

Intelligent Collaborative Information Retrieval

Actively Learning to Classify and Recommend Documents

AUTHORS AND AFFILIATION

Abstract. Next generation of intelligent information systems will rely on cooperative agents for playing a fundamental role in actively searching and finding relevant information on behalf of their users in complex and open environments, such as the Internet. Whereas relevant can be defined solely for a specific user, and under the context of a particular domain or topic. On the other hand shared “social” information can be used to improve the task of retrieving relevant information, and for refining each agent’s particular knowledge. In this paper, we combine both approaches developing a new content-based filtering technique for learning up-to-date users’ profile that serves as basis for a novel collaborative information-filtering algorithm. We demonstrate our approach through a system called *RAAP* (Research Assistant Agent Project) devoted to support collaborative research by classifying domain specific information, retrieved from the Web, and recommending these “bookmarks” to other researcher with similar research interests.

Keywords: Cooperative Information Systems, Software Agents, Collaborative Information Retrieval, Social Filtering, and On-line machine learning algorithms.

1 Introduction

Undoubtedly, in the next generation of intelligent information systems, cooperative information agents will play a fundamental role in actively searching and finding relevant information on behalf of their users in complex and open environments, such as the Internet. Whereas *relevant* can be defined solely for a specific user, and under the context of a particular domain or topic. Because of this, the development of intelligent, personalized, content-based, *document classification* systems is becoming more and more attractive now days. On the other hand, learning profiles that represent the user’s interests within a particular domain, later used for *content-based filtering*, has been shown to be a challenging task. Specially because, depending on the user, the relevant attributes for each class *change in time*. This makes the problem even not suitable for traditional, fixed-attribute machine learning algorithms.

Documents, as well as user’s profiles, are commonly represented as keyword vectors¹ in order to be compared or learned. With the huge variety words used in natural language, we find ourselves with a noisy space that has extremely high dimensionality (10^4 - 10^7 features). On the other hand, for a particular user, it is reasonable to think that processing a set of correctly classified relevant and irrelevant documents from a certain domain of interest, may lead to identify and isolate the set of relevant keywords for that domain. Later on, these keywords or features can be used for discriminating documents belonging to that category from the others. Thus, these user-domain specific sets of relevant features, that we call *prototypes*, may be used to learn to classify documents. It is interesting enough to say that these prototypes may change over time, as the user develops a *particular view* for each class. This problem of personalized learning of text classification, is in fact, similar to the one of on-line learning, from examples, when the number of relevant attributes is much less than the total number of attributes, and the concept function changes over time, as described in [1].

On the other hand, cooperative multi-agent systems have implicitly shared “social” information, which can be potentially used to improve the task of retrieving relevant information, as well as refining each agent’s particular knowledge. Using this fact, a number of “word-of-mouth” *collaborative information filtering* systems, have been implemented as to *recommend* to the user what is probably to be interesting for him or her. This is done based on the ratings that other *correlated users* have assign to the same object. Usually this idea has been developed for specific domains, like “Music” or “Films”

¹ This is called the Vector Model and has been widely used in Information Retrieval (IR) and AI.

² Also called Social Filtering [5]

as in Firefly^a and Movie Critic^b, or for introducing people (matchmaking) as in Yenta[2]. A mayor drawback of these systems is that some of them completely deny any information that can be extracted from the content. This can be somehow convenient for domains that are hard to analyze in terms of content (such as entertainment), but definitely not suitable for textual content-driven environments such as the World Wide Web (WWW). Besides, these systems usually demand from the user, a direct intervention for both classifying and/or rating information.

In this paper we describe a multi-agent system called *RAAP* (Research Assistant Agent Project) that intends to bring together the best of both worlds – Content-based and Collaborative Information Filtering. In *RAAP*, personal agents both helps the user (a researcher) to classify domain specific information found in the WWW, and also recommends these URLs to other researchers with similar interests. This eventually brings benefits for the users as they are recommended only peer-reviewed documents, especially if they perform information and knowledge intensive collaborative work, such as scientific research. Tested in our research laboratory, this system was developed, not only to prove that content-based and collaborative filtering techniques can be combined, but that they are both necessary to improve the results of the overall learning process. In other words, our agents learn both from local and shared resources. Its strategy includes a “*Classification-Recommendation*” feedback process in which the agent suggests the classification of a saved document, and documents’ ratings are extracted from user’s actions such as accepting/rejecting a recommendation, or reviewing the agent’s suggestion. Thus, the better that it classifies the better that it recommends. In the same way, as more users use the system, either collecting information or receiving recommendations from others, the more is the chance to update the user’s prototype for each domain of study, giving the agent a chance for being more accurate in the next classification step.

We introduce a new text classification algorithm, somewhat similar to the version of Rocchio’s algorithm adapted to text classification [3]. A major difference is that, due to the active nature of our system, indexing terms are domain specific and can be added or replaced in the prototype. Also, we perform a special type of relevance feedback [4] every time a document is added to the database, instead of learning from training sets with a fix set of indexing terms. This is made possible because the user implicitly rates documents as interesting or not within a certain class, as shown later in the description of the system

We used our classification algorithm, combined with *Expected Information Gain*, to do *relevant feature selection* for creating a positive prototype of indexing terms for each class. In this paper we will be using phrases, words and stemmed words as indexing terms, where the terms can be drawn from a manually build keyword list or from the documents that the user has saved.

As for the collaborative filtering, a combination of document clustering with the traditional Pearson- r algorithm was implemented. The *user vs. category matrix*, needed for calculating the correlation between a pair of users, is automatically constructed by counting, for that user, the number of times a document is successfully classified into a certain class. We also use a *user confidence* factor that serves as feedback for the recommendation process.

Finally, we give some experimental results and analysis. We also discuss about the similarity of our approach with the on-line learning algorithms in a Mistake Bounded model, widely discussed in Computational Learning Theory, and compare these techniques with traditional IR.

2 Description of RAAP

The Research Assistant Agent Project (*RAAP*) is a system developed with the purpose of assisting the user in the task of classifying documents (bookmarks), retrieved from the World Wide Web, and automatically recommending them to other users of the system, with similar interests. In order to evaluate the system, we narrowed our objectives to supporting a specific type of activity, such as *Research & Development (R&D)* for a given domain. In our experiment, tests were conducted using *RAAP* for supporting research in the *Computer Science* domain.

^a <http://www.agents-inc.com>

^b <http://www.moviecritic.com>

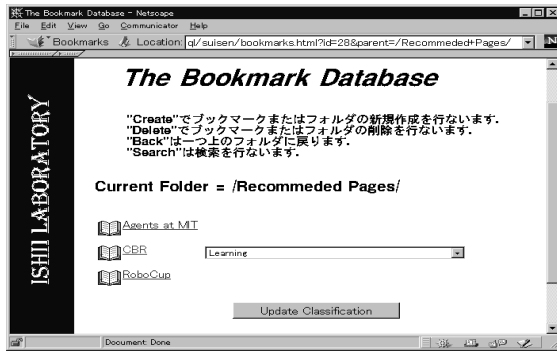


Figure 1 Agent's Suggestion

RAAP consists of a bookmark database with a particular view for each user, as well as a software agent that monitors the user's actions. Once the user has registered an "interesting" page, his agent suggests a classification among some predefined categories, based on the document's content and the user's profiles (Fig. 1). Then the user has the opportunity to reconfirm the suggestion or to change classification into one, that he or she considers best for the given document as shown in Fig.2.

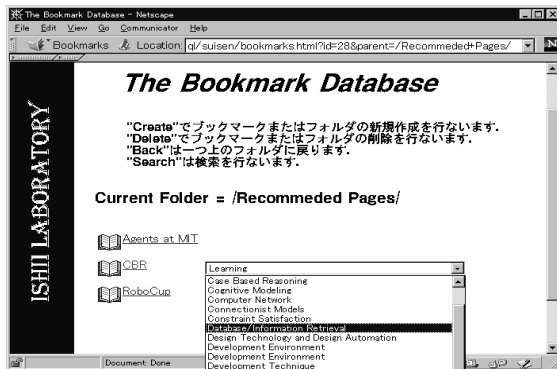


Figure 2 Classification

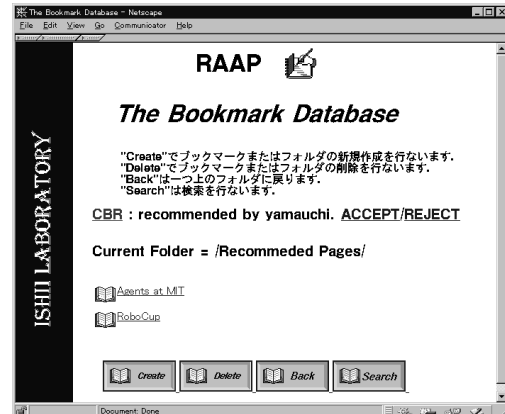


Figure 3 Recommendation

In parallel, the agent checks for newly classified bookmarks, and recommend these to other users that can either accept correct them when they eventually login the system and are notified of such recommendation, as illustrated in Fig.3.

The first time of registration into the system, the user is asked to select the research areas of interest by the time of registration. This information is used to build the initial profile of that user for each class. The agent updates the user's profile for a specific class every time certain number k of documents are successfully classified into it. In that way, RAAP only uses up-to-date profiles for classification, reflecting always the latest interests of the user. During this process the agent learns how to improve its classification and narrow the scope of the people to whom it recommends in a way we shall explain in the following sections.

A flowchart for RAAP's front-end is given in Fig. 4

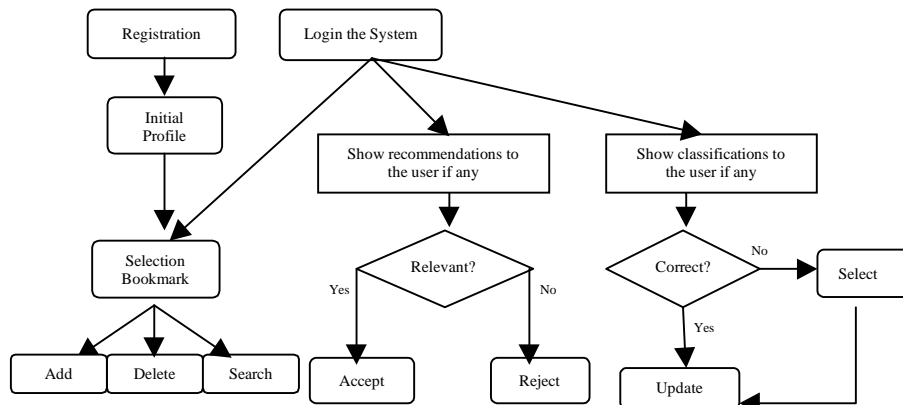


Figure 4 RAAP's Front-End Flowchart

3 Learning to Classify

In the flavor of traditional IR, we must state our classification problem as a combination of two: *Text Categorization* and *Relevance Feedback for Retrieval*. Text Categorization can be defined as assigning predefined categories to free text documents. On the other hand, Relevance Feedback is an application of machine learning to ranked retrieval systems. Such systems begin with an initial query, usually stated by the user, to construct a classifier used to rank documents, distinguishing the relevant from non-relevant ones. Then the user has the option of labeling some of the top ranked documents, that later are given to a supervised learning procedure, which uses them, along with the original request, to produce a new classifier that usually improves the effectiveness of the retrieval.

Text Categorization and Retrieval, although very related, has been treated by the IR community quite separately. Some authors like T. Tokunaga [6] have the opinion that the categorization is more suitable for evaluating measures than retrieval, because it avoids the problem of defining *relevance* of documents that differs from user to user. In the same line of thought, D Lewis states in [7] that for text classification, features must be useful across a wide range of classes, since features typically cannot be created for each user of category. To come across these barriers is precisely what we're aiming at.

Perhaps *information routing* or *filtering*, most recently applied in Internet oriented applications (e.g. directing electronic mail the appropriate folder or recipient), is the concept most suited for our task, in the sense of having both, properties for categorization and retrieval.

3.1 Building the Prototypes

In our system, a **document** belonging to a user u_i is a finite list of terms, resulting of filtering out a stoplist of common English words, from the original document fetched from the WWW using its URL. These documents are either explicitly "bookmarked" or "accepted" by the user. In case of being "rejected", sometimes it can also be "saved" by the user's agent as negative example. Under this notion, and from now on, we will be using the words "document", "page" and "bookmark" without distinction. We will also be using "class", "category" and "topic" in the same way.

Because we are trying to build a document classification system that learns, it is of our interest to keep track of the positive and negative examples among the documents that have been already classified. In RAAP, **positive examples** for a specific user u_i and for a class c_j , are the documents *explicitly registered* or *accepted* by u_i and classified into c_j . Note that accepting a recommended bookmark is just a special case of registering it. On the other hand, **negative examples** are either *deleted* bookmarks, *misclassified* bookmarks or *rejected* bookmarks that happens to be classified into c_j . In the case of rejected bookmarks the document is only classified by the agent – we don't ask the user to reconfirm the classification for rejected bookmarks. In this sense, and as measure of precaution, we only take rejected bookmarks that fall into a class in which the user has at least one bookmark correctly classified (a class in which the user has shown some interest).

Let $C_{i,j}^+$ be the set of all documents classified as positive examples for user u_i and class c_j
 Let $C_{i,j}^-$ be the set of all documents classified as negative examples for user u_i and class c_j

Now for a given user, we want to build a classifier Q for each category, as a list of terms, in order to apply the dot product similarity measure (Eq. 1), widely used in IR for the purpose of ranking a document D respect a given query. The classifier most similar to the document would indicate candidate class.

$$(1) \quad \boxed{\text{sim}(Q, D) = \sum_{\tau \in Q} w(\tau, Q) w(\tau, D)}$$

For term weighting, we chose TFIDF (Eq. 1.1), as it is one of the most successful and well-tested term weighting schemes in IR. It consists of the product of $tf_{\tau,d}$, the term-frequency (TF) of term τ in the document d , by $idf_{\tau} = \log_2(N/dt_{\tau}) + 1$, the inverse document frequency (IDF), where N is the total number of documents of collection and dt_{τ} is the total number of occurrences of τ in the collection. Note that we maintain a different collection for each user.

$$(1.1) \quad w(\tau, d) = \frac{tf_{\tau, d} \left[\log_2 \left(\frac{N}{df_{\tau}} \right) + 1 \right]}{\sqrt{\sum_i \left(tf_{\tau_i, d} \left[\log_2 \left(\frac{N}{df_{\tau_i}} \right) + 1 \right] \right)^2}}$$

Our problem now consists in learning accurate classifiers that reflects the user's interests both in terms of *relevance feedback* and *feature selection*.

3.1.1 Relevance Feedback

Given the unique terms in Q , $P = \{\tau_1, \tau_2, \tau_3, \dots, \tau_r\}$, named P as for Prototype, it is easy to see that both Q and D can be represented as numerical vectors, containing the respective weights of these terms in Q and D , denoted as \vec{Q} and \vec{D} respectively. Their dot product³ is, in fact, the similarity measure explained before. We can also express Q and D as numerical vectors, containing the term frequency of each elements of P within them, denoted respectively, as:

$$Tf(Q) = \langle tf_{\tau_1, Q}, tf_{\tau_2, Q}, tf_{\tau_3, Q}, \dots, tf_{\tau_r, Q} \rangle \quad \text{and} \quad Tf(D) = \langle tf_{\tau_1, D}, tf_{\tau_2, D}, tf_{\tau_3, D}, \dots, tf_{\tau_r, D} \rangle$$

We can now describe mathematically a very simple approach to the process of relevance feedback as:

$$(2) \quad Tf(Q^{i+1}) = Tf(Q^i) + \alpha Tf(D) \quad \text{Where } \alpha = 1 \text{ if } D \in C_{i,j}^+ \\ \alpha = -1 \text{ if } D \in C_{i,j}^-$$

And then, recalculate \vec{Q}^{i+1} based on the values of $Tf(Q^{i+1})$.

Another approach found in the literature, is the Rocchio's Algorithm [3]:

$$(2.1) \quad \vec{Q} = \frac{\beta}{|C_{i,j}^+|} \sum_{j \in C_{i,j}^+} \vec{D}_j - \frac{\gamma}{|C_{i,j}^-|} \sum_{j \in C_{i,j}^-} \vec{D}_j \quad \text{Note that } |C_{i,j}^+| \neq |C_{i,j}^-| \\ \text{Usual values are: } \beta=4 \text{ and } \gamma=4\beta$$

The basic problem of these algorithms and the main reason why we couldn't use them for our system is that *they do not take into account the possibility that the dimension of the vectors \vec{Q} and \vec{D} may change in time*. In other words, that unique terms listed in P can be added or deleted in \vec{Q} order to reflect the user's current interests (feature selection).

Perhaps Rocchio's algorithm can be adapted to be recalculated each time a unique term is added or deleted to P , but the computational cost would be very high. This is without mentioning the size of the complement of the positive examples, used to calculate the negative part of the formula.

Instead, we propose a new algorithm for incrementally building \vec{Q} in order to reflect effectively the user's current interests. We shall give at first a couple of useful definitions:

Definition 1: We define *positive prototype* for a class c_i , user u_j at time t ,

$$P_{i,j}^{(t)+} = \{\tau_1, \tau_2, \tau_3, \dots, \tau_r\}$$

as a finite set of unique indexing terms, chosen to be relevant for c_i , up to time t (see feature selection).

Definition 2: We define *negative prototype* for a class c_i , user u_j at time t ,

$$P_{i,j}^{(t)-} \subseteq P_{i,j}^{(t)+} / \forall \tau \in P_{i,j}^{(t)-}, \exists d \in C_{i,j}^- \wedge (tf_{\tau, d} > 0)$$

as a subset of the corresponding positive prototype, whereas each one of its elements can be found at least once in the set of documents classified as negative examples for class c_i .

³ The dot-product between the numerical vectors of Q and D is denoted as $(\vec{Q} \circ \vec{D})$.

Now we construct the vector of our positive prototype as follows,

At time $t + 1$

if $(P_{i,j}^{(t+1)+} = P_{i,j}^{(t)+}) \{$

$Tf(Q_{i,j}^{(t+1)+}) = Tf(Q_{i,j}^{(t)+}) + Tf(D_{i,j}^{(t)+})$

Update $\vec{Q}_{i,j}^{(t+1)+}$, based on $Tf(Q_{i,j}^{(t+1)+})$

$\} \text{ else } \{$

forall $\tau \in P_{i,j}^{(t+1)+} - P_{i,j}^{(t)+}$ do{

calculate $w(\tau, d)$ for the n - most recently processed

documents $\in C_{i,j}^+$ and update these values in $\vec{Q}_{i,j}^{(t+1)+}$

}}

Where n is the number of documents used as basis for feature selection.

This algorithm is to be applied in the same way for the negative prototype. We can now re-define the similarity measure between a class c_j and a document D as:

$$(3) \quad \boxed{sim_i^t(c_j, D) = (\vec{Q}_{i,j}^{(t)+} \circ \vec{D}_{i,j}^{(t)}) - (\vec{Q}_{i,j}^{(t)-} \circ \vec{D}_{i,j}^{(t)})}$$

Expressing this equation in words, we should say that for a given user u_p , the similarity between a class c_j and an incoming document D at time t , is equal to the similarity of D , with respect of the classifier of the corresponding positive prototype minus the similarity of D , with respect of the classifier of the corresponding negative prototype. This intuitively says that a document is similar to a class if its similar to the class positive prototype and not similar the class negative prototype. It is important to say that the initial positive prototype for each class is a list of selected core keywords from that domain that were integrated into the system to provide at least an initial classification.

Finally, we use a heuristic for RAAP's ranking. This heuristics states that *it is more likely that a new document is to be classified into a class in which the user has shown some interest before*. We chose the class with the highest ranking among these.

3.1.2 Feature Selection

Automatic feature selection methods include the removal of non-informative terms according the corpus statistics. In comparative studies on feature selection for text categorization [8], information gain (IG) is shown to be one of the most effective for this task. Information gain is frequently employed as a term-goodness criterion in the field of machine learning [9,10] .

Given all these interesting properties we decided to use IG for selecting informative terms from the corpus. We re-define the expected information gain that the presence or absence of a term τ gives toward the classification of a set of pages (S), given a class c_j and for a particular user u_i as:

$$(4) \quad \boxed{\begin{aligned} E_{i,j}(\tau, S) &= I(S) - [P(\tau = present)I(S_{\tau=present}) + P(\tau = absent)I(S_{\tau=absent})] \\ \text{where,} \\ I_{i,j}(S) &= \sum_{c \in \{C_{i,j}^+, C_{i,j}^-\}} -P(S_c) \log_2(p(S_c)) \end{aligned}}$$

In Eq. 4, $P(\tau=present)$ is the probability that τ is present on a page, and $(S_{\tau=present})$ is the set of pages that contain at least one occurrence of τ and S_c are the pages belonging to class c .

Using this approach, in RAAP, the user's agent finds the k most informative words from the set S of the n most recently classified documents. As in Syskill & Webert [11] we chose $k=128$ and arbitrary selected $n=3$ for our experiments.

Out of the selected 128 terms, 28 are to be fixed, as they constitute the core list of keywords or a basic ontology for a topic, given for that class as an initial classifier. Within the rest 100 words, we adopt the following scheme for adding or replacing them in the positive prototype:

1. Perform *stemming* [12] over the most informative in order to create the list of terms.
2. Replace only the terms that are in the prototype but not in the list of the most informative terms.
3. As shown in the algorithm for constructing the classifier, update the weights of the newly added or replaced terms with respect of the n documents processed by the agent for this purpose.

We conclude this section saying that even if IG is a computationally expensive process, in RAAP this is drastically improved both by having n low and only updating the weights for the selected terms only with respect of these documents. We also provide a mechanism in which the “memories” of the terms that repeat in time, are left intact, given that their accumulated weight and information value is high.

4 Learning to Recommend

For the purpose of learning to who recommend a page saved by the user, the agent counts with two matrixes. They are the *user vs. category* matrix M_{mn} and the *user's confidence* factor, where m is the number of users in the system and n the number of categories. The first one is automatically constructed by counting, for that user, the number of times a document is successfully classified into a certain class. During the initial registration in RAAP, the matrix is initialized to one for the classes that the user has shown interest.

	John	Kato	Ishii	Joacc	Gina
Information Retrieval	7	7	3	4	7
Temporal Reasoning	4	3	2	0	4
CBR	1	3	7	2	3
Distributed AI	2	5	3	7	2

Table 1User-Category Matrix

The first idea was the user-category matrix to calculate the correlation between a user u_x and the rest, using the Pearson-r algorithm (Eq. 5). Then recommend the newly classified bookmarks to those with highest correlation.

$$(5) \quad \text{correl}(u_x, u_r) = \frac{\sum_{i=1}^n u_{x,i} - \bar{u}_{x,i})(u_{r,i} - \bar{u}_{r,i})}{\sqrt{\sum_{i=1}^n (u_{x,i} - \bar{u}_{x,i})^2 \sum_{i=1}^n (u_{r,i} - \bar{u}_{r,i})^2}}$$

$$\text{where } \bar{u}_{j,i} = \frac{\sum_{i=1}^n u_{j,i}}{n}; j \in \{x, r\}$$

One problem with this approach, is that the correlation is only calculated as an average criterion of likeness between two users, regardless of the relations between the topics. That is, if the agent decides to recommend a bookmark classified into a class x , but it happens to be that its user is highly correlated to another user based on the values in the matrix respect other classes, then the bookmark would be recommended to that user anyway. These classes can be presumably unrelated to the class of the bookmark, which is undesirable since we only want the agents to recommend bookmarks to people that have interest in the topics to which it belongs or in related ones. What we really want is to give more weight in the correlation between users to those classes more related to the class of the bookmark that is going to be recommended. For this reason we introduce the concept of similarity between two classes for a user u_i at time t , as the dice coefficient between the positive prototypes of the classes.

$$(6) \quad \text{rel}_i^t(c_m, c_n) = \frac{2 \times |P_{i,m}^{(t)+} \cap P_{i,n}^{(t)+}|}{|P_{i,m}^{(t)+}| + |P_{i,n}^{(t)+}|}$$

Where $|A \cap B|$ is the number of common terms, and $|A|$ is the number of terms in A.

Given the class of the bookmark c_j , the class similarity vector is defined as:

$$(6.1) \quad \vec{R}_j = \langle rel_i^t(c_j, c_1), rel_i^t(c_j, c_2), \dots, rel_i^t(c_j, c_n) \rangle$$

where $n = \# \text{ of classes}$

Now we multiply this vector by the user-category matrix obtaining a weighted, user-category matrix.

$$(7) \quad WM = \vec{R}_j \times M$$

Using this new User-Category matrix similar to the one shown in Table 1, but with modified (weighted) values, we proceed to calculate the weight between the subject user u_x (who recommends) and the others u_i (candidates to receive the recommendation) as the correlation between them multiplied by the confidence factor between them.

$$(8) \quad Weight(u_x, u_i) = correl(u_x, u_i) * confidence(u_x, u_i)$$

In Eq. 8, the confidence factor of user u_i with respect u_x , is a function with a range between 0.1 and 1. It returns 1 for all new users respect others, and it decreased or increased by a factor of 0.01 every time a bookmark recommended by user u_x is accepted or rejected by u_i respectively. Note that $confidence(u_x, u_i) \neq confidence(u_i, u_x)$. This means that the confidence is not bi-directional, but differs for every combination of pair of users.

For deciding to who recommend we used a threshold of 0.5 for the minimum weight, as well as recommending to at most to $f(n) = \lceil 1/(n-264) \rceil + 5$ number of users, where n is the total number of users in the system. We use $f(n)$ to maintain a reasonable proportion of the number of users that are selected as recipients for the recommendation, respect the number of registered users that at some moment of time can be huge.

To avoid circular references in the recommendation chain, the agents verify that the recommended document is not already registered in the target's database.

5 Experimental Results

In order to evaluate the system we set up an experiment with 9 users in our laboratory. They were asked to interact freely with RAAP during one week, registering home pages only with content relevant to their current research interests. An update of the users' prototype for a certain class was executed every 3 bookmarks classified into that class.

- **Bookmark Classification**

A number of 72 bookmarks were registered and classified by the agents/users into a total of 9 different classes. We give the results of the classification in Table 2.

User ID	Correct	Incorrect	Total	Accuracy (%)	# of Updates
23	4	3	7	57.1	0
25	3	0	3	100.0	0
26	3	7	10	30.0	1
27	12	6	18	66.7	4
28	7	10	17	41.2	2
30	0	2	2	0.0	0
31	4	2	6	66.7	0
32	1	0	1	100.0	0
34	2	6	8	25.0	0

Table 2 Classification Accuracy

The first thing we should notice is that results were quite different for each user. The usage of the system also varied quite a lot. In this experiment only 2 users (ID=27,28) received 2 or more updates of there profiles, having also the biggest amount of registered bookmarks. Given these reasons, let's study these cases more in detail.

For ID=27, an average of 66.7% of accuracy was achieved while 4 profile updates occurred. Before the first update, an error rate of 80% was observed. Afterwards, the error rate pattern was 0%, 50%, 0%, 0% for each respective update. During transition from 0% to 50% the user/agent classified some incoming bookmarks into topics other than those which prototype were updated, thus increasing the error rate. The other times bookmarks were classified mainly into 2 different topics that did received updates.

On the other hand, ID=28 had a somewhat low average of 41.2% of accuracy. Nevertheless, this user had a high error rate of 85.7% before the first update, decreasing to 50% the first time and 42.8% the second time.

We can mention also the case of ID=26, that decreased from an error rate of 40% rate to 33% after the single update was performed. For the cases where there was no updating at all, an average of more than 55% of accuracy was obtained.

- **Bookmark Recommendation**

As we can see in Table 3, the overall acceptance rate was quite high for the majority of the users. In total, there were 74 recommendations, 52 (70%) of which were accepted. Comparing this with the results in bookmark classification, it is easy to realize that, in general, the acceptance rate was lower for those users that didn't receive any update in their profile such as ID=23, 34.

User ID	Accept	Reject	Total	Accuracy (%)
23	3	8	11	27.3
25	6	0	6	100.0
26	9	1	10	90.0
27	13	2	15	86.7
28	7	3	10	70.0
30	8	1	9	88.9
31	2	1	3	66.7
32	4	5	9	44.4
34	0	1	1	0.0

Table 3 Recommendation Acceptance Rate

Out of 70 registered bookmarks only 42 were unique, which means that 30 of them (41.7% of all the bookmarks) actually came from recommendations. This indicates that intelligent information sharing and collaborative filtering occurred in high degree.

Finally, for a more general evaluation, we must point out that for *RAAP* there was no training data available, other than the bookmarks itself. The learning was performed on-line and incrementally throughout the interaction with the system. Thus, *RAAP* cannot be directly compared with traditional of-line, text categorization algorithms. In spite of this, our classification algorithm showed to be satisfactory, to the extent that in the majority of the cases the user didn't even need to rectify the agent's suggestion. The relatively high percentage of accepted recommendations showed that its not only feasible to support collaborated filtering on content-based filtering, but also that with the increase of *relevant* data as product of the recommendations, the classification accuracy is very likely to improve.

6 Discussion

Machine learning for text classification offers a variety of enhancements comparing it with traditional IR. The exponential growth of computing and networking has led to the development of new applications of text classification and retrieval, as well as more demand on its effectiveness. Active (on-line) and passive (of-line) learning are to be combined in order to achieve better results in understanding the complex semantics of natural language.

On the other hand computational agents promise to be the solution for managing, filtering, and sharing information on behalves of their users in open environments such as the Internet. Learning in social structures such as Multi-Agent Systems gives the opportunity to take advantage of common knowledge for cooperative problem solving, ignored some time before in traditional AI.

The combinations of techniques of both areas not only offer us interesting challenges but also introduce new open questions. For instance, how should we evaluate systems like *RAAP*? Are the traditional methods of evaluations in text retrieval such as *precision* and *recall* applicable to

personalized and collaborative information retrieval systems? How do we learn when the concept function changes in time? If an agent cannot classify a document, can it ask another agent to do this task (cooperation & coordination)? These were some questions just to mention a few.

There are many types of applications in which we could apply the techniques introduced in this paper. Besides *RAAP*, intended for collaborative information retrieval, we can think now of an application for *collaborative paper review*. In scientific journals and high level conferences, there is always the problem of deciding what experts, in the different sub-fields of the general domain, are going to review a specific paper, based on its content. On the other hand, papers sometimes belong to several of these sub-fields. Having papers semi-automatically classified and sent (as a recommendation) to the corresponding experts, that can, of course, accept or reject a paper for review, is the aim of such application. Note that it should be enough for the experts to register in the system.

Finally, we must point out the recent advances in Computational Learning Theory (COLT), in the Mistake Bounded model, that may represent the key for solving hard problems such as learning target functions that change in time and the selection of relevant features [14]. This is the case of learning users' profiles. Some recent developments in active learning and multiplicative update algorithms [15], as well as in infinite attribute model, are giving some theoretical support to what has been until now just empirical observations.

7 Conclusion and Future Work

The contributions of this paper are threefold:

- 1) We proposed the combination of content-based information filtering with collaborative filtering as the basis for multi-agent collaborative information retrieval. For such purpose the system *RAAP* was explained in detail.
- 2) A new algorithm for active learning of user's profile and text categorization was introduced.
- 3) We proposed a new algorithm for collaborative information filtering in which not only the correlation between users and also the similarity between topics is taken into account.

Some experimental results that support our approach were also presented. As future work we are looking forward test *RAAP* in broader environments and to compare it with other similar systems, as well as improve the efficiency of both the classification and recommendation processes.

Acknowledgements

This work was supported by the Hori Foundation for the Promotion of Information Science and the Japanese Ministry of Science, Culture and Education (Monbusho).

References

1. Blum, A., "On-line Algorithms in Machine Learning" (a survey). Dagstuhl workshop on On-Line algorithms (June 1996).
2. Foner, L., "A Multi-Agent Referral System for Matchmaking", in *Proceedings of the First International Conference on the Practical Applications of Intelligent Agent Technology (PAAM'96)*, London (April 1996).
3. Buckley, C., Salton, G., et.al.: "The effect of adding relevance information in a relevance feedback environment". In *Proceedings of the 17th International ACM/SIGIR Conference on Research and Development in Information Retrieval* (1994).
4. Salton, G., Buckley, C., "Improving retrieval performance by relevance feedback", *Journal of the American Society for Information Science*, 41, 288-297 (1990).
5. Maes, P., "Agents that Reduce Work and Information Overload", *Comm ACM*, 37, No7 (1994).
6. Tokunaga, T., Iwayama M.: "Text categorization based on weighted inverse document frequency", Technical Report 94-TR0001, Department of Computer Science, Tokyo Institute of Technology (March 1994).
7. Lewis, D.: "Challenges in machine learning for text classification", in *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, P1. New York (1996). ACM
8. Yang, Y., Pedersen, J. "Feature selection in statistical learning of text categorization", *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, (1997).
9. Quinlan, J.R., "Induction of decision trees". *Machine Learning*, 1(1):81-106 (1986)

10. Mitchell T., "Machine Learning" McGraw Hill, 1996
11. Pazzani, M., Muramatsu, J., and Billsus, D., "Syskill & Webert: Identifying interesting websites", In *Proceedings of the American National Conference on Artificial Intelligence (AAAI'96)*, Portland, OR. (1996)
12. Frakes, W., Baeza-Yates, R.: "Information Retrieval: Data Structure & Algorithms" Printice Hall, NJ (1992)
13. Blum, A., Langley, P.: "Selection of Relevant Features and Examples in Machine Learning", *Artificial Intelligence*, 97:245—277, (1997)
14. Blum, A.: "Empirical support for Winnow and Weighted-Majority based algorithm: results on a calendar scheduling domain", *Machine Learning* 26:5—23. (1997)