A MULTI-AGENT ARCHITECTURE FOR DYNAMIC COLLABORATIVE FILTERING

Gulden Uchyigit and Keith Clark
Department of Computing, Imperial College, London SW7 2BZ
Email: {gu1,klc}@doc.ic.ac.uk

Key words: Multi-Agent Systems, Collaborative Filtering, Recommendations

Abstract: *Collaborative Filtering* systems suggest items to a user because it is highly rated by some other user with similar tastes. Although these systems are achieving great success on web based applications, the tremendous growth in the number of people using these applications require performing many recommendations per second for millions of users. Technologies are needed that can rapidly produce high quality recommendations for large community of users.
In this paper we present an agent based approach to collaborative filtering where agents work on behalf of their users to form shared "interest groups", which is a process of pre-clustering users based on their interest profiles. These groups are *dynamically* updated to reflect the user's evolving interests over time. We further present a multi-agent based simulation of the architecture as a means of evaluating the system.

1 INTRODUCTION

The huge amount of Information available in the currently evolving world-wide information infrastructure can easily overwhelm end-users, a situation that is likely to worsen in the future, unless the end user has the ability to filter information based on its relevance. *Content-based filtering* systems infer a user's profile from the contents of the items the user previously rated and recommend additional items of interest according to this profile. In contrast, *Collaborative filtering* systems (Herlocker et al., 1999), (Maes, 1994), (Kautz et al., 1997), (Terveen et al., 1997), work by collecting human judgements (known as ratings) for items in a given domain and correlating people who share the same information needs or tastes. These systems generally have the following problems. They rely on an overlap of user rated items (i.e if the user's did not rate any common items then their profiles can not be correlated). The enormous number of items available to rate in many domains makes the probability of finding user's with similar ratings significantly low. Since, recommender systems are used by a large number of users, correlation based algorithms need to search through a large neighbourhood of user's in real time. An alternative approach and one that we use is *content based collaborative filtering* techniques. Such systems (Basu et al., 1998), (Claypool et al., 1999), (Good et al., 1999) combine both the content and collaborative information. We utilise the content of the items the users have rated to infer their interest profile and we then use these profiles to *dynamically form* interest groups which are continuously updated with changing user interests. These interest groups are smaller to analyse compared with the correlation based algorithms where a large neighbourhood of users need to be analysed every time a collaboration recommendation is to be given.

The paper is structured as follows. We first present the multi-agent architecture of our system, along with a brief description of the functionality of the agents involved. We then explain the method of constructing and updating a user's interest profile. This is followed by a description of the collaborative filtering framework, in which we present our algorithm for dynamicaly updating the different interest groups. This is followed by an overview of the multi-agent based simulation of the framework.

2 SYSTEM OVERVIEW

The personalised TV recommender provides the user with recommendations from online TV guides

and other web accessible TV program information. It creates, learns and modifies the user profile automatically based on user feedback on the programs that they have already seen.

A recommended list of programs is displayed to the user upon request or regularly on a daily basis. The user can also request more detailed descriptions or reviews of the recommended programs. The user can decide to watch recommended programs or make his own entirely different selection. The user is asked to give feedback on the programs that they watched the next time they view a new recommendation.

A network of agents work behind the scenes, completely hidden from the user to model and learn user preferences in order to provide the program recommendations. A pictorial overview of the agent architecture is given in Figure 1. For each user interacting

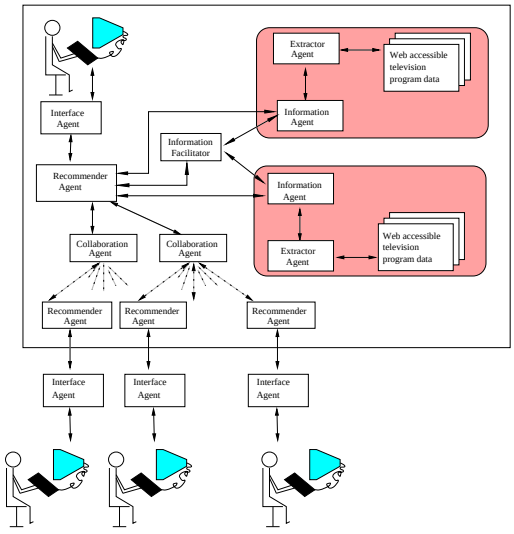

Figure 1: Overview of the agent architecture

with the system there is an associated *interface agent* and a *recommender agent*. The interface agent handles all user interactions with the system. It is also able to map user requests and actions into a form understandable by the recommender agent. The recommender agent, on the other hand, does not directly interact with the user but it is the only agent in the system that knows about the user's viewing preferences. Since the recommender agent has to accomplish numerous complex tasks related both to the user and the other agents, it needs to have a picture of the current state of affairs. In particular its knowledge component has a representation for:

- *the user's world*, the kind of viewing habits a user has, his current viewing interests and his past history with respect to the viewed programs. This information is represented in the form of a user profile.
- *the available agents* and their capabilities. It is able to coordinate a number of agents and knows what tasks they can accomplish, what resources they have and their availability.

The system also supports a collection of information sites. The notion of an information site is used to describe a logical entity that contains a set of WWW sites. At each site there is an *extractor agent* and an *information agent*. The role of the former is to extract Television guide information, the role of the latter is to maintain a database of extracted information. For all the sites there is only one *information facilitator agent* that is able to accept queries from the network of recommender agents. It is then able to route these queries to the information agents that are able to answer the query. The inter-agent communication is based on standard Knowledge Query Manipulation (KQML) (Finin et al., 1997) performatives.

For the collaborative recommendation component one *collaboration agent* exists for each program category. Recommender agents register their user's interest profile with these agents. Collaboration agents use these registered interest profiles to automatically and dynamically create and maintain interest groups within each program category. For example, there is a set of interest groups for *situation comedies*, another set of interest groups for *documentaries* and so on. When one of the users reports that they have unexpectedly enjoyed a program (unexpectedly because the program was not recommended by their agent) their recommender agent will immediately inform the collaboration agent for that program category. This will then route this information to all the recommender agents in the same interest group. Each of these agents will recommend this new program to its user if the user hasn't yet seen it. They can then catch the program on a repeat or on its next episode, if it is a series or a serial. More generally, each time the user gives feedback on the programs they have watched the recommender agent updates the user's profile using descriptions of the viewed programs. This in turn causes the interest groups to be updated. It may be the case that an updated user profile is moved from one interest group to another by the collaboration agent, or it may be used to form the nucleus of a new grouping.

3 USER PROFILING

In this section we briefly describe the format of a user's interest (or viewing) profile. Each user's profile is divided into categories corresponding to program categories such as *dramas*, *comedies*, in total we have 29 fixed categories that a profile can be divided into. Each category in turn is represented as

weighted tuples of keywords (actually word stems) produced from the descriptions of programs the user has liked and disliked. For instance Table 1 is an illustration of a user's preferences in the drama category (i.e the types of drama programs the user likes). The fact that this user has a drama category in his profile indicates his interest in drama programs in general but the existence of "hospital" (i.e hospit) in his profile indicates his *high* interest in hospital drama programs. This type of keyword representation of the

Table 1: User's drama profile

drama	murder	doctor	victim	hospit

user interests has the problem that small amounts of training data will lead to selection of keywords that accidentally appear in the data and that are really irrelevant in discriminating between interesting and uninteresting programs. Using irrelevant features makes the learning task much harder and leads to a lower classification accuracy. As a solution to this problem alternative methods have been proposed to initialise the user profile. For instance in (Pazzani and Billsus, 1997), these keywords are elicited *explicitly* from the user. However, this process of profile initialisation is too tedious for the user and will often result in a user being unable to specify the words that best describe his interests in a particular domain, especially if the user is unfamiliar with this domain. To overcome this problem we use a *rapid profile initialisation method* where the agent automatically and continuously extracts web accessible program information and determines the informative terms that most frequently occur in the descriptions of TV programs for each program category. This is done in the following way. Textual content of program descriptions for different categories are extracted from web sources. The content is then parsed and all *stop-words* (or non informative terms such as "a", "the", "and", "to" etc.) are removed. The words are then stemmed using the Porter stemming algorithm (Porter, 1980) such that the word "computers" and "computing" both stem to the word "comput". We use the expression below to calculate the weight w_j for a word k_j appearing in the description of a particular program category c_p with an analogous expression for the weight of a word for a program not in that category (i.e $\overline{c}_p$). $\overline{c}_p$ is the union of all the program descriptions in categories other than c_p.

$$w_j = (P(c_p|k_j) - 0.5) \times fabs(sig) \qquad (1)$$

Here, $P(c_p|k_j)$ is the probability of a program description belonging to the category c_p given that it contains the word k_j, this is calculated from the simple Bayes formulation

$$P(c_p|k_j) = \frac{P(k_j|c_p)P(c_p)}{P(k_j|c_p)P(c_p) + P(k_j|P(\overline{c}_p))P(\overline{c}_p)} \qquad (2)$$

where, $P(k_j|c_p)$, is the probability that the program description contains the word, k_j given that it belongs to the category c_p. $P(c_p)$ is the probability that program is in category c_p (i.e, the ratio of programs in category c_p to all programs in all the other categories). Similarly $P(k_j|\overline{c}_p)$, is probability that the program description contains the word, k_j given that it belongs to $\overline{c}_p$ and $P(\overline{c}_p)$ is the probability that a program is not in c_p

The sig value is calculated by the formulation:

$$sig = \frac{\frac{h_1}{n_1} - \frac{h_2}{n_2}}{\sqrt{\left(\frac{h_1+h_2}{n_1+n_2}\right) \times \left(1 - \frac{h_1+h_2}{n_1+n_2}\right) \times \left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \qquad (3)$$

where, h_1 is the number of times word k_j occurs in the set of program descriptions in c_p, h_2 is the number of times word k_j occurs in the set of programs of $\overline{c}_p$. n_1 and n_2 are respectively, the number of words in the set of programs in c_p and $\overline{c}_p$. These words are then sorted according to their weights and the top scoring words are selected as the representative keywords for each category. The higher the weight the more significant the word.

As a result of this process each category is described with a set of informative terms (keywords). Table 2 shows *some* of the keywords which have been automatically extracted from the content of random selection of *drama* program descriptions. Recommender agents then use these sets of pre-

Table 2: Feature words

secret	disast	suspens	hostag	doctor
terrifi	mysteriou	affair	drama	polic
evil	trial	victim	emergen	hospit
crimin	murder	viciou	scam	danger

defined keywords to initialise a user profile using user feedback. At first registration the user is asked to rate some programs he has seen within the past two weeks. The agent then uses the descriptions of the programs the user rated and the list of pre-defined keywords to initialise the user profile. For instance consider the program description illustrated in Figure 2, further consider that this program was liked by the user. To initialise the user's profile the agent repeatedly searches for occurrences of the pre-defined keywords from Table 2 within the description of this rated program to produce the subset of keywords of Table 1.

Each time a keyword from Table 2 is found its count is incremented by 1. Finally, a probability is determined for each found keyword using the total number of times it appears in the set of programs which has been liked with the total number of times the keyword appears in the descriptions of any program which has been viewed, and rated for that program category. The

Murder On The Hour.
Lighthearted drama about a hospital doctor who uses his sleuthing skills to help the crack baffling cases. A serial killer appears to be selecting victims who had recently come close to death, but were on their way to making full recoveries. In addition, every murder is taking place on the hour.

Figure 2: A description of a drama program

advantage of the prior keyword selection process is that a user need only rate a few items for the agent to be able to determine which keywords to select for representing in the user profile.

Later on, as the user gives more feedback, the agent uses this feedback to update the user profile to reflect the users changing interests. This involves an updating of the prior probabilities of the keywords within the user profile. For this we use *conjugate priors* (Pazzani and Billsus, 1997). Conjugate priors are a traditional technique from Bayesian statistics to update probabilities from data (Heckerman, 1996).

Classifying new programs
We use the naive Bayesian probabilistic classifier (Duda and Hart, 1973) to determine the probability that a given program description will belong to the set of programs liked or to the set of programs disliked in any given category, given the feature values of the of the program description. To determine the probability that a program description belongs to either the liked or disliked subset of a category, we use Equation 4, which is the naive Bayes probabilistic classifier.

$$v_{NB} = arg \max_{v_j \in \{like, dislike\}} P(v_j) \prod_i P(a_i|v_j) \quad (4)$$

where, v_{NB} denotes the target value of the naive Bayesian classifier, $P(a_i|v_j)$ is the probability that a program description contains the word a_i given that it was *liked* or *disliked*. The naive Bayesian probabilistic classifier has been chosen since previous research suggests that it is faster at learning than other classification methods (Pazzani and Billsus, 1997).

4 BUILDING AND MAINTAINING INTEREST GROUPS

In our system the collaboration agent clusters user's into interest groups based on the similarity of their profiles for each interest category. Users are not clustered based on their entire profile contents since it may be the case that two users have similar tastes in *comedies* but quite different tastes with respect to *sports* programs.

For the process of clustering we adapted the Hierarchical Agglomerative Clustering (HAC) algorithm (Everitt, 1980), (Rasmussen, 1992), (Willet, 1988), to cluster user profiles. The HAC process repeatedly merges the two most similar clusters until only one cluster remains. This results in a *single* hierarchy or *dendrogram* to use the correct terminology.

Our adaption of this algorithm for clustering users based on their profiles is as follows. The first phase involves *initialisation* where each separate interest profile is represented as one element cluster. There then follows the process of merging the two most similar clusters until one of the *two* possible termination conditions are satisfied. Either, the similarity of any two clusters is less than 0.7 or only one cluster remains. For the similarity measure between clusters, we use vector similarity (Salton and Buckley, 1988). Figure 3 is the pseudo-code of our algorithm for clustering the user profiles in a given category. Figure 4 shows the clusters formed for three different categories of interest. At present we keep the similarity levels fixed for every category. One of our future plans is to determine experimentally the *optimal* similarity levels for the different categories. For the formation of the *interest groups*, the collaboration agent generates the clusters that have an internal profile similarity greater than 0.7. In Figure 4 the drama category has three clusters (clusters are determined by horizontal lines in the dendrogram). These are $\{d_1, d_2, d_3, d_4\}$, $\{ d_5, d_6\}$, $\{d_7, d_8, d_9\}$ where, $d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9$ are the individual interest profiles for the drama category of nine people. For maintaining the interest groups the agent reclusters the profiles on a weekly basis. As a result of re-evaluating the clusters, the clusters are updated to reflect the user's changing interests. This may result in new clusters being formed or existing clusters being augmented.

5 SYSTEM EVALUATION

The collaborative recommendation component based on viewing recommendations that come from users with similar tastes requires a reasonable large community of users. We have not got the resources

Input

P=$\{p_1$,p_2,..p_N $\}$

Initialise

– start with clusters that contain a single user profile

C=$\{c_1$,c_2,..c_N $\}$ *a set of clusters*

c_i=$\{p_i\}$ for $1 \leq$ i $\leq$ N

$S = 0$

- Repeat the following steps iteratively until there is only one cluster left **or** $S \leq 0.7$

  for $k = N - 1$ **to** 1 **do**

- identify the two clusters that are most similar

$$S = (c_j, c_m) = arg \max sim(c_j, c_m)$$

where, $sim(c_j, c_m)$ is the cosine similarity

$$\frac{\vec{c_j} \bullet \vec{c_m}}{|\vec{c_j}| \times |\vec{c_m}|}$$

- merge them to form a single cluster

$c^* = \{c_j \cup c_m\}$

- update the clusters

$C_k = C_{k-1} - \{c_{i,j}, c_{i,m}\} + c^*$

Figure 3: Pseudo-code of our clustering algorithm

for such a field trial of the system, so instead we decided to simulate such a large user community, representing each user as an agent (see figure 5). This presents the problem of how to represent the viewing tastes of each user, tastes that have to be approximated to by the user's recommender agent, as it develops its user model. We decided that the way in which the recommender agent models the user, by weighted tuples of words for liked and disliked types of programs in each program category, was a reasonable way to model a user in their simulating agent. This model is hidden inside the simulating agent and is used by that agent to give *viewing* feedback to its recommender agent which is trying to learn this model.

This approach reduced the problem of simulating a user community to the need to generate a set of different user profiles, each of which was reasonable internally consistent, to represent some plausible community of viewers. We did this by randomly choosing for each program category, a set of programs that some viewer might have liked and disliked over a three months viewing schedule. Text descriptions of these programs were then used to build the same user model for that program category that would be built by a recommender agent. We then rejected those pro-

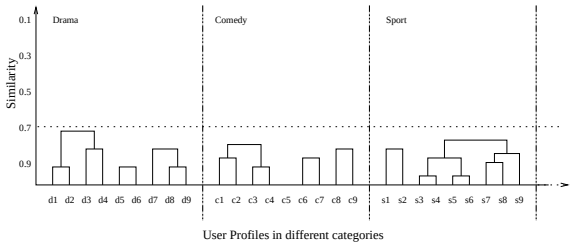

Figure 4: Profile hierarchies for different categories

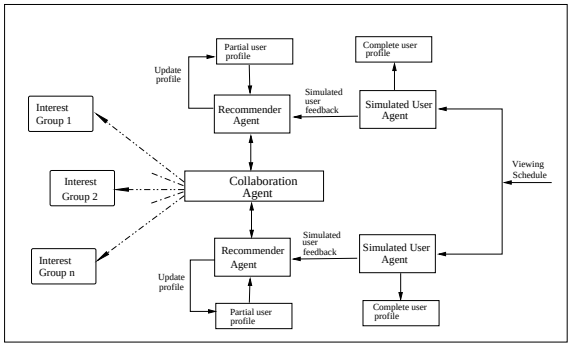

Figure 5: Agent architecture of the simulation process

files in which the liked and disliked profiles were too similar. This is so that simulated user profiles have differing likes and dislikes to represent each interest category. We then constructed a complete user model by randomly selecting a set of likes and dislikes profiles to cover all the program categories.

The next process is to simulate the user feedback process which is used by the recommender agents to learn the user profiles. We start by having the simulated user agent use its assigned user model to classify as liked or disliked each program in a three week viewing schedule and to give this information to its recommender agent. The recommender agent uses this to build a partial model of its user which it then sends to the collaboration agents. The collaboration agents use these partial user models to form the interest groups. The recommender agent also uses its partial user model to recommend new programs to the simulated user agent. The recommended programs are then classified by the simulated user agent using its more complete user model and feedback is given to the recommender agent so that it can update its partial model. These updated partial models are used by the collaboration agent to update the interest groups. In order to test the collaborative recommendation component we make the simulated user agent randomly report non-recommended programs as *liked*. These are programs which were not recommended by the recommender agent but are similar to the complete user model, this in effect

is same as having the user report *serendipitous discoveries*. We can then observe whether and how this new program is disseminated amongst the user recommender agents. By building such a simulation community we were able to evaluate the collaborative recommendations process by observing the accuracy of the recommendations given. Accuracy is the ratio of recommended programs classified as being liked by the simulated user agent to all programs recommended by the recommender agent.We could also observe how the partial user model evolves over time to become a near complete user model and also to observe how these changes will be reflected in formation and updating of the interest groups.

Experimental Results

We conducted a number of simulation experiments to assess the validity of our architecture. One of which was to observe the performance with the changing user interests. To do this we observed what would happen to the performance when there is a sudden change in the user profiles. Figure 6 shows results

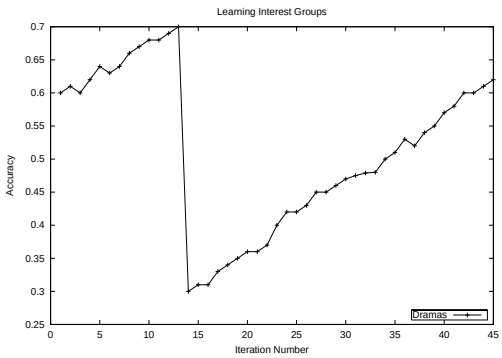

Figure 6: Results of the simulation

of the simulation experiments performed within the drama category. We report the average accuracy of the predictions given to all the interest groups over 45 iterations. At each iteration the partial profiles are re-clustered (i.e the interest groups are updated). After iteration 13 we purposely got the simulated user agents to report programs that are different to the partial profiles. This is like having user's discover new programs that were not recommended to them by their recommender agents. Only some of the simulated user agents had their complete profiles updated with this new program information. At iteration 14 this rapid change within the simulated user's profile is indicated as a sudden drop in the average accuracy of the predictions. But this is followed by a steady increase in the accuracy of the predictions with the other iterations indicating recovery of the interest groups as they are dynamically updated.

REFERENCES

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

Claypool, M., Gokhale, A., and Miranda, T. (1999). Combining content-based and collaborative filters in an online newspaper. In *ACM SIGIR Workshop on Recommender Systems- Implementation and Evaluation.*

Duda, R. and Hart, P. (1973). *Pattern classification and Scene Analysis.* John Wiley and Sons, New York.

Everitt, B. (1980). *Cluster Analysis*. John Wiley and Sons, New York.

Finin, T., Labrou, Y., and Mayfield, J. (1997). Kqml as an agent communication language. In Bradshaw, J., editor, *Software Agents*, pages 291–316. AAAI Press/ The MIT Press.

Good, N., Schafer, J., Konstan, J., Brochers, A., Sarwar, B., Herlocker, J., and Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. In *Sixteenth National Conference on Artificial Intelligence*, pages 439–446.

Heckerman, D. (1996). A tutorial on learning with bayesian networks. Technical report, Technical Report, MSR-TR-95-06 Microsoft Corporation.

Herlocker, J., Konstan, J., Borchers, A., and Reidl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the Conference on Research and Development in Information Retrieval*.

Kautz, H., Seman, B., and Shah, M. (1997). Referral web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65.

Maes, P. (1994). Agents that reduce work and information overload. *Communication of the ACM*, 37(7):31–40.

Pazzani, M. and Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331.

Porter, M. (1980). An algorithm for suffix stripping. *Program (Automated Library and Information Systems)*,, 14(3):130–137.

Rasmussen, E. (1992). *Chapter 16: Clustering Algorithms Information Retrieval, Data Structure and Algorithms*. Prentice Hall.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic retrieval. *Information Processing and Management*, 24(5):513–523.

Terveen, L., Hill, W., Amento, B., McDonald, D., and Creter, J. (1997). Phoaks: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62.

Willet, P. (1988). Recent trends in hierarchic document clustering: a critical review. In *Information Processing and Management*, volume 34.