Content + Collaboration = Recommendation

Joaquín DELGADO and Naohiro ISHII

Department of Intelligence & Computer Science
Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya 466 JAPAN
jdelgado@ics.nitech.ac.jp

Abstract

Building **recommender systems**, that filters and suggests *relevant* and *personalized* information to its user, has become a practical challenge for researchers in AI. On the other hand, the relevance of such information is a user-dependent notion, defined within the scope or context of a particular domain or topic. Previous work, mainly in IR, focuses on the analysis of the **content** by means of keyword-based metrics. Some recent algorithms apply social or **collaborative** information filtering 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) implemented to support collaborative research by classifying domain specific information, retrieved from the Web, and recommending these "bookmarks" to other researcher with similar research interests.

Introduction

Undoubtedly, in the next generation of intelligent information systems, collaborative 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 vectors 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 (Blum 1996).

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 interesting for him or her. This is done based on the ratings that other correlated users have assign to the objects being evaluated. Usually, this idea has been developed for specific domains, like "Music" or "Films" as in *Firefly* (http://www.firefly.net) or for introducing people (matchmaking) as in *Referral Web* (Kautz 1997). Remarkably, some of these systems completely deny that 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 mainly textual and 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, thus "manually" constructing users' profiles.

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.

System Description

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 these 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 and Development (R&D) for a given domain. In our experiment, tests were conducted using *RAAP* for supporting research in the Computer Science domain.

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).

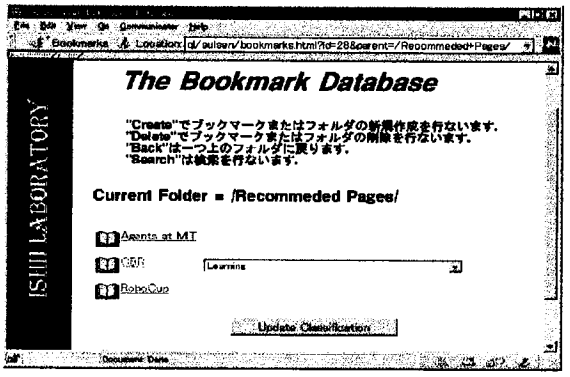

Figure 1 Agent's Suggestion

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.

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

During the first time registration into the system, the user is asked to select the research areas of interest. This information is used to build the initial profile of the user for each class. The agent updates the user's profile for a specific class every *k*-number 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.

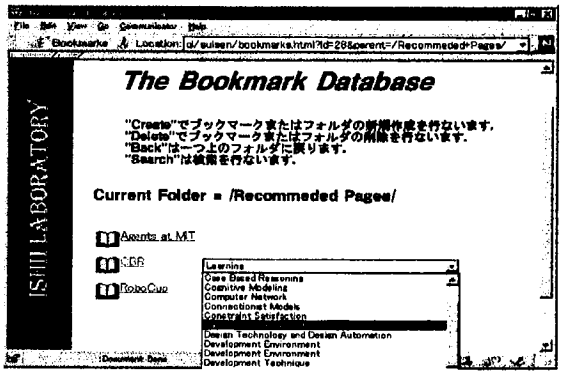

Figure 2 Suggested Classification

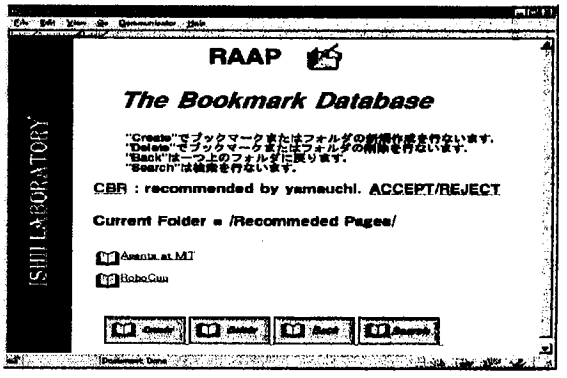

Figure 3 Recommendation

Learning to Classify

In our system, a **document** belonging to a user u_i s as 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 *deleted* bookmarks, incorrectly *classified* 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.

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

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.

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

Relevance Feedback

Given the unique terms in Q, $P=\{\tau_1, \tau_2, \tau_3, \ldots, \tau_r\}$, named P as for Prototype, it is it 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. Their dot product 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:

$$Tf(D) =< tf_{\tau_1,D}, tf_{\tau_2,D}, tf_{\tau_3,D}, \ldots, tf_{\tau_r,D} > \quad \text{and}$$

$$Tf(Q) =< tf_{\tau_1,Q}, tf_{\tau_2,Q}, tf_{\tau_3,Q}, \ldots, tf_{\tau_r,Q} >$$

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

$$Tf(Q^{i+1}) = Tf(Q^{i}) + \alpha Tf(D) \qquad (2)$$

Where $\alpha = 1$ if $D \in C^{+}_{i,j}$

$\alpha = -1$ if $D \in C^{-}_{i,j}$

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

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

$$\vec{Q} = \frac{\beta}{\left|C^{+}_{i,j}\right|}\sum_{j \in C^{+}_{i,j}} \vec{D}_j - \frac{\gamma}{\left|\overline{C^{+}_{i,j}}\right|}\sum_{j \in \overline{C^{+}_{i,j}}} \vec{D}_j \qquad (2.1)$$

Note that $\left|\overline{C^{+}_{i,j}}\right| \neq C^{-}_{i,j}$

Usual values are:
$\beta=4$ and $\gamma=4\beta$

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

Instead, we propose a new algorithm for incrementally building these vectors in order to reflect effectively the user's current interests. For this, at first, we shall give a couple of useful definitions:

Definition 1: *We define **positive prototype** for a class c_i, user u_j at step t, as a finite set of unique indexing terms, chosen to be relevant for c_i, up to step t (see feature selection).*

$$P^{(t)+}_{i,j} = \{\tau_1, \tau_2, \tau_3, \ldots, \tau_r\}$$

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

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

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 $Tf(Q_{i,j}^{(t+1)+})$
$\}$ 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:

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

Lets now express this equation in words. For a given user u_i, the similarity between a class c_j and an incoming document D at step 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 follows the intuitive notion that a document should be 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.

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 (Yang 1997), 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.

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:

$$E_{i,j}(\tau, S) = I(S) - [P(\tau = present)I(S_{\tau = present}) + P(\tau = absent)I(S_{w = absent}) \quad (4)$$

where,

$$I_{i,j}(S) = \sum_{c \in \{C_{i,j}^{+}, C_{i,j}^{-}\}} -P(S_c)\log_2(p(S_c))$$

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 (Pazzani, Muramatsu and Billsus 1996) 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* (Frakes and Baeza-Yates 1992) 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 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.

Learning to Recommend

For the purpose of learning to who recommend a page saved by its user, the agent counts with 2 elements. The *user vs. category* matrix M_{mxn} 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

	John	Kato	Ishii	Joacc	Gina
Information Retriev	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

matrix is initialized to one for the classes that the user has shown interest.

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.

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

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 equation of similarity between two classes for a user u_i at step t, as the dice coefficient between the positive prototypes of the classes (Eq. 6).

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

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:

$$\vec{R}_j = < rel_i^t(c_j,c_1), rel_i^t(c_j,c_2), \ldots, rel_i^t(c_j,c_n) > \quad (6.1)$$

where n = # 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 corresponding confidence factor.

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

In Eq. 8, the confidence factor of user u_i with respect u_j, 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 and to avoid circular references in the recommendation chain, the agents verify that the recommended document is not already registered in the target's database.

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.

As future work we are looking forward test *RAAP* in broader environments and 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.

References

- Blum, A. 1996. On-line Algorithms in Machine Learning (a survey). *Dagstuhl workshop on On-Line algorithms*
- Yang, Y., Pedersen, J. 1997. Feature selection in statistical learning of text categorization", *In Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*.
- Pazzani,M.,Muramatsu,J.,and Billsus, D., 1996. Syskill & Webert: Identifying interesting websites, In *Proceedings of the American National Conference on Artificial Intelligence (AAAI'96)*, Menlo Park, Calif.: AAAI Press.
- Kautz, H., Selman, B. and Shah, M., 1997. The Hidden Web. *AI Magazine*, AAAI Press.
- Frakes, W.,Baeza-Yates,R. 1992. *Information Retrieval: Data Structure & Algorithms*. Printice Hall, NJ.