Mixing Collaborative and Cognitive Filtering in Multiagent Systems

Ramon SANGÜESA, Alberto VÁZQUEZ-HUERGA and Javier VÁZQUEZ-SALCEDA
Software Departament,
Universitat Politècnica de Catalunya
Campus Nord, Mòdul C-6, Despatx 204
C/Jordi Girona Salgado, 1-3
08034 Barcelona
SPAIN
E-mail: [sanguesa | avazquez | jvazquez]@lsi.upc.es,
URL: http://www.lsi.upc.es/~sanguesa

Abstract: The combined use of cognitive and collaborative filtering has been advocated as a means to improve the usefulness of Recommender Systems. This approach has been used in some domains. In this paper we present the ACE Multi-Agent System, a Recommender System which combines both approaches and, in addition, is fully domain-independent. Besides describing the main traits of ACE we also comment on a preliminary experience on the domain of leisure recommendations.

Keywords: Software Agents, Multiagent Systems, Recommender Systems, Collaborative Filtering, User Modeling.

1. Introduction

Efforts towards the automatization of the information filtering process have delivered quite a lot of results but still lacked the degree of personalization that is needed to really find and deliver the rigth information to a *specific* user. Softbots allowed some type of personalized searching by using a model of each user's interests. An agent or a group of agents were in this way able to automatically build powerfull queries from the ones entered by the user and then merge and filter the results, giving to the user a, hopefully more relevant, subset of the found information.

During the last years this idea has evolved from the personalyzed searching agents to the more powerful Recommender Systems (Resnick,1994). While a personalized searching agent tries to find documents similar in content to those ones the user has received before, a Recommender Systems uses also other types of information in order to fine tune the interestingness of sources of information: apart from previous queries by the same user it can also take advantage of other cues as, for instance, time devoted to read each previous document, actions done with it (saving, printing, bookmarking), user's feedback (Pazzani, 1996) as well as the information of similar behaviour and interests of a group of users related to the actual one issuing a query. Even more, Recommender Systems try to work as unobstrusively as possible, and they do issue recommendations even when the user is not querying but just browsing or navigating through information (Lieberman,1995), (Munro,1999) in this way acting more as a Personal Digital Assistant (Maes, 1994). Although there is a wide variety of solutions it can be said that two types of recommender system seem to have appeared. One type is based solely on content and the observations of a single user actions. The second one pools behaviour and interests of a community of users

(Terveen,1997) *including* the recommendations on a set of topics contributed by the same users that form the virtual community. There are both advantages ans disadvantages in each type of system. The first one require less actions on the part of the user. Users in the first systems do not need to contribute with a recommendation issued by the system (as *in Syskill and Webert*, see (Pazzani,1996)) but then they have to rely on the automated indexing and categorization mechanisms used in calculating the relevance or interestingness of information content. Automatic calculation of relevance may be a crude approximation to the categorization and evaluation of information that a good human expert can do while evaluating content. The second type of systems overcomes this last disadvantage by relying on human judgement. In effect, the users in the community issue recommendations on documents and other sources of information that are based on their expertise in the domain, in principle a much more elaborated way of rating information. On the negative side, the quality of a system based on this collaborative approach hinges critically on the quality of each users recommendations. Other problems include the need to have a critical mass of active and qualified users/contributors, the cold-start effect and the decaying involvement of users without any further incentive. Advantages and disadvantages of both approaches have been analysed in (Balabanovic,1997b), and are summarized in Table 1:

Cognitive Filtering	**Collaborative Filtering**
Each time a new document enters into the system, it can be recommended (an analisys of its content can be made).	There's no way to recommend an incoming document until it has been voted by a minimun number of users.
The filtering quality is not affected by the number of registered users and the number and the quality of recommendations made by them.	The filtering quality depends on the number of registered users and the number and the quality of recommendations made by them.
The content-based analisys of a document can only manage textual documents. It's hard to analyze by its content information such as images, the style or the layout of a document, or semanthical information.	It's based on the opinions made by other users. Users are entities with huge power of semantical and visual analysis.
The system only search similar topics to the documents retrieved before. Is hardly to get new interesting topics to the registered users.	The taste of the users usually changes in time, in a very dynamic way. Users commonly recommend new topics.

Table 1. Characteristics of Cognitive and Collaborative Filtering approaches.

Usually both approaches have been used separately. However, there have appeared some systems that mix both approaches, getting better results (see (Balabanovic, 1997a), (Balabanovic,1997b), (Delgado,1998), and (Miyahara,1998)). If we look again on Table 1 we can see the reason: disadventages in one approach can be counterbalanced by the other approach, building a system that keeps the adventages of both approaches while minimizing the disadventages. Several of these systems, however, still do show some limitations in some aspects, and there are some problems still to be solved. For example, there seems to be a contradiction in having a fixed keyword set for the content-based side of the system without expanding it as a response to the introduction of new topics by users through the collaborative side. Another problem worth menctioning is the very restricted domains and solutions offered by the current combined systems that don't seem, in principle, to generalize well to other domains. In this paper we present ACE, a multiagent recommender system that combines dinamically both approaches. In Section 2 we give an overall description of the system's domain of application, its goals and functionalities; in Section 3 we describe the internal

architecture of the system, describing in some detail the implemented solutions; in Section 4 we describe in greater detail the combined approach to document filtering; in Section 5 the problem of user modelling is discussed and our solution described; Section 6 discusses a first test application of ACE and, finally, Section 7 sums upand lists future lines of research, development and application.

2. Description of ACE

The Electronic Cultural Agency (*Agencia Cultural Eletrónica*, ACE) is a system developed for the personalized recomendation of leisure activities offered in a metropolitan area, in this case Barcelona, Spain. The system is now in the process of being integrated in the leisure pages of an electronic newspaper (Diari de Barcelona, http://www.diaridebarcelona.com), giving personalized information to the newspaper's registered readers according to their tastes or interests. Although the system was built to give recommendations on the Barcelona leisure offer, one of our major goals has been to design a recommender system general enought to be capable of recommending not only leisure but also other kind of items.

Our goal was to make the system capable of issuing recommendations with very little information about the recommended items, as for example, is usually the casewith the data available on movie or theatre shows directories or online reviews: oftenly just a short textual description. A pure Cognitive Filtering approach based solely on content wouldn't work properly, as the information it had on the item to recommend was very poor. This was the reason why a collaborative suystem seemed a good idea in the first place. A second one was cited by the newspaper directors which see a collaborative system as a means to user community building.

Figure 1: The ACE system, just after the incoming user has been authentified.

We comment briefly the overall functionalities of the system from the user's point of view. The first time a user enters to the system, he or she must register, choosing an username

and a password. The user can optionally give some information about his or her interests, choosing the kind of leisure he likes from a list (cinema, theatre or other). The required information us as less as possible so as not to bother the user with an annoyingly long questionnaire. This information, however, makes shorter the process of automatic adaptation of the system to the user's interests. A user can decide to byass this step. If the user does so, the system will accept he or she as a new user but it will have to learn his or her interests from scratch. Subsequent accesses of a user to the system only require the introduction of his or her username and password. The system then is able to recover whatever information has been built on the corresponding user profile since the last interaction.

Once ACE has identified the incoming user it shows the window displayed in Figure 1. It's a web page with two frames: the menu frame, on the left side, and the information area, on the right side. The initial options to the user are listed below:

a) *See new recommendations*: by means of this option the system shows to the user the items recommended according to the knowledge it has on the user's interests. All listed items are sorted by the estimation of user interest calculated for each item. If the user chooses one item, ACE displays a brief description of it. This textual description helps the user to decide whether this item is interesting for him. The user has the chance to vote for or against the item, as shown in Figure 2.

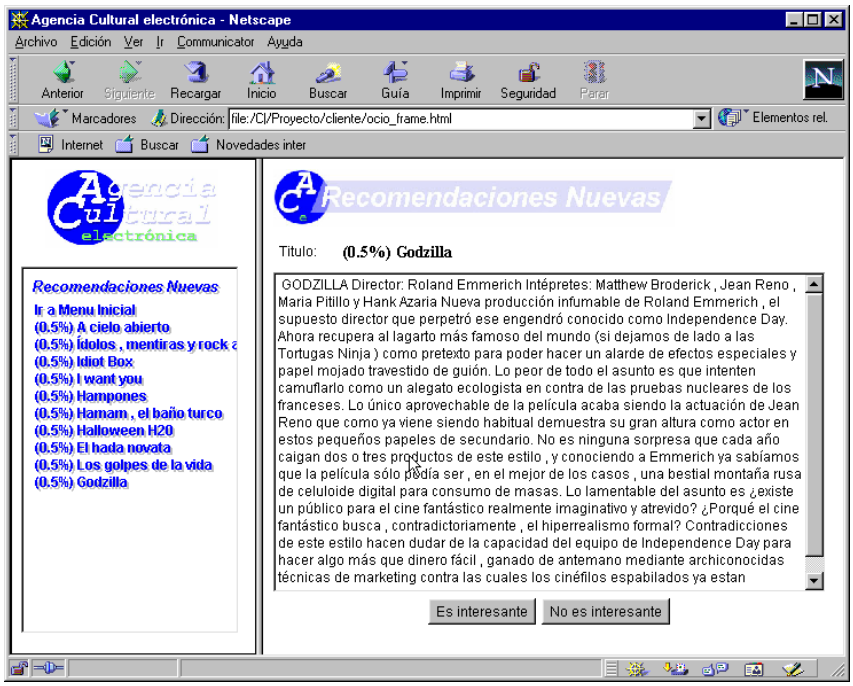

Figure 2: In the left frame, the list of recommended items for the user. In the right frame, the description of one of them, and below the buttons to vote the item.

b) *See previous recommendations*: It shows to the user the items he has voted for previously. The user cannot vote an item twice.

c) *See all the suggestions*: This option shows all suggestions ACE has retrieved for a given user. The user can see the item description by clicking the items of the list, and read the textual description associated with each item. This textual information comes from a repository of leisure activities' descriptions including movie and theatrical reviews that are freely offered by newspaper. If ACE discover a possible interest of the user in that

item then a window prompts the user asking if he wants to vote for or against that item. If some time passes with no action, ACE assumes that the user feels no special interest against or in favour of the item.

d) *See all users' suggestions*: the user can see all the suggestions submitted by the other registered users with similar interests. Again the user has the posibility to make a vote.

e) *Write a suggestion*: This option gives to the user a way to submit a suggestion to the rest of users. A suggestion is a free-text form where any opinion can be expressed. This is in contrast with other systems, where a fixed form or a limited range of voting values is offered. In this way ACE gives more freedom at the expense of further processing of each suggestion.

f) *See the help*: Displays the help.

g) *Exit*: To go out of the ACE system, returning to the main page of the electronic newspaper.

3. Internal architecture of ACE

Agent-based recommender systems tend to associate a single agent for each user, working as an assistant ot him or her. These agents usually are resident to th users computers which are assumed to be continuously connected. In our case, users are supposed to get connected to the system only for short spans of time each day. So it seemed more natural to try to locate as many as possible of the underlying processes on the server. It's important, then, to come up with an architecture that limits the potential explosion in the number of agents present on the server. Figure 3 shows the internal architecture of the system:

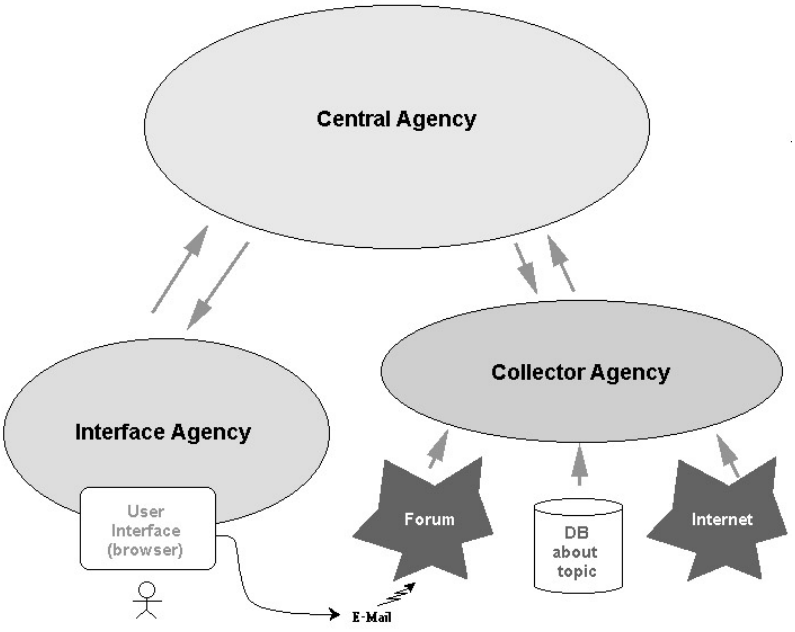

Figure 3. The internal architecture of the ACE system.

The ACE architecture is composed as several Agencies: *Interface*, *Collector* and *Central*. All informations, that is, retrieved documents and user's suggestions (we will call both documents from now on) is sent to the *Central Agency*.

3.1 Interacting with the user: the Interface Agency

The *Interface Agency* is the one which performs the communication among the users and the ACE system. Each user has three personal agents (see Figure 3.1): the *Informant Agent*, which looks into the database for the recommendations suggested by the system to that particular user; the *Feedback Spy Agent*, which gets information by "looking over the shoulder" (Maes,1994) all the actions the user does (navigation among the web pages of the ACE system, voting, etc...) to create a better profile for him or her; and the *Suggestion Spy Agent*, which has as input the suggestions made by the users of the virtual community growing arround ACE. All these agents are not active anytime but only when the user is on-line. Having all the agents continuously active for all registered users leads to a huge number of agents: the number of registered users of an electronic newspaper can grow higher than 10000 users, so the number of agents could be more than 30000. Having only active interface agents for the users that are on-line in a given moment reduces the number of active agents of the system, as all the users do not tend to get connected to the system at the same time. Simultaneously connected users are in the range of 10-900.

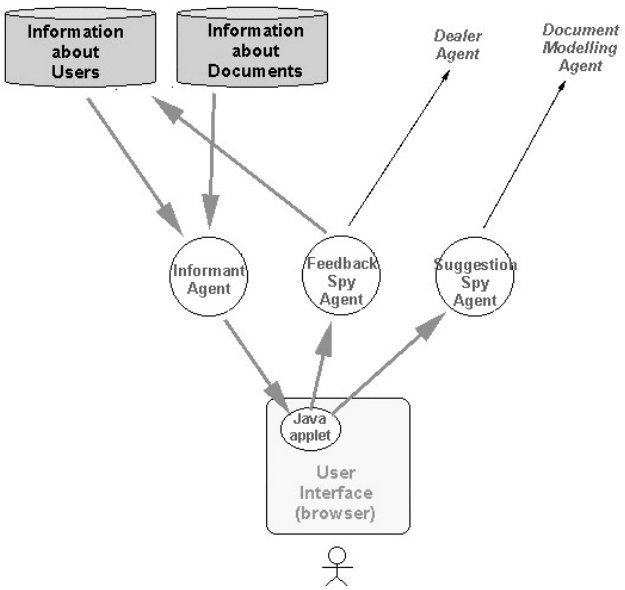

Figure 3.1. The components of the Interface Agency.

The actual architecture is an *information push* architecture: the information recommended to the user is decided before the user asks for it, as opposite to the *information pull* models where user requests are the triggers of the retrieving and filtering processes. So, each time the user enters ACE, the information to recommend is ready. There is no waiting for it.

All the information these agents need no know about the user or the documents is inside a database (in figure 3.1 and following it appears divided in *Information about users* and *Information about documents*). This database works as a blackboard for all agents, and it lets the agents to share information about users and documents instead of having the information duplicated inside each agent. This also contributes to make ACE a lightweight agency.

3.2 Retrieving useful information: the Collector Agency

The *Collector Agency* consists of a group of *Collection Agents* (see Figure 3.2) specialized in the retrieval of different sources of information, such as web pages or public databases. There are special agents such as the *Suggestion Spy Agents*, (that also belong to the Interface Agency) and who have as input the suggestions made by the users of ACE, and another agent, the *Forum Spy Agent*, which search the contributions of the registered users in the electronic newspaper's forums, in order to know better these users. All these documents are sent to the *Central Agency*.

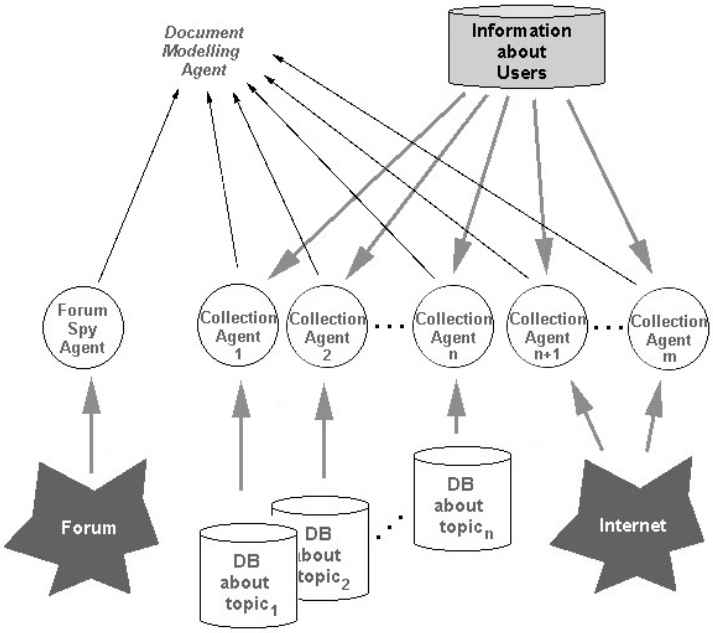

Figure 3.2. The components of the Collector Agency.

3.3 Managing documents and recommendations: the Central Agency

Finally, the *Central Agency* receives information from the *Interface* and *Collector* agencies and filters and delivers the documents to the users. The process begins when any agent in the *Collector Agency* sends some information to the *Document Modelling Agent*. This agent builds the *document model* from the content of the document (as we said, it is a textual description of an item), stores it into the database and sends a note to the *Dealer Agent* to notify the arrival of a new item to recommend. The *Dealer Agent* is the one which makes the filtering of the information, decides which items could be interesting for each user and then "delivers" the items to the user. But at any time there are not active agents representing all the users, so the action of "delivery" is not sending a message to an interface's agent but storing a record into the database, where the *Informant Agent* will extract it later as the information to be recommend to the corresponding user. When the *Dealer Agent* receives an item written by a registered user (a suggestion sent to the ACE system, or a contribution in the newspaper's forums), a note to the *User Modelling Agent* is sent in order to learn more information about that user. *The User Modelling Agent* modifies the model of that user by using the model of the document built by *the Document Modelling Agen*t. ***Here is where the combination of content and collaboration takes place****.* This process is further explained in section 5 of this paper.

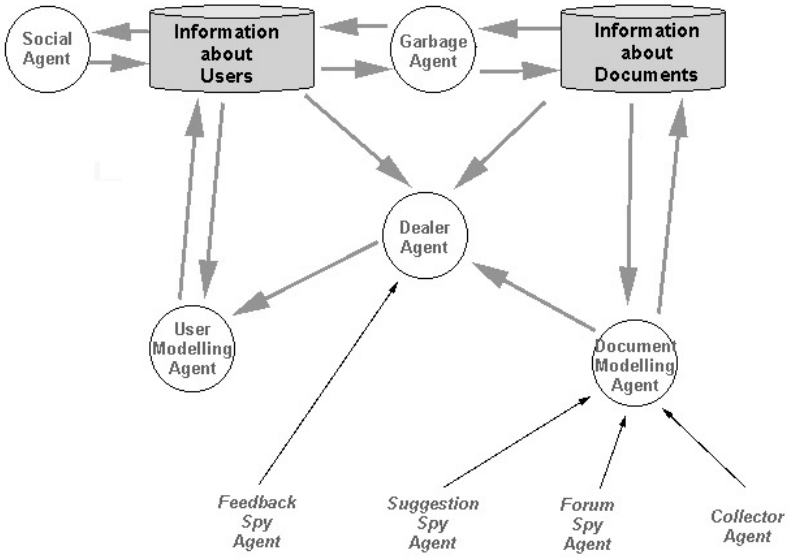

Figure 3.3. The components of the Central Agency

The *Dealer Agent* also receives from the *Feedback Spy Agents* the votes made by the users. Each vote of user *U* to item *I* is useful:

a) to learn more about the user *U*: The *Dealer Agent* sends a note to the *User Modelling Agent*, which will make use of this information to update the model of user *U* stored in the database.
b) to modify the "dealing" of item *I* to other users, taking into acount the new voting.

The ACE architecture has two more agents: The *Garbage Agent* , which gets rid of aged documents in the database, and the *Social Agent*, which clusters users, grouping them by their similar interests extracted from the documents they suggested as well as their voting behaviour.

4. Document Filtering

The ACE system doesn't work with the entire text of an item's description, but with a *document model*. In our system, the document model is the vector-space model (a list of pairs {word, weight}) as it has shown to work properly in systems mixing cognitive and collaborative filtering (see (Balabanovic 1997a, 1997b) and (Miyahara,1998)), composed by the top words of the item's description, that is, the words that "define" this document.

The process follwed for obtaining the top words of a description is divided in 4 steps, done concurrently:

a) Getting the most frequent words: a pair list {word,frequency].
b) Removing unuseful words: there's a list of stop words (in Spanish, less than 220 words), words that give very little or null topic-related information, such as articles, adverbs, conjunctions, prepositions or pronouns.
c) Join different forms of a word into a same word, reducing them to their "stems".
d) Weighting each word with the frequency of the word in the document. Words in a document are not weighted by a TF-IDF (*Term Frequency – Inverse Document Frequency*), as the IDF is computed in our system for each user.

The Document Filtering process is done entirely by the *Dealer Agent*, but is based in information computed by other agents:
- the *document model*, built by the *Document Modelling Agent*.
- the *user model* of his interests, created by the *User Modelling Agent*
- the *clusters of users*, computed by the *Social Agent*.

4.1. Stemming for general contexts

As mentioned before, one of our goals was to design and built a quite general recommender system, as independent of domain as possible. This goal difficults the stemming process. Usually, a thesaurus of topic-related terms is needed to reduce each word to its stem. But having domain independent system makes makes necessary a thesaurus that covers the whole language domain (for instance, a Thesaurus covering all English words). Having a system that should be able to be multilingual (something very important in European Community) makes necessary to have several Thesaurus covering each of them a language (English, Spanish, French, German...).

There are two approaches to solve this:
- Making the system capable of building his own thesaurus, from the documents it receives.
- Using no thesaurus in the stemming process.

The ACE system uses the second approach, as the first one is still under development by other mmebers of our group.

We have developed a new algorithm for stemming, the *fusing words algorithm*, that needs no thesaurus to work. It takes adventage of a property present in most languages: the stem is usually a word's prefix.

Our algorithm is quite simple: given an alphabetically ordered list of words *l* and a new word to add *w*:
- Searches the position where the word *w* should be inserted in *l.*
- Looks into the next words of the list if there are any words *w'* which can fuse to *w*.
- Looks into the previous words of the list if there is a word *w''* the new word *w* can fuse to.

A word *w* can fuse to a word *w'*, if *w'* is prefix of *w*, and if the difference in length of both words are under a certain threshold. The second condition ensures that it's not loosing too much semantic information about the word *w* when it's fused to *w'*. During the testing of the ACE system, the threshold was that length of word *w'* have to be, at least, the 65% of the length of word *w* to let *w* fusing into *w'*.

With the *fusing words algorithm* different forms of the same word fuse into one term, letting other words to enter in the document model. But there are two major problems in this algoritm: 1) different order of the incoming words lead to different results in fusion, and 2) the algorithm cannot fuse words that have a same ancestor, but have differences in their stems(for example, "forget" and "forgotten"). Part of our future work will focus on improving this algoritm and making some benchmarks, comparing it to the stemming.

4.2. Connecting documents with users

The estimation of the interest of an user *u* for a document *d*, $p_t(u,d)$, is given by:

$$p_t(u,d) = k \cdot p_c(u,d) + (1-k) \cdot p_s(u,d)$$

Where $p_c(u,d)$ is the estimation of the interest of user *u* for the content of document *d* (that we will name the content interest), $p_s(u,d)$ is the estimation of the interest of user *u* for the document *d* given the votes made from other users to *d* (we will name that estimation of the social interest), and *k* is a factor to weight the influence of each estimation in the result of the formula.

4.2.1. Content Interest

The estimation of the content interest for user *u* and document *d* is computed by the comparison of the *Document Model* and the *User Model*, both of them lists of pairs {word, weight}. There are some transformations needed to do this comparison, thought: first of all, both word lists are reduced to the words present in both models. Next, the words are put in the same order in both lists, so the word in the position i of the *Document Model* is the same of the word in the position i of the *User Model*. Then, there is an important transformation to be made: while the *User Model* weights its words with the TF-IDF, that is, the product of the word frequency (TF) in all the documents user u has voted, and the inverse document frequency (IDF) of this word computed for each user, the *Document Model* weights its words only by the TF, as IDF is computed for each user and stored in the ACE's database. So, the TF-IDF of the words in the document model is computed. Then the words are dropped out of the lists, so the user and the document are represented by a weight vector. The last transformation is tonormalize both vectors.

Given the two transformed lists (the *User Model* vector, *u'* and the *Document Model* vector, *d'*), the comparison is made with the cosine-similarity measure, defined by the following formula:

$$p_c(u,d) = sim(u',d') = \sum_{i=1}^{n} (u'_i \, d'_i)$$

4.2.2. Social Interest

The estimation of the social interest for user *u* and document *d* is given by:

$$p_s(u,d) = \overline{v}_u + k \sum_{i=1}^{n} w(u,i)(v_{i,d} - \overline{v}_i)$$

Where $\overline{v}_i$ is the mean vote of user *i*, $v_{i,j}$ is the vote of user *i* to document *j*, and *w(u,i)* is a similarity factor between user *u* and user *i*. The *Dealer Agent* finds all the information needed in this formula stored in the database, and is computed by the *User Modelling Agent* and *the Social Agent* as is explained in the following section.

5. Learning an User Model

If we want to be able to suggest interesting information for each particular user of the ACE System we must keep a user model for each one of them. The information used to obtain a user model are the votes the user has made for the documents presented previosly or the suggestions written by that user (that are considered as another document voted as interesting by the user). Votes are useful to get user feedback and make an adaptative model.

There are two ways to get the user vote for each document.

a) *Explicit voting*: the documents presented for the user can be rated. The user has two options: he can decide to make a positive vote or, otherwise he can make a negative vote. The user can decide also not to vote.
b) *Implicit voting*: In that case the System doesn´t wait for the user to issue a vote. ACE implements two techniques for implicit voting:
    1. when a person wastes too much time reading a document in the general list of documents, the system average the time wasted with the text length. If the value is higher than a threshold, a window display is presented to the user, asking if the document is interested for him or not. The system learns to modify the value of the threshold. If the user commonly rate positively the documents presented with implicit vote, it means that this technique works and for this reason the threshold can be lower. On the other way, if the user don´t vote or vote negatively the votes suggested by the System, the threshold can be higher.
    2. The users can write documents (suggestions)about a topic for the rest of users. These documents are used as a implicit positive vote for the user who made that vote.

Using implicit an explicit vote the system has for each user a list of the documents he voted for and against. This information is used to build the *User Model*. The user model representation choosed is a very used representation in the area, the *space vector model*. However we have to remark that ACE does not uses the classical implementation of the space vector model. Our vector is split in two subvectors, one of them keeps content-based information and the other keeping social-based information. The system uses two different techniques to learn each type of sub-vector.

5.1. Learning the content-based part of the user model

The aim of this process is to learn a basic user model using the textual content of the documents that the user has rated previously. The resulting vector is the best group of words than define the user's interests. A window of five documents is taken into account to build the user model. Each document is converted into a vector model representation. Each vector uses a set of words and their frequency for these last five model documents. The classical TF-IDF algorithm is used. Words present in one vector but not used in another document model are included in the five documents with zero frecuency. We build five new vector models with common words, to limit the length of the vectors to 100. To choose these 100 words an Information Gain (Quinlan,1986) measure was used, which is computed with the following formula:

$$E(W,S) = I(S) - \left[P(W = present)I(S_{w=present}) + P(W = \neg present)I(S_{w=\neg present})\right]$$

where

$$I(S) = \sum_{c \in \{+,-\}} - p(S_c)\log_2(p(S_c))$$

W is the actual word and *S* is the set of documents. *P(W=present)* is the probability that *W* was present in a page. $S_{w=present}$ is the set of pages where word W occurs one or more times. S_c are all pages of class *c*. There are two possible classes: the class of documents voted positively and the class of documents voted negatively.

For each word we keep the *tf* (term-frequency): the number of times the word appeared in the documents, and the *idf* (inverse-document-frequency): the number of documents where the same word occurred. The TF-IDF algoritm is applied then to those five vector model corresponding to the window of five documents. We didn´t want to lose the old user model but just to update it when the five new documents came. So, for this reason, we mantain a user model and we apply an algorithm to replace words in the old user model with words included in the new TF-IDF vector being built. The algorithm used is the following for each word included in the new vector TF-IDF:

IF the new word was included in the old model **THEN** the new *tf*, *idf* and *information gain* are averaged with the new values obtained.
IF the new word was not included in the old model but there is space enough to include that word **THEN** the new word is included with the *tf* ,*idf* and its *information gain*.
IF the new word was not included in the old model and there is not space enough **THEN** we replace the new word with another word that has a *tf-idf*Information Gain* higher than that word

5.2. Learning the social part of the user model

This procedure is used to take into account information coming from the activities of the virtual community of users that have probably common interests. The knowledge that these users have is used to make recommendations for other users which seem to have common interests.

The social user model is learnt again by using the *space vector model*. Each vector component is a user who has similar interests and a value of similarity. To discover similar interest between users we used the votes each user has made are used, similar users will be voted in the same way as in the case of documents.

To find the similarity between any two users, a set of documents who have been voted by the two users is built. Then by applying the following formula (Breese,1998):

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$

This formula measures the vector similarity between the two users, *a* is the actual user and *i* is one of the rest of users, and $V_{a,j}$ is the vote *a* has made to the document *j*. This formula computes the cosine of the angle formed by the two vectors. The squared terms in the denominator are used to normalize votes.

If the value obtained is higher than a fixed threshold, user *i* is included in the social part of the vector model for user *a*, and user *a* is included in the social vector of the user *i.*

6. Experimental results

A first implementation of ACE has been tested only on the movie reviews ans recommendations domain on a small comunity of users. The actual implementation does not make use of Forum contributions, this us left for further development. In this first test phases the goal was to check the learning algorithms used to build the user model. Some conditions were set for this experiment:

- Films were showed to the user in a random way. Therefore the films are voted in the same way.
- The users don´t see the system estimation for each film.

These two conditions guarantee that the Recomender System doesn't have any influence on users' purposes and their interests. This type of experiment setting point is explained in more detail for a similar system in (Balabanovic,1997a).

The method used to test the recomender system accuracy was organizing a controlled group of users that interacted with the system for a given time *s*. In order to have reliable estimation of the system's performance these users had to make a high number of votes in order to overcome the cold-start and non-cooperative community effects typical to the collaborative part of any mixed recommendation system. While the users voted, the system split the group of votes into two sets:

- The Vote Set I_a was used to build the content and social model of the user *a*, this set was used to create the examples for training ACE.
- The Vote Set *Pa* were the votes ACE had to predict for each user *a*. This became the test set.

Then, votes in the training set *Ia* were used to predict the votes for the user in *Pa*

By using the models built in this first phase, ACE showed films to each user giving an estimation of the interests of that film for that user. This is a probability of interest of a given user for a given film. So we adopted the following formula to test if the system was performing well (see (Breese, 1998) for further justification for this formula):

$$S_a = \frac{1}{m_a} \sum_{j \in P_a} \left| p_{a,j} - v_{a,j} \right|$$

P_{aj} is the probabilty estimated by the system for an user a and a film j, and V_{aj} is the vote the user a issued for film j. This operation is computed for all the films the user voted for, and is weighted by m_a, the total number of films the user voted. The output obtained measures the error in the System estimation. This formula is applied for all the users in the database and averaged over all of them.

Tests have been done with a set of 25 users, in a trial test that lasted one week. They voted on 40 items randomly chosen from a set of 214 items in the database.

The results are shown in Figure 4:

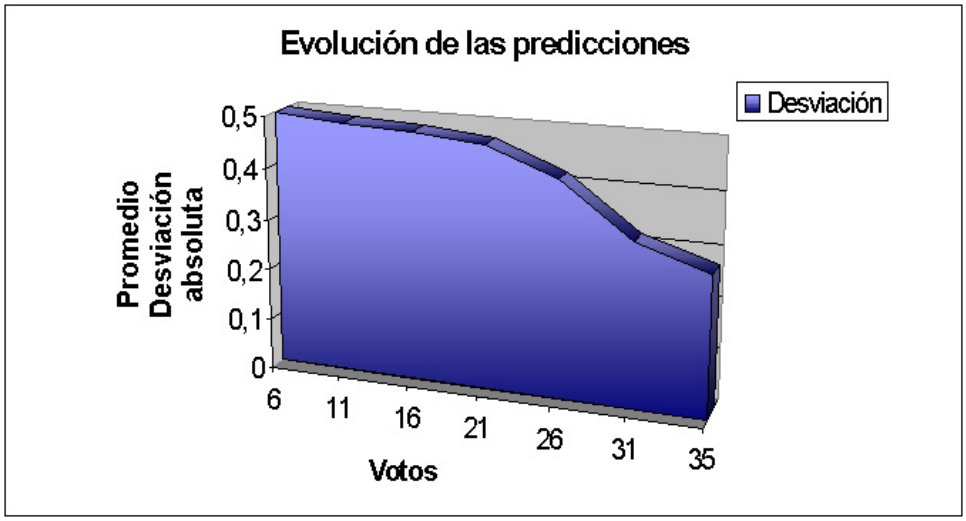

Figure 4: Evolution of the predictive capability in the ACE system.

The graphic shows in the horizontal axis all votes the users did. And in the vertical axis the absolute deviation computed for each user averaged and normalized for all the users. The graphic gives an indication of the system fit to user's interests. It can be seen that performance improves with the number of votes, which is the expected performance for this type of system. Average absolute deviation has a value of 0.27.[1]

7. Conclusions and future work

We think that experimental results show that ACE learns an user model which is useful to make accurate predictions. Probably, a wider virtual community could have allowed the system to home in faster on a lower error prediction rate. The way the users are clustered arround similar interestys is made not only by using just documents that have been of interest to them as other system combining cognitive and collaborative filtering do (Grasso, 1999), (Munro,1999) but also by taking into account explicit recommendations contributed by theusers. This is a novel approach to the best of our knowledge although some point of contacts may exist with the approach proposed by (Olsson,1998) and (Grasso,1999). It has

[1] each user has voted 40 items but the graphic only displays 35 votes due to the last 5 items are used as a test set for the vote 35.

been shown that the system, in fact, improves over time by resorting to that type of combination. In fact, it can be shown that it gives better prediction than by using only content-based or social-based recommendation.

One aspect of the architecture we want to remark is its *lightness*, in the sense previously discussed of imposing little burden to the server and clients by having always the minimum number of active agents at any time.

Another aspect worth mentioning is that ACE is devised so as to be as independent od the domain as possible. No special natural language technique reflecting any linguistic knowledge for the domain has been used as is in other systems. In our view, this approach also reduces the complexity of the calculations for finding similarities in content. Probably using linguistic knowledge about each type of domain (leisure, cinema, directories, etc.) would result in a more precise content retrieval behaviour but at the expense of (a) introducing a lot more of knowledge and processing (b) making the system less domain independent or, at least, language independent. Apparently in its present version there is a limitation in that last aspect in the sense that information about sich things as *stop words* or the method for fusing words is devised for the Spanish language. However, the overall modularization of the system ensures that this aspect can be asily replaced by information for other languages. Further details in that respect will be available in (Sangüesa,1999a) without having to introduce any complex natural language information.

Althought further abstraction is envisaged and this is a priority in our agenda for improvement, we feel that the present distribution of roles and tasks among agents is a first step in the right direction towards a generic architecture for recommendation (Sangüesa, 1999b). We would like to point that each aspect of the filtering and combining system has been made as modular as possible.

Further refinements that we are starting to address include the following ones:

- Test the Forum Spy Agent in a real setting.
- Study changing from binary voting ("I like"/"I don't like the item") to several degrees of voting, giving users the chance to express better their interests. We are trying to get a finer voting scale without introducing too much decision burden to the user (See (Balabanovic,1997b)(Balabanovic,1998) for an interesting discussion on that point).
- Make the system capable of evaluating intself, in order to know at each moment which of the two filtering approaches (the content-based or the social-based) is better, and weighting them automatically.

As we menctioned at the begining of this paper, ACE is now being integrated as a new service for the users of a resl electronic newspaper and it will be soon into another one. We want to take adventage of this new experiences to get new insights about aspectssuch as how the lightweight approach we used can be osed to face scaling un in the number of users.

References

(Balabanovic,1998) Balabanovic, M. An Interface for Learning Multi-topic User Profiles from Implicit Feedback. *Proceedings of the first WorkShop on Recommender Systems*, 1998, Technical Report WS-98-08, AAI-Press, pags. 6-10.

(Balabanovic,1997a) Balabanovic, M. An adaptative web page recomendation service. *Proceedings of the 1st International Conference on Autonomous Agents*, Marina del Rey, California (February 1997), pags. 378-385.

(Balabanovic,1997b) Marco Balabanovic, Yoav Shoham. Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM*, vol. 40, nº 3 (March 1997), pags. 66-72.

(Breese,1998) Breese, J. S.; Heckermann, D.; Kadie, C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, Madison, Winsconsin (July, 1998). Morgan Kaufmann Publisher.

(Delgado,1998) Delgado,J; Ishii, N.; and Ura, T. Intelligent Collaborative Information Retrieval. *6th Iberoamerican Conference on Artificial Intelligence (IBERAMIA 98)*, Lisbon, Postugal (October 1998).

(Grasso, 1999) Grasso, A. Mixing Cognitive and Collaborative Filtering. *Proceedings of the I3net Community of the Future Conference*. Sienna, Italy, October 1999.

(Lieberman,1995) H. Lieberman. Letizia; an agent that assists web browsing. *Proceedings of IJCAI-95*. AAAI Press.

(Maes,1994) Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM* vol. 37, n. 7, pags. 31-40.

(Miyahara,1998) K. Miyahara, T. Okamoto. *Collaborative Information Filtering in cooperative communities*. Journal of computer Assisted Learning 14 (1998), pags. 100-109.

(Mundher, 1998) Mundher,M; and Sen, S. Use of voting schemes to tradeoff user preferences. *Proceedings of the first Workshop on Recommender Systems*, 1998, Technical Report WS-98-08, AAAI-Press, pags. 75-76.

(Munro,1999) Munro, A. J.; Hook, K. And Benyo, D. (eds) (1999) *Social Navigation of Information Space*. CSCW Series. Springer-Verlag, London.

(Olsson,1998) Olsson, T. Decentralised Filtering based on Trust. *Proceedings of the first Workshop on Recommender Systems*, 1998, Tehnical Report WS-98-08, AAAI-Press, pags 84-86.

(Pazzani,1996) Pazzani, M; Muramatsu, M and Billsus, D. Syskill & Webert: identifiying interes web sites. *Proceedings of the Thirteenth National Conference on Artificial Intelligence.* Portland Oregon (1996).

(Quinlan,1986) Quinlan, J. R. Induction of decision trees. *Machine Learning*, 1 (1986), 81-106

(Resnick,1994) Resnick, P.; Iancouvou,N; Sushack, M.; Begrstrom, P. And Riedl,J. Grouplens: An open architecture for collaborative filtering of NetNews. *Proceeedings of the CSCW 1994 Conference*.

(Rucker,1997) Rucker, J; Polanco, M.J. Siteseer: Personalized Navigation for the Web. *Communications of the ACM*, vol. 40, n. 3 (March 1997), pags. 73-75.

(Sangüesa, 1999a) Sangüesa,R.; Vázquez-Salceda, J. And Vázquez-Huerga, A. The ACE Recommender System. Universitat Politècnica de *Catalunya,* Departament de Llenguatges i Sistemes Informàtics. Tech-Report (forthcoming).Barcelona, Spain.

(Sangüesa, 1999b) Sangüesa,R.; Abad,A.; Santanach, F.;Vázquez-Salceda, J. And Vázquez-Huerga. MANDRAS: towards a generic framework for Recommender Systems.(forthcoming) Universitat Politècnica de *Catalunya,* Departament de Llenguatges i Sistemes Informàtics. Tech-Report (forthcoming).Barcelona, Spain.

(Terveen,1997) Terveen, L; Hill, W.; Amento, B; McDonald, D. and Creeter,J. Phoaks: a System for Sharing Recommendations. *Communications of the ACM*, vol. 40, n. 3 (March 1997), pags. 59-62.