.

Procedure

Subjects were tested individually. Each session lasted approximately 15 minutes. Before each session, subjects were asked to first compile a short questionnaire on personal data and their Internet and DVD systems expertise; then the experimenter quickly introduced the system and its functionalities by means of the ibex documentary. Finally the subject was left alone to watch the bear documentary. The subjects were asked to explicitly rate the interest at the end of each additional scene accessed.

Tools

The questionnaire includes 4 items assessing personal data (age, sex, studies and profession) and 2 items measuring the familiarity (in terms of amount of time spent weekly) with Internet and with the DVD. The last two items are based on a Likert scale (none/more than 30 times weekly).

Subjects could select additional scenes from the "additional info" menu. After seeing the additional scene, they were requested to express their interest in it, by rating their agreement with the following statement "*I would suggest to a friend who I know very well, to see the additional scene I have just seen*", on a 5 points Likert scale.

Results

Viewers saw 46.89% of the additional scenes available and the mean rate about the scenes was 3.54 points (the scale ranges from 1 to 5). Table 1 reports the average number of additional scenes seen, divided according to whether they belong to the first, second, third or fourth documentary section, each session consists of 2 consecutive scenes.

	N	Mean %	Cumulative Mean %	Standard Deviation
I section	50	22,89	22.89	28,43
II section	50	5,33	28.22	12,746
III section	50	5,11	33.33	15,094
IV section	50	13,56	46.89	21,91

Table 1

As can be seen, most of the additional scenes were accessed during the first and the last section. Figure 3 shows the mean rate expressed by the whole sample for each additional scene.

Moreover, 78% of the users watched 3.24% of the available additional scenes, this yielding an average of less than 1 additional scene seen every 65 seconds of the documentary duration. The remainder (22% of the users) watched 85.78% of the additional scenes, with an average of 4 scenes during the first 65 seconds, and of 3 scenes during the last 2 minutes of the documentary. These data suggest that it is possible to distinguish two groups of subjects: those who see few additional scenes, with a constant access rate throughout the documentary; and subjects who access many more scenes, mostly during the first and last part of the documentary. In the next section, we will see that this expectation can be confirmed.

The study of the correlations between behavioral indices and the explicit ratings, showed that only the number of additional information navigated per documentary section were significantly related to ratings. Thus, behavior concerning actions like 'stop', 'skip, etc., did not correlate with the interest expressed. To understand these data, it must be noticed that the overall number of those actions, as performed by users, was very low, this way preventing possible differences from emerging.

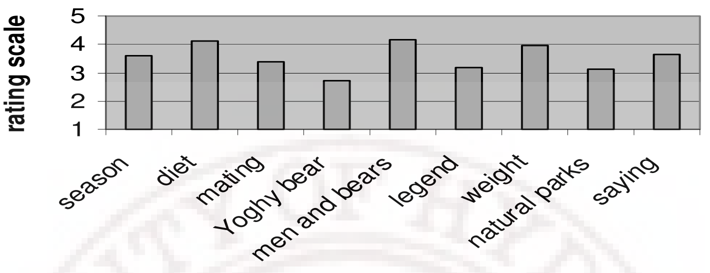


Figure 3: Ratings on the additional scenes of the bear documentary

To see whether it is possible to divide subjects into different categories, a cluster analysis was performed, with the explicit ratings expressed by users as the dependent variable. It turned out that it is possible to group subjects into two categories: those providing high rates and those providing low rates. The first group includes 39 people expressing a rate below 3, whereas the second group comprised the remaining 11 subjects, who expressed a rate between 3 and 5.[1] Table 2 shows the average rate for each additional scene based on the cluster assignment (K-Means Cluster).

	Cluster1 Low rating	**Cluster2** Medium/high rating
RATE_29 (season)	,56	3,55
RATE_32 (diet)	,72	3,09
RATE_28 (mating)	1,18	3,18
RATE_37 (Yogy bear)	1,08	3,00
RATE_38 (men and bears)	2,62	4,45
RATE_40 (legends)	,87	3,27
RATE_31 (weight)	1,59	2,64
RATE_39 (natural parks)	,44	3,00
RATE_30 (saying)	1,21	3,36

Table 2

The differences between the average ratings for each additional scene in the two clusters are all significant except for scene 31 (Weight), as can be seen from Table 3 where Anova results are reported.

[1] With the exception of the scene “Weight”, for which the average is below 3 in both groups, see Table 2.

	Cluster	Error	F	Sig.
	Mean Square	Mean Square		
RATE_29 (season)	76,263	2,007	38,006	,000*
RATE_32 (diet)	48,313	3,142	15,378	,000*
RATE_28 (mating)	34,400	3,237	10,627	,002*
RATE_37 (Yogy bear)	31,731	2,433	13,043	,001*
RATE_38 (men and bears)	29,022	3,874	7,491	,009*
RATE_40 (legends)	49,459	2,678	18,469	,000*
RATE_31 (weight)	9,399	3,958	2,375	,130
RATE_39 (natural parks)	56,410	1,575	35,821	,000*
RATE_30 (saying)	39,976	2,977	13,427	,001*

*Reliable difference, p = 0,05

Table 3

As previously noticed, subjects can be divided according to both the number of additional scenes seen, and to when they accessed them. It is natural to ask whether these data can be confirmed, and, if so, how it relates to the clusters we discovered. To investigate this point, we treated cluster assignment as a new factor (two levels independent variable, 1 = low rating and 2 = medium/high rating), and studied how the two groups fare with respect to the number of additional scenes seen. Table 4 shows the average number of additional scenes seen by the two clusters in the 4 documentary's sections.

		N	Mean	Std. Deviation
I_ section	Low rating	39	1,4615	1,66775
	Medium/high rating	11	4,1818	3,89405
II_ section	Low rating	39	,5385	1,25334
	Medium/high rating	11	,2727	,64667
III_ section	Low rating	39	,2051	,46901
	Medium/high rating	11	1,3636	2,65604
IIII_ section	Low rating	39	1,0256	1,63010
	Medium/high rating	11	1,9091	2,87939

Table 4

As it turns out, people in cluster 2 (medium-high ratings) are those who see more additional scenes, doing so mostly at the beginning and at the end of the documentary. The subjects in cluster 1 (low ratings), on the other hand, tend to see less scenes, and their behavior in this respect doesn't seem to change during the documentary. The results of an analysis of variance, Table 5, shows that these differences are significant.

		df	Mean Square	F	Sig.
I_ section	Between Groups	1	63,491	11,843	,001*
	Within Groups	48	5,361		
	Total	49			
II_ section	Between Groups	1	,606	,455	,503
	Within Groups	48	1,331		
	Total	49			
III_ section	Between Groups	1	11,516	7,005	,011*
	Within Groups	48	1,644		
	Total	49			
IIII_ section	Between Groups	1	6,697	1,748	,192
	Within Groups	48	3,831		
	Total	49			

*Reliable difference, p = 0,05

Table 5

The two clusters behave consistently, according to the within group results, whereas they significantly differ on the number of additional scenes seen during the first and the third section of the documentary.

Finally a discriminant analysis was performed to check accuracy of rating prediction on the basis of the number of additional scenes accessed in the first and second sections of the documentary only. By applying the discriminant function, 92% of the subjects of our sample were correctly classified as belonging to cluster (a) *low rating* or cluster (b) *high rating*. Table 6 reports the discriminant function's coefficients.

	Discriminant function
I_ section	0.973
II_ section	-0.242

Table 6 Standardized Canonical Discriminant Function Coefficients

In the future we will test the classification accuracy of the discriminant function on new users.

Conclusions

An experimental study with 55 subjects was conducted in a condition as close as possible to an advanced video-on-demand setting. The goals of the experiment were to find out whether interactive viewers can be classified in a way similar to the well-known classes of *skimmers* and *readers* and to investigate whether the interest of viewers in the optional scenes of a documentary might be correlated with, and predicted from, their behavior.

In percentage, 78% of the users accessed 3.24% of the additional scenes available, with an average of less than 1 additional scene every 65 seconds of the documentary time-line, whereas the remaining 22% of the users accessed 85.78% of the additional scenes available, with an average of 4 scenes seen during the first 65 seconds, and of 3 scenes in the last 2 minutes of the documentary. These impressionistic findings were confirmed by the discovery, through cluster analysis, of two main groups of viewers: (a) medium/high–rating people who also are additional-scene consumers, seeing about 4 additional scenes during the first section of the documentary and less than 2 in the remaining sections, and (b) low-rating people who also showed a low tendency towards viewing additional scenes (less than 2 during the whole documentary's vision).

These results can be interpreted as evidence for the existence of different iTV viewer categories comparable to the *skimmers* (our group (a)), and to *readers* (group (b)) identified in [Golovchinsky, 1997] among hypertext users. That is, people seeing a low number of additional scenes can be argued to do so because they prefer to follow a traditional, sequential fruition of the media; this resembles Golovchinsky's *readers* who, when confronted with an hypertext, tend to replicate the behavior followed with ordinary texts. On the other hand, the subjects viewing a high number of additional scenes depart from such a 'linear' model, being keener to adapt to, and take advantage of, the characteristics of the new media. In this sense, they are close to *skimmers*, who show a strong interest in following the links the hypertext makes available.

The soundness of this parallelism, however, must be further assessed.[2] Indeed, it is possible that instead of reflecting different viewing styles, the behavior of group (a) and (b) is related to contingent differences in the attitudes of our subjects towards some features of the experimental situation that our study didn't controll for. For instance, subjects could differ with respect to their interest in the topic of the documentary (the bear), or in their level of involvement in the experiment. At present we can only offer impressionistic results from the (written) observations made by the experimenters, who always reported a good level of interest and involvement in the execution of the task. But, clearly, further investigation is required.

To the extent that our suggestion can be confirmed, the found differences can be used to predict viewer's interest, once it is clear which group they belong to on the basis of the number of additional scenes seen. For instance, a viewer who sees few additional scenes in the initial portion of the documentary can be expected to continue to do so, and to express a low degree of interest. As far as our sample goes, this prediction is confirmed by the results from the discriminant analysis. This, again, is not conclusive, since we haven't assessed yet how the discriminant function fares with new users; independent evidence about its predictive power is presently missing.

Finally, we didn't find indications that more indirect, hence more interesting, behavior plays a predictive role. Possibly, our documentary, and/or our experimental design, were not able to elicit a sufficiently rich behavior in terms of pause, stop, skip, etc., actions for differences to emerge. Different experimental conditions, and/or different contents might elicit a richer behavioral repertoire, with significant correlations with interests in additional scenes and follow-ups.

In conclusion, the present study has provided initial evidence for the existence of two groups of ITV-viewers that can be related to the *skimmers/readers* distinction in their attitude towards the media, and found behavioral correlates that can be used to

[2] As an anonymous reviewer pointed out to us.

predict users' interest in additional scenes. Our results need to be confirmed in a more controlled setting, testing the predictive power of the discriminant function on independent data, while also trying to elicit a richer behavioral repertoire.

Acknowledgment

This work has been supported by the TICCA and PEACH projected funded by the Autonomous Province of Trento.

References

iTVMarketer, http://www.itvmarketer.com/deployments/vod_deployments.htm

Claypool M., Phong L., Wased M. and Brown D. Implicit Interest Indicators. In Proceedings of ACM Intelligent User Interfaces Conference (IUI), Santa Fe, New Mexico. 2001 2001.

Golovchinsky G.. Queries? Links? Is There a Difference? In CHI 97 Conference Proceedings, ACM Press, 1997, pp. 407-414.

Nardon M., Pianesi F., Zancanaro M.. *Interactive Documentaries: First Usability Studies*. AH2002: 2nd Workshop on Personalization in Future TV. 2002

Steven M. Drucker and Asta Glatzer and Steven De Mar and Curtis Wong (2002). SmartSkip: consumer level browsing and skipping of digital video content. *Proceedings of the SIGCHI conference on Human factors in computing systems* pp.219--226.

Content Augmentation Aspects of Personalized Entertainment Experience

Nevenka Dimitrova[1], John Zimmerman[2], Angel Janevski[1], Lalitha Agnihotri[1], Norman Haas[3], and Ruud Bolle[3]

[1] Philips Research, 345 Scarborough Rd., Briarcliff Manor, NY 10510, USA
{nevenka.dimitrova,angel.janevski,lalitha.agnihotri}@philips.com

[2] Human-Computer Interaction Institute, Carnegie Mellon, Pittsburgh, PA, USA
johnz@cs.cmu.edu

[3] IBM T.J. Watson, 30 Saw Mill River Road, Hawthorne, NY 10532, USA
{nhaas,bolle}@us.ibm.com

Abstract. We present personalization aspects in information and video content retrieval and augmentation applications. Our goal is to explore the value of linking related content among different media such as TV and Web. Metadata extraction can make a great difference for personalized user experience. Annotations that provide title, genre, description, and cast can be greatly enriched with detailed information at subprogram level, i.e. at the scene and story level. This enhanced metadata can be matched against the user profile for prioritizing information and entertainment.

1 Introduction

For many years, television has provided rich content, delivering news, information and entertainment. More recently, the web has developed as a rich content source, more overwhelming to navigate and master. Linking the related information between these two media, it seemed to us, could create an enhanced, more personalized, TV viewing experience. We set out to explore the possibilities, in this project.

At a high-level, we wanted to explore how information linking and personalization can bring value to users. We call this research direction *Content Augmentation*. As an example, imagine a viewer is watching a TV news story about Brazil. Because a TV program is expensive to produce, the content creators can only afford to play a short story, of around two minutes. After watching the story, a viewer may want to know more about Brazil. A content augmentation application could understand that the news story is about Brazil, and provide the user with appropriate, summarized, and targeted information, as well as references (e.g. web links) for further exploration. In addition, this application could employ a user profile, personalizing the linked content by prioritizing the types of links a user is likely to want. To test this model, we developed a pilot system. We began by focus group-testing several concepts, and, based on the group's reaction, designed and implemented a personal news application (MyInfo) and a movie information retrieval application (InfoSip).

While the majority of TV personalization research conducted today focuses on examining TV shows at a program level, our system utilizes video content analysis to process TV programs at a sub-show level. It first processes the video in order to create content-based metadata at a subprogram level and stores this metadata with the TV program. At the same time, the system extracts relevant web content based on a personal profile. MyInfo extracts specific web content listed in the profile and can display a personalized TV news program based on the web content and prioritized, individual TV news stories. This approach is a step further than existing news retrieval systems in the literature that are focused on broadcast news [Merlino et al. 1997]. InfoSip identifies actors in individual scenes. It then extracts web data, including the latest filmographies and biographies. While watching the movie, users can press a button and see a list of all actors in the current scene. In addition, they can view filmographies based on their TV viewing history. These content augmentation applications offer a new direction for personalization research, where the source of the content is less important than the actual delivered information to the user.

2 Augmented User Experience

Existing TV user experience is based on a 50-year-old tradition. Broadcasters tailor TV programming to capture mass audiences. They used both demographic data on viewers and input from advertisers to determine which programs to play at the various times of day. More recently, with the emergence of niche-based TV channels such as CNN (news), MTV (music), and ESPN (sports), viewers have had a little more control over when they view the content they desire. Until recently, viewers had only printed guides to help them plan their viewing and select TV shows. More recently, electronic program guides allow viewers to browse the program offerings by genres, date, time, channel, title and, in some cases, search using keywords.

Next, we describe the existing TV user experience in a broadcast setting and the potential of connected media delivery modes for enhanced user experience.

2.1 The TV Experience of Current EPG, PVRs, and Recommenders

Current electronic program guides found in products such as DirecTV and EchoStar's digital satellite settop boxes, cable settop boxes from Time-Warner Cable and Cable-Vision, and personal video recorders by TiVo and ReplayTV, offer users several methods for searching and browsing TV listings. All these systems hold one to two weeks worth of TV data, which users can view by time and channel. Higher end systems allow users to browse by show title and keywords. Finally, the TiVo system even offers a recommender to help users find something to watch.

Although TiVo is currently the only commercial product with a recommender, much personalization research has been done in this area. Das and Horst developed the TV Advisor, where users enter their explicit preferences in order to produce a list of recommendations [Das et al. 1998]. Cotter and Smyth's PTV uses a mixture of case-based reasoning and collaborative filtering to learn users' preferences in order to generate recommendations [Cotter et al. 2000]. Ardissono et al. created the Personalized EPG that employs an agent-based system designed for set top box operation [Ardissono et al. 2001]. Three user modeling modules collaborate in preparing the final recommendations: Explicit Preferences Expert, Stereotypical Expert, and Dynamic Expert. And Buczak et al. developed a based recommender that uses a neural network to combine results from both an explicit and an implicit recommender [Buczak et al. 2002]. What all these recommenders have in common is that they only examine program-level metadata. They do not have any detailed understanding of the program, and cannot help users find interesting segments within a TV program.

The Video Scout project we previously developed offers an early view of personalization at a subprogram level [Jasinschi et al. 2001, Zimmerman et al., 2001]. Video Scout employed a GUI construct called "TV magnets" (Figure 1). Users can specify financial news topics and celebrity names. The system then watches TV and extracts segments, searching the contents of talk shows for matching celebrity clips and searching the contents of financial news programs for matching financial news stories.

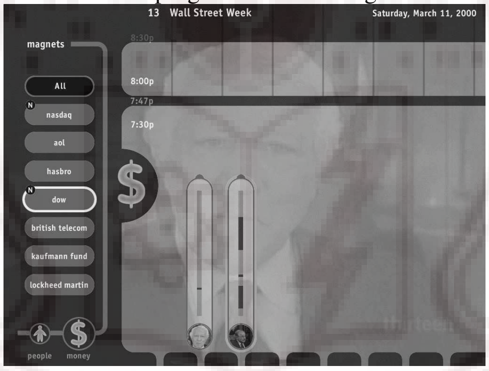


Figure 1. Financial news magnet screen with four stored clips from two TV shows.

2.2 Content Extraction: Beyond High-level Metadata

The technology described in the previous section provides mostly coarse tools for content personalization and navigation. However, with the advancement of video content analysis, indexing, and retrieval, we can offer the consumer a more powerful set of tools for an abundant set of media sources including TV, Web, and radio.

The system diagram in Figure 2 shows the high-level chain of content processing and augmentation. Unannotated or partially annotated content is delivered to the service provider (e.g. content provider, broadcaster) where generic analysis and augmentation is performed. The first step extracts features and summarizes the content and content segments, generating descriptive metadata. A more detailed description of this step is given in Section 3. The generated metadata in conjunction with any preexisting metadata, is then used to augment the content with additional information from web sources. This augmentation is general in that it is not based on any personal profile. Following broadcaster augmentation, the content with the complete metadata is formatted and delivered to the consumer device.

The consumer device ("client" in Figure 2.) has the capability of storing content and metadata, and also contains the user profile in conjunction with the prioritization module. This is used to perform a secondary augmentation with web information, but this time based solely on user preferences. The information obtained is stored together with the content and is presented to users as if it were a part of the original program.

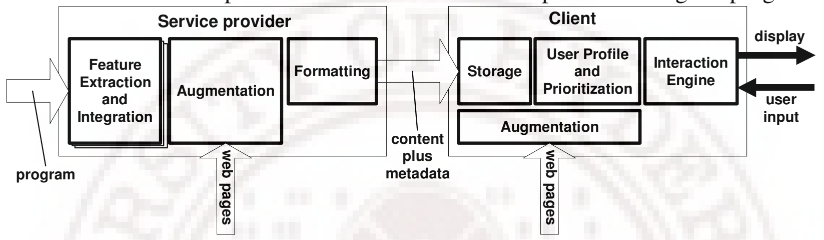


Figure 2. High-level system diagram

2.3 Applications

Detailed metadata extracted locally or delivered from a server or a peer device offer a host of possibilities for personalized applications. We divide these applications into retrieval and augmentation.

There are two distinct modes for a retrieval application: (i) the user proactively searches through the metadata in order to find segments of interest, such as querying a digital library, (ii) a user profile acts as a query against continuously updated metadata of the latest stored news, narrative or other content, such as Video Scout – a content-based PVR [Jasinschi et al. 2001, Zimmerman et al. 2001].

Content augmentation enhances the user experience by automatically utilizing the user profile. We have researched and prototyped two applications: MyInfo and InfoSip. MyInfo delivers news from TV and web channels according to a user profile. To access the news, users select one of the six "content zones": weather, traffic, sports, financial news, headlines, and local events (Figure 3). InfoSip answers most frequently asked questions, such as who, when, where, and what. Using a remote control and the current play context, users indicate their query, such as "Who is that actor?", "What is this song?" and the system overlays the answer onscreen, allowing users to sip relevant information while watching movies (Figure 4).

3 Metadata Extraction for Personalized User Experience

Methods for automatic metadata extraction can be divided into coarse- and fine-grain segmentation and abstraction. In this section, we briefly introduce the methods used.

3.1 Segmentation

We have developed a high-level segmentation method, in which we first find the commercial breaks in a particular news program and then we perform story segmentation. For stories, we use the story break markup (">>>") in the closed captioning. This method has been used in the literature before [Maybury 2000]. For commercials, we use a transcript-based commercial detector. In part, this relies on the absence of closed captioning for 30 seconds or more, and in part, it relies on the news anchors using cue phrases to segue to/from the commercials. We look for onset cues such as "right back", "up next" and "when we return", in conjunction with offset cues, such as "welcome back" and the "new speaker" markup (">>"). We tested commercial detection on four financial news and four talk show programs, totaling 360 minutes, with 33 commercials totaling 102 minutes. Our algorithm detected 32 commercials totaling 104 minutes. Of these, 25 were exactly right. Only one commercial was completely missed. We detected 4 extra minutes spread out over seven commercials. The resulting recall and precision are 98% and 96% respectively.

In addition we have developed a single descent method for story segmentation that relies on multiple cues: audio, visual, and transcript. The single descent method relies on unimodal and multimodal segment detection. Unimodal – within the same modality – segment detection means that a video segment exhibits "same" characteristic over a period of time using a single type of modality such as camera motion, presence of certain objects such as videotext or faces or color palette in the visual domain. Multimodal segment detection means that a video segment exhibits a certain characteristic taking into account attributes from different modalities. Next, we use multimodal pooling as a method to determine boundaries based on applying votes from multiple modalities. In the single descent method, the idea is to start at the top of the stacked uniform segments and perform a single pass through the layers of unimodal segments that in the end yields the merged segments. In this process of descending down through the different attribute lines, we perform simple set operations such as union and intersection. For example, one approach is to start with the textual segments as the dominant attributes and then combine them with the audio and visual segments.

3.2 Summarization

Each broadcast news story must be summarized in order to use (i) the abstracted data for matching against the personal profile and (ii) the summary for presentation browsing. Although there are different forms of summaries, in our case a summary consists of a sentence of text and a representative image (keyframe). Another implicit form of summarization is categorization (assignment to one of six "content zones").

The closed captioning text sent with each frame of the story is collected. Since it is usually mono-case, it is recapitalized. (We used our own algorithm for this, but more thorough algorithms have been developed [Brown et al, 2002]). Then IBM TextMiner document summarizer is applied, to select the single best sentence in the "document" [TextMiner]. "Best" in this context is a weighted metric involving the "salience" (position) of the sentence, its length, and other factors. Usually the first sentence of a news story is selected; this sentence is normally both a comprehensive summary and a good introduction. However, if a non-useful sentence occurs first ("Hello, I'm Dan Rather."), TextMiner often catches these and makes a better selection.

We also use a TextMiner engine for document (story) classification. It works on the basis of frequency of occurrence of words in the story. The classifier was trained off-line with a corpus of labeled exemplar stories.

Finally, we try to find a representative keyframe for each news story. Each story is composed of an anchor shot followed by *reportage* shot(s). The anchor shot is the same for all stories, so it does not provide any value. In order to select an image from only the *reportage*, we developed an anchorperson detector, based on the Kolmogorov-Smirnoff test. Basically, it finds the face, which is the largest face in view in the majority of the frames of the entire news program. Further, we have empirically determined that outdoor shots are more important for news stories than indoor shots. We use the indoor/outdoor detector developed by Naphade [Naphade et al. 2002]. Subject to these constraints, we select a frame that is deemed to be the most "interesting" by the algorithm that considers other attributes such as color, etc.

3.3 Person Identification

A rich "frequently asked question"-answering application relies on manual annotation or automatic detectors. For example, to answer the "who is this person" question in a movie, documentary, or home video, we need to know the people that are present in the each of the scenes. The challenge is to robustly identify persons from different views, distance, lighting conditions, in various background noise conditions. We have used automatic face and voice identification methods for this task [Li et al. 2001].

A person identification approach is constructed based on the joint use of visual and audio information. First, in the *analysis* phase we perform visual analysis for detection, tracking, and recognition of faces in video. Face trajectories are first extracted and the Eigenface method is used to label each face trajectory as one of the known persons in database. Due to the limitation of existing face recognition techniques and the complex environmental factors in our experimental data, the recognition accuracy is not high. Next we employ audio analysis, which operates by speaker identification. Both audio and visual analysis have their advantages under different circumstances, and we studied how to exploit the interaction between them for improved performance. In the fusion phase, two strategies have been employed. In the first strategy, the *audio-verify-visual fusion* strategy, speaker identification is used to verify the face recognition result. The second strategy, the *visual-aid-audio fusion* strategy, consists of using face recognition and tracking to supplement speaker identification results.

3.4 Generic Detectors

We have developed a number of other detectors that in the future could be integrated into the system. For example, a natural scene vs. graphics detector can be used to differentiate information-rich natural image from a relatively information-poorer synthesized graphic [Naphade et al. 2002]. Other event detectors can be of great value, for example, highlights in sports, explosions, and music segment detectors.

4 Personalized News

Personalization provides one of the greatest benefits of the MyInfo application. However, it is also one of the greatest risks. From our focus group sessions we learned that users must feel that they are in control of the system, or they will quickly abandon the application. MyInfo personalizes the segmented and summarized news (see section 3.2) in two ways. For the Internet data, the system parses and extracts information from web sites as specified in the user profile. For the TV news stories, the application prioritizes individual stories based on both topics of interest listed in the user profile and on cues broadcasters use to indicate the importance of a story.

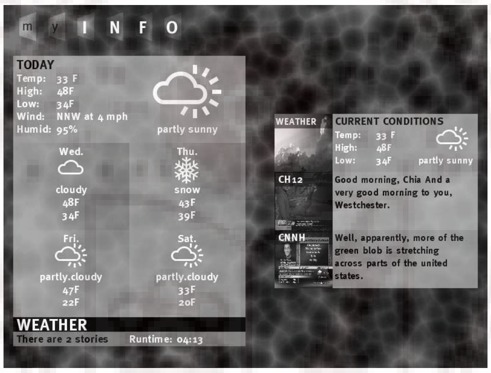
4.1 Personal Profile

In designing the personalization of the Internet content, we focused on providing users with maximum information using minimal input. The traffic section of the profile contains the user's home address. In addition, it contains a set of destinations and "hot spots". Destinations include towns or prominent structures such as malls, stadiums, airports, etc. Hot spots include points of constriction like bridges and tunnels, which notoriously have traffic delays. Once selected, the system extracts web traffic information on the specific hot spots and on the major roads between the users home and the selected destinations. For sports, the profile contains team names. As a default, once a zip code has been entered, MyInfo begins to track the teams in the local area. When users select the sports zone, they see the latest scores and upcoming events for the monitored teams. For financial news, the profile contains the names of indexes, stocks, and mutual funds. For local events, it contains a set of keywords describing events users like most. The system prioritizes a listing of local events based on their distance from the user's home and the match to the keywords.

Personalized web information improves the TV news experience in two ways. First, it makes it faster for users to get information. For example, if users just want to know the current temperature or a stock price, the information is a single button push away. They don't even have to wait for the news anchor to tell them. Second, the Internet-extracted data is of more benefit to users than traditional TV content because it is much more relevant to their information needs. For example, the local TV news can only afford to devote so many minutes of broadcast time each day for traffic information. This prevents them from relaying information on all routes during each clip. Often, they skip the specific routes that are important to an individual user. The personalized web data takes care of this issue, while still allowing users to see the traditional TV traffic news to get an overview of the worst spots in their area.

MyInfo collects personalized web information through Information Extraction (IE) [Janevski et al. 2002]. IE is a process of extracting structured data from a relatively

unstructured input (e.g. English free-form texts or a web page). An IE system contains one or more *rules* that describe generic ways of processing natural language texts and extracting the correct data. For a specific input document and output data, the rules are instantiated as *tasks*. In general, rules have a set of parameters and specific parameter values that define a task.
Our system obtains personalized web data using specific web IE tasks that depend on the user profile. For example, the current weather conditions and a weather forecast can be obtained from the weather.com site with a URL that contains a particular zip code. Here, the rule specifies how to extract the weather information, but the zip code defines a particular task with a customized page location. Similarly, the traffic information can be found for a geographic area. Once an area has been selected, information for the user's hot spots is extracted by instantiating an IE task with a list of hot spots of interest and for the particular time of the day/week. The system can extract information on different routes for the morning/afternoon and weekend commutes.


m y I N F O

TODAY
Temp: 33 F
High: 48F
Low: 34F
Wind: NNW at 4 mph
Humid: 95%
partly sunny

Wed. cloudy 48F 34F
Thu. snow 43F 39F
Fri. partly.cloudy 47F 22F
Sat. partly.cloudy 33F 20F

WEATHER
There are 2 stories Runtime: 04:13

WEATHER
CURRENT CONDITIONS
Temp: 33 F
High: 48F
Low: 34F
partly sunny

CH12
Good morning, Chia And a very good morning to you, Westchester.

CNNH
Well, apparently, more of the green blob is stretching across parts of the united states.

Figure 3. The Weather content zone screen.

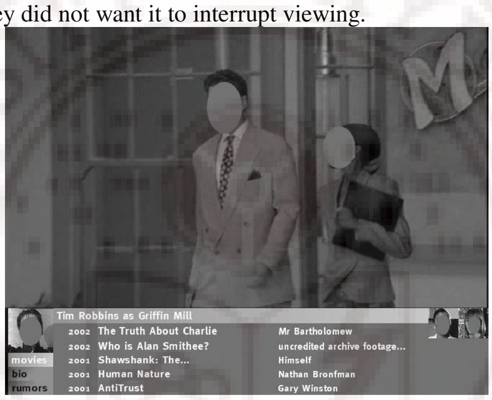
4.2 Prioritization

MyInfo further personalizes TV news stories by prioritizing them. The system tries to balance topics users have specified in the profile with cues the broadcaster uses to indicate a story's importance. Use of the broadcaster information is very important. Users have no way of predicting every kind of news story that might be important to them. They may know they are interested in China, and therefore add this topic to their profile. However, it is hard for them to predict major events that affect many people, such as earthquakes, elections, etc. By allowing the broadcasters' editorial content decisions to play a role, users get a much better mix of information.

MyInfo determines broadcaster importance of a story from three different characteristics: (i) duration, (ii) location in the newscast, and (iii) teaser announcing a story will play later in the broadcast. Since broadcast time is limited, a longer story will be more important. Location in the broadcast and use of a teaser are subtler. The most important TV news stories generally appear at the beginning; however, broadcasters place other stories they think many viewers want to see at the end. Then they use teasers to keep the viewers from switching channels. At this time, our measurement of broadcaster importance is far from perfect; however, using this value in conjunction with stories that match the user profile creates a richer viewing experience.

5 InfoSip: Personalized/Augmented Narrative

InfoSip is an example of a "frequently asked questions" answering application. It unobtrusively serves actor information related to the scene. During focus group testing, participants indicated that they wanted supplementary information for movies and TV shows, but they did not want it to interrupt viewing.



Tim Robbins as Griffin Mill

2002	The Truth About Charlie	Mr Bartholomew
2002	Who is Alan Smithee?	uncredited archive footage...
2001	Shawshank: The...	Himself
2001	Human Nature	Nathan Bronfman
2001	AntiTrust	Gary Winston

Figure 4. Screenshot of the InfoSip showing actor information.

With InfoSip, users interact by selecting a specific query. InfoSip uses predefined categories of questions/buttons such as "who", "where", "what", "when", "why", and "how much". For example, users press the "who" button to ask "who's that actor?". This displays a list of all of the actors in the current scene using annotated data from person identification (see section 3.3) and supplemental data about each one obtained through web IE (Figure 4). Web IE works better than metadata loaded on products such as DVDs, because it always contains the latest information. Filmography information is personalized, based on the user's viewing history. Highlighting the movies in which users have seen this actor increases the chances that they will remember why this person looks familiar.

6 Conclusions

In this paper we presented personalization aspects for content augmentation applications that combine content from multiple media sources. Our pilot applications MyInfo and InfoSip show promise that the technology has come of age. Web Information

Extraction and the segmentation, indexing, and retrieval of video at a subprogram level offer new tools for TV personalization developers. These technologies can improve the viewing experience by better understanding the TV content and by retrieving related material that is more focused at individual users. In the future we plan to evaluate our pilot applications with real users, continue developing video and web retrieval and extraction algorithms and generate more content augmentation concepts.

References

Ardissono, L., Portis, F., and Torasso, P.: Architecture of a System for the generation of personalized Electronic Program Guides. Proceedings of UM '01: Workshop on Personalization in Future TV, Sonthofen, Germany, (2001)

Brown, E.W., and A. R. Coden, A.R.: Capitalization Recovery for Text. A. R. Coden, E.W. Brown, and S. Srinivasan (eds.), Information Retrieval Techniques for Speech Applications, Springer, New York, (2002) 11-22

Buczak, A., Zimmerman, J., Kurapati, K.: Personalization: Improving Ease of Use, Trust, and Accuracy of a TV Show Recommender. Proceedings of AH '02 Workshop on Personalization in Future TV, Malaga, Spain, May (2002) 1-10

Cotter P. and Smyth, B.: PTV: Intelligent Personalized TV Guides. Proceedings of AAAI '00, Austin, TX, USA, (2000) 957-964

Das D. and ter Horst, H.: Recommender Systems for TV. Recommender System, Papers from the 1998 Workshop, Madison, WI. Menlo Park, CA: AAAI Press, (1998) 35-36

Dimitrova, N. Agnihotri L., and Jasinschi R., Temporal Video Boundaries, in Video Mining, A. Rosenfeld, D. Doermann, D. Dementhon eds., Kluwer Academic Publishers, 2003, 63-92.

IBM Intelligent Miner for Text™

Janevski A. and Dimitrova, N.: Web Information Extraction for Content Augmentation. Proc. IEEE Int. Conference on Multimedia and Expo, Switzerland, Aug., (2002), pp 389-392

Jasinschi, R., Dimitrova, N., McGee, T., Agnihotri, L., Zimmerman, J. Video Scouting: An Architecture and System for the Integration of Multimedia Information in Personal TV Applications, ICASSP, Salt Lake City, UT, USA, May 7-11 (2001) 1405-1408

Li, D., Wei, G., Sethi, I.K., and Dimitrova, N.: Person Identification in TV Programs, Journal on Electronic Imaging, special issue on Storage, Processing and Retrieval of Digital Media, October (2001), pp 930-938

Merlino, A., Morey D., Maybury, M.: Broadcast navigation using story segmentation. Proceedings of ACM MM '97, ACM Press, November (1997) 381-388.

Maybury, M. (ed.) February 2000. News On Demand. CACM. Volume 43(2), pp 32-34.

Naphade, M. R., Kozintsev I., and Huang, T.S.: Factor Graph Framework for Semantic Video Indexing. IEEE Trans. on Circuits & Systems for Video Technology, 12(1) (2002) 40-52

Zimmerman, J., Marmaropoulos, G., and van Heerden, C. Interface Design of Video Scout: A Selection, Recording, and Segmentation System for TVs. Proceedings of Human Computer Interaction Intl (HCII) New Orleans, LA, USA, August 5-10, (2001) 277-281

User Modeling as a goal in itself: an Artificial Companion for the Elderly

Judith Masthoff and Kees van Deemter

University of Brighton, UK

Judith.Masthoff@brighton.ac.uk
Kees.van.Deemter@itri.brighton.ac.uk

Abstract. User modeling tends to be seen as a "means to an end": something that is needed to adapt systems to users, and that we want to make as painless and effortless for the users as possible. This paper describes how user modeling can be a goal in itself, a pleasurable activity that provides users with somebody to have meaningful conversations with. The example of interactive Television as an artificial companion for the elderly is explored. In addition, it is shown how techniques for knowledge editing as put forward by the Natural Language Generation community can help to make user modeling easy and pleasant.

1 Background

The importance of social support
The elderly constitute an increasing proportion of the population in the UK and Europe. Increasingly, elderly people live alone, and have a small social network [1]. A poor social network carries an increased risk of developing dementia [2] and depression [3], and has a life-shortening effect [4]. Conversely, friendship and conversation have been shown to have a clear positive impact on health and well-being [5].

Television plays a central role in the life of an elderly person: a recent study in the UK shows that older people are the heaviest consumers of television, with the 65+ group watching a staggering average of 5 hours 14 minutes a day [6]. The same study shows that almost all of those aged 75+ watch television every day (96%). Television is watched to alleviate loneliness [7], and lonely people watch more television [8]. However, this is a vicious circle, as television watching tends to *reduce* social involvement [9]. Conversely, people who indicate being happy when engaged in social activity, tend to report being bored and unhappy when watching television [10].

Interactive TV as an artificial companion
In this paper, we introduce a possible future system called Artificial Companion for the Elderly (ACE). ACE is an attempt at turning the negative effects of TV watching around, by changing the TV from a piece of broadcasting technology into a stimulating *companion* whom you can have conversations with. The advent of digital television, and in particular its potential for interactivity provides the opportunity to

pursue this idea. It opens up the possibility of personalized, adaptive experiences delivered in such a way that the viewers' awareness of the world around them is enhanced and their general alertness is boosted, which will in turn make it easier for them to participate in meaningful interactions with other people. Our target user group is the (very) elderly user.

But surely, only a human companion can help?
There is evidence that pets alleviate depression in older adults, particularly those with minimal social support [11]. There has been quite some work on artificial pets: for example, the Sony dog, the Omron cats, the Tamagotchi, the Norns, the PlayStation "Pet in TV". These pets build up a simple repertoire of behaviors, based on how you take care of them. Artificial pets are being used in therapy [12]. Of course, a pet cannot replace a human, and neither can an interactive TV. We will, however, work towards the construction of an artificial but human-like companion, somebody to have (limited) two-way conversations with.

Conversations, not done before?
The last decades have seen a considerable amount of research on dialogue systems, based on a variety of input and output media. These systems tend to focus on performance of a concrete task, such as making travel reservations [13], or obtaining information on bank account balances [14]. Useful though systems of this kind may become, their aim is not to entertain, educate, and integrate their users. ACE, by contrast, does not support the elderly through performing a task, but through conversation. It needs to keep them entertained, show interest and curiosity, and encourage them to be actively engaged. The well-known example of the *Eliza* system (which models a Rogerian psychotherapist, [15]) shows that simple mechanisms can go a surprisingly long way. ACE will go beyond *Eliza* (and its 'chatterbot' heirs) by tapping into recent research in gerontology, personalization, human-computer interfaces (HCI) and natural language processing (NLP). A particular source of inspiration is the recent wave of work on (embodied) conversational agents, which try to mirror the way in which personality and emotion are expressed linguistically (and through speech, gesture, gait, gaze, facial expression). See [16, 17, 18] for examples.

HCI / Gerontechnology challenges.
Difficulties in handling computers and other modern technology increase with age [19]. This is partly due to a change in cognitive and perceptual abilities with age [20], and partly to the phenomenon of *technology generations*: the experience with technology in adolescence and young adulthood influences how users approach current technology [20]. A key challenge therefore is to keep interactions with the TV manageable even for elderly users. Our take on this problem will be to structure the interaction with the TV by letting ACE generate questions that the user will subsequently answer (using either speech or a remote control), in line with work on so-called conversational interfaces [21]. In addition, ACE will need to be able to cope with a large variation in perceptual and cognitive abilities. It will build on work done on "Design for all" (also called "Universal Access"), e.g. [22]. Other major issues will

be the integration of ACE within the normal TV broadcast, and the encouragement to activity of normally passive viewers.

2 Envisaged Application

ACE could support conversations in a narrow domain, such as the weather, which may be modeled thoroughly. This approach would, however, tend to lead to very limited, impersonal, conversations, which would not provide the viewers the support and entertainment we have in mind. A more promising approach would be to concentrate on a wide domain that is of great interest to many elderly users: *the daily news* [6]. If we engage the elderly user with the daily news, their awareness of the world around them and their ability to interact with their peers will be enhanced. The difficulty is that it will be impossible to enter all the relevant facts in a format that ACE can understand. To get around this problem, we will -- roughly in the spirit of the Semantic Web -- allow existing newsreel to be selectively *annotated*, based on a dedicated ontology. For each event reported, the ontology might allow, for example, specification of the *type* of the event (e.g., political/sporting/etc.), its *beginning and end* time, whether it is generally viewed as *good or bad*, and so on. A new annotation tool based on such an ontology would allow a specialist to make aspects of the daily news transparent to ACE, allowing it to ask intelligent questions about it even though its understanding is limited, e.g., 'Has an earthquake of this magnitude occurred earlier during your lifetime?' Needless to say, as much as possible of the annotation will be performed automatically (e.g., [23]).

3 Modeling the elderly viewer

To be able to relate the news to the interests of the elderly user, the television will need to build up a detailed user model. This will not be done in the usual indirect way (i.e., by monitoring the user's behavior), but by involving the elderly user in the construction of a user's knowledge base (UKB) that contains their family tree and/or other aspects of their personal history. As a result, ACE will often know the answers to the questions that it asks (e.g., the user was alive during the 1995 earthquake in Kobe), enabling it to respond intelligently. Crucially, the construction of the UKB will mimic human conversation (with ACE asking questions), aiming to be both user-friendly and entertaining. At any time, there will be many holes in the UKB, and therefore multiple questions competing to be asked. ACE will carefully select which questions it asks:

- It will not ask too many questions at any time, and relate them as much as possible to the broadcast. For instance, if a news item is related to younger people, then ACE could ask whether the viewer has children. If a news item mentions France, and if the viewer is British, then ACE could ask whether the viewer has ever visited France.
- It will mainly ask questions that make a substantial contribution to extending its knowledge base. As observed in [24], for some questions the answers are very

likely. For instance, when asking a British viewer whether they have ever visited Burundi, the answer is very likely to be ‘no’, and therefore the answer is not likely to add much to the UKB. Similarly, for viewers in the US the ‘have you ever visited France’ question might be replaced by a ‘have you ever been to Europe’ question. In contrast to [24], however, for ACE it is not always bad to ask questions it can guess (or even knows) the answer to. For instance, in the earthquake example mentioned above, ACE knows the viewer was alive during the Kobe earthquake, and therefore the answer to the question ‘Has an earthquake of this magnitude occurred earlier during your lifetime?’ should be ‘yes’. Asking this question makes the viewer think, which is a good thing in a conversation. Besides, the viewer might not remember the Kobe earthquake, and a ‘no’ response could therefore lead to an interesting discussion. In a sense, some questions could serve to add information to the UKB about whether the viewer remembers something or not. A lack in memory could indicate a lack of interest in a topic.

- Follow-up questions could be asked depending on the viewer’s response. For instance, if they have visited France, ACE could ask when was the last time they were over there. This is related to maintaining narrative flow. Our conversations do not tend to jump from topic to topic: rather we speak some time about one topic before switching to another. Topics that are salient (by just having been discussed in the broadcast, or in a previous question) will have a higher likelihood of being asked about. If the answer to the question about visiting Europe was ‘no’, a question could be asked about ‘the viewer ever having been abroad’.
- The information already in the knowledge base will allow ACE to personalize its questions. For instance, if ACE already knows that the viewer has a granddaughter of say 16 years old, then it can ask whether it is expected that she will go to university (after a news item on university tuition fees).
- It will only ask questions that are easy to answer using the television remote control (so, questions that have a limited number of possible answers, or have a number as answer). This means that it cannot just ask the viewer a question like ‘What job does your son have?’ If it wants to get this information, then it will need to ask gradually over time a number of questions such as ‘Does your son do manual work?’ Which question it asks will depend on the likelihood of the answer, which may depend on the knowledge it has already gathered. For instance, if it knows the son studied at university, then it is less likely to start with the ‘manual work’ question.

Letting a user construct a model of herself can be viewed as a special case of *knowledge editing*, which is defined as the task of building up, extending and modifying a knowledge base. (The knowledge base in this case is the UKB of course.) Knowledge editing has been made easier in recent years by the advent of a new paradigm, which uses Natural Language Generation [25] to clarify the content of the knowledge base to the user and, more importantly in the present setting, to offer the user options for modifying it. Probably the most elaborate version of this idea is the approach called What You See Is What You Meant (WYSIWYM), developed by Richard Power and colleagues at the University of Brighton [26]. WYSIWYM has been applied across a range of applications, including Question Answering (in a domain of Maritime Law) [27] and Document Authoring (in a pharmaceutical domain among other things) [28]. The idea of WYSIWYM is to express the content of the knowledge base in natural language text, and to allow users to modify the knowledge base by interacting with this text. In the case of our application, for example, the UKB could contain the information that Mrs. Smith has a son (∃x: Son(MrsSmith,x)), which a language generator might simply express as 'You have a son'. (Note that the same information would be expressed differently when said to someone who is not the parent.) A sentence of this kind, which expresses information in the UKB, is called a feedback text. Since sons have birth dates, the system can now offer the user the choice of entering the birth date of the son. We intend to go beyond this "standard" use of WYSIWYM in the following ways:

- Following [29], options for modifying the UKB will be offered in the form of questions to the user. For example, instead of offering the user a menu-based interface for specifying the son's birth date, the system will ask a question, e.g., 'In what year was your son born?' The user can then use the remote control to enter the year whereupon the system can use its understanding of the situation to interpret the response (e.g., '57' will be interpreted as 1957).
- Although experiments are confirming that WYSIWYM makes knowledge editing easier, the present application requires more than ease of use: to achieve the goals of section 1, knowledge editing has to become pleasurable and stimulating. For this reason, any information expressed by the system needs to be expressed colloquially, and preferably in a style mirroring that of the user (e.g., 'When was your son borne? In what year was that?'). This can be done by exploiting the UKB, which can contain information about the user's dialect, personality, etc, in the same way as it contains information about their family tree. Thus, the UKB will influence the form as well as the content of the things that the system says.

The resulting system would have some important similarities with recent Dialogue Systems, which allow users to enter queries by allowing them to answer questions [30]. The common element here is to give the system more initiative than was done by older Dialogue Systems, thereby bypassing the need for (error prone) Natural Language Interpretation. A difference with most current Dialogue Systems, however, is that ACE would tailor its output (such as the linguistic expression of style, personality and emotion) to a specific user. Experiments will have to determine the optimal 'mix' of modalities: current WYSIWYM systems use written text throughout, but it may well prove more effective, in the present application, to use speech when asking questions of the user (cf. [30]), especially if the user is visually impaired. Similarly, speech input might prove more effective than input via remote control.

4 Conclusions

Traditionally user modeling has been used to *enable* adaptation of content, presentation and interaction (nine different *purposes* of user modeling are mentioned in [31]). For instance, Adaptive Hypermedia and Recommender systems (as used in Electronic Program Guides) rely on user modeling. Smart approaches to user modeling have been developed, involving stereotypes, observation of user actions, and explicit user feedback. Having *conversations* to build up a user model is not new in itself: it has, for instance, been applied in [32] to improve the interaction with a system that recommends restaurants. Systems of this kind perform user modeling to make the interaction with the user (e.g., the making of the restaurant recommendation) as effective as possible. We propose to do the opposite, using the conversation (and hence the user modeling) is a goal in itself: pleasant conversations are the goal. As a consequence, the domain has to be a lot wider, and natural language generation has to be more expressive than in more conventional approaches.

The application of an Artificial Companion for the elderly has been introduced as an example of such a new class of applications. It has been indicated how the naturalness of the conversations can be achieved by using a combination of natural language generation techniques and personalization.

We have only just started this work, and the user modeling and its user interface will need to be specified in more detail. Additionally, it will be vital to empirically establish the impact of ACE. In particular, it needs to be investigated whether ACE is easy enough to have a conversation with, whether it really feels like having a conversation with a person (e.g., involving a version of the Turing test), whether the conversation is on a deep enough level, and whether the interaction has the long-term effects that we are aiming at (i.e., enhanced mental activity and social involvement).

References

1. Knipscheer, C.P.M., Jong-Gierveld, J. de, Tilburg, T.G. van, and Dijkstra, P.A. (Eds.) (1995). Living arrangements and social networks of older adults. Amsterdam: VU University press.
2. Fratiglioni L., Wang H.X., Ericsson K., Maytan M., and Winblad B. (2000). Influence of social network on occurrence of dementia: A community-based longitudinal study. The Lancet 355(9212): pp 1315-1319.
3. Roberts, R. E. (1997). Prevalence and correlates of depression in an aging cohort: The Alameda County Study. Journal of Gerontology--Series B: Psychological Sciences & Social sciences, 52B, pp252-258.
4. Blazer, D.(1982). Social support and mortality in an elderly community population. American journal of epidemiology, 115. pp 684-694.
5. Carter, P., and Everitt, A. (1998). Conceptualising practice with older people: Friendship and conversation. Ageing and Society, 18 (1), pp79-99.
6. Hanley, P. (2002). The numbers game: Older people and the media. Independent Television Commission publication.
7. Rook, K..and Peplau, L. (1982). Perspectives on helping the lonely. In L. Peplau & D. Perlman (Eds.), Loneliness: A sourcebook of current theory, research and therapy. New York: Wiley.
8. Canary, D., and Spitzberg, B. (1993). Loneliness and media gratification. Communication research, 20, pp800-821.

9. Brody, G. (1990). Effects of television viewing on family interactions: An observational study. Family relations, 29, pp216-220.
10. Kubey, R, and Csikszentmihalyi, M. (1990). Television and the quality of life: How viewing shapes everyday experience. Hillsdale, NJ: Lawrence Erlbaum.
11. Garrity, T., Stallones, L., Marx, M., Johnson, T. (1989). Pet ownership and attachment as supportive factors in the health of the elderly. Anthrozoos, 3 (1), pp35-44.
12. Drexler, M. (1999). Pet robots considered therapy for elderly. http://www.cnn.com/TECH/ptech/9903/25/robotcat.idg/
13. Potamianos, A. Ammicht, E. and Kuo, H-K. (2000) Dialogue management in the Bell Labs communicator system. In proceedings of ICSLP-2000.
14. Bagga, A., Stein, G., Strzalkovski, T. (2000). FidelityXPress: a multi-modal system for financial transactions. Procs. of 6th conf. On Content-Based Multimedia Information Access.
15. Weizenbaum, J. (1965) A computer program for the study of natural language communication between man and machine. Communications of the ACM, 9, pp36-45.
16. Fleischman, M. and Hovy, E. (2002). Towards emotional variation in speech-based natural language generation. In Proceedings of the second Int. Natural Language Generation Conference (INLG-02), July. Harriman, NY, USA
17. Walker, M.A., Cahn, J.E, and Whittaker, S.J. (1996). Linguistic style improvisation for lifelike computer characters. In Proceedings of the AAAI Symposium on AI, ALife, and Entertainment. Portland.
18. De Carolis, B., Carofiglio, V, Pelachaud, C., and I.Poggi (2001). Interactive information presentation by an embodied animated agent. In Proceedings of the International Workshop on Information Presentation and Natural Multimodal Dialogue. Verona, Italy, Dec. 2001, pp 19-23.
19. Czaja, S. (1996). Aging and the acquisition of computer skills. In W.A. Rogers, A.D. Fisk, and N. Walker (Eds.), Aging and skilled performance: Advances in theory and applications, pp201-220. Mahwah, NJ: Lawrence Erlbaum.
20. Docampo-Rama (2001). Technology generations handling complex user interfaces. Ph.D. Thesis Eindhoven University of Technology. Eindhoven, The Netherlands.
21. Jellyvision. The Jack Principles. http://www.jellyvision.com/
22. Stephanidis, C., and Emiliani, P.L. (1999). 'Connecting' to the information society: a European perspective. Technology and Disability Journal, 10, pp21-44.
23. Humphreys, K. (1997). Event coreference for information extraction. Proceedings of the ACL/EACL 1997 Workshop On operational factors in practical, robust anaphora resolution. pp75-81. Madrid.
24. Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A., and Riedl, J. (2002). Getting to know you: Learning new user preferences in recommender systems. Proceedings of the Intelligent User Interface conference. pp. 127-134.
25. Reiter, E. and Dale, R. (2000). Building Natural language Generation Systems. Cambridge University Press, Cambridge, UK.
26. Power, R. and Scott,D. (1998). Multilingual Authoring using Feedback Texts. In Procs. of ACL/Coling conference, Montreal.

27. Piwek, P., Evans, R., Cahill, L. and Tipper, N. (2000). Natural Language Generation in the MILE System. In Proceedings of the 'IMPACTS in NLG' Workshop, Schloss Dagstuhl, Germany, July 2000.
28. van Deemter, K. and Power, R. (In press). High-level authoring of illustrated documents. To appear in Natural Language Engineering, 9 (2), June 2003.
29. Good. A. (2001). Improving the usability of WYSIWYM. Final year student project. University of Brighton.
30. Veldhuijzen van Zanten, G., Bouma, G., Sima'an, K., van Noord, G., and Bonnema, R. (1999). Evaluation of the NLP components of the OVIS2 spoken dialogue system. In: van Eynde, Schuurman and Schelkens (Eds.), Computational Linguistics in the Netherlands 1998, Rodopi Amsterdam, pp 213-229.
31. UM97 Reader's guide. http://w5.cs.uni-sb.de/~UM97/guide.html [Accessed 5-2-2003].
32. Thompson, C.A., Göker, M.H., and Langley, P. A personalized system for conversational recommendations. http://www.cs.utah.edu/nlp/thompson-apa.pdf. [Accessed on 5-5-2003].

US007051352B1

(12) **United States Patent**
Schaffer

(10) **Patent No.: US 7,051,352 B1**
(45) **Date of Patent: May 23, 2006**

(54) **ADAPTIVE TV PROGRAM RECOMMENDER**

(75) Inventor: **J. David Schaffer**, Wappingers Falls, NY (US)

(73) Assignee: **Koninklijke Philips Electronics N.V.**, Eindhoven (NL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/498,271**

(22) Filed: **Feb. 4, 2000**

(51) **Int. Cl.**
H04N 7/16 (2006.01)
H04N 5/445 (2006.01)

(52) **U.S. Cl.** **725/39**; 725/9; 725/14; 725/46; 706/45

(58) **Field of Classification Search** 725/38, 725/9–14, 44–47; 706/45; 705/10, 14
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,706,121	A	11/1987	Young	358/142
5,410,344	A	4/1995	Graves et al.	348/1
5,444,499	A	8/1995	Saitoh	348/734
5,534,911	A	7/1996	Levitan	348/1
5,585,865	A	12/1996	Amano et al.	348/731
5,635,989	A	6/1997	Rothmiller	348/563
5,704,017	A *	12/1997	Heckerman et al.	706/12
5,758,257	A *	5/1998	Herz et al.	725/116
5,758,259	A *	5/1998	Lawler	725/45
5,768,422	A *	6/1998	Yaeger	382/228
5,790,935	A *	8/1998	Payton	725/91
5,801,747	A *	9/1998	Bedard	725/46
5,828,419	A	10/1998	Bruettte et al.	348/563
5,848,396	A *	12/1998	Gerace	705/10
5,867,226	A *	2/1999	Wehmeyer et al.	725/46
5,867,799	A	2/1999	Lang et al.	707/1
5,880,768	A	3/1999	Lemmons et al.	348/1
5,987,415	A	11/1999	Breese et al.	704/270
6,005,597	A *	12/1999	Barrett et al.	725/46
6,317,722	B1 *	11/2001	Jacobi et al.	705/14
6,412,012	B1 *	6/2002	Bieganski et al.	709/232
6,637,029	B1 *	10/2003	Maissel et al.	725/46
6,727,914	B1 *	4/2004	Gutta	715/719
6,871,186	B1 *	3/2005	Tuzhilin et al.	705/26

FOREIGN PATENT DOCUMENTS

EP	0572090	12/1993
EP	0669760	8/1995
EP	0682452	11/1995
EP	0721253	7/1996
EP	0725539	8/1996
EP	0735749	10/1996
EP	0774866	5/1997
EP	0836320	4/1998
EP	0840504	5/1998
EP	0854645	7/1998
FR	2726718 A1 *	5/1996
GB	2289782	11/1995

(Continued)

OTHER PUBLICATIONS

Billsus et al, "Learning Probabilistic User Models", http://www.dtki.de/~bauer/um-ws/Final-Versions/Billsus/ProbUserModels.html.

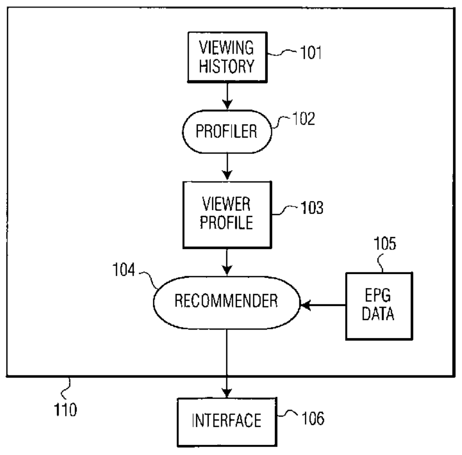
(Continued)

Primary Examiner—Chris Grant
Assistant Examiner—Son P. Huynh
(74) *Attorney, Agent, or Firm*—Edward W. Goodman

(57) **ABSTRACT**

A system for recommending television programs makes use of probabilistic calculations and a viewer profile to create a recommendation. The probabilistic calculations preferably are in the form of Bayesian classifier theory. Modifications to classical Bayesian classifier theory are proposed.

36 Claims, 4 Drawing Sheets



FOREIGN PATENT DOCUMENTS

GB	2325537		11/1998
WO	WO9413107		6/1994
WO	WO9627840		9/1996
WO	WO 9746006	A1 *	12/1997
WO	WO9748228		12/1997
WO	WO9748230		12/1997
WO	WO9821878		5/1998
WO	WO9853609		11/1998
WO	WO9856173		12/1998
WO	WO9901984		1/1999

OTHER PUBLICATIONS

Breese et al., "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI, Jul., 1998, Morgan Kaufmann Publisher.

Pazzani et al., "Learning and Revising User Profiles: The Identification of Interesting Web Sites", Machine Learning 27, pp. 313-331, 1997.

* cited by examiner

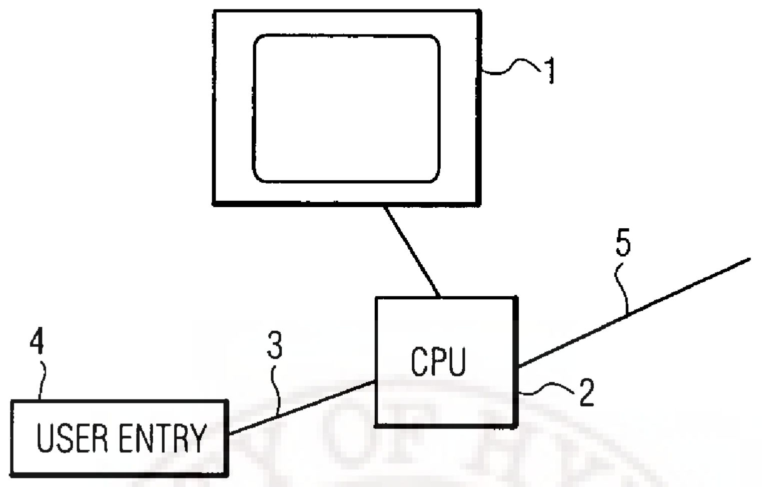

1
5
4
3
CPU
2
USER ENTRY

FIG. 1

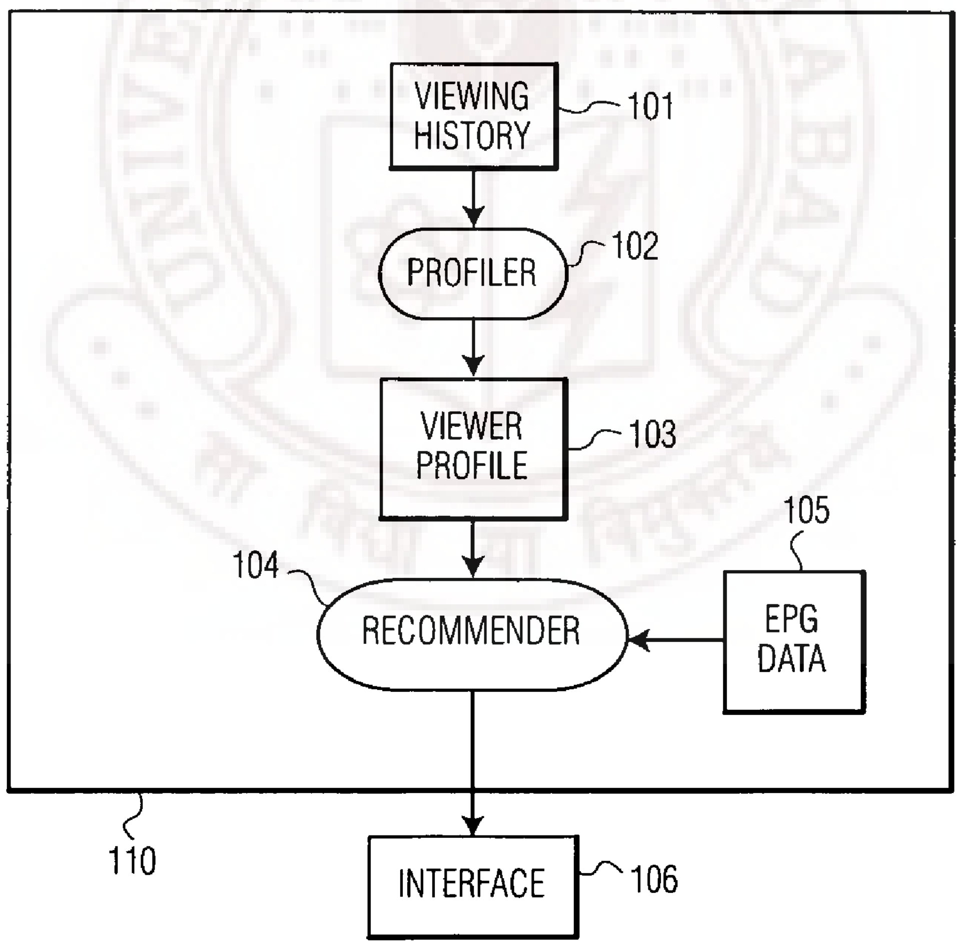

VIEWING HISTORY
101
PROFILER
102
VIEWER PROFILE
103
105
104
RECOMMENDER
EPG DATA
110
INTERFACE
106

FIG. 2

```
LET S BE ALL SHOWS NOT WATCHED IN THE 7 DAYS PRIOR TO
        AND INCLUDING THE CURRENT DAY.
LET N BE 1.

FOR EACH TV SHOW WATCHED
        ENTER THE WATCHED SHOW IN THE VIEWING HISTORY
        AS A POSITIVE EXAMPLE.
        SELECT A SUBSET S OF SHOWS NOT WATCHED
        SELECT AT RANDOM N SHOWS FROM SET S AND ENTER
        THEM IN THE VIEWING HISTORY AS NEGATIVE EXAMPLES.
        IF AN EXPLICIT VIEWER PROFILE IS AVAILABLE, THEN THE
        RANDOM SELECTION CAN BE BIASED AWAY FROM SHOWS
        "LIKED" AND TOWARDS SHOWS "NOT LIKED."
```

FIG. 3

FIELD	DESCRIPTION
$DATE	YYYYMMDD
$AIR_TIME	HHMM (E.G. SOME VALUE IN THE RANGE 0000-2359)
$STATION_SIGN	4 CHARACTERS (E.G. WABC)
$TITLE	120 CHARACTERS (E.G. "ANTIQUES ROADSHOW")
$DESC	120 CHARACTERS (E.G. "SKULLY VISITS ALIEN SPACECRAFT")
$GENRE	20 CHARACTERS (E.G."SCIENCE FICTION")
$ACTORS	120 CHARACTERS (E.G. "JOHN DOE, JANE DOE")
$DIRECTORS	120 CHARACTERS (E.G. "JOHN HITCHCOCK")
$HOSTS	120 CHARACTERS (E.G. "JOHN HOST")
$PRODUCERS	120 CHARACTERS (E.G. "JANE RICH, JOHN MONEYBAGS")
$WRITERS	120 CHARACTERS (E.G. "JOHN POET")

FIG. 4

TOTALIPROGRAMS	55	55
DAYTIMEIMON2100	5	0
DAYTIMEIMON2200	6	1
DAYTIMEITUE2200	4	1
DAYTIMEIWED2000	4	0
DAYTIMEIWED2200	6	0
STATIONIWABC	10	1
STATIONIWNBC	30	0
STATIONIWNYW	13	0
TITLEI20/20	5	0
TITLEIDATELINE NBC	11	0
TITLEIMLB PLAYOFFS	10	0
TITLEIPAID PROGRAMMING	0	5
GENREIANIMATED	0	4
GENREIBASEBALL	13	1
GENREICOMEDY	4	8
GENREIFOOTBALL	4	0
GENREIMAGAZINE	18	0
GENREINEWS	22	4
GENREIREALITY	4	1
GENREISITUATION	3	5
GENREISPORTS EVENT	15	2
GENREITALK	18	3
DESCIALLY	6	0
DESCIIF	6	0
DESCIMCBEAL	4	0
DESCIALTERNATE	5	0
DESCIGAME	4	0
DESCILINEUP	5	0

DESCILOCAL	4	0
DESCINECESSARY	6	0
DESCIPRIMETIME	5	0
DESCIPROGRAMMING	4	0
TV RATINGITV14	6	3
TV RATINGITVG	0	4
SEX RATINGIN	51	52
LANGUAGE RATINGIN	51	50
ADVISORYIADULT SITUATIONS	0	8
ADVISORYILANGUAGE	0	5
MPAA RATINGIR	0	4
STAR RATINGI**+	0	5
PROGRAM LANGUAGEIENGLISH	51	52
COUNTRY OF ORIGINIUNITED STATES	0	7
EPISODE TITLEIGAME	7	0
EPISODE TITLEILEAGUE	4	0
EPISODE TITLEINEW	5	0
EPISODE TITLEISERIES	9	0

FIG. 5

$$T = k(C+) + k(C-)$$
$$P(C+) = k(C+)/T$$
$$P(C-) = k(C-)/T$$

FIG. 6A

$$P(fi|C+) = k(fi|C+)/k(C+)$$
$$P(fi|C-) = k(fi|C-)/k(C-)$$

FIG. 6B

$$P(C+|x) = P(x|C+)P(C+)/P(x)$$
$$P(C-|x) = P(x|C-)P(C-)/P(x)$$

WHERE

$$P(x) = P(x|C+)P(C+) + P(x|C-)P(C-)$$

$$P(x|C+) = \prod_{i=1}^{n} P(fi|C+)^{xi} \ (1-P(fi|C+))^{1-xi}$$

n = number of features in profile
fi = the I^th feature in the profile is a bit string
of length n, where the I^th bit indicates the
x = {0,1}^n presence (1) or absence (0) of
feature fi in the program

FIG. 6C

ADAPTIVE TV PROGRAM RECOMMENDER

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to recommending television shows based on a user profile.

2. Description of the Related Art

U.S. Pat. No. 5,758,259 shows a method for identifying a preferred television program based on a "correlation" between the program and predetermined characteristics of a user profile. The term "correlation" as used in the patent does not appear to relate to the mathematical concept of correlation, but rather, is a very simple algorithm for assessing some similarity between a profile and a program.

SUMMARY OF THE INVENTION

It is an object of the invention to improve techniques of automatic program recommendation.

This object is achieved by using a probabilistic calculation, based on a viewer profile. The probabilistic calculation is preferably based on Bayesian classifier theory.

The object is further achieved by maintaining a local record of a viewer history. The local record is preferably incrementally updatable. The local record is advantageous for privacy reasons, and can be contrasted with methods, such as collaborative filtering, which would require viewer history information to be uploaded to a central location. The use of incremental updates is advantageous in minimizing storage requirements.

It is a still further object of the invention to improve the classical Bayesian classifier technique.

In one embodiment, this object is achieved by noise filtering.

In another embodiment, this object is achieved by applying a modified Bayesian classifier technique to non-independent feature values.

Further objects and advantages of the invention will be described in the following.

Bayesian classifiers are discussed, in general, in the textbook of Duda & Hart, "Pattern Recognition and Scene Analysis" (John Wiley & Sons 1973). An application of Bayesian classifiers to document retrieval is discussed in "Learning Probabilistic User Models", by D. Billsus & M. Pazzani.

BRIEF DESCRIPTION OF THE DRAWING

The invention will now be described by way of non-limiting example with reference to the following drawings, in which:

FIG. **1** shows a system on which the invention may be used;

FIG. **2** shows major elements of an adaptive recommender;

FIG. **3** shows pseudo code for a viewing history generator;

FIG. **4** shows a table of key fields;

FIG. **5** shows a viewer profile;

FIG. **6***a* shows a prior probability calculation;

FIG. **6***b* shows a conditional probability calculation; and

FIG. **6***c* shows a posterior probability calculation.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. **1** illustrates hardware for implementing the invention. The hardware includes a display **1**, some type of processor **2**, some type of user entry device **4** connected to the processor **2** via some type of connection **3**, and some type of link **5** for receiving data, such as television programming or Electronic Programming Guide ("EPG") data. The display **1** is commonly a television screen, but could be any other type of display device. The processor **2** may be a set-top box, a PC, or any other type of data processing device, so long as it has sufficient processing power. The user entry device **4** may be a remote control unit and the connection **3** may be an infrared connection. If the processor is a PC, the user entry device will commonly be at least plural, e.g., a keyboard and mouse. The user entry device may also be touch sensitivity on the display. The connection **5** to the outside world could be an antenna, cable, a phone line to the Internet, a network connection, or any other data link. Equally well, connection **5** could connect to a memory device or several memory devices.

FIG. **2** illustrates major elements of an embodiment of an adaptive recommender. These elements preferably reside as software and data in a medium **110** readable by a data processing device, such as CPU **2**. The elements include a viewing history data structure **101** that gives input to profiler software **102**. The profiler software, in turn, produces the viewer profile **103**. The terms "user profile" and "viewer profile" shall be used interchangeably herein. The viewer profile serves as an input to recommender software **104**. The recommender software also uses, as an input, the EPG data structure **105**, that contains features describing each show, such as title, channel, start time and the like. An output of the recommender **104** appears on a user interface **106** where a user can interact with it.

The viewer history data structure includes selected records from the EPG database. EPG databases are commercially available, for instance, from Tribune Media Services. Those of ordinary skill in the art may devise other formats, possibly with finer shades of description. The selected records minimally correspond to TV shows watched by the viewer. It is assumed that these records have been deposited in the viewing history data structure **101** by software that is part of the user interface and knows what shows the viewer has viewed. Preferably, the software allows recording of a user watching more than one show in a given time interval, as users often switch back and forth during commercials, and so forth. Preferably, also, the software records a program as watched regardless of how long it was watched; and if a show was watched or whether it was taped for later viewing.

The preferred viewing history format assumes the presence of both positive and negative records in the viewing history. This is needed because the goal is to learn to differentiate between the features of shows that are liked and those not liked. FIG. **3** shows pseudo code for collecting the viewing history.

Let the notation C+ denote the set of positive (i.e., watched) shows and C− denote the negative (i.e., not watched) shows.

The viewer profile includes a number of feature value counts. These counts are incremented whenever new entries are deposited in the viewer history. Usually, each program has several feature values. Accordingly, the deposit of a program in the viewer history causes the update of counts associated with all feature values associated with that program.

The incremental updatability of this type of history is advantageous because it allows for ongoing adaptation of the viewer history without a large amount of storage or computing effort being required.

In addition to the count of the number of positive and negative entries (k(C+), k(C−)), a count of occurrences of individual features is also kept among the positive and negative examples (k(fi|C+), k(fi|C−)) where fi denotes feature i and k(fi|C+) denotes the number of shows in set C+ that possess feature fi. The feature set includes entries in the EPG records extracted from selected key fields, an example of which is shown in Table 1 which is FIG. **4**.

A partial example of an embodiment of such counts is presented in Table 2, shown in FIG. **5** to illustrate the idea. The list as presented in FIG. **4** has six columns to save space, but, in fact, the list has only three columns, with the bottom part of Table 2 being presented next to the top part. Each row of Table 2 has four pieces of data, i.e., a feature type and a feature value in the first column, a positive count in the second column, and a negative count in the third column. The positive count indicates the number of times a program having that feature value has been watched. The negative count indicates the number of times a program having that feature value has not been watched.

A television program schedule normally includes several, if not many, programs for every time slot in every day. Normally, the user will only watch one or two of the programs in any given time slot. If the viewer profile contains a list of ALL the programs not watched, the number of programs not watched will far exceed the number of programs watched. It may be desirable to create a method for sampling the programs not watched. For instance, as the processor assembles the viewer profile, the processor may chose a single not-watched program at random from the weekly schedule as a companion for each watched program, as suggested in the pseudo-code of FIG. **3**. This design will attempt to keep the number of positive and negative entries in the viewing history about equal so as not to unbalance the Bayesian prior probability estimates, discussed below.

It is not generally desirable to choose a companion program from the same time slot as a watched program. Experiment has shown that the combined time and day feature value is typically the strongest or one of the strongest predictors of whether a particular program will be preferred. Thus, another program at the same time as the watched program may well be a second or third choice program, while a program at a totally different time may be very undesirable. Accordingly, it is preferred to choose the companion program at random from the program schedule of the entire week that includes the watched program.

Since time and day feature values for a program are often so important in determining whether a program will be of interest to a user, it is typically undesirable to consider two programs of identical content to be the same if they are shown on different days and/or at different times. In other words, a particular episode of a series may be strongly preferred if it is shown at 8 p.m. on Tuesday, while the same episode of the same series may be completely undesirable if it is shown at 10 a.m. on Monday. Thus, the episode at 10 a.m. should be considered a different program from the episode at 8 p.m., even though the content of the two are identical.

As more and more shows are viewed, the length of the profile will tend to grow larger and larger. To combat this,

and to keep the focus on features that are effective discriminators, the following are recommended:

periodic reviews of the features in the viewer profile, and removal of words that appear to be frequent and not very discriminating.

In general, those of simple tastes, e.g., those who only like to watch football, will be fairly easy to recommend for after taking of a viewer profile for a relatively short time. For those of more complex preferences, it will take longer for the viewer history to be sufficiently meaningful to make good recommendations. These latter people, however, are those who are probably most in need of a recommendation.

In the final analysis, viewer histories will always be ambiguous. Recommendations of shows based on such histories will always contain a margin for error. The recommendations can, at best, be said to have some probability of being correct. Therefore, probabilistic calculations are useful in analyzing viewer profile data to make recommendations.

The preferred embodiment of the recommender uses a simple Bayesian classifier using prior and conditional probability estimates derived from the viewer profile. How recommendations are shown to viewers is not defined here, yet it will be assumed that one can capture the viewer's response to them, at least observing whether or not they were watched.

Below, a 2-class Bayesian decision model is discussed. The two classes of TV shows of interest are:

C1—shows that interest the viewer

C2—shows that do not interest the viewer

Other classes might be used showing more shades of interest or lack thereof.

In contrast with the classes of interest listed above, viewing history obtains information only on the classes:

C+—shows the viewer watched

C−—shows the viewer did not watch

Determining which shows a user watched or did not watch is outside of the scope of this application. The user might enter a manual log of which shows he/she watched. Alternatively, hardware might record the user's watching behavior. Those of ordinary skill in the art might devise numerous techniques for this. It should be possible to consider shows as watched even if they are watched only for a short time, as a user may be switching back and forth between several shows, trying to keep track of all of them.

Inferences may be made about classes C1 and C2 based on observations, but these inferences will always contain an element of uncertainty. The Bayesian model will compute the prior probabilities P(C+) and P(C−) directly from the counts in the viewer profile in accordance with FIG. **6***a*. In other words, the assumption will be that shows not watched are those the viewer is not interested in, and that the shows watched are the ones that the viewer is interested in.

The conditional probabilities, that a given feature, fi, will be present if a show is in class C+ or C−, are then computed in accordance with FIG. **6***b*. These calculations can be performed once a day during times that the TV is not being viewed and stored in the viewer profile.

Recommendations for upcoming shows can be computed by estimating the posterior probabilities, i.e., the probability that a show is in class C+ and C− given its features. Let x be a binary vector (x**1**, x**2**, . . . , xi, . . . , xn) where i indexes over the features in the viewer profile, and where xi=1 if feature fi is present in the show being considered for recommendation, and **0** if not. For the exclusive features, like day, time, and station, where every show must have one and only one feature, the index i will be taken to indicate the value present in the show being considered, provided that this value is also present in the profile. Otherwise, novel exclusive features will not enter into the calculations. For non-exclusive features, the index i will range over all values present in the profile; non-exclusive features novel to the considered show will not contribute to the calculations. The posterior probabilities are estimated in accordance with FIG. **6***c*.

With these estimates in hand, a show will generally be recommended if P(C+|x)>P(C−|x) and the "strength" of the recommendation will be proportional to P(C+|x)−P(C−|x). One potential problem with this scheme is that some conditional probabilities are likely to be zero. Any zero in a chain multiplication will reduce the result to zero; hence, some means for eliminating zeros is needed. The Billsus and Pazzani article referenced above, presents a couple of schemes, including simply inserting a small constant for any zero that occur.

One method for dealing with zeroes in the conditional probability multiplication chain would be as follows. One can choose a heuristic of 1000. If the number of shows in the viewing history is less than 1000, then the value of 1/1000 can be substituted for zero. If the number of shows in the viewing history is greater than 1000, the correction can be

$$\frac{k_i+ +1}{k+ +2}$$

Where

k_i+ is the number of watched shows having feature I

k+ is the total number of watched shows.

This is what is called the Laplace correction in the Billsus and Pazzani article. This Laplace correction must also be done for the not watched shows.

Alternative schemes may be devised by those of ordinary skill in the art.

Classical Bayesian theory would require the use of all accumulated elements of the list of FIG. **5** in making a recommendation. Nevertheless, in some instances, it may be useful to use a noise cutoff, eliminating features from consideration if insufficient data about them appears in the list. For instance, if a particularly feature did not appear in more than some given percentage of shows considered, whether in negative or in positive count, it might be ignored in determining which recommendation to make. Experimentally it was found that a cutoff of 5% was far too large.

Rather than use a percentage, one embodiment of the noise cutoff would use the viewer profile itself to determine the cutoff. This embodiment would first take a subset, or sub-list, of the viewer profile relating to particular feature types. For instance, a sub-list might advantageously comprise all of the elements of the viewer profile relating to the feature types, i.e., time of day and day of the week. Alternatively, in another example, the sub-list might advantageously comprise all of the elements of the viewer profile relating to channel number. Generally, the feature type or types chosen should be independent feature types, in other words, feature types which do not require another feature type to be meaningful.

The sub-list is then sorted by negative count, i.e., by number of shows having a particular feature value and not being watched. The highest negative count in this sorted list can be viewed as the noise level. In other words, since, in the preferred embodiment, the "not watched" shows are chosen at random from the week's program schedule, any not watched time slot can be considered to be noise.

Thus, any feature having both a positive and a negative count at or below the noise level need not be considered in the Bayesian calculation in making a recommendation. This example of noise level thresholding uses a particular feature, e.g., day/time as one for determining noise cutoff. In general, any feature that is uniformly randomly sampled by the negative example sampling procedure may be chosen by those of ordinary skill in the art for the calculation of the noise threshold.

The calculations of FIGS. **6***a*–**6***c* are advantageous in that they require fairly low computing power to complete and are therefore readily adaptable to modest hardware, such as would be found in a set-top box.

"Surprise Me" Feature

Recommendations according to the above-described scheme will be programs having a preponderance of features that are present in shows that have been watched. The viewer profiles accumulated will not yield any meaningful recommendations with respect to shows having few features in common with watched shows. Accordingly, optionally, the recommender may occasionally recommend shows at random, in a "surprise me" feature, if such programs have relatively few features in common with watched shows.

Using the User Profile in Other Domains

Once a user profile is developed, the recommendation techniques of the invention might be used to recommend other types of items, such as movies, books, audio recordings, or even promotional materials, such as tee-shirts or posters.

Non-Independence of Features

The classical assumption in the domain of Bayesian classifier theory is that all features are independent. Therefore, if a features is, say, often present in positive shows, but is missing from a show being considered for recommendation, the fact should count against the show. However, this may yield undesirable results for the current application.

For example, let us assume that there are five day/time slots indicated in the user profile as being most watched. Let us assume further that a particular show being evaluated falls within one of those five slots. The calculation of FIG. **6***c* would then give rise to an increase in probability for the day/time slot that matches and a decrease for the four day/time slots that do not match. Intuitively, it appears that the latter decrease is not reasonably related to an accurate determination of probability for the show in question. The different values of day/time are not independent—as every show has one and only one value, so the values a show does not have should not count against it.

To remedy this deficiency in the classical Bayesian approach, it is proposed to designate features into two types: Set 1 and Set 2. If a feature is designated Set 1, the Bayesian calculation will ignore any non-matching values of the feature. If the feature is designated Set 2, then the normal Bayesian calculation, per FIG. **6***c*, will be done.

Normally in a television application, Set 1 would include day/time; station; and title. Some features which have values only for a few shows, e.g., critic ratings, should also be set 1, because too many shows would be non-matching merely because critics tend to rate only a tiny percentage of shows.

Set 2, for television shows, would normally include all features that can have several values per show, such as actor. From reading the present disclosure, other modifications will be apparent to persons skilled in the art. Such modifications may involve other features which are already known in the design, manufacture and use of television interfaces and which may be used instead of or in addition to features already described herein. Although claims have been formulated in this application to particular combinations of features, it should be understood that the scope of the disclosure of the present application also includes any novel feature or novel combination of features disclosed herein either explicitly or implicitly or any generalization thereof, whether or not it mitigates any or all of the same technical problems as does the present invention. Applicants hereby give notice that new claims may be formulated to such features during the prosecution of the present application or any further application derived therefrom.

The word "comprising", "comprise", or "comprises" as used herein should not be viewed as excluding additional elements. The singular article "a" or "an" as used herein should not be viewed as excluding a plurality of elements.

What is claimed is:

1. A data processing device comprising:

at least one input for receiving data including
viewer profile data; and
data regarding a television program;
a medium readable by the data processing device coupled to the input, said medium storing said viewer profile data; and
a processor, the processor being adapted to perform the following:
calculating a probability that the television program is a desired one; and
supplying a recommendation regarding the television program based on the probability,

wherein the processor maintains the viewer profile in accordance with a data structure comprising:
a list of feature values; and
for each element of the list, a respective number of times programs having that feature value were watched, and a respective number of times programs having that feature value were not watched,

and wherein the processor is further arranged to perform the following, each time a user watches a new program,
first adding, to the list, feature values or counts of such feature values, associated with that new program;
selecting at least one companion program to the new program, the companion program being selected at random from a program schedule, which companion program has not been watched; and
second adding, to the list, feature values of the companion program, or counts of such feature values.

2. The data processing device of claim **1**, wherein the processor is further arranged to perform the following, each time a user watches a new program: first adding, to the list, feature values or counts of such feature values, associated with that new program.

3. The data processing device of claim **1**, wherein the input is a network connection.

4. The data processing device of claim **1**, wherein calculating comprises using a Bayesian classifier.

5. The data processing device of claim **4**, wherein the processor is further adapted to subject the viewer profile to a noise threshold calculation prior to using the Bayesian classifier.

6. A data processing device comprising:

at least one input for receiving data including
viewer profile data; and

data regarding a television program; and
a processor, the processor being adapted to perform the following:
calculating, using a Bayesian classifier, a probability that the television program is a desired one; and
supplying a recommendation regarding the television program based on the probability,

wherein the processor is further adapted to subject the viewer profile to a noise threshold calculation prior to using the Bayesian classifier,

and wherein
the viewer profile data comprises
a list of feature values;
a respective negative count for each element of the list, the negative count indicating a number of times programs having that feature value have not been watched;
a respective positive count for each element of the list, the positive count indicating a number of times programs having that feature value have been watched;
the noise threshold calculation comprises
selecting a sub-list comprising at least feature values having at least one specific type of feature;
choosing the highest negative count in the sub-list as the noise threshold;

the recommendation comprises a program selected from a group having at least one feature value having a positive or negative count in the viewer profile, which count exceeds the noise threshold.

7. The data processing device of claim **6**, wherein the specific type comprises a day and time of day feature type.

8. The data processing device of claim **6**, wherein the specific type comprises a station identification feature type.

9. The data processing device of claim **6**, wherein the viewer profile data comprises a plurality of respective counts of programs watched, each respective count indicating how many programs watched had a respective feature.

10. The data processing device of claim **9**, wherein calculating comprises calculating a probability that the television program is in a particular class.

11. The data processing device of claim **10**, wherein the class is one of
programs the viewer is interested in, and
programs the viewer is not interested in.

12. A data processing device comprising:

at least one input for receiving data including
viewer profile data; and
data regarding a television program; and
a processor, the processor being adapted to perform the following:
calculating, using a Bayesian classifier, a probability that the television program is a desired one; and
supplying a recommendation regarding the television program based on the probability,

wherein the processor is further adapted to subject the viewer profile to a noise threshold calculation prior to using the Bayesian classifier,

and wherein subjecting the viewer profile to the noise threshold further comprises using observations gathered by a known random process to estimate a reasonable noise threshold.

13. A data processing device comprising:

at least one input for receiving data including
viewer profile data; and
data regarding a television program; and
a processor, the processor being adapted to perform the following:

calculating a probability that the television program is a desired one; and

supplying a recommendation regarding the television program based on the probability,

wherein calculating the probability comprises:

computing a prior possibility, of whether a program is desired or not;

computing a conditional probability of whether a feature fi will be present if a show is desired or not; and

computing a posterior probability of whether program is desired or not, based on the conditional probability and the prior probability.

14. The data processing device of claim **13**, wherein it is assumed that programs watched are programs that the viewer is interested in.

15. The data processing device of claim **13**, wherein the processor is further adapted to provide a recommendation regarding an additional item, other than a television program, based on the viewer profile.

16. A data processing device comprising:

at least one input for receiving data including

viewer profile data; and

data regarding a television program; and

a processor, the processor being adapted to perform the following:

calculating a probability that the television program is a desired one; and

supplying a recommendation regarding the television program based on the probability,

wherein

the viewer profile comprises a list of features types and values for such feature types;

the feature types are selected from at least two sets, including

a first set of feature types whose values are deemed non-independent; and

a second set of feature types whose values are deemed independent; and

calculating a probability comprises

applying a Bayesian classifier calculation corresponding to feature types from the second set; and

applying a modified Bayesian classifier calculation corresponding to feature types from the first set.

17. The data processing device of claim **16**, wherein

with respect to features of the first set, the modified Bayesian classifier calculation considers only feature values that match with a show being classified.

18. A computer readable medium having computer-executable instructions stored thereon for performing the method comprising:

calculating a probability that a television program is a desired one, based on a viewer profile and data regarding the television program; and

supplying a recommendation regarding the television program based on the probability,

wherein the computer readable medium further embodies the viewer profile, the viewer profile being embodied as a data structure comprising:

a list of feature values; and

for each element of the list, a respective number of times programs having that feature value were watched,

and wherein the software is further arranged to perform the following, each time a user watches a new program,

first adding, to the list, feature values or counts of such feature values, associated with that new program;

selecting at least one companion program to the new program, the companion program being selected at random from a program schedule, which companion program has not been watched; and

second adding, to the list, feature values of the companion program, or counts of such feature values.

19. The computer readable medium of claim **18**, wherein the computer-executable instructions is further arranged to perform the following, each time a user watches a new program: first adding, to the list, feature values or counts of such feature values, associated with that new program.

20. The computer readable medium of claim **18**, wherein the computer readable medium embodies the data regarding the television program.

21. The computer readable medium of claim **18**, wherein calculating comprises using a Bayesian classifier.

22. The computer readable medium of claim **21**, wherein the computer-executable instructions, is further adapted to subject the viewer profile to a noise threshold calculation prior to using the Bayesian classifier.

23. A computer readable medium having computer-executable instructions stored thereon for performing the method comprising:

calculating, using a Bayesian classifier, a probability that a television program is a desired one, based on a viewer profile and data regarding the television program; and

supplying a recommendation regarding the television program based on the probability,

wherein the computer-executable instructions is further adapted to subject the viewer profile to a noise threshold calculation prior to using the Bayesian classifier,

and wherein

the viewer profile data comprises

a list of feature values;

a respective negative count for each element of the list, the negative count indicating a number of times programs having that feature value have not been watched;

a respective positive count for each element of the list, the positive count indicating a number of times programs having that feature value have been watched;

the noise threshold calculation comprises

selecting a sub-list comprising at least feature values having at least one specific type of feature;

choosing the highest negative count in the sub-list as the noise threshold;

the recommendation comprises a program selected from a group having at least one feature value having a positive or negative count in the viewer profile exceeding the noise threshold.

24. The computer readable medium of claim **23**, wherein the specific type comprises a day and time of day feature type.

25. The computer readable medium of claim **23**, wherein the specific type comprises a station identification feature type.

26. The computer readable medium of claim **23**, wherein the viewer profile data comprises a plurality of respective counts of programs watched, each respective count indicating how many programs watched had a respective feature.

27. The computer readable medium of claim **26**, wherein calculating comprises calculating a probability that the television program is in a particular class.

28. The computer readable medium of claim **26**, wherein the class comprises at least one of programs the viewer is interested in and programs the viewer is not interested in.

29. A computer readable medium having computer-executable instructions stored thereon for performing the method comprising:

calculating, using a Bayesian classifier, a probability that a television program is a desired one, based on a viewer profile and data regarding the television program; and

supplying a recommendation regarding the television program based on the probability,

wherein the computer-executable instructions is further adapted to subject the viewer profile to a noise threshold calculation prior to using the Bayesian classifier,

and wherein subjecting the viewer profile to the noise threshold further comprises using observations gathered by a known random process to estimate a reasonable noise threshold.

30. A computer readable medium having computer-executable instructions stored thereon for performing the method comprising:

calculating a probability that a television program is a desired one, based on a viewer profile and data regarding the television program; and

supplying a recommendation regarding the television program based on the probability,

wherein calculating the probability comprises:

computing a prior possibility, of whether a program is desired or not;

computing a conditional probability of whether a feature fi will be present if a show is desired; and

computing a posterior probability of whether program is desired or not, based on the conditional probability and the prior probability.

31. The computer readable medium of claim **30**, wherein it is assumed that programs watched are programs that the viewer is interested in.

32. The computer readable medium of claim **30**, wherein the computer-executable instructions is further arranged to provide a recommendation regarding an additional item, other than a television program, based on the viewer profile.

33. A computer readable medium having computer-executable instructions stored thereon for performing the method comprising:

calculating a probability that a television program is a desired one, based on a viewer profile and data regarding the television program; and

supplying a recommendation regarding the television program based on the probability,

wherein

the viewer profile comprises a list of features types and values for such feature types;

the feature types are selected from at least two sets, including

a first set of feature types whose values are deemed non-independent; and

a second set of feature types whose values are deemed independent; and

calculating a probability comprises

applying a Bayesian classifier calculation corresponding to feature types from the second set; and

applying a modified Bayesian classifier calculation corresponding to feature types from the first set.

34. The computer readable medium of claim **33**, wherein with respect to features of the first set, the modified Bayesian classifier calculation considers only values that match with a show being classified.

35. A data processing method comprising performing the following operations in a data processing device:

first receiving data reflecting physical observations, which data includes a list of feature values and observations about feature values, some of which feature values are independent and some of which are not;

second receiving data about an item to be classified, the data about the item to be classified including feature values;

maintaining a division of the data reflecting physical observations into at least two sets, including

a first set including those feature values which are deemed not independent; and

a second set including those feature values which are deemed independent;

performing a probabilistic calculation on the data reflecting physical observations and the data regarding an item to be classified including:

applying a Bayesian classifier calculation with respect to feature values relating to the second set; and

applying a modified Bayesian classifier calculation with respect to feature values relating to the first set

presenting a conclusion regarding the item to be classified to a user based on the probabilistic calculation.

36. The method of claim **35**, wherein the modified Bayesian classifier calculation comprises ignoring feature values from the data reflecting physical observations when those feature values are not present in the data regarding the item to be classified.

* * * * *

PERGAMON

Expert Systems with Applications 25 (2003) 331–341

Expert Systems with Applications

www.elsevier.com/locate/eswa

Personalizing information appliances: a multi-agent framework for TV programme recommendations

Wei-Po Lee*, Tsung-Hsien Yang

Department of Management Information Systems, National Pingtung University of Science and Technology, Nei-Pu Hsiang, Pingtung, Taiwan, ROC

Abstract

It has been advocated to develop information appliances to provide ubiquitous Internet information access. After the invention of digital set-top-box, television is expected to become one of the most popular information appliances because of the tremendous digital multi-media programmes it can broadcast. However, as in the World Wide Web, the available TV programmes and their correspondingly electronic information within the increasing digital channels may cause the problem of information overload. Though the audience has more alternatives while choosing programmes, he also has to spend more and more time to read the on-line information about the programme contents or to browse different channels in order to decide what to watch. One way to overcome such a problem is to build intelligent recommender systems to provide personalized information services. By analyzing the information collected from the user, a personalized recommender system is able to reason his personal preferences and then choose the programmes for him. This paper presents a multi-agent framework in which a decision tree-based approach is proposed to learn a user's preferences. The experimental studies concentrate on how to recommend programmes of films and news to a user, and on how the system can adapt to a user's most recent preferences. The results and analysis show the promise of our system.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Intelligent agent; Personalization; Machine learning; Recommender system; Information appliance

1. Introduction

Following the explosive advances of Web technologies, information can now flow around the world through the standard protocols and be rendered into a variety of electronic devices, such as personal digital assistants, mobile phones, etc. It is thus advocated to develop information appliances that emphasize the integration of computers, communications and consumer electronics, to provide ubiquitous information access, in addition to their originally specialized functions (Lewis, 1998; Want & Borriello, 2000). Though it is convenient to access on-line information by the information appliances, the problem of information overload caused by the exponentially increasing electronic information on the Internet still has to be overcome. Our work will study this issue and focus on the appliances for multi-media broadcasting as experimental examples.

Among others, television has long dominated the traditional home entertainment market because it has changed the way information is presented, from the text-based environment to a multi-media world. At present, a television itself is not a true information appliance, but with the invention of digital set-top-box, television is expected to become a certain kind of popular information appliance soon, due to the tremendous digital multi-media programmes it can broadcast. However, as in the World Wide Web, the available TV programmes and their correspondingly electronic information within the increasing digital channels lead to the problem of information overload. Though the audience has more alternatives while choosing programmes, he also has to spend more and more time to read the on-line information about the programme contents or to browse different channels in order to decide what to watch. For programmes a certain audience is not interested in, he cannot help watching them for a short while in order to filter them out. Consequently, a general audience will spend most of his leisure time browsing the channels rather than watching the programmes.

One way to overcome the above problem is to develop intelligent recommender systems to provide personalized information services. For the multi-media broadcasting

* Corresponding author.
E-mail address: wplee@mail.npust.edu.tw (W.-P. Lee).

0957-4174/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0957-4174(03)00074-5

environment such as television, an ideal solution to the user is to have a personal identifier (like the SIM card in the mobile phone) that can record his most recent preferences and work as a filter for channel selection. Therefore, whenever a user equips his identifier into a special device (e.g. remote control) to turn on the television, the programmes available at that time are sorted automatically and presented to him based on his preferences. In other words, the first programme presented to a user is presumed to be the one he likes the most among the programmes currently broadcasted; the next one is the second preferred, and so on. In this way, the audience can immediately access the programmes he is interested in and then save much of the time spent on reading the information about the contents or browsing the channels to decide what to watch.

In this paper, we describe how we adopt an agent-based methodology to develop a prototype system for personalized TV programme recommendations. Two types of programmes are considered: one is the film, and the other, the news. To deal with different programmes, our system includes five agents: a *film agent* to implicitly observe a user's surfing behavior and then record what he watches, or to gather a user's preferences in an explicit way; a *news agent* to collect Internet news for user's identification; a *learning agent* to analyze and model a user's interests; a *collaboration agent* to look for other users with similar tastes for sharing the information collected; and an *adaptation agent* to evaluate the performance of the system and to adapt to the user's newly changed interests. The experimental results show that our learning agent can learn a user's interests in real time and give sensible recommendations. In addition, the *k*-nearest neighbor method is implemented as a baseline for performance comparison. Furthermore, we also show that for different situations in which a user chooses a programme with lower priority rather than the top one suggested by the system, the learning agent can be resumed accordingly and it continuously learns the most up-to-date preferences for the user.

2. Background

In general, there are two kinds of approaches in developing a recommender system: personalized (content-based) and socialized (collaborative) ones. In a personalized system, a user's personal information is first collected, then the system reasons the user's preferences by analyzing and modeling the personal information available. The ensuing user profile is recorded by the system for further recommendations. To collect information, the system can ask a user to manually indicate his interests by giving some examples with personal ratings. Alternatively the system can implicitly observe and record the user's operating behavior, and then analyzes the content the user has read in order to build a model to distinguish between what he likes and what he does not.

The most important issue in the personalization work is to construct a computational model for each individual user to predict his preferences for incoming information. With the user model working as a filter, only the information considered as user-interested can reach the end user. Therefore, the work of recommendation can be regarded as classification: using the information already known to build a model to predict the events unknown. The model may be a set of rules, a decision tree, or a set of numerical numbers representing the corresponding weights for some specific features. Some machine learning approaches have been applied to construct users' profiles. In Balabanovic and Shoham (1997), a statistic-based approach *tfidf* (term frequency and inverse document frequency (Salton, 1989)) was used to build a user's profile to recommend Web pages. In Joachims, Freitag, and Mitchell (1997), a reinforcement learning method was employed for Web page recommendations, and in Mooney and Roy (2000), for book recommendations. In this work, we use a decision tree-based approach to develop a learning agent to model a user's interests for TV programme selections, and to explore different programme features for better recommendations.

Instead of building a personal model to predict how a user likes an item, a collaborative system recommends items to a specific user according to the evaluations from other users with similar tastes. In other words, the system does not analyze what a user likes but taking the opinions of others. In such a system, the similarity of the users is measured by certain correlation criterion, and then a *k*-nearest neighbor method is used to find some users with similar interests for a specific user. The prediction of an unknown item for a user is thus based on the combination of the ratings of his nearest neighbors. Good et al. (1999) and Shardanand and Maes (1995) present systems of this kind.

Some hybrid methods are proposed to overcome the problems with the above two kinds of recommender systems, for example, the combined techniques for personalization and collaboration used in Balabanovic and Shoham (1997), Basu, Hirsh, and Cohen (1998), and Smyth and Cotter (2000). In Smyth and Cotter (2000), a personal profile is compiled from the TV guide for programme recommendations. Our work presented here differs from theirs not only in the learning approach used, but also in the situations of two different types of programmes broadcasted (i.e. films and news). Most importantly, our work includes a specially designed mechanism for system adaptation, with which a user's profile can be re-learnt to correct the prediction errors and to meet the user's most recent preferences.

3. A multi-agent framework for personalized TV programme recommendations

The major tasks of a personalized recommender system include collecting a user's personal interests, building a model to describe the information collected, and adapting to the user's most recent interests. In this work, we extend the agent-based framework presented in our previous work (Lee, Liu, & Lu, 2002) to design and implement a prototype system for the future TV-based information appliances. In this system, each agent autonomously performs a specific task, and different agents work simultaneously to achieve the overall task in a distributed way. Fig. 1 describes the system architecture. As can be seen, three types of agents for information gathering, user modeling, and system adaptation are included for the above tasks.

For the two types of programmes (i.e. film and news) considered here, the system uses an arbitrator, based on the user's habits, to select the appropriate type to recommend. In our current implementation, the system uses a weekly timetable. It is divided into time units of 10 min, to record how often the user watched each type of programmes in each time unit. In this way, when a user would like to watch TV at a certain time, the system firstly determines the priorities of different programme types according to the user's habits recorded, and orders the channels predicted as user-interested within each programme type by the corresponding user model. The system then recommends the channels to the user in the above order.

3.1. Information gathering

In order to reason the user's preferences, the first task a personalized system has to perform is to collect his personal information. For the task of TV programme recommendations, two types of programmes are considered here: one is the film and the other is the news. Therefore, the *film agent* and *news agent* shown in Fig. 1 are designed to collect information about films and news, respectively. The film agent operates in different ways: it can present the user a form (as shown in Fig. 2) that allows him to explicitly point out his interests; it can also provide a browser-like environment that links the user to different movie sites, and implicitly records and analyzes the contents the user has read. In the latter case, the film agent measures the time he stays to view the content about the film presented in a certain page, to recognize to what extent a user is interested in it. In our current implementation, the agent directs the user to the on-line movie site *allmovies* (http://www.allmovieguide.com) and records his browsing behavior. Another alternative for the film agent is to directly record the film programmes a user has chosen to watch, and then to retrieve the on-line information for further analysis. For the news programmes, the news agent in Fig. 1 extracts the on-line daily news from the CNN news site (http://www.cnn.com), re-organizes the news, before presenting them to the user by the interface shown in Fig. 3. The user can mark the news to indicate his interests. The above agents both play the role of information collectors; a user can activate them arbitrarily to provide personal information to the system.

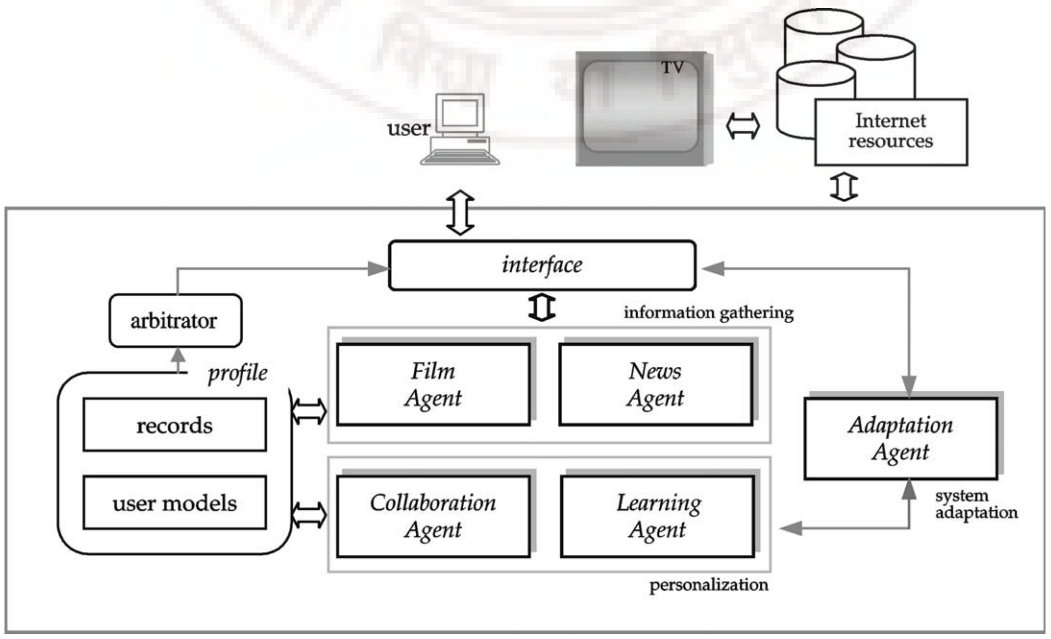
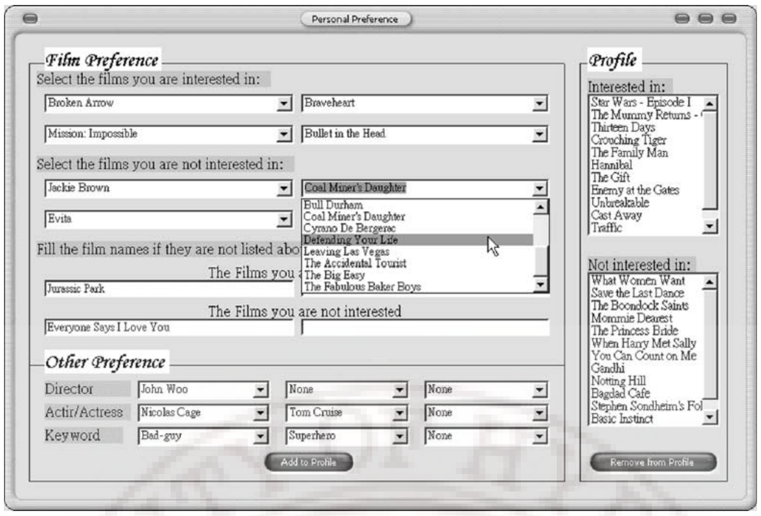


Fig. 1. The system architecture for TV programme recommendations.



Film Preference

Select the films you are interested in:

Broken Arrow | Braveheart

Mission: Impossible | Bullet in the Head

Select the films you are not interested in:

Jackie Brown | Coal Miner's Daughter

Evita

Bull Durham
Coal Miner's Daughter
Cyrano De Bergerac
Defending Your Life
Leaving Las Vegas
The Accidental Tourist
The Big Easy
The Fabulous Baker Boys

Fill the film names if they are not listed abo

The Films you

Jurassic Park

The Films you are not interested

Everyone Says I Love You

Other Preference

Director	John Woo	None	None
Actir/Actress	Nicolas Cage	Tom Cruise	None
Keyword	Bad-guy	Superhero	None

Profile

Interested in:

Star Wars - Episode I
The Mummy Returns -
Thirteen Days
Crouching Tiger
The Family Man
Hannibal
The Gift
Enemy at the Gates
Unbreakable
Cast Away
Traffic

Not interested in:

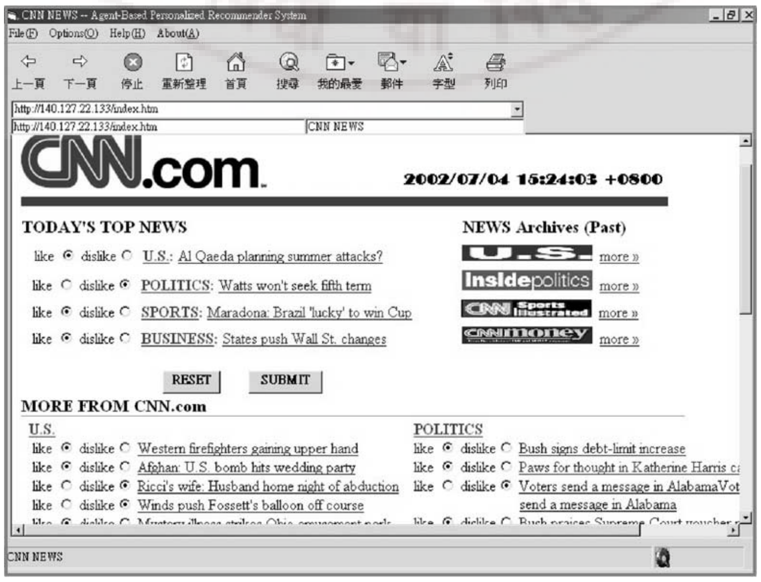
What Women Want
Save the Last Dance
The Boondock Saints
Mommie Dearest
The Princess Bride
When Harry Met Sally
You Can Count on Me
Gandhi
Notting Hill
Bagdad Cafe
Stephen Sondheim's Fol
Basic Instinct

Fig. 2. Interface presented by the film agent for information collection.

In this system, a film programme is represented as a feature vector of genre, director, cast, and plot, because they often are the most decisive factors a general audience used to determine whether he likes a specific film or not. The plot here means the keywords describing the film content and available from the on-line database. Consequently, a film program is represented as the combination of the above four types of terms. For example, the film *Batman* is represented as ⟨genre: fantasy, genre: action, director: Tim Burton, cast: Jack Nicholson, cast: Michael Keaton, keyword: revenge, keyword: superhero, etc.⟩. All terms appearing in the training examples are defined as candidate features (attributes) in the learning procedure and used to construct the user models. In addition, as the standard work in text mining, each piece of news extracted here is represented as a word vector in which the feature words are determined by the *tfidf* method, after the common words are removed and the stemming procedure is performed.

3.2. User modeling

The second type of agents in Fig. 1 is to model a user's preference from the information recorded in his profile. As is shown in Fig. 1, two agents, including a learning agent and a collaboration agent, are developed for user modeling. The learning agent takes the responsibility of building a classifying system that consists of a set of prediction models (or classifiers) for a user, by using the most recent


CNN NEWS -- Agent-Based Personalized Recommender System
File(F) Options(O) Help(H) About(A)
上一頁
下一頁
停止
重新整理
首頁
搜尋
我的最愛
郵件
字型
列印
CNN.com
RESET
SUBMIT
CNN NEWS

http://140.127.22.133/index.htm

CNN NEWS

2002/07/04 15:24:03 +0800

TODAY'S TOP NEWS

[✓ like] dislike U.S.: Al Qaeda planning summer attacks?
like [✓ dislike] POLITICS: Watts won't seek fifth term
[✓ like] dislike SPORTS: Maradona: Brazil 'lucky' to win Cup
[✓ like] dislike BUSINESS: States push Wall St. changes

NEWS Archives (Past)

U.S. more »
Insidepolitics more »
CNN Sports Illustrated more »
CNNmoney more »

MORE FROM CNN.com

U.S.

[✓ like] dislike Western firefighters gaining upper hand
[✓ like] dislike Afghan: U.S. bomb hits wedding party
like [✓ dislike] Ricci's wife: Husband home night of abduction
like [✓ dislike] Winds push Fossett's balloon off course

POLITICS

[✓ like] dislike Bush signs debt-limit increase
[✓ like] dislike Paws for thought in Katherine Harris ca
like [✓ dislike] Voters send a message in AlabamaVot send a message in Alabama

Fig. 3. Interface presented by the news agent for information collection.

items collected and recorded in the personal profile as training examples. In modeling a user, the examples of films and news collected are trained separately to build models for their corresponding types of programmes. Currently, the learning agent acts in a passive way; it is activated when the performance of recommendation is lower than a pre-defined threshold.

The learning agent is implemented by a decision-based approach which is one of the most popular and practical methods for inductive inference. The decision learning is a useful method for approximating discrete-valued target functions, and is thus suitable for building models to recognize a user's interests here. In this approach, the model learnt is represented as a decision tree; it classifies an instance by sorting it down on the tree from the root node to a certain leaf node that specifies the category of the item. For the vector representation mentioned above, a node could be a test for the appearance of a feature keyword and this node has two branches indicating whether a film (or a piece of news) includes this keyword. After a series of attribute testing, the leaf node directs the item to one of the classes pre-defined. The construction of the above tree classifier is based on the measure of a statistical property that is called information gain for each attribute. This property measures how well a given attribute separates the training examples according to their target classification. The most informative attribute among those not yet considered in the path from the root node is then used and associated to the current node. The details about the decision tree learning can be found in Quinlan (1993).

Though decision trees can perfectly separate training examples by a covering tree, their predictive performance for the test cases is not as good as in the training phase. This overfitting phenomenon appears mainly in the situations when the number of training examples is too small to form a representative sample set or when there is noise in the data. As a result, the tree is overly specialized to the training data. To avoid the overfitting cases, we adopt the method of adaptive resampling for multiple trees. This method is to reduce the variance between training and testing cases, and it has been observed that inducing multiple decision trees from the same training set is an efficient approach. In such an iterative approach, training examples are firstly sampled from the original training set and used to construct a tree. Then this tree is given a vote for determining the categories of other items. After that, the training examples are replaced by the newly sampled cases, and a new decision tree is constructed again. This process continues and the category with most votes determines the final answer. It has been shown that when the number of induced decision trees increases, the variance can gradually be reduced and the corresponding prediction performance can be enhanced (Breiman, 1996). In the sampling phase mentioned above, the technique of adaptive resampling usually performs better than others: instead of randomly sampling a training set in each iteration, this technique increases the probabilities to be chosen for those cases previously misclassified (Weiss et al., 1999).

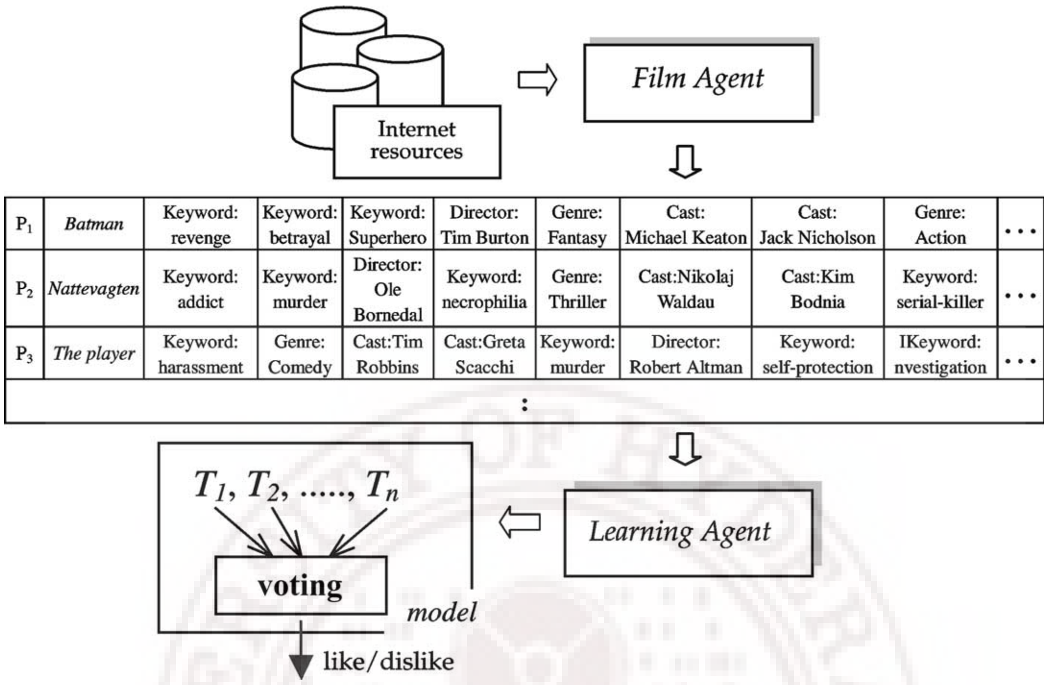
3.2.1. Learning a user's preferences on films

As mentioned above, a film can be described by its features including genres, directors, actors, plot, and so on. Different types of features are thus explored. Firstly, we use the plot of a film as the representative feature. In the preliminary experiments, we used the introductory note of a film as its corresponding plot, and employed the traditional text analysis approach (i.e. *tfidf*) to extract words to constitute a feature vector for a film. Yet, as is found, the introductory note is generally too short to extract representative terms. Therefore, we decide to directly use the keywords associated with a film (available from the on-line film database) to represent its plot and to be the features. For example, the film *Jurassic Park* is represented as ⟨attack, clone, dinosaur, experiment, genetic, etc.⟩ in our system. In addition to the plot (keywords), we also experience other features to describe a film, including the combination of genre, director, actor, actress of the film, and the hybrid of the above two ways. The comparison for the above three representations are shown in the experiments.

For simplicity, in our current implementation, the system only predicts whether a user is interested in a specific film or not, rather than the degree in which he favors the film. Both implicit and explicit ways for information collection are provided and the examples collected are maintained in a user's profile for later training. Ten decision trees are included, and their voting results are used to determine whether a user likes a specific film programme or not. That is, a programme will be predicted as user-like if it receives at least five tree votes for this class. Fig. 4 shows part of the user's profile and its functional relationships with different agents.

3.2.2. Learning a user's interests on news

In addition to the film programmes, the system also considers the channels that broadcast daily news. In our implementation, the nature of the news channels is more comprehensive and all-inclusive than specific topics, so the contents of different news channels in all are regarded as similar. Hence, the task here is to determine whether the system should recommend news channels to a user. Yet to decide which news channel to watch is left to the user himself (for example, a general user may favor a specific reporter or the overall policy or style of a channel). To predict this situation, the decision tree learning approach is used to construct the user model. With the learnt model, the system can extract the daily news from the Internet and then classify the news into classes of user-interested and user-non-interested. If the number of daily news predicted as user-interested has exceeded a certain threshold, the news channels will then be taken into



P_1	*Batman*	Keyword: revenge	Keyword: betrayal	Keyword: Superhero	Director: Tim Burton	Genre: Fantasy	Cast: Michael Keaton	Cast: Jack Nicholson	Genre: Action	. . .
P_2	*Nattevagten*	Keyword: addict	Keyword: murder	Director: Ole Bornedal	Keyword: necrophilia	Genre: Thriller	Cast:Nikolaj Waldau	Cast:Kim Bodnia	Keyword: serial-killer	. . .
P_3	*The player*	Keyword: harassment	Genre: Comedy	Cast:Tim Robbins	Cast:Greta Scacchi	Keyword: murder	Director: Robert Altman	Keyword: self-protection	IKeyword: nvestigation	. . .
:										

Fig. 4. The flow of user modeling related to the films.

consideration by the system for recommendation when the user would like to watch TV.

Instead of directly constructing a single classifier to categorize all news documents, our system decomposes the overall classification task into several subtasks in which an individual classifier is developed, respectively, for each category of news, for example business, sports, or politics. For each category, the news cases recorded in the profile are used as training examples to build the corresponding classifier. In this way, the personal model for news prediction in a user profile includes several classifiers for different categories of news. Each piece of Internet news extracted is sent to the corresponding classifier according to its category originally assigned by the news site, and then the classifier predicts the user's interests for this news. All the news predicted as user-interested from different classifiers will be accumulated and used to determine whether the news channels will be recommended. Fig. 5 illustrates the part of the profile related to the news and the flow of learning a personal model.

3.2.3. Collaborative recommendation

In the content-based approach described above, the system evaluates available TV programmes for a user according to the ones ranked and recorded in his personal profile. Yet, in the real world application, there still are some problems to be overcome. The first is about the training examples required in the learning procedure. On the one hand, the system needs many examples that have been rated by the user in order to construct an accurate personal model; on the other hand, from the user's point of view, rating items is a laborious job, so the few ratings required the better for him. The second problem with the content-based approach is its lack of the variety of items: the newly recommended items tend to be similar to the ones he liked in the past and consequently the user is restricted to watch certain types of programmes.

One solution to the above two problems is to adopt the collaborative approach in which the evaluation of a new item depends on how similar users liked it, but not based on the personal model any more. Typically each user is given a set of other users with most similar preferences by a certain matching strategy. And in order to predict how this user will like an unfamiliar item, the system will combine the evaluation results for this item from the given set of users. In this way, the information (i.e. the ratings of items) collected from different users can be shared and unfamiliar regions of the item space can be automatically explored. However, as is known, the collaborative approach has problems of its own, for example the need of accurate measurement of user similarity, or the consideration for the newly introduced items that have not yet been rated (Balabanovic & Shoham, 1997; Mooney & Roy, 2000). Therefore, in our work, the collaborative approach is not only taken but also used as a complementary way for information gathering. The collaboration agent shown in Fig. 1 is designed to collect training examples from other similar users.

3.3. System adaptation

In our system, two classes of preferences are defined: user-like and user-dislike. With the multiple decision trees

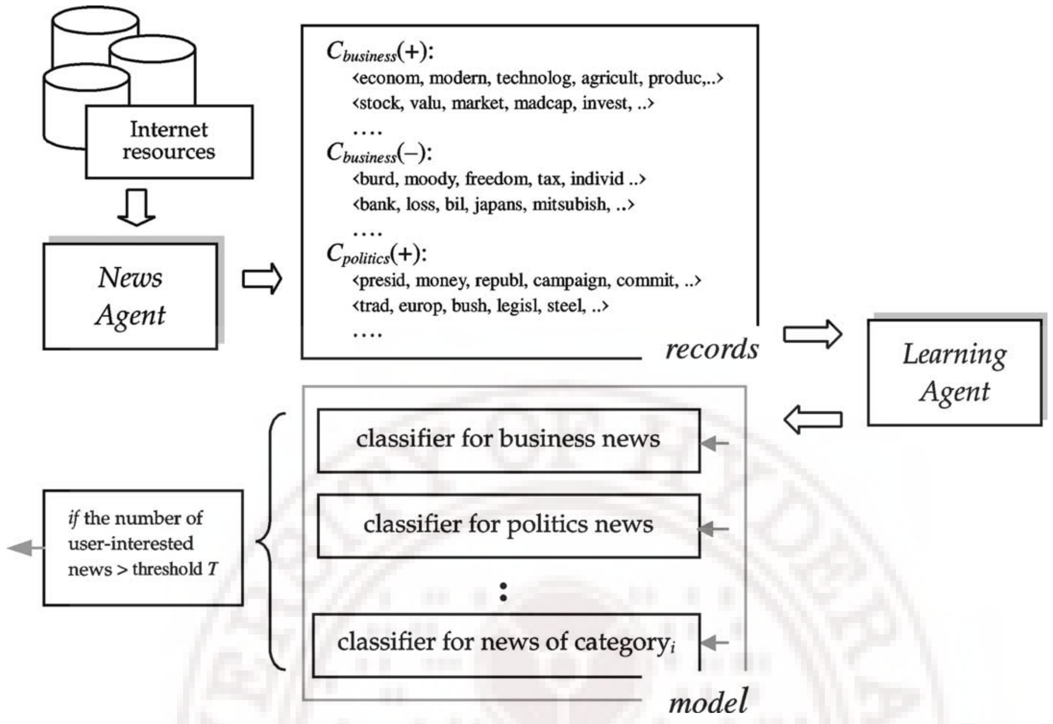

Internet resources
News Agent
$C_{business}(+)$:
<econom, modern, technolog, agricult, produc,..>
<stock, valu, market, madcap, invest, ..>
....
$C_{business}(-)$:
<burd, moody, freedom, tax, individ ..>
<bank, loss, bil, japans, mitsubish, ..>
....
$C_{politics}(+)$:
<presid, money, republ, campaign, commit, ..>
<trad, europ, bush, legisl, steel, ..>
....
records
Learning Agent
classifier for business news
classifier for politics news
classifier for news of category$_i$
model
if the number of user-interested news > threshold T

Fig. 5. The flow of user modeling related to the news. The words in the news vectors have been stemmed.

learnt, the system can classify film programmes into appropriate classes, and in the user-like class the rank (i.e. the recommending order) of a programme is determined by the number of tree votes received (a tree vote here means a programme has been predicted as user-like by one tree). However, in the real world application, a user could choose any programme rather than the one with the highest priority. Therefore, it is essential to have a mechanism to dynamically modify the personal model over time. The adaptation agent is thus designed for this purpose.

The user model will be modified for two situations. In the first situation when the programme chosen by the user is predicted into the correct class (i.e. user-like), but it is not on the top of preference rank, the system has to modify the user's model to raise the rank of this programme up to the top to match the user's decision. To deal with this situation, the system adds a copy of the programme as a training example to the user's profile and evokes the learning agent to re-construct a personal model. If the newly developed model still cannot give this programme the highest priority, the above procedure is repeated until it reaches the top rank. The later experiments will evaluate such a method and also examine whether this method will affect the overall prediction performance of the system.

The second situation means the cases that the programme chosen by the user is classified into the wrong class. This is mainly due to occasional changes of a user's interests, in which the model built earlier is not able to characterize the most up-to-date interests any more, and the system performance is thus decline. Unlike the first situation described above, the prediction errors occur here and therefore, the examples used in the previous training phase have to be gradually substituted. Under such circumstances, the system uses this programme to replace the oldest training example in the user's profile and re-build the personal model. As the decision tree learning approach can build new models in real time in our application case, the learning procedure is thus activated for each faulty prediction.

As is mentioned, our system considers all news channels as the same kind. It only determines whether to recommend the news channel to a user; the order of the different news channels is pre-defined by the user. Hence, for the news programmes, the re-learn procedure is activated manually. The user can arbitrarily add new example or remove the old ones from the profile to derive a new model.

4. Experiments and results

In order to assess the developed system, this section describes the experiments conducted. In the experiments, we concentrate on evaluating three different agents including the learning, collaboration, and adaptation agents. For the learning agent, we first investigate whether a prediction model can be learnt by the proposed decision tree-based mechanism, and then compare the results with the well-known k-nearest neighbor method. For the collaboration agent, a set of experiments is conducted to examine the corresponding effects in sharing information. And for the adaptation agent, we analyze how it monitors the user's feedback and how it can adapt to a user's most recent preferences.

4.1. Evaluation of the learning agent

To evaluate the performance of our learning agent, we implemented a decision tree-based mechanism to develop models of preferences for users. In the first set of experiments, we used the plot of a film to be the representative feature: each film was represented by its corresponding keywords. In this way, a decision tree was constructed from the keywords within the training set. In the training phase, different numbers of examples (including 60, 80, 100, and 120 training examples) were used to develop trees: a half of them were positive examples and the other half, negative. After the training, each tree model was tested by another 40 cases not contained in the training set for further evaluation. Table 1 shows the test results in which each data point is averaged from 10 independent experiments (with different training sets from different users). The test results indicate that in average 73.8–77.3% accuracy was obtained. This may be satisfactory as there were only 60–120 training examples used in the learning phase and they distributed in different genres. Though using sparse samples to reason about a user's preferences in a large space could result in a relatively loose user model, it is nevertheless a more realistic situation.

After showing that our learning agent is capable of developing tree classifiers to model a user's preferences, we compare it with the well-known k-nearest neighbor method. Three different values of k (1, 3, and 5) were used, and for each k, different numbers of reference examples were provided as in the above decision tree experiments. The k-NN-1, k-NN-3, and k-NN-5 in Table 1 are the results. As can be seen, in our user modeling cases, the decision tree learning outperforms the k-nearest neighbor method.

The above decision tree approach was also employed to learn classifiers for news categorization. As is analyzed, each classifier here is developed for a specific category of news; it is to predict whether or not a user is interested in a certain piece of newly available news assigned to this category by the news Web site. Four sets of experiments were performed for four categories of news, including business, politics, sport, and US domestic, as expository examples. For each category, different numbers of training cases collected by the news agent were used to construct classifiers. Table 2 shows the prediction performance of 40 test cases. The performance is similar to those of decision tree-based text mining work (Mlademic, 1999). It shows the feasibility of this approach in news classifications.

Table 1
A comparison of the decision tree and k-nearest neighbor methods

	Number of training examples			
	60	80	100	120
Decision tree	73.8	74.7	76.1	77.3
k-NN ($k = 1$)	67.5	67.2	69.3	68.5
k-NN ($k = 3$)	67.2	67.4	70.4	69.4
k-NN ($k = 5$)	66.7	66.8	70.4	68.8

Table 2
The results for news categorizations

	Number of training examples			
	60	80	100	120
Business news	76.1	77.2	78.6	80.4
Politics news	80.5	81.7	81.1	81.8
Sport news	79.4	81.4	81.1	81.6
US domestic news	75.0	74.7	77.5	79.3

4.2. The Effect of adaptive resampling

In order to explore the effect of adaptive resampling, a series of experiments were conducted. In addition, we used different film representations to investigate the most appropriate features in modeling a user's preferences. As the user's models for films and news recommendations are developed by the same approach, the experiments for comparing decision trees with and without adaptive resampling were performed only for film programmes. The same technique can be applied to news classifications.

Table 3 shows the test results in which each value is the accuracy of prediction averaged over 10 independent runs. In each single run, 120 training examples and 40 test examples were used. The first and the second rows are the results by the traditional decision tree method and the one with resampling as described, respectively. In the experiments of 10 trees with adaptive resampling, one additional copy of each misclassified case was added to the original example pool so that its corresponding probability to be chosen as a training example for the next iteration was increased. By comparing the decision tree methods with and without advanced techniques, we can observe that the method with adaptive resampling for 10 trees gives slightly better performance.

For each of the above two methods, three different representations were used for a film: the plot keywords, the combination of genre, director, and cast, and the hybrid

Table 3
The performance of different methods in building decision trees

	Plot (keywords)	Genre, director, cast	Hybrid
Original	77.3	78.6	78.5
With resampling	78.4	79.8	81.1

of the above two ways. Though there seems to be no significant difference in prediction performance among the three representations, during the experiments we have noticed that the hybrid representations can give more consistent results for different experimental runs. The reason could be that users normally have their own ways to choose films, for example, some tend to focus on the plots, but others on the directors. Therefore, using the hybrid of different features can mediate various cases and then obtain more stable results.

4.3. Evaluation of the collaboration agent

As is indicated, this system includes a collaboration agent that collects film programmes rated by other users with similar tastes, to reduce a user's load in rating items as training examples. To evaluate this collaborative approach, we chose a set of users (each has at least 120 rating records) from a popular on-line database (http://www.cs.umn.edu/research/GroupLens/) that contains records of many people's preferences on movies. Here, the similarity between any two users is defined as the subtracting result of the number of movies rated as the same by the two users and the number of movies rated as different. If the measuring result exceeds a certain threshold (i.e. 20 in our experiments), their preferences on movies are considered as similar, and by which the nearest neighbor of a certain user can be determined. To examine the effect of information sharing described above, we conducted three sets of experiments. In the first and second sets of experiments, 60 and 80 training examples were used to learn each user's preferences, respectively, while in the third set, 60 examples from each user and another 20 from his nearest neighbor were used. Table 4 presents the test results. As can be seen, both sets of experiments with 80 training examples can obtain similar results and they are better than the one with 60 examples. It shows that using the collaborative approach to automatically collect more information from similar users is an efficient way to reduce a user's load in building his personal preference model.

4.4. Analysis of the adaptation agent

To evaluate the adaptability of our system, we conduct two kinds of experiments to examine how the system works

Table 4
The results of experiments with and without the use of collaborative approach

	Number of training examples		
	60	80	60 + 20
Accuracy (%)	71.9	74.3	74.1

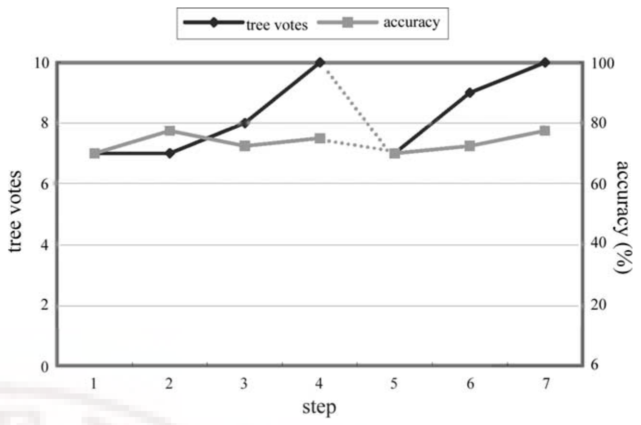


Fig. 6. An example shows that the rank of a chosen programme can be raised to match the user's selection, and the overall system performance did not decline. The user chose two different programmes with seven tree votes at step 1 and step 5, respectively.

in practice. The first was for the fine-tune situation (as described in Section 3.3) in which the goal was to increase the positive tree identifications for the specific programme chosen by a user. A straightforward way is to gradually add copies of this programme to the profile as the training examples to increase its corresponding effect in developing a new classifier. The method described in Section 3.3 was thus implemented to increase the priority of the specific programme to achieve this goal. Fig. 6 shows two experimental cases in which the user chose a programme with seven tree votes (rather than the ones with 10 votes) from the programmes in the class of user-like at step 1, but its rank can be successfully raised afterwards. At the simulation step 7, the user chose the other programme with seven tree votes again. As can be seen, with the strategy used, the rank of the programme chosen can be gradually raised to the top again. The other curve in Fig. 7 shows that the overall system performance (i.e. prediction

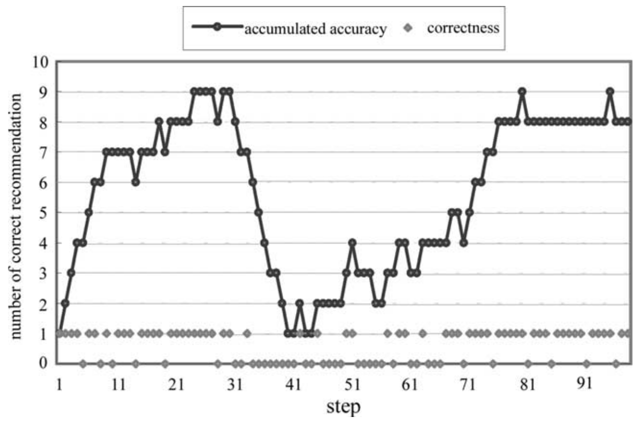


Fig. 7. An example illustrates how the system performance changes during the 100 consecutive recommending steps.

accuracy) remained the same during the experiment; it did not decline because of the method used.

The second experiment was to deal with the situation when the prediction fault occurred. Different from the above fine-tune case, here the programme misclassified was used to substitute the training example firstly added to the user's profile. Fig. 7 shows an example of how the system performance changed during the 100 consecutive recommending steps for a certain user. In this figure, each data point in the upper curve indicates the number of correct cases accumulated from the most recent 10 recommendations, and the data point at the lower part of the figure, the correctness of each single prediction. As can be seen, with the mechanism of adaptation, the system can keep its performance at a certain level. In this example, the user suddenly changed his preferences into completely different ones at the 30th step (it was simulated by exchanging the positive and negative cases) and it caused the decline of system performance. For each fault, the adaptation agent resumed the learning agent to build a new model. After most of the original training examples were replaced by the new ones, the model learnt could then represent the user's current interests and the system performance was thus improved.

5. Conclusions and future work

In this paper, we emphasize the importance of providing personalized information services for the future information appliances to overcome the problem of information overload along with the exponentially increasing digital information. We also suggest that developing personalized recommender systems is a promising way to achieve the task. Therefore in this work, we investigate the appliance for multi-media broadcasting as an example. To achieve the personalization, we present a framework that mainly includes three parts: information gathering, user modeling, and system adaptation. The first part is to collect a user's personal information in both implicit and explicit ways; the second part focuses on how the system can know the individual user by reasoning his preferences; and the third part emphasizes the system's adaptation ability in response to the user's feedback. The decision tree approach with a technique in reducing the overfitting phenomenon is employed in user modeling and a series of experiments are conducted to examine the efficiency. Also the k-nearest neighbor method is implemented for performance comparison. As there are two different types of TV programmes considered here, an arbitrator is designed for coordination. In addition, the system is evaluated to show how it can re-build a user's model when the user does not choose the programme with the highest recommending priority. Experimental results have shown the promise of our proposed framework and its potential for the Internet-based information applicances.

As is explained, using sparse samples to model user preferences could lead to loose results in prediction, but it is a more realistic situation. In the experiments presented, the collaborative method has shown the possibility to overcome this problem. In our current implementation, a binary decision making strategy is employed and the instances are, therefore, categorized into two extremely different classes of user-like and user-dislike. Under such circumstances, instances that are slightly different could be dispatched to different groups and such a categorization makes no difference with that of two instances completely different. One possible way to remedy the above situation is to evaluate products in terms of different levels of preferences, for example, to ask the user to rank products by a five-scale criterion. The other alternative is to use the fuzzy technique (Yager & Zadeh, 1992) to develop decision trees. Both of them could reduce the evaluation bias and obtain more reliable and robust models. Now we have been conducting more experiments to address this issue.

Based on the work presented, we have also been extending the framework to investigate how to recommend programmes in the broadcasting environment including more other types of programmes. For multiple types of programmes, more complicated arbitrators are to be considered. Another important issue is to examine the possibility of applying the personalization techniques to more other information appliances.

References

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

Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: using social and content-based information in recommendation. *Proceedings of National Conference on Artificial Intelligence*, 714–720.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., & Riedl, A. (1999). Combining collaborative filtering with personal agents for better recommendations. *Proceedings of National Conference on Artificial Intelligence*, 439–446.

Joachims, T., Freitag, D., & Mitchell, T. (1997). WebWatcher: a tour guide for the World Wide Web. *Proceedings of International Joint Conference on Artificial Intelligence*.

Lee, W.-P., Liu, C.-H., & Lu, C.-C. (2002). Intelligent agent-based systems for personalized recommendations in Internet commerce. *Expert Systems with Applications*, *22*(4), 275–284.

Lewis, T. (1998). Information appliances: gadget netopia. *IEEE Computer*, *January*, 59–68.

Mlademic, D. (1999). Text-learning and related intelligent agents: a survey. *IEEE Intelligent Systems*, *14*(4), 44–54.

Mooney, R. J., & Roy, L. (2000). Content-based book recommending using learning for text categorization. *Proceedings of the ACM International Conference on Digital Libraries*, 195–204.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. San Mateo: Morgan Kaufmann.

Salton, G. (1989). *Automatic text processing*. Reading, MA: Addison-Wesley.

Shardanand, U., & Maes, P. (1995). Social information filtering: algorithms for automating. "Word of Mouth". *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 210–217.

Smyth, B., & Cotter, P. (2000). A personalised TV listings service for the digital TV age. *Knowledge-Based Systems*, *13*, 53–59.

Want, R., & Borriello, G. (2000). Survey on information appliances. *IEEE Computer Graphics and Applications, special issue on Information Appliances*, *20*(3), 24–31.

Weiss, S. M., Apte, C., Dameran, F. J., Johnson, D. E., Oles, F. J., Goetz, T., & Hampp, T. (1999). Maximizing text-mining performance. *IEEE Intelligent Systems*, *14*(4), 63–69.

Yager, R. R., & Zadeh, L. A. (1992). *An introduction to fuzzy logic applications in intelligent systems*. Boston: Kluwer Academic Publishers.

From "Machine Learning: Proceedings of the Twelfth International Conference" (ML95)

Fast Effective Rule Induction

William W. Cohen
AT&T Bell Laboratories
600 Mountain Avenue Murray Hill, NJ 07974
wcohen@research.att.com

Abstract

Many existing rule learning systems are computationally expensive on large noisy datasets. In this paper we evaluate the recently-proposed rule learning algorithm IREP on a large and diverse collection of benchmark problems. We show that while IREP is extremely efficient, it frequently gives error rates higher than those of C4.5 and C4.5rules. We then propose a number of modifications resulting in an algorithm RIPPER*k* that is very competitive with C4.5rules with respect to error rates, but much more efficient on large samples. RIPPER*k* obtains error rates lower than or equivalent to C4.5rules on 22 of 37 benchmark problems, scales nearly linearly with the number of training examples, and can efficiently process noisy datasets containing hundreds of thousands of examples.

1 INTRODUCTION

Systems that learn sets of rules have a number of desirable properties. Rule sets are relatively easy for people to understand [Catlett, 1991], and rule learning systems outperform decision tree learners on many problems [Pagallo and Haussler, 1990; Quinlan, 1987; Weiss and Indurkhya, 1991]. Rule sets have a natural and familiar first order version, namely Prolog predicates, and techniques for learning propositional rule sets can often be extended to the first-order case [Quinlan, 1990; Quinlan and Cameron-Jones, 1993]. Certain types of prior knowledge can also be easily communicated to rule learning systems [Cohen, 1994; Pazzani and Kibler, 1992].

One weakness with rule learning systems is that they often scale relatively poorly with the sample size, particularly on noisy data [Cohen, 1993]. Given the prevalence of large noisy datasets in real-world applications, this problem is of critical importance. The goal of this paper is to develop propositional rule learning algorithms that perform efficiently on large noisy datasets, that extend naturally to first-order representations, and that are competitive in generalization performance with more mature symbolic learning methods, such as decision trees. The end product of this effort is the algorithm RIPPER*k*, which is competitive with C4.5rules with respect to error rates, scales nearly linearly with the number of training examples, and can efficiently process noisy datasets containing hundreds of thousands of examples.

2 PREVIOUS WORK

2.1 COMPLEXITY OF RULE PRUNING

Many of the techniques used in modern rule learners have been adapted from decision tree learning. Most widely-used decision tree learning systems use an *overfit-and-simplify* learning strategy to handle noisy data: a hypothesis is formed by first growing a complex tree which "overfits" the data, and then simplifying or *pruning* the complex tree [Quinlan, 1987; Mingers, 1989]. Usually (but not always) such pruning strategies improve error rates on unseen data when the training data is noisy [Quinlan, 1987; Mingers, 1989; Schaffer, 1992]. A variety of methods have been proposed to prune trees, but one effective technique is *reduced error pruning (REP)*. REP can be easily adapted to rule learning systems [Pagallo and Haussler, 1990; Brunk and Pazzani, 1991].

In REP for rules, the training data is split into a *growing set* and a *pruning set*. First, an initial rule set is formed that overfits the growing set, using some heuristic method. This overlarge rule set is then repeatedly simplified by applying one of a set of *pruning operators*; typical pruning operators would be to delete any single condition or any single rule. At each stage of simplification, the pruning operator chosen is the one that yields the greatest reduction of error on the pruning set. Simplification ends when applying any pruning operator would increase error on the pruning set.

REP for rules usually does improve generalization performance on noisy data [Pagallo and Haussler, 1990; Brunk and Pazzani, 1991; Weiss and Indurkhya, 1991; Cohen, 1993; Fürnkranz and Widmer, 1994]; however, it is computationally expensive for large datasets. In previous work [Cohen, 1993] we showed that REP requires $O(n^4)$ time, given sufficiently noisy data; in fact, even the initial phase of overfitting the training data requires $O(n^2)$ time. We then proposed an alternative overfit-and-simplify method called Grow that is competitive with REP with respect to error rates, and was an order of magnitude faster on a set of benchmark problems.

We also showed that Grow was asymptotically faster than REP on random data—if one assumes that Grow's hypothesis is approximately the same size as the target concept. However, Cameron-Jones [1994] later showed that Grow systematically overfits the target concept on noisy data. This has an adverse effect on Grow's time complexity and as a result Grow also requires $O(n^4)$ time asymptotically.

In another response to the inefficiency of REP, Fürnkranz and Widmer [1994] proposed a novel learning algorithm called *incremental reduced error pruning (IREP)*. IREP was shown experimentally to be competitive with both REP and Grow with respect to error rates, and much faster than either; in fact, on 18 of 20 benchmark problems, IREP was faster than the initial step of overfitting the data.

In this paper, we will take as our point of departure the promising results obtained by Fürnkranz and Widmer with the IREP algorithm. Our initial goal was simply to replicate their results, to evaluate IREP on a broader set of benchmarks, and to compare IREP to more mature tree and rule induction methods. In the course of doing this, we discovered that IREP's generalization performance could be considerably improved, without greatly affecting its computational efficiency. In the remainder of the paper we will describe our implementation of the original IREP algorithm, and give evidence that it affords room for improvement. We will then outline three modifications: a new metric for guiding its pruning phase, a new stopping condition, and a technique for "optimizing" the rules learned by IREP. Taken together these modifications give generalization performance that is comparable to C4.5 and C4.5rules [Quinlan, 1994] on a large set of diverse benchmarks. The modified learning algorithm, however, still scales well with the number of training examples. The current implementation can efficiently handle training sets of several hundred thousand examples.

```
procedure IREP(Pos,Neg)
begin
  Ruleset := $\emptyset$
  while Pos$\neq\emptyset$ do
    /* grow and prune a new rule */
    split (Pos,Neg) into (GrowPos,GrowNeg)
      and (PrunePos,PruneNeg)
    Rule := GrowRule(GrowPos,GrowNeg)
    Rule := PruneRule(Rule,PrunePos,PruneNeg)
    if the error rate of Rule on
      (PrunePos,PruneNeg) exceeds 50% then
        return Ruleset
    else
      add Rule to Ruleset
      remove examples covered by Rule
        from (Pos,Neg)
    endif
  endwhile
  return Ruleset
end
```

Figure 1: The IREP algorithm

2.2 INCREMENTAL REDUCED ERROR PRUNING

The IREP rule-learning algorithm is described in detail by Fürnkranz and Widmer [1994], but we will summarize it below. IREP tightly integrates reduced error pruning with a separate-and-conquer rule learning algorithm. Figure 1 presents a two-class version of this algorithm. (In the two-class Boolean case a "rule" is simply a conjunction of features, and a "rule set" is a DNF formula.) Like a standard separate-and-conquer algorithm, IREP builds up a rule set in a greedy fashion, one rule at a time. After a rule is found, all examples covered by the rule (both positive and negative) are deleted. This process is repeated until there are no positive examples, or until the rule found by IREP has an unacceptably large error rate.

In order to build a rule, IREP uses the following strategy. First, the uncovered examples are randomly partitioned into two subsets, a *growing set* and a *pruning set*. In our implementation, the growing set contains 2/3 of the examples.

Next, a rule is "grown". Our implementation of GrowRule is a propositional version of FOIL [Quinlan, 1990; Quinlan and Cameron-Jones, 1993]. It begins with an empty conjunction of conditions, and considers adding to this any condition of the form $A_n = v$, $A_c \leq \theta$, or $A_c \geq \theta$, where A_n is a nominal attribute and v is a legal value for A_n, or A_c is a continuous variable and θ is some value for A_c that occurs in the training data. GrowRule repeatedly adds the condition that maximizes FOIL's information gain criterion until the rule covers no negative examples from the growing dataset.

After growing a rule, the rule is immediately pruned. To prune a rule, our implementation considers deleting any final sequence of conditions from the rule, and chooses the deletion that maximizes the function

$$v(\mathit{Rule}, \mathit{PrunePos}, \mathit{PruneNeg}) \equiv \frac{p + (N - n)}{P + N} \qquad (1)$$

where P (respectively N) is the total number of examples in *PrunePos* (*PruneNeg*) and p (n) is the number of examples in *PrunePos* (*PruneNeg*) covered by *Rule*. This process is repeated until no deletion improves the value of v.

The IREP algorithm described above is for two-class learning problems. Our implementation handles multiple classes as follows. First, the classes are ordered. In the experiments described below the ordering is always in increasing order of prevalence—*i.e.*, the ordering is $C_1, \ldots, C_k$ where C_1 is the least prevalent class and C_k is the most prevalent. Then, IREP is used to find a rule set that separates C_1 from the remaining classes; this is done with a single call to IREP where *PosData* contains the examples labeled C_1 and *NegData* contains the examples labeled C_2, C_3, ..., or C_k. Next, all instances covered by the learned rule set are removed from the dataset, and IREP is used to separate C_2 from classes $C_3, \ldots, C_k$. This process is repeated until a single class C_k remains; this class will be used as the default class.

We also extended the rule learning algorithm to handle missing attributes as follows: all tests involving the attribute A are defined to fail on instances for which the value of A is missing. This encourages the learner to separate out the positive examples using tests that are known to succeed.

2.3 DIFFERENCES FROM FÜRNKRANZ AND WIDMER'S IREP

This implementation differs from Fürnkranz and Widmer's in several details. In pruning rules, our implementation allows deletions of any final sequence of conditions, whereas Fürnkranz and Widmer's implementation allows only deletions of a single final condition. Our implementation also stops adding rules to a rule set when a rule is learned that has error rate greater than 50%, whereas Fürnkranz and Widmer's implementation stops when the accuracy of the rule is less than the accuracy of the empty rule.[1]

More importantly, our implementation supports missing attributes, numerical variables and multiple classes. This makes it applicable to a wider range of benchmark problems.

3 EXPERIMENTS WITH IREP

Experiments with IREP showed that it is indeed fast. Results for one representative artificial problem[2] are summarized in the first graph in Figure 2; the CPU time needed by C4.5rules is also shown.[3] The results are shown on a log-log scale; recall that polynomials appear as lines on such a plot, with the slope of the line indicating its degree. C4.5rules scales roughly as the cube of the number of examples, whereas IREP scales almost linearly. Extrapolating the curves suggests that it would require about 79 CPU years for C4.5rules to process the 500,000 example dataset, which IREP handles in around seven CPU minutes.

Although we have used an artificial concept with an extremely large number of training examples to demonstrate these issues, similar performance issues also arise on natural datasets, as the two smaller graphs of Figure 2 demonstrate.

For reference, the first graph in Figure 2 also shows the curves kx^3 and $y = kx \log^2 x$. Fürnkranz and Widmer's formal analysis of IREP predicts a running time of $O(m \log^2 m)$, where m is the number of examples, on

[1]Actually, Fürnkranz and Widmer described two pruning algorithms. The first, which they called IREP, prunes according to Equation 1, and stops when $p/(p+n) < N/(P+N)$. The second, which they called IREP2, prunes according to the metric $v(\mathit{Rule}, \mathit{PrunePos}, \mathit{PruneNeg}) \equiv \frac{p}{p+n}$ and stops when $p/(p+n) < 1/2$. Our experiments confirmed the conclusion of Fürnkranz and Widmer that IREP generally outperforms IREP2; however, we also discovered that IREP's performance was noticibly improved by adopting IREP2's stopping condition.

[2]The concept $ab \vee bcd \vee defg$ with 12 irrelevant binary attributes, 20% classification noise, and uniformly distributed examples. CPU time was measured on a MIPS Irix 5, configured with 8 150 MHz R4400 processors and 1Gb of memory. Since IREP is a randomized algorithm (because of its random partitioning of the examples) the curve for IREP is the average of 10 trials.

[3]The time for C4.5rules ignores the time needed to run C4.5. However, C4.5 is generally much faster than C4.5rules; on this problem, C4.5 requires less than 400 CPU seconds to handle the 500,000 example dataset. The run-time of C4.5 is generally comparable to that of IREP.

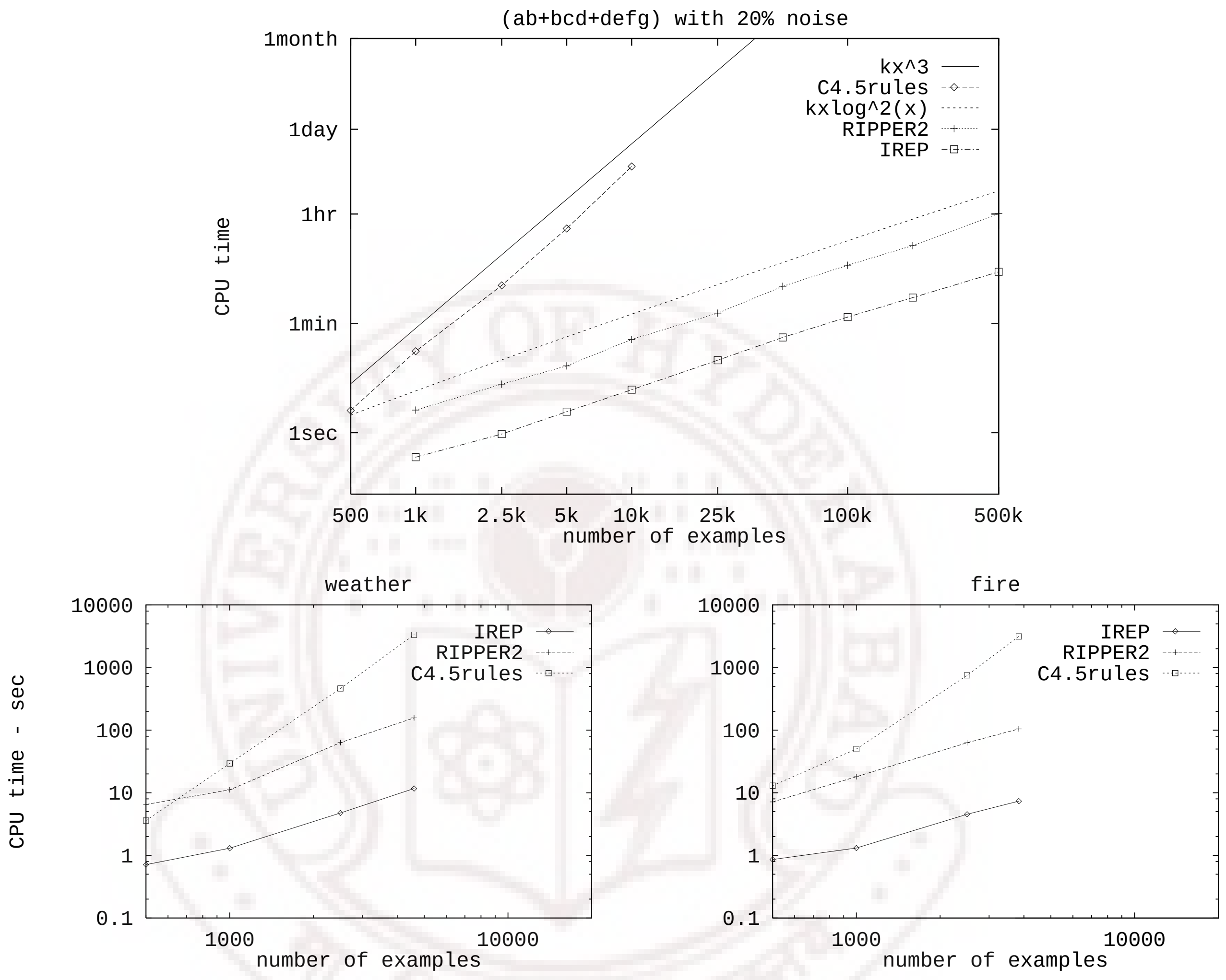


Figure 2: CPU times for C4.5rules, IREP, and RIPPER2

any dataset that contains a fixed percentage of classification noise. Our results are consistent with this prediction. Analysis similar to Fürnkranz and Widmer's also predicts the cubic behavior shown by C4.5rules.

Although IREP is efficient, experiments on real-world datasets showed that the generalization performance of IREP offered substantial room for improvement. We compared IREP to C4.5 and C4.5rules on a diverse set of benchmark problems, summarized in Table 1. Where a test set associated with the benchmark is indicated, we ran C4.5 and C4.5rules once, and ran IREP 10 times and averaged. Where no test set is indicated, we ran 10 different 10-fold cross-validations for all the algorithms and averaged the results. Due to space considerations we will focus on comparisons to C4.5rules, since it also learns rule sets; however, the performance of C4.5 and C4.5rules on these datasets was similar. We used C4.5 Release 6 [Quinlan, 1994], and the most recent version of C4.5rules [Quinlan, 1995].

The left-hand graph of Figure 3 contains one point for each benchmark problem, positioned so that IREP's error rate is the x-axis position and C4.5rules' error rate is the y-axis position. Thus for points below the line $y = x$ IREP's performance is inferior to C4.5rules, and for points above the line IREP's performance is better. From the graph one can readily see that IREP does worse than C4.5rules more often than it does better; specifically, IREP's error rate is higher 23 times, lower 11 times, and the same 3 times.

Of course, it may be that IREP is in fact as likely to outperform C4.5rules as the converse on problems from this test suite, and that the won-lost-tie ratio of 11-23-3 is due to random variation in the error esti-

Table 1: The 37 benchmark problems used in the experiments, with size of training and testing sets; number of classes; number of nominal (n) and continuous (c) attributes; and a brief description. Starred problems are from the UC/Irvine Repository.

Name	Train	Test	Classes	Attributes		Description
AP1-10	999	–	2	85-130n		text categorization (10 problems)
audiology∗	226	–	24	60n		medical diagnosis
bridges1-5∗	106	–	2-6	6n	1c	mech. engineering (5 problems)
iris∗	150	–	3		4c	flower classification
labor∗	57	–	2	8n	8c	labor negotiations
promoters∗	106	–	2	57n		DNA promoter sequences
sonar∗	208	–	2		60c	sonar signal classification
ticket1-3	556	–	2	78n		text categorization (3 problems)
ui	373	–	18	10n		text-to-speech subproblem
coding1∗	5000	15000	2	15n		DNA coding sequences
fire	3225	608	8		11c	risk of forest fires
market	3181	1616	2	3n	7c	market analysis
mushroom∗	3988	4136	2	22n		random split of mushroom data
netwk1	2500	1077	2		30c	predict equipment failure
netwk2	2600	1226	2		35c	predict equipment failure
ocr	1318	1370	2	576n		image classification
segment∗	1133	1177	7	19n		image analysis
splice∗	1614	1561	3	60n		split of DNA splice-junction data
thyroid∗	2514	1258	5	22n	7c	medical diagnosis
vidgame	1484	1546	2	10n		decide if game moves are random
voting∗	300	135	2	16n		congressional voting records
weather	1000	4597	2		35c	weather prediction

mates. Using a nonparametric sign test [Mendenhall *et al.*, 1981, page 578], one can determine that the probability of observing a ratio this one-sided would be just under 0.05 if IREP had a 50/50 chance of bettering C4.5rules on problems in this test suite. We can thus conclude with 95% confidence that C4.5rules outperforms IREP on this test suite.[4]

It is also evident from the graph that IREP seldom does much better than C4.5rules, and not infrequently does much worse. It is not obvious how to best aggregate measurements across learning problems, but one method is to consider the average value of the ratio

$$\frac{\textit{error rate of IREP}}{\textit{error rate of C4.5rules}}$$

For this set of problems the average of this ratio is 1.13, if one discounts a single extreme outlier; thus on average IREP's error rates are about 13% higher than those of C4.5rules. (This average is 1.52 if one includes the *mushroom* dataset—on this benchmark C4.5rules obtains an error of 0.2% to IREP's 3.1%.)

As an additional point of reference, we also ran propositional FOIL without any pruning mechanism. The ratio of the error rate of the hypothesis obtained by "overfitting" the data with propositional FOIL to the error rate of C4.5rules is 1.17 excluding the *mushroom* dataset, and 1.14 overall. Finally, we ran IREP2 (also described by Fürnkranz and Widmer [1994]) and IREP with Fürnkranz and Widmer's stopping condition. The average ratio for IREP2 was 1.15 without the *mushroom* dataset, and 1.14 overall. For IREP with the more restrictive Fürnkranz and Widmer stopping condition, the average ratio was 1.71 without *mushroom* and 2.08 overall. The best won-loss-tied record of any of these three systems relative to C4.5rules was 17-20-0, achieved by propositional FOIL without pruning. To summarize, on average, all of the IREP variants performed substantially worse than C4.5rules, and none of the IREP variants performed substantially better than simply overfitting the data.

There is also evidence that IREP fails to converge on some natural datasets. One example is the well-known KRK-illegal problem [Muggleton *et al.*, 1989;

[4]More precisely, we can conclude that C4.5rules outperforms IREP in this sense: if a problem is drawn at random from this test suite and its error rate is measured as described above, then with probability greater than 0.5, the measured error rate of C4.5rules will be lower than that of IREP.

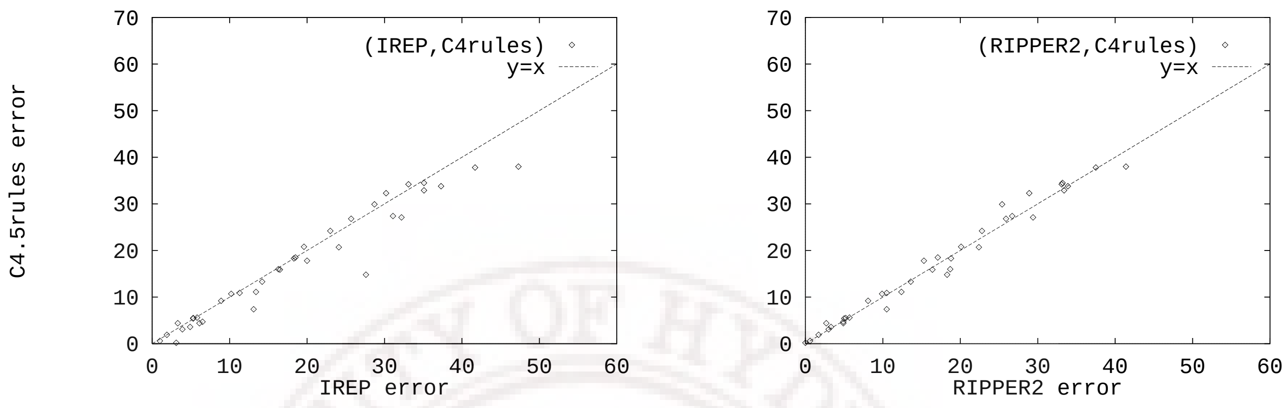


Figure 3: Comparison of generalization performances: C4.5rules *vs.* IREP and RIPPER2.

Quinlan, 1990]. We encoded a propositional version of this problem, and implemented a data generator.[5] Without noise, IREP reliably learns an approximate theory with an error rate of 0.6% from as few as 100 examples; however, IREP does not improve this error rate even if as many as 100,000 examples are given. In contrast C4.5rules reliably produces a perfect theory from only 5000 examples. Artificial examples can also be constructed which show non-convergence to a greater extent; for example, IREP obtains an error of 9.5% given anywhere between 100 and 100,000 noise-free examples of the concept $ab \vee ac \vee ade$. This is worrisome behavior for an algorithm whose main strength is that it efficiently handles very large numbers of examples.

4 IMPROVEMENTS TO IREP

Based on our experiments with IREP, we implemented three modifications to the algorithm: an alternative metric for assessing the value of rules in the pruning phase of IREP; a new heuristic for determining when to stop adding rules to a rule set; and a postpass that "optimizes" a rule set in an attempt to more closely approximate conventional (*i.e.*, non-incremental) reduced error pruning.

4.1 THE RULE-VALUE METRIC

The occasional failure of IREP to converge as the number of examples increases can be readily traced to the metric used to guide pruning (given above in Equation 1). The preferences encoded in this metric are sometimes highly unintuitive; for instance (assuming that P and N are fixed) the metric prefers a rule R_1 that covers $p_1 = 2000$ positive examples and $n_1 = 1000$ negative examples to a rule R_2 that covers $p_1 = 1000$ examples and $n_1 = 1$ negative example; note, however, that R_2 is highly predictive and R_1 is not. We thus replaced IREP's metric with

$$v^*(Rule, PrunePos, PruneNeg) \equiv \frac{p-n}{p+n}$$

which seems to have more intuitively satisfying behavior.

4.2 THE STOPPING CONDITION

Our implementation of IREP stops greedily adding rules to a rule set when the last rule constructed has an error exceeding 50% on the pruning data. This heuristic often stops too soon given moderate-sized samples; this is especially true when learning a concept containing many low-coverage rules. Our assessment of the problem is that for low-coverage rules, the estimate of error afforded by the pruning data has high variance; thus in learning a series of small rules, there is a good chance that one of the rules in the series will have its error rate incorrectly assessed at more than 50%, causing IREP to stop prematurely. Put another way, IREP seemed to be unduly sensitive to the "small disjunct problem" [Holte *et al.*, 1989].

Our solution to this problem is the following. After each rule is added, the total *description length* of the rule set and the examples is computed. The new version of IREP stops adding rules when this description length is more than d bits larger than the smallest de-

[5] Our propositional encoding is the one that would be constructed by LINUS [Džeroski and Lavrac, 1991], and we used a uniform distribution to generate KRK positions.

scription length obtained so far, or when there are no more positive examples. In the experiments of this paper we used $d = 64$. The rule set is then simplified by examining each rule in turn (starting with the last rule added) and deleting rules so as to reduce total description length.[6]

Together, the revised rule-value metric and stopping heuristic substantially improve IREP's generalization performance. Unlike the original IREP, the modified version of IREP (henceforth IREP∗) converges *KRK-illegal* and the artificial concept $ab \vee ac \vee ade$. IREP∗'s won-lost-tied record against IREP is 28-8-1; thus with high confidence ($p > 0.992$) one can state that IREP∗ outperforms IREP on problems from this test suite. The error ratio to C4.5rules is also reduced from 1.13 (or 1.52, including *mushroom*) to 1.06 (or 1.04, including *mushroom*.) IREP∗'s won-lost-tied record against C4.5rules is 16-21-0.

4.3 RULE OPTIMIZATION

The repeated grow-and-simplify approach used in IREP can produce results quite different from conventional (non-incremental) reduced error pruning. One way to possibly improve IREP∗'s incremental approach is to postprocess the rules produced by IREP∗ so as to more closely approximate the effect of conventional reduced error pruning. For instance, one could re-prune each rule so as to minimize the error of the complete rule set.

After some experimentation we developed the following method for "optimizing" a rule set $R_1, \ldots, R_k$. Each rule is considered in turn: first R_1, then R_2, etc, in the order in which they were learned. For each rule R_i, two alternative rules are constructed. The *replacement for* R_i is formed by growing and then pruning a rule R'_i, where pruning is guided so as to minimize error of the entire rule set $R_1, \ldots, R'_i, \ldots, R_k$ on the pruning data. The *revision of* R_i is formed analogously, except that the revision is grown by greedily adding conditions to R_i, rather than the empty rule. Finally a decision is made as to whether the final theory should include the revised rule, the replacement rule, or the original rule. This decision is made using the MDL heuristic.[7] Optimization is integrated with IREP∗ as follows. First, IREP∗ is used to obtain an initial rule set. This rule set is next optimized as described above. Finally rules are added to cover any remaining positive examples using IREP∗. Below, we will call this algorithm RIPPER (for <u>R</u>epeated <u>I</u>ncremental <u>P</u>runing to <u>P</u>roduce <u>E</u>rror <u>R</u>eduction.).

Optimization can also be iterated by optimizing the rule set output by RIPPER and then adding additional rules using IREP∗; we will call this algorithm RIPPER2, and in general use RIPPER*k* for the algorithm that repeatedly optimizes k times.

[6]To briefly summarize our MDL encoding scheme: the method used for encoding a set of examples given a theory is the same as that used in the latest version of C4.5rules [Quinlan, 1995]. One part of this encoding scheme allows one to identify a subset of k elements of a known set of n elements using

$$S(n,k,p) \equiv k \log_2 \frac{1}{p} + (n-k) \log_2 \frac{1}{1-p}$$

bits, where p is known by the recipient of the message. Thus we allow $||k|| + S(n,k,k/n)$ bits to send a rule with k conditions, where n is the number of possible conditions that could appear in a rule and $||k||$ is the number of bits needed to send the integer k. As in C4.5rules [Quinlan, 1994, page 53] the estimated number of bits required to send the theory is multiplied by 0.5 to adjust for possible redundancy in the attributes.

Table 2: Summary of generalization results

	won-loss-tied *vs* C4.5rules	error ratio to C4.5rules[a]		
IREP[b]	9-28-0	2.08	1.71	1.93
IREP2	11-25-1	1.15	1.15	1.22
IREP[c]	11-23-3	1.51	1.13	1.20
IREP∗	16-21-0	1.04	1.06	1.09
RIPPER	20-15-2	0.98	1.01	1.03
RIPPER2	21-15-1	0.97	0.99	1.01

[a]Format: all datasets; all datasets except *mushroom*; all datasesets except *mushroom* and weighting similar datasets together.
[b]Using Fürnkranz and Widmer's stopping criterion.
[c]As described in Section 2.3.

4.4 GENERALIZATION PERFORMANCE

RIPPER noticibly improves generalization performance over IREP∗. Its won-lost-tied record against IREP∗ is 28-7-2, a significant improvement ($p > 0.9986$). The error ratio to C4.5rules is also reduced: excluding *mushroom*, the error ratio is 1.06 for IREP∗ and 1.01 for RIPPER, and including *mushroom*, the error ratio is 1.04 for IREP∗ and 0.982 for RIPPER. RIPPER's won-lost-tied record against C4.5rules is 20-15-2.

One additional stage of optimization gives some fur-

[7]More precisely, a variant of R_i is evaluated by inserting it into the rule set and then deleting rules that increase the total description length of the rules and examples. The total description length of the examples and the simplified rule set is then used to compare variants of R_i.

ther benefit. RIPPER2 reduces the error ratio to C4.5rules to 0.995 excluding *mushroom*, or 0.968 including *mushroom*, and RIPPER2's won-lost-tied against C4.5rules is improved to 21-15-1. RIPPER2 is not statistically significantly better than C4.5rules; however, RIPPER2 is certainly quite competitive on the problems in this test suite. To make this concrete, let q be the probability that RIPPER2's measured error rate will be less than or equal to that of C4.5rules on a problem taken at random from the test suite. The won-lost-tied record of 21-15-2 means we can be 93% confident that q is at least 0.5, 95% confident that q is at least 0.488, and 99% confident that q is at least 0.431.

The right-hand graph in Figure 3 gives a more detailed comparison of the error rates of RIPPER2 and C4.5rules, and Table 2 summarizes some of the generalization results given in this section.

One problem with averaging error ratios is that when the actual error rates are very small, ratios tend to have extreme values. (This is the reason why we have reported all averages with and without the *mushroom* dataset: for this dataset the actual error rates range from 0.0% to 3.1% and the ratios range from 0.0 to 17.5.) The following remarks may help reassure readers of the stability of our comparison:

- If groups of similar datasets are weighted together,[8] then the average ratio of RIPPER2 to C4.5rules is 0.957. If *mushroom* is excluded, then the weighted average ratio is 1.005.
- If the two largest and the two smallest ratios are excluded, then the average ratio of RIPPER2 to C4.5rules is 0.986. (The ratio for *mushroom* is one of the four extreme values.)
- The average difference between RIPPER2's error rate and C4.5rules' error rate is -0.1%.
- The won-loss-tied record of RIPPER2 to the C4.5 decision tree learner (with pruning) is 23-12-2. The average ratio of RIPPER2 to C4.5 with pruning is 0.964 with *mushroom*, and 0.991 without.

4.5 EFFICIENCY OF RIPPERk

Importantly, none of the modifications we have described have a major effect on computational efficiency. Figure 2 also shows how RIPPER2 scales with the number of examples on three concepts: one artificial concept, and two of the larger and noisier natural datasets in our test suite. The fact that the lines for RIPPER2 and IREP are parallel shows that the modifications we have introduced affect only the constant factors, and not the asymptotic complexity of the algorithm. The constant factors for RIPPER2 are also still reasonably low: RIPPER2 requires only 61 CPU minutes to process 500,000 examples of the artificial concept of Figure 2. RIPPERk is also quite space efficient, as it requires no data structures larger than the dataset.

In previous work [Cohen, 1993] we sought formal explanations for the efficiency or inefficiencies of REP and other rule-pruning algorithms. While space does not permit such an analysis here, we would like to present some of the intuitions as to why RIPPERk is so much faster on large noisy datasets.

The basic strategy used by RIPPERk to find a ruleset that models the data is to first use IREP$*$ to find an initial model, and then to iteratively improve that model, using the "optimization" procedure described in 4.3. This process is efficient because building the initial model is efficient, because the initial model does not tend to be large relative to the target concept, and because the optimization steps only require time linear in the number of examples and the size of the initial model.

C4.5rules also constructs an initial model and then iteratively improves it. However, for C4.5rules, the initial model is a subset of rules extracted from a unpruned decision tree, and the improvement process greedily deletes or adds single rules in an effort to reduce description length. C4.5rules repeats this process for several different-sized subsets of the total pool of extracted rules and uses the best ruleset found as its hypothesis; the subsets it uses are the empty ruleset, the complete ruleset, and randomly-chosen subsets of 10%, 20%, ..., and 90% of the rules.

Unfortunately, for noisy datasets, the number of rules extracted from the unpruned decision tree grows as m, the number of examples. This means that each initial model (save the empty model) will also be of size proportional to m, and hence if m is sufficiently large, *all* of the initial models will be much larger than the target hypothesis. This means that to build a theory about the same size as the target concept always requires many (on the order of m) changes to the initial model, and at each step in the optimization, many (on the order of m) changes are possible. The improvement process is thus expensive; since it is a greedy search, it is also potentially quite likely to miss finding

[8] "Weighting similar datasets together" means that the ratios for the ten *AP* datasets, the five *bridges* datasets, the three *ticket* datasets and the two *network* datasets are each averaged together before being averaged with the ratios for the remaining seventeed datasets.

the best ruleset.[9]

In summary, both RIPPER*k* and C4.5rules start with an initial model and iteratively improve it using heuristic techniques. However, for large noisy datasets, RIPPER*k* generally seems to start with an initial model that is about the right size, while C4.5rules starts with an over-large initial model. This means that RIPPER*k*'s search is more efficient. We conjecture also that RIPPER*k*'s search is also more effective on large noisy datasets. (RIPPER2 generally seems to do better compared to C4.5rules on larger datasets; in particular for datasets with no more than 150 examples, the average ratio of RIPPER2 to C4.5rules is 1.051, and for datasets with more than 150 examples, the average ratio of RIPPER2 to C4.5rules is 0.944.)

5 CONCLUSIONS

Incremental reduced error pruning (IREP) is a recent rule learning algorithm that can efficiently handle large noisy datasets. In this paper we have presented some experiments on a large collection of benchmark problems with an extended implementation of IREP which allows continuous variables and multiple classes. We showed that IREP does not perform as well as the more mature (but also more expensive) rule learning algorithm C4.5rules.

We also proposed a series of improvements to IREP that make it extremely competitive with C4.5rules, without seriously affecting its efficiency. IREP$*$ incorporates a new metric to guide rule pruning and an MDL-based heuristic for determining how many rules should be learned. RIPPER*k* adds to this *k* iterations of an optimization step that more closely mimics the effect of non-incremental reduced error pruning.

IREP$*$ and RIPPER*k* were shown statistically to be clear improvements over IREP on problems from our test suite. RIPPER2 is also extremely competitive with C4.5rules; in fact on 22 of 37 problems in the test suite RIPPER2 achieves error rates lower than or equivalent to those of C4.5rules.

However, on noisy datasets, RIPPER*k* is much more efficient than C4.5rules. It scales nearly linearly with the number of examples in a dataset; in contrast C4.5rules scales as the cube of the number of examples. This asymptotic improvement translates to speedups of several orders of magnitude on problems of modest size (up to a few thousand examples), and the ability to effectively process datasets containing several hundreds of thousands of noisy examples.

[9]This situation should be contrasted to decision tree pruning, in which even a large tree can be pruned efficiently and, in certain senses, optimally; for instance, the pruned tree with the lowest error on a pruning set can be found in linear time.

Acknowledgements

Much of this research was conducted during a visit to the University of Sydney, which was funded by a grant to Ross Quinlan from the Australian Research Council. This research and this paper have benefitted substantially from numerous helpful discussions with Ross Quinlan, Nitin Indurkhya, and other members of University of Sydney machine learning community. The author is also grateful for valuable suggestions from Jason Catlett, for comments from Jason Catlett and Haym Hirsh on a draft of the paper, and for the suggestions made by two anonymous reviewers.

References

(Brunk and Pazzani, 1991) Clifford Brunk and Michael Pazzani. Noise-tolerant relational concept learning algorithms. In *Proceedings of the Eighth International Workshop on Machine Learning*, Ithaca, New York, 1991. Morgan Kaufmann.

(Cameron-Jones, 1994) Michael Cameron-Jones. The complexity of Cohen's Grow method. Unpublished manuscript, 1994.

(Catlett, 1991) Jason Catlett. Megainduction: a test flight. In *Proceedings of the Eighth International Workshop on Machine Learning*, Ithaca, New York, 1991. Morgan Kaufmann.

(Cohen, 1993) William W. Cohen. Efficient pruning methods for separate-and-conquer rule learning systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1993.

(Cohen, 1994) William W. Cohen. Grammatically biased learning: learning logic programs using an explicit antecedent description language. *Artificial Intelligence*, 68:303–366, 1994.

(Džeroski and Lavrac, 1991) Sašo Džeroski and Nada Lavrac. Learning relations from noisy examples. In *Proceedings of the Eighth International Workshop on Machine Learning*, Ithaca, New York, 1991. Morgan Kaufmann.

(Fürnkranz and Widmer, 1994) Johannes Fürnkranz and Gerhard Widmer. Incremental reduced error pruning. In *Machine Learning: Proceedings of the*

Eleventh Annual Conference, New Brunswick, New Jersey, 1994. Morgan Kaufmann.

(Holte *et al.*, 1989) Robert Holte, Liane Acker, and Bruce Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989. Morgan Kaufmann.

(Mendenhall *et al.*, 1981) William Mendenhall, Richard Scheaffer, and Dennis Wackerly, editors. *Mathematical Statistics with Applications.* Duxbury Press, second edition, 1981.

(Mingers, 1989) John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2), 1989.

(Muggleton *et al.*, 1989) Stephen Muggleton, Machael Bain, Jean Hayes-Michie, and Donald Michie. An experimental comparison of human and machine learning formalisms. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, New York, 1989. Morgan Kaufmann.

(Pagallo and Haussler, 1990) Giulia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1), 1990.

(Pazzani and Kibler, 1992) Michael Pazzani and Dennis Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9(1), 1992.

(Quinlan and Cameron-Jones, 1993) J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In Pavel B. Brazdil, editor, *Machine Learning: ECML-93*, Vienna, Austria, 1993. Springer-Verlag. Lecture notes in Computer Science # 667.

(Quinlan, 1987) J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

(Quinlan, 1990) J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3), 1990.

(Quinlan, 1994) J. Ross Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann, 1994.

(Quinlan, 1995) J. Ross Quinlan. MDL and categorical theories (continued). In *Machine Learning: Proceedings of the Twelfth International Conference*, Lake Taho, California, 1995. Morgan Kaufmann.

(Schaffer, 1992) Cullen Schaffer. Sparse data and the effect of overfitting avoidance in decision tree induction. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, California, 1992. MIT Press.

(Weiss and Indurkhya, 1991) Sholom Weiss and Nitin Indurkhya. Reduced complexity rule induction. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991. Morgan Kaufmann.

TV Content Recommender System

Srinivas Gutta, Kaushal Kurapati, KP Lee, Jacquelyn Martino, John Milanski, J. David Schaffer, John Zimmerman

Philips Research
345 Scarborough Rd.
Briarcliff Manor, NY 10510
{srinivas.gutta, kaushal.kurapati, kp.lee, jacquelyn.martino, john.milanski, dave.schaffer, john.zimmerman}@philips.com

Abstract

The plethora of content available to the consumer has become overwhelming. Increasing amounts of information are being disseminated through terrestrial broadcast, satellite, and cable leading to an information overload. Common modes of searching for TV programs currently in existence include: TV-guide, PreVue channel and rudimentary search tools available through satellite dish TV programming service. These tools are general-purpose in nature and are not specifically tailored to the individual viewer's taste. Towards that end we advance in this paper a recommender system that searches for TV programs based on their likes/dislikes through implicit personalization techniques.

Introduction

Today, most consumers face an exhausting task of having to find something to watch on TV that fits their interests. Currently, the main modes for searching and identification of relevant TV content are the following: (a) browsing through pages of the 2D grid-format paper TV guide, (b) waiting on the PreVue channel, which is an automated scrolling version of the paper TV-guide grid and (c) using the rudimentary search tools that are available with a satellite dish TV programming service. The specific problems associated with them include the inability of the viewers to weed out irrelevant content, non-interactivity and difficulty in navigation. An alternative way is to provide viewers with a personalized means to provide intuitive user interfaces and the ability to filter program information. We propose a combination of the following two approaches to assist the users:

Search Engines & Information Visualization - Provide them with easy-to-use tools for search and present the abstract information in a way that is intuitive and easy to comprehend.

Recommender Systems - Provide them with a system that tracks and recognizes their preferences and organizes the TV program content accordingly.

Background

Researchers in information visualization have taken several approaches to visualize multidimensional data in various application contexts [Card *et al*., 1999]: finding movies and homes, news articles in a database, and web pages on the Internet. The specific problems associated with them include the use of scatter plots which are not particularly helpful in visualizing text-based TV program data, due to occlusion problems and the use of immersive visualization techniques leading to slower response time of the system, and loss of user orientation. The literature is rich with descriptions of other visualization systems for scientific and data mining applications; however, researchers have not focussed their attention on the consumer domain, which is what our work addresses.

The research in adaptive systems aims to build recommender systems that help the user in filtering information based on the user's profile. Several such systems have been built in recent years to help users deal with various sources of information [Etzioni, 1999]. However, these systems have a major source of information missing, the TV. The TV Advisor by Das and Horst [Das and Horst, 1998] is one example of a recommender system for TV found in the literature. They make use of explicit techniques to generate recommendations for a TV viewer. Such techniques require the user to take the initiative and explicitly specify their interests, in order to get high quality recommendations. Implicit techniques, on the other hand, provide a non-intrusive approach to lessen the burden on the user by inferring the user's preferences from the use of a TV set.

Application Prototype

Our application prototype aims to be a 'smart electronic program guide (smart EPG)' that enables a user to search and browse through a TV programs database. It is 'smart' because it maintains an adaptive user profile and makes recommendations of TV programs, computed according to the profile. The application can be divided into 3 parts as described below.

The search environment provides the tools for the user to formulate a search for retrieval from a TV-programs database. We organize the valid search criteria along ‘strings’ or ‘bracelets’, which represent individual dimensions of the multi-dimensional TV programs database. Each ‘bracelet’ is a grouping of complementary ‘beads’ where each ‘bead’ is a visual representation of information contained in the database. In our prototype, we have 7 bracelet categories: day of the week, time of the day, program genres, channels, keywords, user profile names and saved searches. This notion of beads evolves from ancient prayer beads or the abacus, where the beads served as information holding units and were useful for counting. The navigation and selection of search criteria takes place through a standard remote control.

The overview environment is concerned with the visual representation of the search results. The results are TV shows matching the search criteria, retrieved from the database. Our approach to visualization was to map this abstract information in a manner that parallels the human tendency to put physical objects closer when they are important and to let them fade into the background as their importance decreases. We use depth as a cue to achieve this notion. A tunnel model was used for displaying the results comprising of rings, each of which serves as a placeholder for the recommended TV shows. The TV-shows that are highly recommended for a user will be displayed closer to the user, on the first few rings, than those that are not.

In the current version of our prototype, user profiles can be used as search criteria to generate system recommendations of TV content. The search results are visualized, in the overview environment, in the order of high to low relevancy based on a desirability score that is computed according to the person’s profile. The scores are the output of either of two recommender engines that we have built as part of our research. Both of our approaches to generating TV program recommendations are based on implicit profiles of TV viewers. At the current time, we pursued approaches that could deal with incremental updates of the viewing history, and inconsistencies in the indexing of shows in the TV show data. An implicit profile is built from the viewing history of a TV viewer. The viewing history is a list of shows that a viewer has watched (positive examples) and not watched (negative examples). The implicit nature of our profiling method stems from the fact that the process does not involve any explicit interaction with TV viewers, regarding their likes and dislikes, other than collecting information about what shows have been watched.

The first of our TV program recommender systems uses the Bayesian classifier [Billsus and Pazzani, 1996] approach to compute the likelihood that the viewer will like or dislike a particular TV program. We approach the problem with a 2-class Bayesian decision model, where a show belongs to either the class, watched, or the class, not watched. The user profile, in the Bayesian context, is a collection of attributes (or features) together with a count of how many times an attribute occurs in positive and negative examples. From this profile, we first compute the prior probability that a show belongs to a particular class and then the conditional probability that a given feature will be present if a show is in either of the two classes. Using these probabilities we finally compute the a posteriori probability for a new show, given its feature set, that it belongs to a particular class.

The second approach construct rules for classifying shows given a *training set* of positive and negative shows that are part of the TV viewing history. We begin by deriving a decision tree (DT) which is then decomposed into rules for classifying the shows. The decision tree employed is Quinlan's C4.5 [Quinlan, 1993] that uses an information-theoretic approach based on entropy. C4.5 builds the decision tree using a top-down, divide-and-conquer approach: it first selects an attribute, then divides the training set into subsets characterized by the possible values of the attribute, and follows the same procedure recursively with each subset until no subset contains objects from more than one class. The single-class subsets correspond to the leaves of the decision tree, while a node indicates that a further test needs to be performed on that show to determine which class the show belongs to. When a new show, which is not part of the training set, is encountered, the DT is parsed to obtain a probabilistic class distribution for the show and the class with the highest probability is the predicted class.

Future Work

We have tested our application with potential users in order to get an appraisal of our visualization and personalization techniques. We got very encouraging endorsements of our concepts. Users also gave helpful suggestions for improvement, which are under consideration.

References

[Billsus and Pazzani, 1996] D. Billsus and M. Pazzani. Revising User Profiles: The Search for Interesting Web Sites. *Proceedings of 3rd International Workshop on Multistrategy Learning*. AAAI Press, 1996.

[Card *et al*., 1999] Stuart Card, Jock Mackinlay, and Ben Shneiderman. *Readings in Information Visualization.* Morgan Kaufmann Publishers Inc., San Francisco, California, 1999.

[Das and Horst, 1998] Duco Das and Herman ter Horst. Recommender Systems for TV. *Proceedings of 15th AAAI Conference*, Madison, Wisconsin, July 1998.

[Etzioni, 1999] Oren Etzioni, Jorg Muller, Jeffrey Bradshaw. Proceedings of the 3rd Annual Conference on Autonomous Agents, ACM Press, 1999.

[Quinlan, 1993] Ross J. Quinlan. C4.5*: Programs for Machine Learning*. Morgan kaufmann Publishers Inc., San Francisco, California, 1993.