LEARNING USER PREFERENCES FOR PERSONALIZED AND COLLABORATIVE WEB SEARCHING

Roberto OKADA
School of Project Design, Miyagi University,
1 Gakuen Taiwacho Kurakawgun Miyagi 981-3298, JAPAN

Abstract: With the explosion in the amount of information on Internet, finding desired information put additional cognitive and repetitive burdens on the user. In order to overcome these drawbacks, we propose a layer of adaptive and collaborative agents between the layer of users and the layer of distributed information sources like Internet. The personalization is realized by learning the user model and using it at query formulation. Machine learning and information retrieval techniques were utilized to learn the user preferences and to provide support for well-formed personalized query reformulation. However, for learning agents working individually, they face two problems: (i) serendipity, i.e. they cannot deal properly with situations previously unseen in the past; and (ii) cold-start: they spend some time to relearn about new situations.

In order to deal with these problems, we add a layer of collaboration between the agents, where the selection of peers is based on the trust relationship among them. The collaborative aspect is obtained by exchange of information learned by the individual agents.

Keywords: Agent, Learning, User preference , Customization, Collaboration

1. INTRODUCTION

As it has been arguing that the standard Web search services are far from ideal, many researches are seeking for a better way to tackle the ever growing WWW. Various ranking algorithms used to evaluate the relevance of documents to the query are impractical [1]. This is because the information given by the user is too few to give good estimation.

Most search tools still use keywords to specify queries. One factor limiting the precision of queries is that users do not submit well-focused queries. In general, queries get more precise as more words are added to them. Unfortunately, the average number of words in a query submitted to is 1.5, barely enough to narrow in on a precise set of documents [2].

One way to improve effectiveness is to better represent the information need by adding useful terms to the query, e.g. by relevance feedback, a well known technique in information retrieval, where terms occurring in known relevant documents are added to the query. Existing systems with query reformulation support work well for closed information spaces.

However, the Internet information space cannot be considered in this category. Another drawback of existing query reformulation is that the support they provide are not customized to user’s tastes.

In order to overcome these drawbacks, we propose a layer of adaptive and collaborative learning agents between the layer of users and the layer of distributed information sources like Internet. Agents are semi-autonomous continuously running computer programs that perform some task on user’s behalf. The personalization is realized by learning the user model and using it at query formulation. Machine learning and information retrieval techniques were utilized to learn the user preferences and to provide support for well-formed personalized query reformulation.

A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$. In our case, the task $T$ is to find relevant information for the user, the experiences $E$ are represented by the set of documents the user have read in the past and the performance $P$ is measured by metrics like precision or utility.

In this paper, we first present a new query reformulation support, in order to provide customized search results. Our approach is based on the assumption that the information about the query that are not specified in the user’s input may be obtained from the results of the previous or past queries.

For learning agents working individually, they face
two problems [3]: (i) serendipity, i.e. they cannot deal properly with situations previously unseen in the past; and (ii) cold-start: they spend some time to relearn about new situations.

In order to deal with these problems, we add a layer of collaboration between the agents, where the selection of peers is based on the trust relationship among them.

This paper is organized as follows. In section 2, we first review the problems related to web searching, the existing solutions and their drawbacks. Next, we introduce the design of our customized and collaborative web search multi-agent system. Section 3 outlines the realization of the single agent scenario, where we report mainly on the user profile generation mechanism and query reformulation support. Section 4 starts by reviewing the problems of “slow start learning curve” for the single agent scenario. Then, in order to overcome these problems, we introduce a layer of collaboration among agents, based on trust relationships. In section 5, the implementation and evaluation is explained, while in section 6 the conclusions and an outline of our further work is given.

2. PERSONALIZED AND COLLABORATIVE WEB SEARCH SUPPORT

2.1. Search problem formulation

Fig. 1 shows the retrieval model and the user-retrieval system interaction flow.

![User-search engine interaction](image)

Figure 1: User-search engine interaction

Traditional retrieval systems e.g. Smart[4], TREC[5], WWW search engines, build an index of a collection of documents. The retrieval is done by consulting this index containing information about the collection. The user express his information needs as a set of keywords $q_0$ to send to the search engines. In response, he gets from the engines a number of suggested search results, $sol(q_0)$. The document information space covered by this retrieval system is shown at the right of Fig. 1. We regard the information space of documents as expressed in terms of the well-known vector space model [4]. In Fig. 1, every point in the document information space represents either a query or a document vector, where the similarities between two vectors are proportional to the distance between them, this is, the closer, the more similar. For the retrieval, both the documents and the queries are mapped in the same vector space in order to calculate their similarities and retrieve the most similar ones.

○ represent relevant documents to the query, and × the irrelevant ones. In case that the query is not well formulated, many irrelevant documents are returned. One way to focus the search is reformulating the query by adding related terms, where this process is repeated until the user find the documents of interests.

For the query reformulation, a variety of formulas have been studied, e.g. Ide Regular:

$$q_{ref} = q_0 + \sum_{Drelev} D - \sum_{Dirrelev} D$$

The Roccio’s feedback approach[6], expressed in vector space terms is: “the final query vector $q_{ref}$ is the initial query vector $q_0$ moved towards the centroid of the relevant documents and away from the centroid of the non-relevant documents”, as shown in Fig. 1.

2.2. User needs and loads for web searching

2.2.1. User needs

We can classify the information needs into mainly two cases: the user is searching for: (a) one document in particular, (b) a number of documents related to some area of interest. Here, we deal with the second case, where we assume long-term-interest search.

2.2.2. User load

We classify the user loads into (a) cognitive load and (b) repetitive load, both appearing at the different stages of the searching process:

1. At query formulation stage: (1a) the cognitive
load, due to the fact that the user does not know how to express the query, specially in unfamiliar domains, (1b) the repetitive load, i.e., the composition of a long query every time.

2. At search result selection or relevance feedback: (2a) the cognitive load, i.e. the user may not be sure how to do the proper selection, (2b) the repetitive load, where the user has to go checking through hopeless results.

Among the factors causing the above loads, one is that some keywords are manifold, so irrelevant documents are included in the search results, reducing the precision. This situation is further worsened for those keywords that encompass too broad of the search space e.g. computer [1].

2.3. Existing solutions and their drawbacks

Regarding personalized or customized information, there are works on Learning Interface Agents on Internet. In order to complement works on Web agents for autonomously browsing, e.g. Letizia[7], Web-Watcher[8], LIRA[9], or filtering on behalf of the user, e.g. Amalthaea[10], Musag[11], works by Maes[3] group at MIT, we focus on adaptive agents for web searching, which provide support for customizing the query formulation and retrieval.

With respect to the drawbacks of existing query reformulation support e.g. Yahoo, InfoSeek, Bhatia[12]: (i) they are limited to closed information spaces, (ii) the support they provide are not customized to user’s tastes.

Regarding research on collaboration, there are some works, such as the collaborative interface agent for mail handling[13], GroupLens[14], FireFly[15], Fab[16]. Our work is characterized in that we share information from experiences and not application-dependent rules or rated items.

2.4. Our proposal: personalized query reformulation on open information spaces

We are dealing with an open and distributed information space like Internet, where it is not possible to have complete knowledge of such a huge information network.

Our approach consists in building a user profile based on documents acknowledged either as relevant or irrelevant. Then, this profile is used for query reformulation. The user model is obtained by learning through the interactions with the user, in order to use these experiences for optimizing future search and trying to adapt to the user’s personal needs.

In this case, the reformulation is under incomplete knowledge, i.e. we don’t need to have knowledge about the whole open information space. Instead, we are trying to “infer” from the local personal profile in order to move the query closer to the possible relevant documents in the distributed information spaces.

2.5. Design and system overview

The personalization is realized by learning the user model and using it at query formulation, while the collaborative aspect is obtained by exchange of information between agents with similar interests.

The aim of learning is to provide with the ability to record its success (relevant suggestions) and failures (irrelevant suggestions), and thus infer knowledge useful for increasing its performance over time.

The system’s main components, shown in Fig. 2 are: (a) user agent (UA), one per user; (b) retrieval manager agent (Man), specialist in a given topic or domain; and (c) existing search engines and document servers.

![Figure 2: Multi-agent architecture for personalized and collaborative web searching](image)

- The user agent’s (UA) main function is to maintain the user’s profile, which represents user’s interests, and use it to provide support for customized query reformulation. In our approach, information retrieval techniques were utilized to learn the model of the user preferences, thus creating the personalization layer at the user site. The personalization is obtained by creating a “personal index” of documents acknowledged
The retrieval manager agent (Man) is a kind of meta-search engine, but with learning and collaborative capabilities. The Man receives a user-customized query from the UA, and passes it to commercial Web search engines, e.g., Lycos, InfoSeek, AltaVista. They can adapt the query to the operators supported by the different engines, represented as \( q_1, q_1' \) in Fig. 2. We are interested also in making these Man specialists in a given domain or topic.

3. SINGLE AGENT SUPPORT

In this section, the objective is to assist users in personalized query reformulation for getting information adapted to user preferences. The basic idea is:

1. Build a user profile expressed as a set of terms expressing his interests, from acknowledged documents;
2. Provide support for query reformulation based on this profile. Techniques from information retrieval and machine learning, like vector space model and unsupervised learning are used. In this paper, we review the basic mechanisms involved. More details can be found in [17].

3.1. Iterative cycle of interaction user-agent

The interaction from users consists in providing feedback on the suggested search results. This rating is used to update his personal agent’s profile and are also forwarded back to the retrieval manager agents, which will use them to adapt their topic profiles.

Based on this knowledge acquired, the agents will interact with users to reduce the cognitive and repetitive loads, explained in section 2.2.2.

Fig. 3 shows the interaction between the user and the user agent. The UA do the following: (1) collect the documents acknowledged by the user; (2) compute all pair-wise inter-document similarity and cluster the documents; (3) extract related words; (4) display the clustering results at query formulation to the user; (5) refine the query; (6) get the user’s feedback to acknowledged documents; and (7) repeat step 1-6.

3.1.1. Profile generation

We adopt the well known vector space model [4] used in information retrieval, explained before. This support is based on the association hypothesis: “if an index term is good at discriminating relevant from non-relevant documents, then any closely associated index term is also likely to be good at this” [18].

Using existing clustering algorithms, a profile can be divided into clusters, where each cluster contains documents having a number of common terms. The average sum of all the document vectors in a cluster is used as its representative (also called “cluster centroid”). The whole profile is represented by the union of all its cluster centroids in the form of a term-cluster matrix.

This user model is implemented, from the clusters of interest extracted above - as a personal individual dictionary for that user, containing the query-word and the associated words related to the query-word. In the associated words there are two sets: one set for related-words and a second set of undesired-words, extracted from documents with positive feedback and negative feedback, respectively. The dictionary is composed of a set of

\[
\{ \text{related-words}, \{ \text{undesired-words} \} \}
\]

3.1.2. Query reformulation and retrieval

The query reformulation support is provided via the user interface shown in Fig. 4. The system suggests related terms to the query input by the user and let the user select the terms to add. These suggested terms are extracted from user’s past interactions, so the terms in the profile represent the user interest.
3.2. Improvement in quality of results

3.2.1. Evaluation metrics

Two common measures of retrieval effectiveness are:

**True Recall**: defined as the ratio of the total number of relevant elements in the space to the total of relevant results returned by the search. It cannot be calculated for Web space, because the total number of relevant links change quickly and is practically unknowable.

**True Precision**: defined as the ratio of relevant elements returned to the total number of elements returned. It is too arduous to calculate in environments like WWW, because it would mean examining all of the links returned by a service, which may number in the thousands or millions.

We are more interested in the increase in precision than recall, because most search engines will retrieve pages with adequate recall, but with poor precision. However, such traditional metrics cannot be properly used in open environments like Internet. Thus, for a study of relevance of links returned by the search engines, a new measurement of relevancy must be defined. Another problem is that these traditional metrics do not reflect properly the relevancy of suggested links. To overcome, a new metric which take into account what counts as “relevant” and “useful” is defined. To express the usefulness, the links are classified into several relevancy categories, such as: duplicate links; inactive links; irrelevant links (category 0); technically relevant links (category 1) are links satisfying the query but not useful; potentially useful links (category 2) are links of conceivable use; and most probably useful links (category 3) are links clearly useful to almost everyone conducting the search.

About what measure to be used to compare the performance of the search services, our criteria is to measure for their ability to put relevant pages within the first N links returned by a query, which we called Utility. Utility is defined as the normalized ratio of the weighted sum of the relevant links considered to the total number of links considered. This metric is based on the fact that it is better for a good link to be higher in the ranked list than lower. The user will see and evaluate the links listed first, so pages located there should be given more weight.

For the calculation of the numerator of the utility metric, we use a weighting scheme, where we take into account both (i) the category the returned link belongs to, and (ii) its position in the ranked list. It is important to remember that our metric have to give higher relevance to good links (ideally, category 3) ranked at the top of the list. Thus, we assign the values of 1/2 for category 1 (technically relevant, but not no useful), 1 for category 2 (relevant) and 2 for category 3 (highly relevant). These values are then multiplied by weights according to their location that give more value to links earlier in the list. The denominator is a normalizing factor based on the total number of links in consideration.

3.2.2. Evaluation for single agent scenario

We have shown the validity of our approach, where the detailed results are given in [17]. By doing the comparison for different queries and different engines it is possible to see that the results with the agent support are closer to the user preferences than in the case without support. The experiments suggest that this approach of forming a user model and using it at query formulation and refinement time provides an effective way to discover resources in a large universe of documents, eliminating bordering unnecessarily the user as well as eliminating the waste of
time and network resources.

4. COLLABORATIVE AGENTS SUPPORT

In the previous section, we have focus on the interaction “one user ↔ one UA ↔ search engines”, with focus on the user profile learning mechanism. The problem with the learning is the slow learning curve. In other words, when the system is faced with a new situation, it takes some time to learn before start being useful. In this section, our aim is to assist the information search process by collaborating with trusted peers, in order to avoid re-learning what others already have learned, i.e. to improve the learning curve.

The user profile for a single user agent, generated in section 3 is:

\[
\text{Profile}(user_k) = \langle \text{history}, \text{dictionary} \rangle \\
\text{history} = \{< q_i, \text{location} \_ \text{doc}(q_i) > \} \\
\text{dictionary} = \{< q_i, \text{rel} \_ \text{term} \_ \text{cluster}(q_i) > \}
\]

where history contains the locations of documents the user have acknowledged in the past in response to a given query \(q_i\), and the dictionary the set of clusters of interests which represent user’s interest.

It is important to note that for a previously unseen situation, Profile does not contain related information. Despite the success reported for learning user’s interests and providing support at query reformulation, some drawbacks remains. A major problem of the learning approach is that they require a sufficient amount of time (the training phase) before they can start being useful, i.e. present a slow increasing learning curve (known as cold-start). Even worse, their competence is limited to situations similar to those they have encountered in the past, so they don’t know how to deal with previously unseen situations (known as serendipity). Even worse, their competence is limited to situations similar to those they have encountered in the past, so they don’t know how to deal with previously unseen situations (known as serendipity). The agents of different users thus have to go through similar experiences before they can achieve a minimal level of competence, although there may exist other agents that already possess the necessary experience and confidence [13].

We will focus on the collaboration between agents in order to improve the learning curve. In this section, we will introduce additional information to the user’s profile to allow collaborative work.

4.1. Framework for collaboration

In the previous section, we have discussed about the case in which each user has its own locally situated profile and utilizes only his own profile at query reformulation support. As it is not possible that a single user’s profile becomes all mighty, then there will be cases in which this single profile cannot provide the necessary support. In such cases, one solution is that user agents belonging to different users to collaborate with other users distributed over the network. Such a system can elegantly handle discovery of new keywords and thus the learning curve increases faster. Also, one user agent can collaborate with multiple agents, thereby obtaining more than one perspective about a given user’s topic of interest.

Collaboration, however, increases the response time, but this delay is compensated by the better quality of query reformulation support. When faced with an unfamiliar situation, an agent consults its peers who may have the necessary knowledge to help it [13].

In a collaborative situation, a particular agent may not have any prior knowledge, but there may exist a number of agents who do. So, instead of each agent re-learning what other agents have already learned through experience, agents can simply ask for help in such cases.

In our approach, agents: (1) learn from relevance feedback; i.e. based on the relevance feedback from the user, the agents update their internal values of trust, confidence and profile, to be explained in subsection 4.3.; (2) build a trust relationship, i.e. we add \(\text{trust}_\text{peer} = \{\text{query}_i, \text{peer} \_ \text{list(query)}\}\) to the Profile\((user_k)\); and (3) maintains a trust table and use these weights to decide collaborative peers.

4.2. Interaction protocol between agents

In continuation, we explain how the different agents interact and cooperate with each other to satisfy the user request and decrease his load, as well as to gain efficiency in the gathering process. Here, we describe the process of locating information on the web based on our system, named CASIG.

We concentrate on how the User Agent map from the user requirement to the most suitable set of peers capable of providing help about the information the user is looking for.

Notation

- \(A = \{a_i\}\) : a set of agent \(a_i\)
- agent \(a_i = \langle id, loc, K, P \rangle\), where
id is identifier, loc is location, K is its knowledge/ability, P is the set of known peers, with a trust relationship.

- descr = <τ, ρ, η, cost>, where
  - τ is Topic, ρ is quantity, η is quality (which can be expressed as a vector of parameters with components like precision, recall, completeness),
  - cost of obtaining this information.

- message = type(α₀, α₄, X), where
  - α₀ is originator, α₄ is destination, X contains additional information (for simplicity, sometimes we omit writing α₀, α₄)
  - type: request, response

Assumptions

1. We assume that each agent "knows" its ability to answer about a given topic τ, i.e. when asked about K(τ), it responds with descr = <τ, ρ, η, cost>.
2. In order to contact the Man (or Index), its location is needed. We assume that is possible to get it from somewhere e.g. by asking to other agents.
3. In order to avoid the broadcasting of one UA requesting help to all other agents, we adopt a two-level communication: the UA ask first its local Manager.

Step 0. UA receives user request as keywords, and then convert keywords → τₓ

Step 1. for each known "trusted peer" (either other UAs or Man), UA sends the message: req-for-advice(τₓ)

Step 1'. The requested peer calculate and respond: respond-advice(τₓ) = <τ, ρ, η, cost> for

Step 2. UA receives responses, select and assign priorities to them.

Step 3. UA sends to the "selected" peers: request(τₓ)

Step 4. UA receives from "selected peers": response(τₓ)

Step 5. UA order the responses and present this to the user.

Step 6. the user will select from the set of candidate solutions, used as relevance feedback for update of trust values.

4.3. Learning from relevance feedback

We define some useful parameters, some of them indicate the degree of relationship between agent, i.e.:
1. level of trust: among same type of agents,
2. level of confidence: about its own capacity,
3. personal profile: to data necessary for personalized behaviour of his UA. These parameters are updated based on user's relevance feedback, in order to change the trust relationship among agents, their self-confidence about a given topic and their profiles.

- the update of trust of this UA to the agent agentₓ which comes with the candidate solution
  \[ trust(agentₓ) = trust(agentₓ) + β \times f \]

- the update of the agent's level of confidence to answer queries regarding topic τ
  \[ conf(agentₓ, τ) = conf(agentₓ, τ) + γ \times f \]

- the update of the user's personal profile, which are representation of the user interests.
  \[ profile = profile + δ \times f \times τ \]

τ is topic; β is the learning rate or the sensitivity of trust(agentₓ) to user feedback f; γ, sensitivity of confidence(agentₓ, τ) to f; δ, sensitivity of profile to f.

5. IMPLEMENTATION AND EVALUATION

5.1. Implementation

The multi-agent system is actually implemented in the real Internet environment - is build in C and using the libwww. It serves to show the benefits of machine learning and the social aspect of collaboration among agents, in order to share acquired information.

This system incorporates one user agent per user, several meta-index like managers, and the existing web including search engines, e.g., Lycos, InfoSeek, AltaVista and the document servers distributed all over the Internet. In the implemented system, when the user invokes his personal user agent, it opens a personal log file to keep a memory of the documents which the user finds interesting. This user agent also maintains a table of trusted peers, their location and
Table 1: Sample of profiles for user agents UA1, UA2, UA3 in organization A, and UA4, UA5 and UA6 in organization B

<table>
<thead>
<tr>
<th></th>
<th>UA1</th>
<th>UA2</th>
<th>UA3</th>
<th>UA4</th>
<th>UA5</th>
<th>UA6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>computer</td>
<td>agent</td>
<td>project</td>
<td>control</td>
<td>recognition</td>
<td>network</td>
</tr>
<tr>
<td>2</td>
<td>software</td>
<td>messaging</td>
<td>agent</td>
<td>traffic</td>
<td>learning</td>
<td>system</td>
</tr>
<tr>
<td>3</td>
<td>resource</td>
<td>system</td>
<td>network</td>
<td>research</td>
<td>network</td>
<td>research</td>
</tr>
<tr>
<td>4</td>
<td>user</td>
<td>flexible</td>
<td>member</td>
<td>network</td>
<td>neural</td>
<td>japanese</td>
</tr>
<tr>
<td>5</td>
<td>lab</td>
<td>asynchronous</td>
<td>system</td>
<td>scheme</td>
<td>large</td>
<td>picture</td>
</tr>
<tr>
<td>6</td>
<td>goal</td>
<td>mail</td>
<td>japanese</td>
<td>parameter</td>
<td>apply</td>
<td>click</td>
</tr>
<tr>
<td>7</td>
<td>framework</td>
<td>software</td>
<td>English</td>
<td>atm</td>
<td>algorithm</td>
<td>computer</td>
</tr>
<tr>
<td>8</td>
<td>lot</td>
<td>design</td>
<td>user</td>
<td>include</td>
<td>high</td>
<td>background</td>
</tr>
<tr>
<td>9</td>
<td>agent</td>
<td>message</td>
<td>based</td>
<td>arrival</td>
<td>telnet</td>
<td>page</td>
</tr>
<tr>
<td>10</td>
<td>research</td>
<td>based</td>
<td>link</td>
<td>bucket</td>
<td>scale</td>
<td>complete</td>
</tr>
<tr>
<td>11</td>
<td>English</td>
<td>lab</td>
<td>lab</td>
<td>ingress</td>
<td>system</td>
<td>data</td>
</tr>
<tr>
<td>12</td>
<td>Institute</td>
<td>existing</td>
<td>dynamic</td>
<td>input</td>
<td>method</td>
<td>recognition</td>
</tr>
<tr>
<td>13</td>
<td>Team</td>
<td>proposal</td>
<td>version</td>
<td>call</td>
<td>convergency</td>
<td>question</td>
</tr>
<tr>
<td>14</td>
<td>Internet</td>
<td>function</td>
<td>resource</td>
<td>topic</td>
<td>pattern</td>
<td>character</td>
</tr>
<tr>
<td>15</td>
<td>Utilize</td>
<td>intelligent</td>
<td>flexible</td>
<td>generating</td>
<td>obtained</td>
<td>technology</td>
</tr>
<tr>
<td>16</td>
<td>Effectively</td>
<td>version</td>
<td>paper</td>
<td>real</td>
<td>character</td>
<td>study</td>
</tr>
<tr>
<td>17</td>
<td>Implement</td>
<td>team</td>
<td>software</td>
<td>mechanism</td>
<td>propagation</td>
<td>control</td>
</tr>
<tr>
<td>18</td>
<td>Member</td>
<td>request</td>
<td>technology</td>
<td>interested</td>
<td>coefficient</td>
<td>contact</td>
</tr>
<tr>
<td>19</td>
<td>Requirement</td>
<td>making</td>
<td>related</td>
<td>leaky</td>
<td>data</td>
<td>future</td>
</tr>
<tr>
<td>20</td>
<td>People</td>
<td>user</td>
<td>communication</td>
<td>term</td>
<td>back</td>
<td>topic</td>
</tr>
<tr>
<td>21</td>
<td>Dynamic</td>
<td>Japanese</td>
<td>computer</td>
<td>modeling</td>
<td>response</td>
<td>develop</td>
</tr>
<tr>
<td>22</td>
<td>Project</td>
<td>orchestration</td>
<td>result</td>
<td>output</td>
<td>calculation</td>
<td>algorithm</td>
</tr>
</tbody>
</table>

their trust relationship. This trust relationship is updated based on user feedback after the evaluation of retrieved documents. When this user agent receives the query as a set of keywords from the user, it first selects the agents with high “trust values” and sends them the query. The trusted user agents look at their personal log file for documents retrieved in the past related with the query in question, while the trusted managers select some search engines to send this request. This selection is done based on a “trust table” with the list of known search engines, their location and a weight indicating how successful they were in the past answering about this query. Manager receives the pointers to candidate solutions from the selected search engines, re-prioritize them and return this list of pointers to candidate solutions as an HTML document. This document is presented to the user. Then he selects one hyperlink causing the document to be retrieved. The user evaluates this document and provides a positive or negative feedback. This causes the update of the weights of the trust relationships of this user agent to the agent which proposed this solution, the trust weights of the manager to the search engines as well as the user’s personal profile.

5.2. Experimental Setting

For the evaluation of the benefits of collaborative support, we have conducted an experiment with two organizations (static group of users and their corresponding agents), named organization A and B, as shown in Fig. 2. The organization A is composed of seven users with their corresponding user agents (UA) and one manager ManA, while the organization B is composed of eight users and their user agents and a manager ManB.

We allow a learning period for all the user agents, working in a single setting, where the user profiles are build based on documents acknowledged by their corresponding users. For the experiment, we have selected three user agents from organization A, named UA1, UA2 and UA3 and other three user agents UA4, UA5 and UA6 from organization B, as depicted in Fig. 2. A sample of the profiles for the user agents of organization A is shown in the left portion of Table 1, while the corresponding to the organization B is shown in the right part of Table 1.

As an example, let’s assume that user1 has the intention of searching for documents about “computer network” but only gives as query input qo = network. We also assume that his corresponding user agent UA1 does not have in its profile knowledge about “network”. In this setting, as UA1 has no previous experience on “network”, it cannot provide support. Thus, the utility value for the returned results is very
low, as $UA_1$ cannot provide the necessary support for query formulation. In order to help $UA_1$ to deal with situations previously unseen in the past, we allow to ask help to other user agents. In this example, $UA_3$ from organization A or $UA_6$ from organization B can provide this support. The candidate selection is based on the trust relationship between them. The trust value of the agent which comes with the proper suggestion is incremented, according to the formulas explained in section 4.3.

5.3. Experimental results for collaborative search

The results of the utility values for different queries are summarized in Table 2, where $q_0$ is the query without collaboration and the other queries are based on suggestions provided by other agents, i.e. collaborative support for searching.

<table>
<thead>
<tr>
<th>query</th>
<th>utility (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$ : network</td>
<td>30.01</td>
</tr>
<tr>
<td>$q_1$ : computer network</td>
<td>37.18</td>
</tr>
<tr>
<td>$q_2$ : $q_1$ + system</td>
<td>51.24</td>
</tr>
<tr>
<td>$q_3$ : $q_2$ + research</td>
<td>52.41</td>
</tr>
<tr>
<td>$q_4$ : $q_3$ + project + paper + information</td>
<td>58.41</td>
</tr>
</tbody>
</table>

Table 2: Some sample queries, where $q_0$ and $q_1$ are in a single agent environment, while the other queries are with collaborative support

Taking the average of the experiments, we obtain an example of learning curve (user satisfaction vs. no. of examples) for the cases without and with collaboration is shown in Fig. 5. Here, the user satisfaction is proportional to the utility metric. In the single agent scenario, when faced with an previously unknown situation, takes some time to learn, and during this learning or re-learning period, the utility is low and oscillatory, as we can notice in Fig. 5. If we allow collaboration between agents, this problem is solved.

6. CONCLUSION AND FUTURE WORKS

With the explosion in the amount of information on Internet, finding desired information put additional cognitive and repetitive burdens on the user. In order to overcome these drawbacks, we have proposed a layer of adaptive and collaborative agents between the layer of users and the layer of distributed information sources like Internet. In section 3, our focus was on the customization of information searching by learning a user profile. We have proposed a method for query reformulation adapted to user interests. The building or user profile is based on recursive clustering of documents acknowledged by the user. At query reformulation, this profile is used to add related terms to the initial query.

While the learning approach give satisfactory results, one drawback is the slow learning curve when it faces a previously unseen situation. In order to overcome, in section 4, we have focus on the collaboration between agents of different users. When faced with an unfamiliar situation, an agent consult its "trusted" peers, instead of broadcasting the request to every known agents. This trust relationship between agents is also learned through the successive interactions.

Each of our user agents maintains: (1) history information (i.e. where to find information related to a given topic), (2) personal dictionary information (i.e. the query and related words, as explained in section 3 for customizing information search). In our approach, the user agents of different users - by sharing both types of information - improve the learning curve and are able to face new situations effectively.

Our future work includes: (i) regarding UA, the use of phrases in the user profile, calculated by word correlation; (ii) at the RMA the topic specialization and the interaction adapted to the search engine operators, and (iii) the improvement of inter-agent collaborative work.
Acknowledgements

The author would like to thank Prof. Eun-Seok Lee, Prof. Tetsuo Kinoshita and Prof. Norio Shira-tori for their support and helpful comments.

REFERENCES