A Hybrid Learning Approach for TV Program Personalization

Zhiwen Yu, Xingshe Zhou, and Zhiyi Yang

School of Computer Science, Northwestern Polytechnical University
Xi'an, P.R.China, 710072
yuzhiwen77@yahoo.com.cn

Abstract. The rapid growth of communication technologies and the invention of set-top-box (STB) and personal digital recorder (PDR) have enabled today's television to receive and store tremendous programs. The abundance of TV programs precipitates a need for personalization tools to help people obtain programs that they really want to watch. User preference learning plays an important role in TV program personalization. In this paper, we introduce a hybrid user preference learning approach for TV program personalization. The learning architecture is designed to integrate multiple learning sources for preference learning, which are explicit input/modification, user viewing history, and user real-time feedback. Among those, learning from user viewing history and learning from user real-time feedback are described in detail. The experimental results proved that the hybrid learning approach outperforms the learning method merely adopting user real-time feedback.

1 Introduction

The rapid growth of communication technologies, such as broadcast, satellite, and Internet, etc. has created abundant channels transmitting programs to today's television set. TV viewers are confused how they can find programs they really like from the thousands of programs.

For hundreds of channels, browsing printed TV guide and channel surfing may take a long time. Most users have no patience to use these ways. Recently, electronic program guides (EPGs) have become available. While EPGs allow viewers to identify desirable programs more efficiently than conventional printed guides, they still lack of intelligence. TV viewers still have to look for interesting programs manually.

New means should be introduced to provide viewers with what they really want to watch in an intelligent and transparent manner. TV program personalization can fulfill this purpose. Three key technologies are needed to implement TV program personalization, which are feature representation, user profile learning, and recommendation technology. The capability to model and learn user interests is at the heart of TV program personalization systems. Since interests of a user is changing as time goes by, the factor, which most affects the personalization quality, is whether the user profile really reflects the user preference.

This paper introduces a hybrid learning approach for TV program personalization. The learning architecture is designed to integrate multiple learning sources for preference learning, which are explicit input/modification, user viewing history, and user real-time feedback. Among which, learning from user viewing history and learning from user real-time feedback are described in detail.

2 Related Work

How to update the user profile to express the preference up to date for TV personalization or recommendation is a difficult problem. Much recent work has been done in this area. Existing methods on preference learning can be divided into three categories based on different learning sources:

- *Explicit input* User explicitly inputs or modifies his preference manually, typically by clicking on items in a Graphical User Interface. It is simple and accurate, but cannot adapt to user's interests changing.
- *Explicit feedback* User feeds back his evaluation to what system had recommended explicitly. It is clear, but need the system to provide interfaces.
- *Implicit feedback* System watches the user behavior and analyzes the user's viewing history (automatic user profile adaptation).

The explicit techniques (explicit input and explicit feedback) can reflect abrupt interest changes. But when watching TV, user rarely expresses his/her preferences to the system actively. Further more, the explicit techniques are 'static', and cannot adapt to changing user tastes. TV-Advisor [1] employs explicit feedback to adapt user interest changing. PTV [2] does not include a 'dynamic' learning algorithm that tracks a person's changing TV preferences over time.

The implicit learning is capable of reflecting gradual interest changes, and can adapt to user's changing interests. But it is lack of reflection to abrupt interest changes. The methods based on viewing history analysis generally apply probability statistics, which only identify a program "watched" or "non-watched", and cannot know how much the user likes a program or a feature. For most cases, the user can decide if he/she likes the program until he/she has watched the program for several minutes. The methods based on viewing behavior use average update. Once the user watched a program, the methods apply the same update to the related features. If the viewing history statistics are not considered, it is difficult to update reasonably and avoid the misunderstanding for the user's behavior.

Based on these reasons, many researchers have proposed hybrid or integrated learning strategies for TV personalization, which have better performance than single learning method. Philips TV recommender [3] encapsulates three user information streams: implicit viewing history, explicit preferences, and feedback information on specific shows into adaptive agents for program recommendation. P-EPG [4] adopts both implicit and explicit feedback for user modeling.

We believe that combining different learning techniques in an appropriate and intelligent way can achieve better user profile approaching to user's real interests. Our learning architecture is designed to integrate multiple learning sources including explicit input and implicit feedback. For implicit profiling, history-based learning and real-time feedback based learning are combined.

3 Learning Architecture

The learning architecture is designed to integrate multiple learning sources for preference learning. The schematic of the system is shown in Fig. 1. Three knowledge sources are used to update user profile: explicit input/modification, user viewing history, and real-time feedback. Explicit input/modification means a user inputs interests when registration or modifies preference after log in through Graphic User

Interface. The user viewing history is a list of TV programs that a viewer has watched (positive examples) or rejected (negative examples). The real-time feedback is the feedback information on specific programs, which is tracked by the system automatically and instantly.

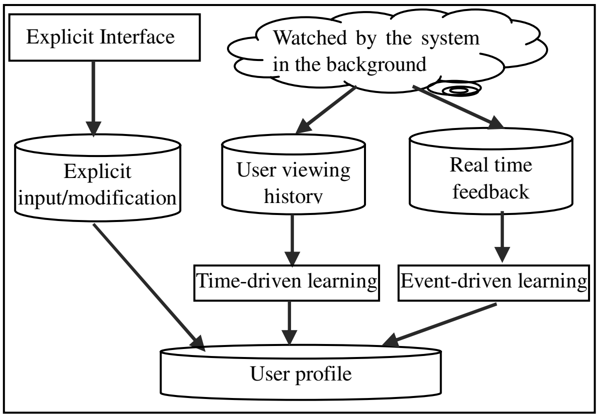

Fig. 1. Learning architecture

Profile update through explicit input/modification can be done easily and directly. While profile learning through user viewing history and real-time feedback is much more complicated. A time-driven learning algorithm is designed for update user profile through user viewing history. Real-time feedback is taken as input by an event-driven learning algorithm for update user profile. The profile update is done through the modification of the preference features and their weights respectively. We describe these two algorithms in detail in Section 4 and Section 5.

4 Learning From User Viewing History

The profile learning by using user viewing history is a time-driven algorithm, that is, the learning algorithm runs at a regular interval, for example, once in a week. The learning context is illustrated as Fig. 2. The user viewing history is stored in two files; one contains positive examples (a list of programs the viewer has watched), the other contains negative examples (a list of programs the viewer has rejected). The examples are tab separated information about the programs and user's viewing attributes.

When the learning occasion arrives, the learning algorithm will startup. It firstly reads data from user viewing history files. And then it uses VSM (Vector Space Model) [5] as the feature extraction and object information presentation method to represent them. With the logic data representation done, the kernel learning algorithm is launched to update user profile.

The general model of the learning algorithm is based on relevance feedback [6], which is an effective and efficient information retrieval technique that can be used to adjust user profile approaching to user's real preferences. The update algorithm is defined as follows:

$$W_i' = \alpha \times W_i + \beta \times F_{pos} - \gamma \times F_{neg} \qquad (1)$$

W_i' is the updated weight of feature f_i; W_i is the initial weight of feature f_i. α, β and γ are the feedback parameters to be set. The well-known and effective feedback parameters is instantiated by Rocchio as α=1, β=2, and γ=0.5. F_{pos} and F_{neg} represent the weights of positive feedback and negative feedback to feature f_i respectively.

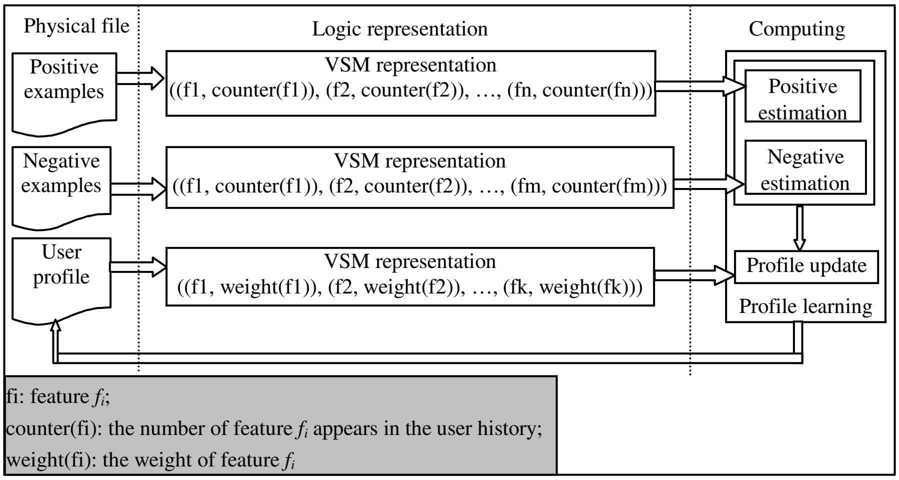
Physical file
Logic representation
Computing
Positive examples
Negative examples
User profile
VSM representation ((f1, counter(f1)), (f2, counter(f2)), …, (fn, counter(fn)))
VSM representation ((f1, counter(f1)), (f2, counter(f2)), …, (fm, counter(fm)))
VSM representation ((f1, weight(f1)), (f2, weight(f2)), …, (fk, weight(fk)))
Positive estimation
Negative estimation
Profile update
Profile learning
fi: feature f_i;
counter(fi): the number of feature f_i appears in the user history;
weight(fi): the weight of feature f_i

Fig. 2. Learning from user viewing history

The positive estimation (F_{pos}) and negative estimation (F_{neg}) were performed by using Naïve Bayes classifier [7] approach. All programs are divided into two classes, *watched* or *rejected*, that is, positive feedback examples class *PC*, and negative feedback examples class *NC*. F_{pos} and F_{neg} equals the conditional probabilities of feature *fi* in class *PC* and *NC*:

$$F_{pos} = P(f_i \mid PC) \tag{2}$$

$$F_{neg} = P(f_i \mid NC) \tag{3}$$

Supposing $n(PC, f_i)$ means the times of feature *fi* occurring in *PC*, while $n(NC, f_i)$ means the times of feature *fi* occurring in *NC*. $n(PC)$ and $n(NC)$ denote the sum of times of all features occurring in *PC* and *NC* respectively. $|V_1|$ and $|V_2|$ denote the total number of features occurring in *PC* and *NC* respectively. According to Lidstone's Law of succession [8], $P(f_i \mid PC)$ and $P(f_i \mid NC)$ can be estimated as follows:

$$P(f_i \mid PC) = \frac{n(PC, f_i) + \lambda}{n(PC) + \lambda \mid V_1 \mid} \tag{4}$$

$$P(f_i \mid NC) = \frac{n(NC, f_i) + \lambda}{n(NC) + \lambda \mid V_2 \mid} \tag{5}$$

λ is a positive number, normally between 0 and 1.

5 Learning From User Real-Time Feedback

The profile learning by using user real-time feedback is an event-driven algorithm. When a real-time feedback (we mean a switch between channels or programs) happens, the learning algorithm will startup.

Usually the user will only watch what he/she likes unless he/she accidentally switches to something he/she does not like. We should eliminate these accidental switches' impact on user profile learning. Generally, these accidental wrong switches can be filtered from the user's viewing history by a TST (Trashy Switch Time). This TST is a threshold value, for example, we can set TST = 2s, and therefore all those viewing pieces in the user's viewing history which has a viewing time less than TST will be considered as accidental wrong switches. Through TST those accidental wrong switches can be filtered from the user's viewing history.

However, to different contents, TST has different significance. For example, 2s is long for an advertisement lasting 5s, but it is very short for a film lasting 2 hours. So the ratio of user's real watching time to the content's total duration time is significant. We define T_r as user's real watching time, and T_t as total duration time of a specific program. We assume that if $\frac{T_r}{T_t}$ is larger than a threshold μ (such as 0.01), the user really likes the content (the user gives implicit positive feedback); otherwise, the user dislikes it (the user gives implicit negative feedback).

The learning algorithm is depicted as follows:

(1) If T_r < TST then $W_i' = W_i$, that is not to update the user preference, because the user's switch is an accidental switch.

(2) If T_r ≥ TST, which means the user's switch is significative, then for those features in the metadata of the program just watched,

(I) If a feature already exists in the profile, its weight is modified according to the following formulae:

$$W_i' = \begin{cases} W_i + \omega \times \varphi \times f(i) & \varphi \geq \mu \\ W_i - \omega \times \varphi \times f(i) & \varphi < \mu \end{cases} \tag{6}$$

$$\varphi = \frac{T_r}{T_t} \in [0,1] \tag{7}$$

$$f(i) = \frac{I_{\max} - i}{I_{\max}} \qquad 1 \leq i \leq I_{\max} \tag{8}$$

where w is the learning rate which indicates the sensitivity of the profile to user feedback. If the user think the feedback is very important and wants the profile learning to be fast, then w can be set larger, other wise w can be set smaller. j is the ratio of user's real watching time (T_r) to the program's total duration time (T_t). φ can be considered as the user's evaluation to the program which he/she has viewed. $f(i)$ reflects the influence of the order of the feature in user's viewing history to the weight update process. The more important features with larger weight reasonably have more influence on the profile learning. So $f(i)$ should decrease with increasing order of the feature. In the expression of $f(i)$, i is the order of feature f_i in the user's profile; $I_{\max}$ is the maximum of i, in other words, it is the total number of features in the user's profile.

(II) If a feature does not exist in the profile, calculate the feature's weight in accordance with the following formula:

$$W_i = \begin{cases} \omega \times \varphi \times f(i) & \varphi \geq \mu \\ -\omega \times \varphi \times f(i) & \varphi < \mu \end{cases} \tag{9}$$

here W_i is the initial weight of the new feature f_i, w and j have the same meanings as above. Since feature f_i is not in the profile before, so $f(i)$ can't be calculated as above, we define it as a default value e. If the absolute value of calculated w_i is higher than a preset threshold ξ (that is, $|W_i| > \xi$), we will add it to the user's profile, otherwise discard it, because it is too trivial.

6 Experimental Result

In information retrieval, system's quality is evaluated by *precision* and *recall* [9]. Since the efficacy of user profile learning directly influences the performance of personalization (that is precision and recall), we can evaluate the learning efficacy of our hybrid learning approach by using these two criterions. Given a time interval, let *interested* denote the program set which the user is interested in the interval and *recorded* denote the program set which the system has recorded in the same time interval, then *precision* and *recall* can be defined as follows:

$$precision = \frac{recorded \cap \text{int}\,erested}{recorded} \tag{10}$$

$$recall = \frac{recorded \cap \text{int}\,erested}{\text{int}\,erested} \tag{11}$$

Since the two measures are often conflicting, for instance, increasing the number of *recorded* tends to increase *recall* but decrease *precision*, we use *F1,* which integrates both *precision* and *recall*, to evaluate our system. *F1* is defined as follows [9]:

$$F1 = \frac{2 \times recall \times precision}{recall + precision} \tag{12}$$

We made the performance comparison with the learning method exploited in our previous personalization system [10], which captures user preferences from user real-time feedback similar to the method described in Section 4 of this paper. The experimental result is shown in Fig. 3. The experiment consists of 8 sessions. The number of simulated broadcast programs ranges from 50 to 100 in each session. Comparing the curves of *F1* obtained by exploiting our hybrid learning approach and previous learning method, we can see that the hybrid learning approach proposed in this paper is generally superior to our previous learning method.

7 Conclusion

This paper introduces a hybrid approach to capture user preference for TV program personalization. The learning architecture is designed to integrate multiple learning sources for preference learning. The Naïve Bayes classifier approach and relevance feedback are closely combined in the learning approach. The experimental result proved that this hybrid learning approach is effective for user preferences observation in TV program personalization systems. The specific algorithms presented here can be applied in more general area of information personalization or recommendation, such as e-mail, XML documents, E-Commerce, and web etc.

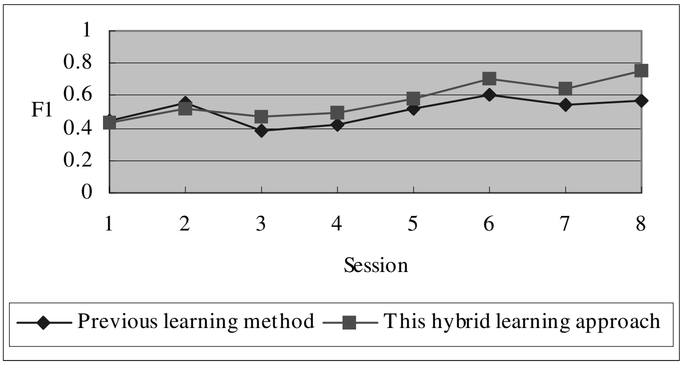

Fig. 3. Experimental result

Acknowledgments

This work was supported by the National Science Foundation of China, and the Doctorate Foundation of Northwestern Polytechnical University of China.

References

1. Das, D., ter Horst, H: Recommender Systems for TV. Proc. of AAAI. (1998)
2. Cotter, P., Smyth, B.: PTV: Personalised TV Guides. Proc. of the 12th Conf. on Innovative Applications of Artificial Intelligence, IAAI 2000, Austin, Texas (2000)
3. K. Kurapati, S. Gutta, D. Schaffer, J. Martino, J. Zimmerman: A Multi-Agent TV Recommender. Proc. of the User Modeling 2001: Personalization in Future TV Workshop, Sonthofen, Germany (2001)
4. Ehrmantraut M., Harder T., Wittig H., Steinmetz R.: The Personal Electronic Program Guide - Towards the Pre-Selection of Individual TV Programs. Proc. of CIKM'96, Rockville, Maryland, USA (1996) 243-250
5. G Salton: Automatic Text Processing: The transformation, analysis, and retrieval of information by computer. Addison-Wesley, Massachusetts, USA (1989)
6. J.J.Rocchio: Relevance feedback in information retrieval. The Smart System--Experiments in Automatic Document Processing. Prentice Hall (1971)
7. Joachims T.: A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. Proc. of the 14th Intl Conf. on Machine Learning (1997) 143-151
8. Lidstone, G.J: Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. Transactions of the Faculty of Actuaries, 8:182-192.
9. C.J. van Rijsbergen: Information Retrieval. Butterworths, second edition (1979)
10. Zhou X.S., Yu Z.W., Gu J.H., Wu X.J., Zhang Y.: A Multi-Agent System for Personalized and Private Service in PDR. Proc. of ITCC2003, Las Vegas, Nevada, USA (2003) 635-639