Chapter 5

TV PERSONALIZATION SYSTEM

Design of a TV Show Recommender Engine and Interface

John Zimmerman[1*], Kaushal Kurapati[2*], Anna L. Buczak[3*], Dave Schaffer[4], Srinivas Gutta[4], and Jacquelyn Martino[5*]

[1] *Carnegie Mellon University*
[2] *IBM Corporation*
[3] *Lockheed Martin Corporation*
[4] *Philips Research*
[5] *MIT*
* *Work done while at Philips Research*

Abstract: The arrival of PVRs (Personal Video Recorders)—tape less devices that allow for easy navigation and storage of TV content—and the availability of hundreds of TV channels in US homes have made the task of finding something good to watch increasingly difficult. In order to ease this content selection overload problem, we pursued three related research themes. First, we developed a recommender engine that tracks users' TV-preferences and delivers accurate content recommendations. Second, we designed a user interface that allows easy navigation of selections and easily affords inputs required by the recommender engine. Third, we explored the importance of gaining users' trust in the recommender by automatically generating explanations for content recommendations. In evaluation with users, our smart interface came out on top beating TiVo's interface and TV Guide Magazine, in terms of usability, fun, and quick access to TV shows of interest. Further, our approach of combining multiple recommender ratings—resulting from various machine-learning methods—using neural networks has produced very accurate content recommendations.

Key words: Personalization, TV recommender, Interactive TV, User Interface, TV Interface, Electronic Program Guide (EPG), Trust

1. INTRODUCTION

The increase in TV viewing options from digital cable and digital satellite has created a world where many US homes have access to hundreds of channels. In addition, the arrival of Personal Video Recorders (PVRs) such as TiVo™ and ReplayTV™ has begun to change how people watch TV. PVRs allow easy navigation of TV-show schedules through electronic program guides (EPG) for selection and storage on hard disks. In observing PVR users, we noticed that within two to three days they shifted from watching live to stored TV programs almost exclusively. So now, instead of having to select a single program to watch from hundreds of channels, PVR users instead select a small set of TV shows to store from the tens of thousands broadcast each week.

In order to address this looming content selection overload issue and improve the TV show selection and viewing experience, we created a TV show recommender to help users find something "good" to watch. During its creation we pursued three related research themes. First, we developed a recommender engine that could track users' TV-preferences and deliver accurate content recommendations. An accurate recommender can extract shows users want to see from an ocean of less interesting fare. Second, we designed a user interface that allows easy navigation of selections and supports inputs required of the recommender engine. A recommender without an easy to use interface can potentially make finding something *good* to watch more difficult than not using a recommender. Finally, we explored the importance of gaining users' trust in the recommender, because if users don't trust a recommender, they will not use it or pay for it as a feature or service.

Our recommender engine combines two user information streams: explicit and implicit. The explicit stream allows users to take control by specifying their preferences and enables the system to begin working right away. The implicit stream unobtrusively observes users to learn their preferences. Implicit methods reduce the amount of work required of users to get recommendations and allow for changes in taste to be captured over time. We obtain accurate recommendations by fusing both recommenders' outputs using neural networks. Our interface runs on a touch-screen remote control. The use of a finger as a floating cursor greatly improves efficiency over traditional remote control interfaces that employ jumping highlights. In addition, our interface allows users to select how much or how little interaction they want with the recommender, making them feel *in control*. Finally, we build user trust by providing (i) accurate recommendations, (ii) flexible access to the recommender, and (iii) appropriate feedback on why users might want to try a *new* TV show.

Our research on both the engine and interface follows a user-centered iterative design process: analyze, design, and evaluate (repeat). At this point our work is still in the preliminary stages. We have focused on a longer-term evaluation with a small set of test subjects instead of a larger, statistically significant study.

2. RELATED WORK

2.1 Related Work on TV Recommender Systems

With respect to recommendation engines, the TV Advisor of Das & Horst (1998), PTV system of Cotter & Smyth (2000), EPG work of Ardissono et al. (in this volume), and our prior work Gutta et al. (2000), and Kurapati et al. (2001) stand out as some of the earliest TV recommender systems. In addition, TiVo and Predictive Networks offer commercially available systems that provide TV recommendations. For a history of the evolution of personalized electronic program guides, refer to Smyth & Cotter (in this volume).

The TV Advisor of Das & Horst (1998) employs explicit techniques to generate recommendations for a TV viewer. Such techniques require individual users to take the initiative and explicitly specify their interests in order to get high quality recommendations. Although this method is a good first step and one that is easy to implement in a set-top box, it burdens users who want minimal interaction with the recommender. Moreover the system is static and does not allow evolution of user profiles over time.

Cotter & Smyth's PTV (2000) uses a mixture of case-based reasoning and collaborative filtering as a means of learning users' preferences in order to generate recommendations. Initially users state their preferences about channel, genre, and viewing time while registering with the system, similar to the explicit recommender of Das & Horst. However, the heart of the PTV recommender infers users' preferences as they enter their feedback on TV shows they have watched. Also, PTV is Internet based, requiring the users to log on to a web site in order to see their recommendations. This approach removes users from the TV viewing environment thereby questioning the system's ability to be useful in the real setting.

Case-based recommendation relies on input of programs that users liked in the past (Balabanovic & Shoham 1997, Hammond et al. 1996, Smyth & Cotter 1999). Collaborative filtering (Balabanovic & Shoham 1997, Billsus & Pazzani 1998) recommends TV shows that other users, having characteristics similar to a given user's profile, liked. The advantage of collaborative filtering is that it does not need content descriptions (titles are sufficient). However, collaborative filtering does not completely protect users' privacy since information about a user’s likes and dislikes is used to make recommendations to other users. Additionally, collaborative filtering cannot work for programs that are completely new, not known to at least one of the viewers. This often happens with TV when new programs and made-for-TV movies are broadcast.

Ardissono et al. (in this volume) created the Personalized EPG that employs an agent-based system designed for set-top box operation. Three user modeling modules collaborate in preparing the final recommendations: Explicit Preferences Expert, Stereotypical Expert, and Dynamic Expert. The Explicit Preferences Expert deals with preferences declared by the users during initial setup. The Stereotypical Expert exploits users’ personal data known to the system and the explicit preferences stated in order to classify individual users into one of the lifestyle (stereotypical) groups. The Dynamic Expert analyzes users’ watching behaviors, and based on it builds and adapts its model of the user. The Personalized EPG recommender is a very interesting mixture of explicit, stereotypical and implicit user preferences allowing the final recommendation to take advantage of the three complementary methodologies involved. This work further validates hybrid approaches to generating content recommendations.

One of the earliest commercially available TV recommenders comes from TiVo, Inc. TiVos generate personalized recommendations that are displayed to users. Their recommender learns by tracking which programs users choose to record and user feedback of “thumbs-up” or “thumbs-down” to indicate how they feel about TV shows (on a 1 to 7 scale).

Predictive Media, Inc. provides another commercially available recommender system. They use a mixed model approach that combines statistical analysis, expert systems, and neural networks to generate content recommendations. Their system is primarily implicit in nature. Predictive Media’s recommender results can be plugged into any UI for presentation. This decoupling lessens the usefulness of a TV recommender in our opinion.

All of the techniques described above share one common characteristic: users only see a very limited number of recommendations. A unique feature of our system is that it creates a prioritized list of *all* TV shows. This allows users to browse both highly rated and lowly rated programs, and to actively tune the recommender and improve its performance.

2.2 Related Work on Interface Design

With respect to TV recommender interfaces, Double Agents by Meuleman et al. (1998), Personalized Contents Guide by Lee et al. (2001), TV Scout by Baudisch & Brueckner (2001), Time-pillars by Pittarello (in this volume), the interface explorations of Barneveld and Setten (in this volume), and the commercial PVR by TiVo all stand out.

Double Agents, designed by Meuleman et al. (1998) employs animated characters to help users find programs of interest. Each character represents a different genre. When a highly recommended program is being broadcast, the character representing the genre of this program changes its posture using human-like behavioral characteristics to let users know something *good* is on. This model can work well when working with the relatively small number of shows being broadcast live, however, unlike our interface it would have a problem with the one to two weeks worth of shows that PVRs present.

The Advanced Contents Guide of Lee et al. (2001) offers an interface where users can select from recommended TV shows. This interface uses user inputs such as selection, fast-forward, and rewind to learn user preferences, but unlike our system it offers users no direct input to the recommender. The interface presents a list of recommended programs by genre. However, the interface at present has only been used with data from four channels based on two weeks of data. It would be interesting to see if it can support hundreds of channels.

The TV Scout interface by Baudisch & Bruekner (2001) offers a fully featured TV show recommender interface that progressively reveals its filtering features to users over time. Users produce queries to find shows of interest, and the system learns what they like by tracking their queries. Unlike our interface, TV Scout only allows users to organize programs by time. However, this limitation may be only an artifact of a limited implementation as opposed to a limitation of the actual interface design. A more difficult problem is that the interface runs on a PC and has a very strong PC look and feel.

Time-pillars by Pittarello (in this volume) provides users with a 3D world populated by pillars that appear similar to architectural wonder, *Traian's column*, found in Rome. Each pillar represents a TV channel, with individual shows ordered by broadcast time spiraling up from the bottom to the top of the pillar. The Time-pillars interface seems most concerned with generating a pleasurable user experience where as our interface focus more on balancing a pleasurable user experience that also reduces the amount of time spent searching for something interesting to watch.

The interface explorations by Barneveld and Setten (in this volume) offer great insights into users' expectations. The conducted user research through participatory design and surveys to help clarify both which features users want and what those features might look like. They specifically looked at Predictions (how recommendations appear on the screen), Feedback (how users explicitly enter preferences to the recommender), and Explanations (how recommendations are justified to users). Their designs for feedback do not support the flexible interaction with the recommender our participants demanded (Section 5.2); however, their evaluation of explanations completely supports our design of the reflective history interface element (Section 7.1).

TiVo currently offers a TV show recommender interface as a commercial product in the United States and in England. Users can enter up to three thumbs up or thumbs down to indicate how much they like or dislike a TV program. To see what programs have been recommended, users have two options. First, they can check the hard disk contents and view programs TiVo has recorded for them that they did not explicitly request. Second, users can navigate to a list of highly recommended programs coming up in the next two weeks. This list never exceeds 100 program titles.

TiVo's recommender interface has two limitations. First, the interface does not allow users to access the recommender when filtering programs by genre such as Action-Movies, or by time such as *show me the recommended programs that are on now*. Second, the interface does not offer users any indication of why a program has been recommended. While there is much to complain about the limitations of TiVo's recommender interface, it must be noted that the interface is quite easy to use, and it is available on store shelves today.

2.3 Related Work on Trust

Related work with respect to trust literature includes Herlocker et al.'s (2000) visualization of collaborative filtering recommendation; Lerch & Prietula's (1989) work on trust of machine advice; Fogg & Tseng's (1999) elaboration on the elements of computer credibility; Cassel & Bickmore's (2001) exploration of trust of virtual characters that participate in small talk; and Wheeless & Grotz's (1977) research on the use of self disclosure in developing trust.

Herlocker et al. (2000) built on the theory that providing explanations to collaborative filtering recommenders increases users acceptance. They prototyped and user tested several visualizations explaining how collaborative filtering works. They found that bar charts showing nearest neighbors were very effective in communicating how a recommender system works and in offering confidence in the recommendation.

Looking more abstractly at the relationship between computers and people, Lerch & Prietula (1989) conducted a study to measure the difference between people's trust of machines and their trust of people. They found that people trusted advice from expert computer systems about as much as advice from novice humans. Expert humans were the most trusted. They also noted that bad advice given early in the use of a system had a strong, adverse effect on trust.

Fogg & Tseng (1999) synthesized the research done on computer credibility. They noticed that in the past people thought of computers as “virtually infallible” but that people’s trust in machines had greatly eroded. They proposed new conceptual frameworks for and offered methods for evaluating computer credibility.

Bickmore & Cassel (2001) explored the use of conversational agents to help people trust computers. They created a virtual reality realtor who would engage people in small talk before trying to sell a house. They discovered that small talk was effective in increasing the trust of extroverted people but that it had almost no effect on introverts.

Wheeless & Grotz (1977) researched the relationship between trust and self-disclosure. They created a method for measuring individualized trust and compared it to many different kinds of self-disclosure. Their results indicate that both intentional disclosure and increased amounts of disclosure create a higher level of trust.

3. RECOMMENDER ENGINE

Our recommender engine contains several components (Figure 5-1). We currently use both Bayesian and Decision Tree (DT) methods to produce implicit recommendations. Test subjects keep paper diaries detailing their viewing histories and based on these implicit recommendations are calculated. The explicit recommender allows users to directly input their preferences using two different interfaces (see Section 5.2). When we feed meta-data describing upcoming shows into the system, each of the different recommenders generates a rating for each show. An artificial neural network fuses the outputs of the different recommenders into a single set of improved recommendations.

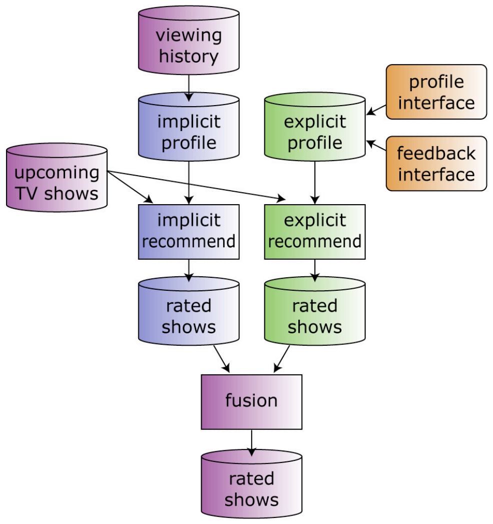


Figure *5-1*. Recommender system architecture.

3.1 Metadata

Our recommender engine relies on metadata from Tribune Media Services. Everyday we downloaded an updated TV-schedule for the next 12 days with channels (85+) and times associated with the Briarcliff Manor, New York area. Currently we have approximately four years of data, enabling us to re-run our recommenders on historic data as needed.

The metadata describes each TV-show using 205 fields, many of which are often not populated or relevant. We selected 21 fields for the implicit recommender, including: title, broadcaster, air time, air date, unique show key (indicating format such as series, movie, etc.), genres (comedy, action, adventure, news, etc.), actors, producers, directors, writers, hosts, synopsis, year, language, and content rating (mature, children, etc.). A subset of 14 fields from the 21 inputs to the implicit recommender is shown in Table 5-1.

Table *5-1*. Metadata sample

Field	Value
<Program Code>	EP1151270151
<Title>	Friends
<Short Title>	Friends
<Episode Title>	The One With the Wedding
<Synopsis>	Rachel serves as a bridesmaid in her ex-fiancé Barry's wedding.
<Genres>	Situation, Comedy
<Channel>	WPIX (NBC Affiliate)
<Air Time>	2000

<Air Date>	20020912 (Sep. 12, 2002)
<Actors>	Jennifer Aniston, Courteney Cox, Lisa Kudrow, Matt LeBlanc, Matthew Perry, David Schwimmer
<Producers>	Marta Kauffman
<Directors>	
<Writers>	
<Language>	English

3.2 Implicit Recommenders

The implicit recommenders generate profiles based on users' viewing histories. The implicit nature stems from the fact that users need only watch TV. We selected our two implicit methods, one based on Bayesian statistics (Kurapati et al. 2001) and one on Decision Trees (Gutta et al. 2000), for three main reasons. First, both methods work with relatively noisy data. We knew that users would occasionally watch shows they do not like and would sometimes not watch show they do like. Second, both methods do not require sharing histories with a central server. In conducting evaluations of the implicit recommenders and of sample user interfaces, all users liked the fact that our system protected their privacy. Third, both methods can run in a processor and memory-constrained environment such as a set-top box or TV.

3.2.1 The Viewing History

We assume that in a final product, such as a set-top box or TV, viewing histories can automatically be collected. For our study, however, we chose to use paper diaries for reasons of both cost and accuracy. First, it is very expensive to build custom set-top boxes for a small set of users. In addition, requiring users to use a device different from their current TV viewing setup seemed disruptive to their normal TV viewing behavior. We also considered a small device to collect all RF commands (Radio Frequency waves are a means of wireless communication—in this case between a remote control and a TV set) but the problem with this is that a single missed command would be impossible to find and would render any collected data meaningless. Therefore, we asked participants in our study to record which programs they watched each day into a paper journal, and we collected these journals each month. The two classes of TV shows of interest include: *C1* (shows that interest the viewer) and *C2* (shows that do not interest the viewer). However, practically we obtain information only on the classes: C+ (shows the viewer watched) and C- (shows the viewer did not watch), which are approximations of classes C1 and C2.

A major element of uncertainty lies in determining what is 'not-watched' by users. For each watched show—which can be ascertained clearly—there are numerous ones ‘not-watched’. The key question is how to sample the universe of shows to appropriately populate the ‘not-watched’ class, C-. We recognize that users could be asked to provide information on shows they do not like, thus gathering information directly on shows belonging to class, C2 (shows disliked by the user). However, when we consider the pure implicit method, we have to rely on a sampling technique to populate class C-. To approximate class C-, one option is to select a not-watched show broadcast in the same time-slot as the watched show. An inference that the watched show is preferable to the not-watched shows seems reasonable in this case. Unfortunately, this scheme prevents learning preferred viewing times.

We chose instead to sample one not-watched show for each watched show from the previous week of shows, randomly selecting the day, time, and channel. This helped preferred viewing times to become statistically discernable; however, the issue of how many not-watched shows to sample is one that could profit from more research. We chose a one-to-one sampling ratio because it makes the Bayesian a priori probability of each class equal. Since we sample uniformly in time, the not-watched class represents a sample of the general run of shows being broadcast and therefore the implicit recommenders should find patterns of features that distinguish what is liked from this general run of material.

For three years we collected viewing histories on eight participants. However, because participants relocated during the study we do not have a full three years for each individual. Participants were recruited from the community around Briarcliff Manor, New York. They ranged in age from late teens to early sixties and were of mixed gender. None of the participants worked for Philips Research. Because of the small number of subjects, the results presented in this paper are preliminary in nature.

3.2.2 Bayesian Recommender

Our Bayesian implicit recommender uses the Bayesian classifier (Billus & Pazzani 1998, Duda & Hart 1973) approach to compute the likelihood that the user will like or dislike a particular TV program. We approached the problem with a 2-class Bayesian decision model, where a show either belongs to the class “watched”, or the class “not watched”. Ideally, we would like to have ground truth information on whether a TV-show was *liked* or *not liked* by the user, but the implicit profiling technique allows us to have information only on the classes watched and not watched. The user profile, in the Bayesian context, is a collection of features together with a count of how many times each occurs in the positive and negative examples (i.e. classes C+ and C-). An excerpt from a user profile is shown in Table 5-

2. Note that a feature is an attribute-value pair and that only a small sample has been shown.

Table 5-2. User profile excerpt

Feature	Watched (C+)	not-watched (C-)
Total_programs	66	66
Station: ABC	10	56
Station: BBC	20	10
Title: BBC News	10	2
Actor: Jim Carrie	2	0
Genre: comedy	1	7
Keyword: murder	5	1

Our Bayesian model computes the prior probabilities directly from the counts in the viewer profile. Then we compute the conditional probabilities that a given feature, f_i, will be present if a show is in class $C+$ or $C-$: $P(f_i|C+) = k(f_i|C+)/k(C+)$, where $k(f_i|C+)$ represents the number of shows in which feature f_i is present given that the show belongs to class watched (C+) and $k(C+)$ represents the count of all shows belonging to class watched (C+). The recommendation scores for upcoming shows are computed by estimating the aposteriori probabilities, i.e. the probability that a show is in class $C+$ and $C-$ given its features: $P(C+|\mathbf{x})$, where $\mathbf{x}$ is the feature vector. For details on the computation, we refer to Kurapati et al. (2001)

3.2.3 Decision Tree Recommender

Like the Bayesian recommender, our Decision Tree (DT) recommender uses the "watched" and "not watched" shows. The decision tree employs Quinlan's C4.5 (Quinlan 1983, 1991) and uses an information-theoretical approach based on entropy. C4.5 builds the decision tree using a top-down, divide-and-conquer approach: select an attribute, divide the training set into subsets characterized by the possible values of the attribute, and follow the same procedure recursively with each subset until no subset contains objects from more than one class. The single-class subsets correspond to leaves of the decision tree.

The DT has nodes and leaves where nodes correspond to some test to be performed and leaves correspond to the two classes: "watched" and "not watched". Testing an unknown show involves parsing the tree to determine which class the unknown show belongs to. When computing a recommendation for each TV show, a probability is computed that the show will be in the class "watched" and the class "not watched" based on the shows it correctly classified during training. However, if at a particular decision node we encounter an unknown attribute value, so that the outcome of the test cannot be determined, the system then explores all possible outcomes and combines the resulting classifications: the class with the highest probability is assigned as the predicted class.

As decision trees become large, with increasing viewing histories, they become difficult to understand because each node in the tree has a specific context established by the outcomes of tests at antecedent nodes. We then rewrite the tree as a collection of rules, which is usually more accurate than a decision tree (Gutta et al. 2000).

3.3 Explicit Recommender

Based on participants' request for interaction with the recommender and our need for a recommender that could work quickly "out of the box", we developed an explicit recommender (Kurapati et al. 2001) that relies on user inputs. By default our system assigns a neutral 4 for all channels, genres, days, and times of day. Individual users can then indicate their preference on a 1 to 7 scale (1 for "hate the feature", 7 for "love the feature") for as many or as few features as they prefer. We compute an explicit recommender score as follows:

$$E = wDayT * rDayT + wCh * rCh + (1/K) * \sum wGenre_i * rGenre_i \qquad (1)$$

where *rDayT* is the profile rating for the feature daytime, *rCh* - profile rating for channel, $rGenre_i$ - profile rating for i-th genre describing a given show (from *K* genres describing it; in Tribune Media Services metadata up to six genres describe a show) and the corresponding heuristically determined weights are: *wDayT* = 0.1, *wCh* = 0.2 and $wGenre_i$ = 0.7 (for each genre). Users have two methods of inputting their preferences as described in Section 5.2.

4. INCREASING ACCURACY THROUGH FUSION

Results of testing the explicit and two implicit recommenders with real users (Kurapati et al. 2001) suggested reasonable recommender accuracy. However different recommenders seemed to perform well for different users with no easy way to pre-match individual recommenders to individual users. Further analysis revealed that different recommenders performed well for very different sets of shows. As a follow-up we attempted to improve accuracy by *fusing* (combining) recommender outputs using a neural network. We chose to use a neural network because it might detect correlations that simple heuristics cannot. We believe that such an approach improves the robustness of our system.

The fusion system combines the following individual recommenders:

1. Implicit Bayesian based on individual view history
2. Implicit Bayesian based on household view history
3. Implicit Decision Tree based on individual view history
4. Implicit Decision Tree based on household view history
5. Explicit

The individual and household view histories were used separately in order to determine whether a TV recommender could just do with one profile per box in a household or if we needed to make fine grain distinctions between individual household members.

We developed two approaches to fusion. The first method uses a single Radial Basis Function (RBF) neural network for all the users. We trained this network on partial data from a subset of our eight viewing history participants. This approach has the advantage that it can be developed using ground truth data from a small set of participants to improve accuracy for a large set of users in a deployed product. The second approach uses a custom network for each user. This neural network was trained on each of our viewing history participants. This approach ensures that the network is responsive to a particular user's characteristics. For example, certain participants obtained better recommendations using an individual profile while others did better using a household profile. Also the importance of explicit recommenders varied from participant to participant. A challenge this second approach faces is that it requires ground truth data from all users. This is not likely in a final product as we do not expect users to be willing to inform the system if they actually liked each show they watched and did not want to see each show the system selects as "not-watched".

4.1 Radial Basis Function Neural Networks

We chose Artificial Neural Networks because of their ability to recognize patterns in the presence of noise and from sparse and/or incomplete data. They perform matching in high-dimensional spaces, effectively interpolating and extrapolating from learned data. We chose Radial Basis Function (RBF) networks (Moody & Darken 1989) because they are universal approximators and train rapidly; usually orders of magnitude faster than back propagation. Their rapid training makes them suitable for applications where on-line, incremental learning is desired such as a set-top box observing TV viewing. The ways we used RBF networks are explained in Sections 4.3 and 4.4.

4.2 Evaluation

In order to generate a set of ground truth data, we asked our eight viewing history participants to rate a set of approximately 300 TV shows. At the time of this evaluation one of our participants had recently relocated, so we have data on seven people, referred to as A, C, D, F, G, H and I. Using Likert scales—an attitude measurement method that allows users to rate on a discreet scale of “strongly agree” to “strongly disagree”—participants rated each show as “would watch” (1), “may watch” (0.5) and “wouldn’t watch” (0). In addition, participants could rate a show as “do not know” (DNK) when they were not familiar with its title. Only the shows known to the user were utilized (there were 1348 such shows).

We compared fusion results using three metrics: Hit Rate (*HR*), False Positive Rate (*FPR*), and Mean Squared Error (*MSE*). Hit Rate and False Positive Rate were computed for only those shows that were crisply classified by the user either as 0 (wouldn’t watch) or 1 (would watch). For the shows classified as 0.5 (may watch) it is disputable whether they should be recommended or not. For computing Hit Rate and False Positive Rate, a threshold value needs to be chosen. A higher threshold value will lead to both lower Hit Rate and False Positive Rate, and a lower threshold will result in higher Hit Rate and False Positive Rate.

The main advantage of using the Mean Squared Error is that it can be computed for shows with any ground truth rating (yes-1, no-0, and may be-0.5). Its additional advantage is that it does not require a determination of a threshold value, which can be quite cumbersome. This metric suits our goal of producing a ‘degree of match’ for each show with a profile, rather than a hard yes or a no vote on a particular show. In the end, based on user evaluation, our recommender interface displays all shows, ordering them from most to least recommended.

4.3 Fusion Method 1: Single RBF Net for All Users

RBF networks with five inputs, one output and a varying number of hidden nodes were trained. We used scores from the five individual recommenders—listed at the beginning of Section 4—as inputs to the RBF network, which produces fused recommendations as the output. The higher the output value, the higher the recommender rating assigned to a show. We used 15% to 40% of data from subjects *A*, *C*, and *D* as a training set. This represents 26% of the whole data set. For the networks’ cross-validation, we used 14% to 45% of data from subjects *D*, *F*, and *G*. This represents 13% of the whole data set. The remaining data was employed for the end tests (testing recall). Data from users *H* and *I* were neither used in training nor in cross-validation as these users had no household values. The best performance of RBF networks in terms of Hit Rate (HR) and False Positive Rate (FPR) was obtained by cross-validation for a network with 15 hidden nodes. A threshold of 0.5 was used to compute HR and FPR.

Figure 5-2 depicts the hit rate for all the subjects. Hit rate obtained by fusion is on average 29% higher then the average of the hit rates from nine separate recommenders.

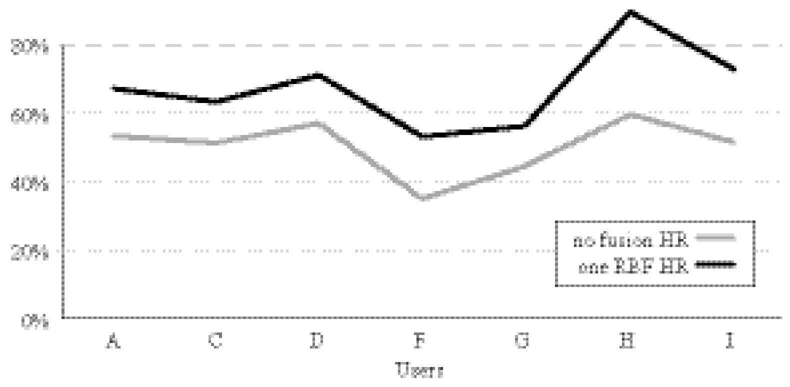


Figure *5-2*. Comparison of no fusion Hit Rate and one RBF net Hit Rate.

Figure 5-3 shows the FPR for each of the users. The lower the FPR, the better the final recommendation. FPRs are mixed, since for certain users a lower FPR is obtained by fusion, for others the average of nine recommenders gives an improved FPR. On average the fused FPR is higher by 5% than that obtained by the average of recommenders. False positive rates cannot however be compared by themselves without comparing hit rates since both are interrelated, and linked by the threshold value. By using a higher threshold value, FPR can be easily lowered; this of course causes hit rate to get lower as well. Therefore if for fusion the hit rate is better on average by 29%, and FPR is worse on average by 5%, still the combined fusion results are more advantageous than the average of recommenders.

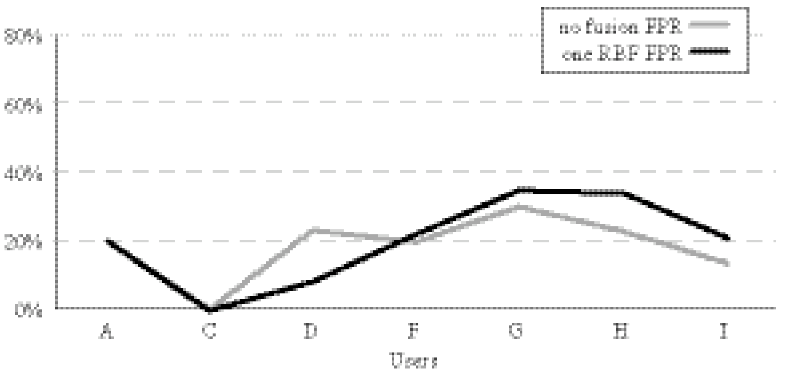


Figure *5-3*. Comparison of no fusion FPR and one RBF net FPR.

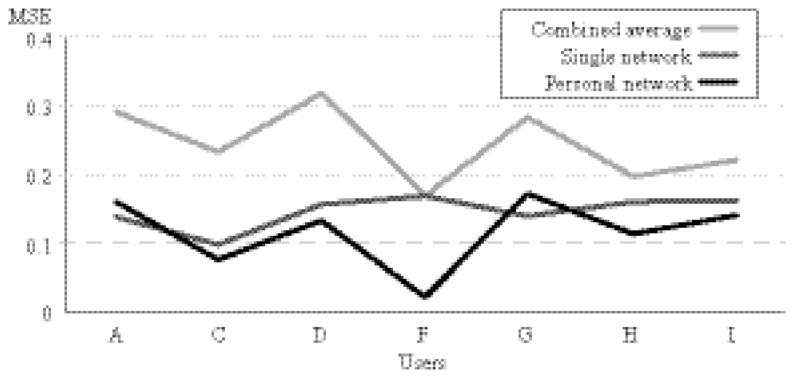


Figure *5-4*. Comparison of Mean Squared Error (MSE) of Different Methods.

The top curve on Figure 5-4 shows the average MSE for nine recommender combinations (all five recommenders plus each implicit method combined with the explicit recommender); the middle curve shows the single-network MSE. The single-net MSE varies for different users from 0.1 to 0.17 per user while the average MSE for nine recommender combinations are much higher (it varies from 0.17 to 0.32 per pattern). For all users, with the exception of user *F*, fused MSE performs better than the average of non-fused results. For user F, the average MSE of non-fused results is as good as the fused MSE. This user is however an outlier whose behavior is completely different from the other users. This is the only user for whom recommenders based on individual viewing history were superior to recommenders based on household history (Kurapati et al. 2001) and the only one for whom the Explicit Recommender is the least reliable. It is remarkable that for such an outlier the single network still works so well.

4.4 Fusion Method 2: Separate RBF Network per User

We also trained a separate RDF network for each individual participant. Approximately 40% of the data were used for training, 15% for cross-validation, and the rest for testing. The number of hidden units for each user's fusion network was determined by checking the performance of a set of trained RBF networks on the cross-validation set. The network that gave the best performance in terms of HR and FPR on the cross-validation set was the one chosen.

The overall performance of fusion can be characterized by MSE that is shown as the bottom curve in Figure 5-2. The separate networks give excellent results, consistently better than the average of recommendations: MSE of the fused results varies for different users from 0.02 to 0.17 (compared with 0.17 to 0.32 for the average of individual recommenders).

It is worth noting the superior performance of fusion for user F. Since training took place on the data from this outlier, RBF network weights have stabilized to better describe this user, resulting in the smallest MSE of all. This shows that separate networks work especially well for outliers who cannot be well described by a common (stereotype) network.

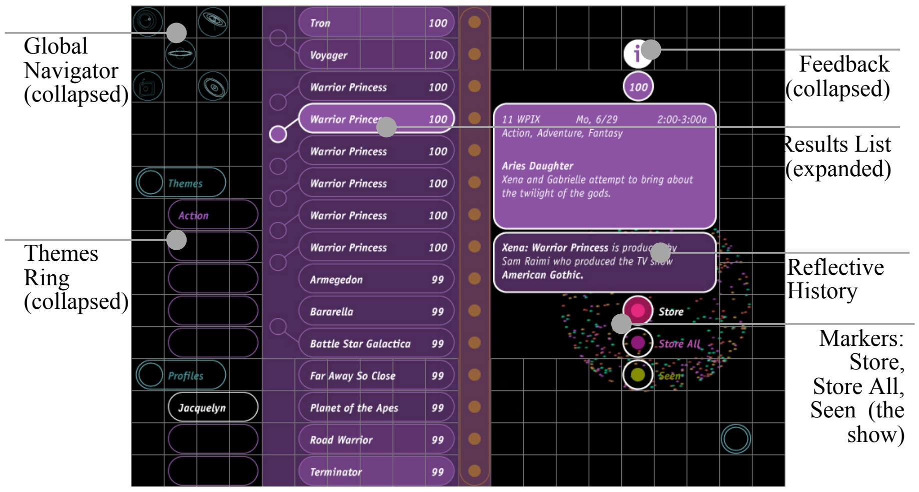
5. INTERFACE DESIGN

We began our design by reviewing participant feedback from evaluations of two previous interfaces. Three issues emerged. First, users complained that both our earlier designs and current commercial products required too many button clicks and navigation of too many screens to complete a single task. Second, users wanted more flexible interaction with the recommender. Some users wanted to have almost no interaction; some users wanted to take complete control; and some users wanted to have more limited interaction with the recommender. Third, users displayed a lack of trust with the recommender. When we recommended shows they regularly watched, they felt the recommender worked well. However, when we recommended a show they had never heard of, instead of thinking the recommender had found something new to watch, they assumed it was broken. The design of the Touch and Drag interface focused on addressing these three user interface issues.

5.1 Ease of Use

Traditional TV interfaces, including our two previous designs, force users to maneuver a jumping highlight across a series of screens making the completion of a single task quite time consuming. For example, if users want the fifth item on a list, they must press the cursor-down button four times and then press OK. We chose to address this problem by (i) placing our interface on a *touch-screen remote control* where users could employ their finger as a floating cursor, and (ii) designing a series of *expanding and collapsing interface elements* so individual tasks could be solved on a single screen.

We designed the screens to read from left to right (Figure 5-5). The far left contains a *Themes Ring* and below this the *Profile Ring*. By accessing these elements, users can indicate who they are and what kind of content they are searching/browsing for. Moving right, users encounter the *Results List* with a default sort by recommender rating. To the right of the *Results List* are *Details* for the highlighted TV show and tools (called *Markers*) for taking action such as storing or access to the feedback interface.



Tron	100
Voyager	100
Warrior Princess	100
Warrior Princess	100
Warrior Princess	100
Warrior Princess	100
Warrior Princess	100
Warrior Princess	100
Armegedon	99
Bararella	99
Battle Star Galactica	99
Far Away So Close	99
Planet of the Apes	99
Road Warrior	99
Terminator	99

11 WPIX Mo, 6/29 2:00-3:00a
Action, Adventure, Fantasy
Aries Daughter
Xena and Gabrielle attempt to bring about the twilight of the gods.

Xena: Warrior Princess is produced by Sam Raimi who produced the TV show American Gothic.

Figure 5-5. Interface screen with expanded *Results List.*

Figure 5-5 and Figure 5-6 with the expanded *Themes Ring* offer an example of the collapsing and expanding elements. When users touch the collapsed *Themes Ring* (Figure 5-5) it automatically expands to provide users access to this tool. At the same time the *Results List* collapses (Figure 5-4) to create more room on the screen. In addition, the details and other tools disappear as they only operate on the highlighted show, which is no longer visible.

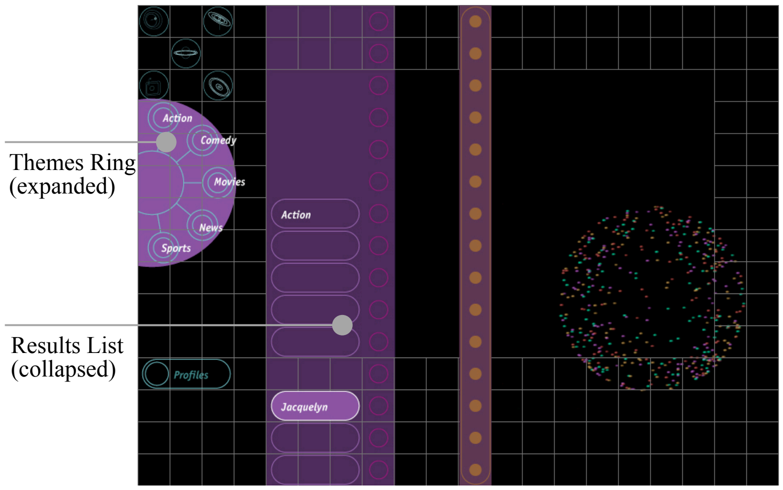


Figure *5-6*. Screen with expanded *Themes Ring.*

5.1.1 Touch, Drag, and Stroke

The design offers three interaction methods: “Touch”, “Drag”, and “Stroke”. Touch allows users to select different elements on the screen. For example, if users want to record a TV show, they can touch the *Marker* element labeled “Store” (Figure 5-5). To record multiple episodes of the same show, users can touch the *Marker* labeled “Store All.”

Figure 5-6 offers an example of the Drag interaction. In order to see all of the items on the *Themes Ring*, users drag their finger clockwise or counter clockwise, causing the ring to rotate. When they find a desired theme, they drag it onto the collapsed *Results List*. The Drag action not only reduces the number of steps needed to complete a task, it also creates a more naturalistic interaction. If users wish to see Action shows on the *Results List*, they drag the Action element there.

Many other recommenders address the problem of long lists of shows by hiding shows that fall below a threshold or by displaying only the top N number of shows. However, our view history participants were uncomfortable with this approach. They did not trust any recommender enough for it to eliminate possible selections. Therefore our interface presents all of the next weeks shows matching the *Themes Ring* selection, sorted from highest to lowest rated. This means users often must deal with very long lists. To address this problem we developed the Stroke interaction. When users want to rapidly scroll the list, they stroke their finger up or down on the list. When the finger is lifted, the system sets a velocity and then slowly decelerates. If users see an item they are interested in scroll by, they touch the list to stop it from moving. This idea works somewhat like the wheel of a bicycle that has been turned upside down. People can Stroke the wheel to make it spin. They can then either watch it decelerate or touch the wheel to stop it.

Use of expanding and collapsing elements combined with the use of a finger as a pointing device greatly reduces the number of screens users need to navigate and the number of steps (button clicks) needed to complete a single task. For example, consider the task of browsing for movies. Using the Touch and Drag interface, users navigate two screens (Global Navigator and Find TV) and perform three Touches and three Drags in order to produce a personalized list of movies. Using TiVo, users need to visit five screens and perform a minimum of fourteen button clicks in order to see a alphabetical listing of movies.

5.2 Flexible Recommender Interaction

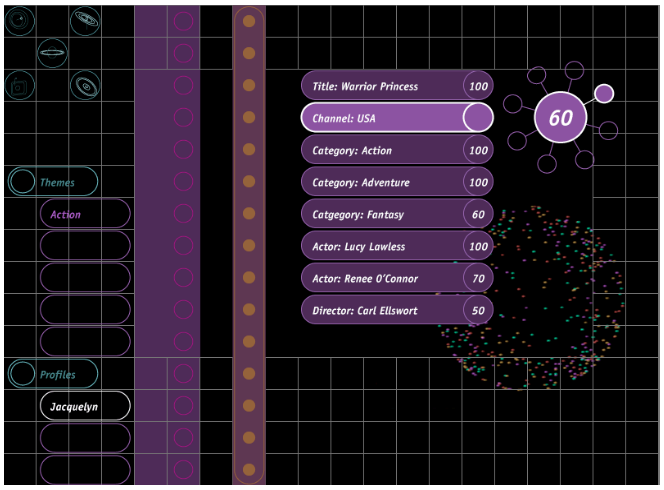
During evaluation of the earlier designs and of the implicit recommenders, some users expressed a desire to interact more directly with the system. This led in turn to our definition of three different user groups and two interfaces to communicate with the explicit recommender.

5.2.1 Do it For Me Users

The implicit recommender works well for the *Do it for me* users. This recommender monitors viewing history and then makes recommendations. All a user needs to do is watch TV. In addition, the Touch and Drag interface automatically displays upcoming programs sorted by recommender rating. Placing highly recommended programs at the top of the list reduces the number of shows users need to browse.

5.2.2 Let's Do it Together Users

The *Feedback* interface (Figure 5-7) supports the *Let's do it together* users and the *Let me drive* users. This interface allows users to access the part of their explicit profile that corresponds to the currently highlighted show. When users see a recommendation they disagree with or when they just want to better understand why a show has a certain rating, they can expand the *Feedback* interface and view or modify the ratings.

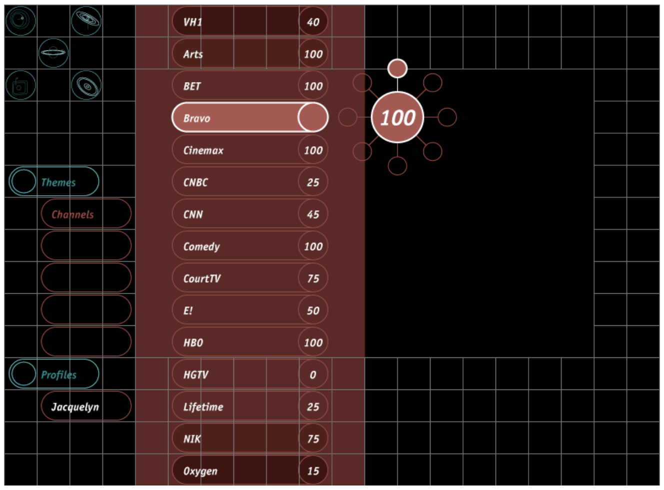


Title: Warrior Princess 100
Channel: USA
Category: Action 100
Category: Adventure 100
Catgegory: Fantasy 60
Actor: Lucy Lawless 100
Actor: Renee O'Connor 70
Director: Carl Ellswort 50

Figure 5-7. Feedback interface expanded.

5.2.3 Let Me Drive Users

Let me drive users can take more control by selecting Profiles on the *Global Navigator*. This transitions them to the Profiles interface (Figure 5-8). Here they can rate all program attributes on a 0 to 100 scale. The system initially gives all items a neutral rating of 50. Users can quickly look through the listed items and change as few or as many as they want.



VH1 40
Arts 100
BET 100
Bravo
Cinemax 100
CNBC 25
CNN 45
Comedy 100
CourtTV 75
E! 50
HBO 100
HGTV 0
Lifetime 25
NIK 75
Oxygen 15

Figure 5-8. Profiles interface.

The interface supports flexible access to the recommender without the need for users to select a novice or expert mode. Users that want little or no interaction are free to simply watch TV. Users that want some interaction can choose how much or how little control they want to take by using the *Feedback* interface and the Profiles screen.

6. TESTING EASE-OF-USE

After completing the Touch and Drag interface design, we performed a *proof of concept* test. Our questions included:

1. Does the use of collapsing and expanding elements improve ease of use over nested menu systems?
2. Does the use of a finger as a floating cursor improve ease-of-use over a jumping highlight and traditional four-cursor remote control?
3. How do users feel about a large touch-screen remote control?

For this evaluation we recruited six subjects (not the participants in our viewing history study), three men and three women, ranging in age from 21 to 60. They were all moderate to heavy TV viewers. Each participant rated items for the explicit recommender in order to generate individual recommendations.

We chose a task-based approach to the test, asking subjects to complete two common TV tasks using TV Guide Magazine™, Touch and Drag, and TiVo™. Task 1 asked users to find three upcoming movies they *wanted* to watch. Task 2 asked participants to find the next broadcast of the TV show *Roseanne*. Participants tested only one system at a time, starting first with TV Guide, then Touch and Drag, and finally TiVo. They were each allowed to play with one of the systems and then asked to complete both tasks. When all tasks had been completed on all systems, we asked subjects to answer qualitative questions about the three systems. One of the challenges of testing came from the recommender. TV guide magazine clearly has no recommender, and in addition, since TiVo does not employ an explicit recommender, we had no way of tuning this device to personalize the content for users.

Subjects found the Touch and Drag interface much easier than either TV Guide magazine or TiVo at finding a list of movies they wanted to watch. The reduced number of screens and clicks along with the personalized listing of movies significantly reduced the amount of time users spent with our system compared to TiVo. TiVo was also handicapped by the fact that the list of movies and individual descriptions of movies appear on separate screens, forcing users to constantly move back and forth between screens.

TV guide was most significantly handicapped by a lack of a single listing for all movies and offered no method of filtering for the channels an individual participant received.

Finding the next broadcast of *Roseanne* proved easiest on TiVo; however, Touch and Drag took only slightly longer. TiVo's on screen keyboard made title input quite easy. Our system suffered from lack of labeling on our Sort tool that allows users to resort the *Results List* by title instead of the default sort by rating. Once participants discovered this tool, they had no problems completing the task. TV Guide proved nearly impossible for this task. Only one of the six participants completed this, and he ended up looking at every page until he found *Roseanne*.

Participants rated twelve qualitative questions using 1 to 5 Lickert scales Responses show that users preferred Touch and Drag and particularly found it attractive and fun to use (Table 5-3). One participant, when playing with the *Stroke* interaction on the *Results List*, told the evaluator, "Don't show this to Vegas." He found the stroke action similar to slot machines and immediately associated it with fun.

Table *5-3*. Participant responses to qualitative questions

Question		**TV Guide**	**Touch & Drag**	**TiVo**
Q1	Frequency of use	3.33	4.33	3.67
Q2	Usefulness for finding shows of interest	2.83	4.33	4.00
Q3	Confidence in using system at home	4.17	3.42	3.42
Q4	Ease of finding shows of interest	3.33	4.58	3.67
Q5	Ease of use of system	3.33	4.17	3.83
Q6	Feeling in control of system	3.33	4.17	3.83
Q7	Adequate information structure	2.83	2.92	3.33
Q8	Intuitiveness of UI	3.08	3.67	3.25
Q9	Attractiveness of info layout	3.00	4.5	3.67
Q10	Fun to use	1.33	4.67	4.00
Q11	Information quantity	4.17	3.83	4.00
Q12	Ease of learning	3.50	4.00	4.25
Average Score		**3.19**	**4.05**	**3.74**

Participants offered differing views on the touch-screen versus remote control. Some liked the touch-screen and felt it would be the best system to use at home. However, others stated that they would rather see the information on the TV screen. They suggested that the Touch and Drag interface be adapted to work with a standard remote or even a remote control with a track pad similar to laptop computers.

The results of this evaluation offered insights into how we might reshape the design. However, it is important to note that this test only offered users a few minutes to use each system. For a more thorough evaluation, we would need a much more robust system we could place in users' homes.

7. IMPROVING TRUST

During evaluation of our earlier recommenders and interfaces we encountered a difficult problem. When our system recommended programs participants regularly watched, they thought it worked great. However, when the system recommended programs they did not know, participants felt the recommender was "broken". This created a serious problem. A recommender that only finds programs users know they like will have limited value. In order to create a recommender that provides real value it must help users find new shows to watch and present these new shows in a way that they might try them.

Experimental evidence has shown that providing explanations of how a recommender works can help to improve user acceptance of a recommender system (Herlocker et al 2000). The Touch and Drag interface allows users to glimpse how the recommender is working through the feedback interface (Figure 5-5) and the Profiles screen (Figure 5-6). However, these interfaces do not specifically address new shows.

Looking deeper into the value of new shows we were quite surprised to discover that people generally *do not want to be challenged* when selecting programs to watch. The majority of television viewing takes place in the evening (Kubey & Csikszentmihalyi 1990), at a time when people want to relax after a hard day. People often choose to watch television at this time because it makes them *feel* relaxed (Kubey & Csikszentmihalyi 1990). In addition, much of television watching is nonselective (Lull 1990). Viewers *coast* from one show to another, watching whatever program comes on.

We found a good analogy in food. People do not want to try new food at each meal. Often they want *comfort food*, particularly when they are stressed such as at the end of a workday. Menus offer people a chance to try something new by listing the main ingredients. When people try a new food, they have some expectation of what it will be like based on foods they have eaten in the past.

7.1 Reflective History

Based on the food analogy we developed the *Reflective History* (Figure 5-3), an interface element designed to motivate people to try a new TV show when they were in the mood. The element needed to be subtle, since the mood to try new things is rare. We wanted it to tempt the user as opposed to cramming new programs down their throats. This element appears when "new" and "highly rated shows" are highlighted in the results list.

We generate these conversational sentences by comparing users' viewing histories used by the implicit recommender with the metadata about upcoming TV shows. Our system looks for highly rated new programs (programs not already in a user's viewing history). Next, it searches for a common person between the new TV show and TV shows the user has already seen. When it finds an appropriate match, the system generates a conversational sentence using the following structure: <NewProgram> <NewTask> <Person> who <OldTask> <OldProgram>. Table 5-4 offers examples of the text strings for <Task>, <NewTask>, and <OldTask>.

Table 5-4. Text strings used in reflective history

Task	NewTask	OldTask
Director	is directed by	directed the TV show
Producer	is produced by	Produced the TV show
Writer	is written by	wrote
Actor	stars	plays <Character> in

Example: *Boston Public* stars Jeri Ryan who plays Seven of Nine in *Star Trek: Voyager*.

The reflective history sentence uses a conversational structure; making it sound like something one friend might say to another. This builds on Reeves and Nash's theory that people interact with computers as if they were people (Reeves & Nass 1999). The sentence reveals some of what the system knows about the user. This is a type of self-disclosure, which can help to build trust (Wheeless & Grotz 1977). The short, conversational sentence also works well with our interface design. Its size and location make it easy for users to ignore. However, its appearance and disappearance as new shows move into the highlight position on the results list make it easy to find for users who are in the mood to try something new.

8. CONCLUSIONS

The Touch and Drag personalization system integrates an accurate TV show recommender engine and an easy to use interface that supports the way people want to watch and store TV shows. In addition, our interface design offers appropriate insights into how the recommender engine works, helping people both refine and trust the recommendations they receive. Preliminary testing of the previous and current recommender engines and interfaces indicate that we are moving in the right direction.

Our recommender engine employs a robust design, combining both an explicit and implicit recommender. Our explicit recommender allows users to enter and modify their preferences to allow for both a system that can run out of the box, and to allow users to *feel in control*. The interface offers two methods for entering and modifying preferences and in addition, preloads all ratable items with a neutral value of fifty. The combination of the two interfaces and the pre-rated items allows the interface to seamlessly support users that want very little interaction with the recommender as well as users that want to really take control. This easy access encourages users to slowly tweak the explicit recommender until they get the results they want.

Our implicit recommender monitors all TV shows users watch, adjusting recommendations based on current and changing tastes. This recommender supports users that want *no* interaction with the recommender. By simply watching TV the system can learn their likes and dislikes. The viewing history also provides an opportunity for increasing user trust. By mining the viewing history, the user interface can subtly recommend *new* programs based on people that the programs have in common. Our conversational approach builds on the person-to-person model of self-disclosure to improve users trust.

Our touch-screen based interaction makes both browsing and profile management easy, efficient, and fun. The collapsing and expanding interface elements along with the use of a finger as a floating cursor help users to complete tasks on a single screen. The improvement in efficiency allows users to get back to doing what they like most: watching TV. The naturalistic style through the *Touch*, *Drag*, and *Stroke* interactions helps to clarify the relationships between the different interface elements and to make the interface as entertaining to use as watching an actual TV show.

Finally, we generate accurate recommendations by fusing the results of both the implicit and explicit recommenders using a radial basis neural network. We developed two RBF network-based approaches to fusing recommendations. The first method trains the network to be responsive to all users while the second one employs a separate RBF net for each user. Both methods achieve a beneficial combination of the strengths of individual recommenders.

The results of both approaches are very encouraging and superior to the average of recommendations. For most users separate networks perform the best because they are well adapted to a given user's characteristics. The most remarkable separate network performance is obtained for an outlier whose individual recommender characteristics were almost the opposite from the mainstream user. For this user a separate net gives much superior results to the single (stereotype) network. For the mainstream users the difference between separate and single nets is much less pronounced.

The power of the single-network method is that such a fusion network can be developed based solely on the data from subjects in our study, whose behavior conforms to the mainstream, and then deployed in the field where it can perform well for users it has not encountered earlier. Later it could be tuned to a specific user behavior by using feedback data collected by the system. We should however stress that the method developed is a proof-of-concept demonstration and testing on a larger user population is needed to obtain statistically significant results.

This increased accuracy works with the whole interface design to help improve user trust of the recommender. In addition, the improved accuracy helps bring the *best* TV shows for users to the top of long lists of shows, making it much faster and more interesting for users to find something they *want* to watch.

ACKNOWLEDGEMENTS

We would like to thank Jeanne de Bont, Henk Lamers, Guy Roberts, Lira Nikolovska, Lesh Parameswaran, John Milanski, and Jenny Weston of Philips Design for their assistance in design and evaluation of the interface. We would also like to thank Henk van der Weij of Big Camp in Eindhoven, the Netherlands, for his assistance with the implementation of the Touch and Drag interface.

REFERENCES

1. L. Ardissono, C. Gena, P. Torasso, F. Bellifemine, A. Chiarotto, A. Difino, B Negro: 'User Modeling and Recommendation Techniques for Personalized Electronic Program Guides'. This Volume.
2. B. Smyth and P. Cotter: 'The Evolution of the Personalized Electronic Program Guide'. This Volume.
3. M. Balabanovic and Y. Shoham: 1997, 'FAB: Content-Based Collaborative Recommender'. *Communications of the ACM* 40(3), pp. 66-72.
4. J. V. Barneveld and M. V. Setten: 'Designing Usable Interfaces for TV Recommender'. This Volume.
5. P. Baudisch and L. Bruekner: 2001, 'TV Scout: Guiding Users from Printed TV Program Guides to Personalized TV Recommendation'. *Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems: Workshop on Personalization in Future TV*, Malaga, Spain, pp. 151-160.
6. T. Bickmore and J. Cassell: 2001, 'Relational Agents: A Model and Implementation of Building User Trust'. *Conference on Human Factors in Computing*, Seattle, WA, USA, pp. 80-87.
7. D. Billsus and M. J. Pazzani: 1998, 'Learning Collaborative Information Filters', *Fifteenth International Conference on Machine Learning*, Wisconsin, USA, pp. 46-54.
8. P. Cotter and B. Smyth: 2000, 'PTV: Intelligent Personalized TV Guides'. *Seventeenth National Conference on Artificial Intelligence*, Austin, TX, USA, pp. 957-964.
9. D. Das and H. ter Horst: 1998, 'Recommender Systems for TV'. Technical Report WS-98-08 *Recommender System, Papers from the 1998 Workshop*, Madison, WI. Menlo Park, CA: AAAI Press, pp. 35-36.
10. R. Duda and P. Hart: 1973, *Pattern Recognition and Scene Analysis*. New York: John Wiley & Sons.

11. B.J. Fogg, H. Tseng: 1999, 'The Elements of Computer Credibility'. *Conference on Human Factors in Computing Systems*, Pittsburgh, PA, USA, pp. 80-86.
12. S. Gutta, K. Kurapati, K.P. Lee, J. Martino, J. Milanski, D. Schaffer and J. Zimmerman: 2000, 'TV Content Recommender System'. *Seventeenth National Conference on Artificial Intelligence*, Austin, TX, USA, pp. 1121-1122.
13. K.J. Hammond, R. Burke and K. Schmitt: 1996, 'A Case-Based Approach to Knowledge Navigation'. In D. Leake (eds.): *Case-Based Reasoning Experiences, Lessons and Future Directions*. Cambridge, MA: MIT Press, pp.125-136.
14. J. Herlocker, J. Konstan and J. Riedl: 2000, 'Explaining Collaborative Filtering Recommendations'. *Conference on Computer Supported Cooperative Work*, Philadelphia, PA, USA, pp. 241-250.
15. R. Kubey, M. Csikszentmihalyi: 1990, *Television and the Quality of Life: How Viewing Shapes Everyday Experiences*. Hillsdale, NJ: Lawrence Erlbaum Associates.
16. K. Kurapati, S. Gutta, D. Schaffer, J. Martino and J. Zimmerman: 2001, 'A Multi-Agent TV Recommender'. *Eighth International Conference on User Modeling: Workshop on Personalization in Future TV*, Sonthofen, Germany, http://www.di.unito.it/~liliana/UM01/kurapati.pdf.
17. H. Lee, J. Nam, B. Bae, M. Kim, K. Kang and J. Kim: 2001, 'Personalized Contents Guide and Browsing based on User Preference'. *Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems: Workshop on Personalization in Future TV*, Malaga, Spain, pp.131-150.
18. F. J. Lerch and M. J. Prietula: 1989, 'How do we Trust Machine Advice?'. In G. Salvendy and M. J. Smith (eds.): *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*. Amsterdam, the Netherlands: Elsevier Science Publishers B. V., pp. 410-419.
19. J. Lull: 1990, *Inside Family Viewing: Ethnographic Research on Television's Audiences*. New York: Routledge.
20. J. Moody, C.J. Darken: 1989, 'Fast Learning in Networks of Locally Tuned Processing Units'. *Neural Computation* 1(2), pp. 281-294.
21. P. Mueleman, A. Heister, H. Kohar and D. Tedd: 1998, 'Double Agents - Presentation and Filtering Agents for a Digital Television Recording System'. *Conference on Human Factors in Computing Systems*, Los Angeles, CA, USA, pp. 3-4.
22. F. Pittarello, 'Time-pillars World: A 3D Paradigm for the New Enlarged TV Information Domain'. This Volume.
23. Predictive Media, Inc. - http://www.predictivemedia.com/
24. J. R. Quinlan: 1983, 'Learning Efficient Classification Procedures and their Application to Chess End Games'. In R. S. Michalski, J. G. Carbonell and T. M. Mitchell (eds.): *Machine Learning: An Artificial Approach*, Vol. 1. Palo Alto, California: Morgan Kaufmann Publishers Inc.
25. J. R. Quinlan: 1991, *C4.5: Machine Learning Programs*, Palo Alto, California: Morgan Kaufmann Publishers.
26. B. Smyth and P. Cotter: 1999, 'Surfing the Digital Wave: Generating Personalised TV Listings using Collaborative Case-Based Recommendations'. *International Conference on Case-Based Reasoning*, Springer-Verlag, Germany, pp. 561-571.
27. B. Smyth and P. Cotter: 'The Evolution of the Personalized Electronic Program Guide'. This Volume.
28. TiVo, Inc. - http://www.TiVo.com
29. Tribune Media Services - http://www.tms.tribune.com/
30. L. Wheeless and J. Grotz: 1977, 'The Measurement of Trust and Its Relationship to Self-disclosure'. *Communication Research* 3(3), pp. 250-257.

31. J. Zimmerman and K. Kurapati: 2002, 'Exposing Profiles to Build Trust in a Recommender'. *Conference on Human Factors in Computing Systems*, Minneapolis, MN, pp. 608-609.