Implicit: An Agent-Based Recommendation System for Web Search

Alexander Birukov
Department of Information and Communication Technology
University of Trento - Italy
birukou@dit.unitn.it

Enrico Blanzieri
Department of Information and Communication Technology
University of Trento - Italy
enrico.blanzieri@unitn.it

Paolo Giorgini
Department of Information and Communication Technology
University of Trento - Italy
paolo.giorgini@unitn.it

ABSTRACT

The number of web pages available on Internet increases day after day, and consequently finding relevant information becomes more and more a hard task. However, when we consider communities of people with common interests, it is possible to improve the quality of the query results using knowledge extracted from the observed behaviors of the single users. In this paper we propose an agent-based recommendation system for supporting communities of people in searching the web by means of a popular search engine. Agents use data mining techniques in order to learn and discover users' behaviors, and they interact one another to share knowledge about their users. The paper presents also a set of experimental results showing, in terms of precision and recall, how agents interaction increases the performance of the overall system.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering, relevance feedback,search process*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

General Terms

Design, Experimentation

Keywords

Multi-agent system, recommendation system, personalized web search, implicit culture, data mining, information retrieval

1. INTRODUCTION

World Wide Web contains a huge amount of web pages. According to ISC Internet Domain Survey [13], in July 2004 there were 285,139,107 hosts on the Internet and this number increased by 22% since January 2004 (233,101,481).

Given these numbers, it results clear that the complexity of finding relevant information in the web increases day after day. Approximately 56.3% of the Internet users search the web at least once per day [14] and analyzing their behaviors, one can notice that they rarely (33% according to iProspect survey) look at the second page of the results provided by the search engine.

Authority-based search engines [6] are one of the most powerful web search tools. However, they have shortcomings such as the lack of personalization. Gori and Witten [12] state that "[...]the need to protect minorities can only be addressed within new paradigms; new, personalized views of the web that supplement today's horizontal search services. Different users may merit different answers to the same query[...]". Recommendation systems are an example of such personalization.

A typical recommendation system accepts queries from a user and exploits knowledge about his/her needs, behavior patterns, search profiles and content information in order to make personalized recommendations on items. In the area of web search, two are the main classes of recommendation systems: systems that deal with the content of the web pages [8, 22] and systems that use a collaborative approach [15]. The result information are used in both cases to create suggestions for the user.

Approaches that apply agents and multi-agent systems to search the web have been presented in literature. The main idea is to use a software agent that assists its user during the web search [8, 16, 23]. The agent tracks the user browsing and builds the user profile in order to anticipate items of interest. Coalitions of several agents are also used to answer the queries of single or multiple users [19] and specific mechanisms such as auction protocol and reward techniques are used to realize the collaboration among the agents [24]. Other approaches [5, 7, 25] propose personal agents acting on behalf of their users, collaborating one another and with the major goal of improving the user browsing. In some of the systems considered so far, the user is asked to do an extra work during the search, for instance he/she needs to specify the areas of interest or analyze a lot of results about similar searches. Often there are also specific restrictions, such as for example the ability of using only certain part of pre-defined knowledge or ontologies.

In this paper we present *Implicit*, a multi-agent recommendation system based on the concepts of Implicit Culture. Implicit Culture [3] is a generalization of Collaborative Filtering [20], which is a technique of producing personal recommendations using similarities between users' ratings.

Implicit Culture means that a new member of a community is induced to behave similarly to the other members without the need of expressing explicitly the knowledge of the community. Our system is mainly intended to improve the web search of a community of people with similar interests. When a user submits a query, *Implicit* suggests specific information exploiting previous observations about the behavior of other users when they asked similar queries. Each user has his/her own personal agent able to interact with the personal agents of other users. The system implements a collaborative approach that provides to the querying user suggestions from the members of the community in addition to results provided by a search engine. Moreover, users are not forced to perform any extra work during the search.

The rest of the paper is organized as follows. Section 2 gives a brief description of what Implicit Culture is and presents a class of systems to support Implicit Culture. In Section 3, we show the general architecture of the proposed system, whereas in Section 4, we present some experimental results. Section 5 reviews related work and Section 6 draws some conclusion and future work.

2. IMPLICIT CULTURE AND SICS

This section presents an overview of the general idea of Implicit Culture and Systems for Implicit Culture Support (SICS).

A group of agents working within a community exploits a great amount of knowledge and skills. Knowledge can be either explicit (when it is possible to describe and share it through documents and/or information bases) or implicit (when it is embodied in the capabilities and the abilities of the community members). When a new agent comes to the community it faces the problem of acquiring the necessary knowledge. In other words, in order to act in the community a newcomer has the problem of acquiring the knowledge that is behind the behaviors of the members of the community, namely the culture of the community. When the environment is under control it is possible to achieve this goal without requiring the agent to know about the group and its behavior. The relation between two groups of agents such that the agents belonging to a group behave consistently with the "culture" of the agents belonging to another has been defined Implicit Culture [3].

For example, let us consider a new member of a lab that needs to browse the papers submitted during the current year by other members of the same lab. Let us suppose also that the list of publications is located on the laboratory intranet and each member knows where it is, but the new one does not. If the personal agent of the newcomer is able to provide him/her with this link and he/she accesses the material, then it is possible to say that new member behaves in accordance with community culture and that the Implicit Culture relation is established.

The general architecture of Systems for Implicit Culture Support consists of the following three basic components:

- *Observer*, the part of SICS that stores in database information about actions executed by the user;

- *Inductive module*, that analyzes the stored observations and implements data mining techniques to discover patterns of user behavior;

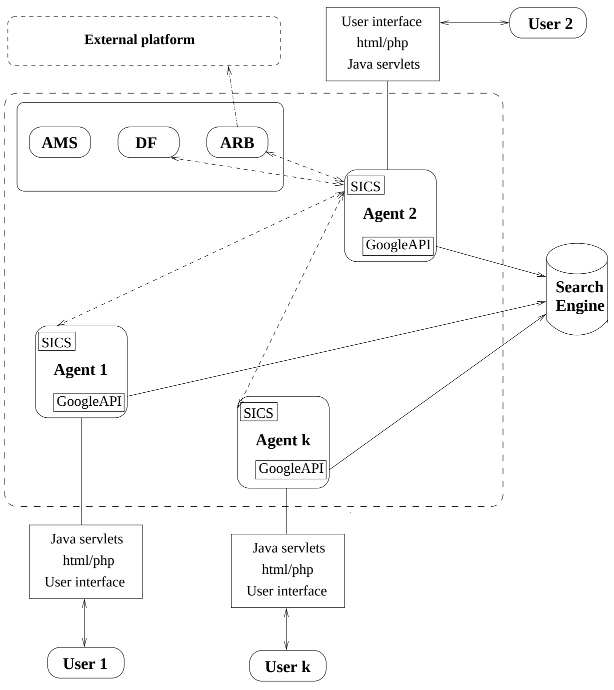

Figure 1: The architecture of the system. ***Personal agents*** **process queries from *users* and interact with each other to exchange links; *SICS* is a part of personal agent that is responsible for the recommendation creation process; *GoogleAPI* allows agent to query Google search engine; *Agent Management System (AMS)* exerts supervisory control over the platform. It provides agent registration, search, deletion and other services; *Directory Facilitator (DF)* provides agents with other personal agents' IDs. *Agent Resource Broker (ARB)* deals with links to the services available on the other platforms.**

- *Composer*, that exploits the information collected by the observer and analyzed by the inductive module in order to produce better suggestions to its user or to other agents.

In the inductive module we use data mining techniques in order to extract interesting patterns from the user behavior. There are several approaches that can be exploited. Clustering can be applied in order to get knowledge about the correlations in the observations. For instance, agents can be clustered by interests and past actions of their users. Alternatively, we can apply association rules techniques, like apriori [1] for learning association rules between the actions. Clusters and rules are used by the composer module.

The goal of the composer is to propose links such that the agents would accept them. Composer consists of two modules. In general, the goal of the first one is to find prospective actions that satisfy the theory (namely the clusters or association rules). The second module deals with selection of the data related to the found actions. More details on the structure and the implementation of the composer and SICS are given in work of Blanzieri et. al [5].

In our application each agent applies the SICS to establish an Implicit Culture relation with the other agents of the environment [3, 5].

```
global result
for all message in INBOX do
  if (message.type == 'query') then
    result := nil
    if (query.sender == user) then
      google-search(query.sender,query.keyword,result.links)
      inform(self, user, result.links)
    end if
    SICS.internal-search(query.sender,query.keyword,result.links)
    SICS.external-search(query.sender,query.keyword,result.agents)
    if (query.sender == user) then
      if (result.agents == nil) then
        add(DF,result.agents)
      end if
      for all agent in result.agents do
        request(self,agent,query.keyword)
      end for
      inform(self, user, result.links)
    else
      inform(self, query.sender, result.links)
      inform(self, query.sender, result.agents)
    end if
  else if (message.type == reply) then
          if (message.content == resource-link) then
            add(resource-link, result.links)
          else if (message.content == agent-ID) then
                  add(agent-ID, result.agents)
                end if
          end if
        end if
  else if (message.type == 'feedback') then
          add(feedback,observations)
        end if
  end if
end for
```

Figure 2: Activities schema of a personal agent search.

3. SYSTEM STRUCTURE

In this section we give a brief description of *Implicit*. *Implicit* is an agent-based web search system that implements the notion of Implicit Culture. The system is implemented using JADE (Java Agent Development Framework) [2], a FIPA-compliant [11] framework for multi-agent systems development. The architecture of the system is presented in Figure 1. Each user is assisted by his/her own personal agent to search the web. The personal agents incorporate and use the SICS module in order to produce recommendations, and each of them is able to communicate and interact with external information sources, and in particular with Google [6]. The major goal of each personal agent is to propose to its user and to other agents links to web pages that are considered relevant for their search. Personal agents can use different internal and external sources of information. The personal agents are software agents running on the server side, whereas on the client side there is a html/php user interface that prompts the input keyword, displays the results and collects feedback information.

Within the platform each agent interacts with the other agents and with its user by exchanging messages. Each personal agent can process several types of messages. *Query* message contains information about the user or other agents inquiries. The basic content of a message is the keyword of the search. *Reply* message contains the recommended link to the web page or the ID of another agent. This message is sent as an answer to a query. At the end a *feedback* message will be sent to all the agents involved in the search. A feedback message contains an accepted link to the web page or the ID of an agent that has suggested the link. Figure 3

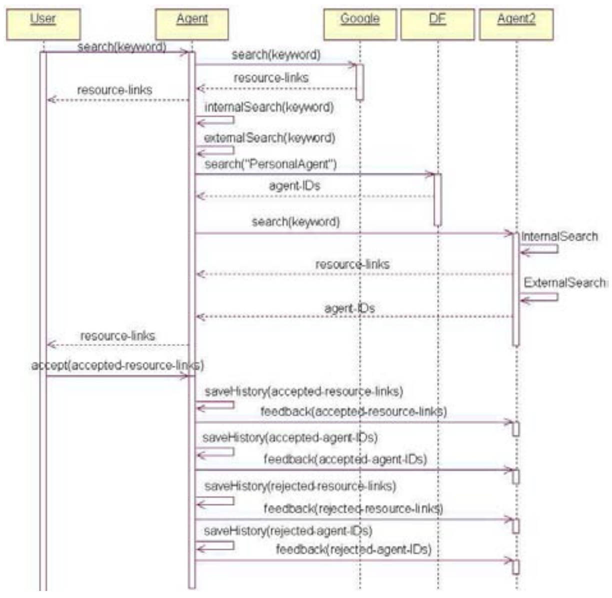
User
Agent
Google
DF
Agent2
search(keyword)
search(keyword)
resource-links
resource-links
internalSearch(keyword)
externalSearch(keyword)
search("PersonalAgent")
agent-IDs
search(keyword)
InternalSearch
resource-links
ExternalSearch
agent-IDs
resource-links
accept(accepted-resource-links)
saveHistory(accepted-resource-links)
feedback(accepted-resource-links)
saveHistory(accepted-agent-IDs)
feedback(accepted-agent-IDs)
saveHistory(rejected-resource-links)
feedback(rejected-resource-links)
saveHistory(rejected-agent-IDs)
feedback(rejected-agent-IDs)

Figure 3: Sequence diagram of agents interactions.

depicts the sequence diagram of message passing between two agents participating in a search. *Agent* is the personal agent of the user and *Agent 2* is the agent queried by *Agent*. Table 1 contains the scheme of interaction, in particular the protocols used in the communication between the system actors.

In the present implementation, the agent performs three types of search in the following order: first Google search (when the query comes from the user), then Internal search and finally, External search. During Google search the agent queries Google search engine in order to obtain links for the keyword chosen by the user. In the Internal search the SICS module generates links to the web pages, using information about observed past user actions. External search also uses SICS, but in this case the goal of the SICS is to propose agents to contact. The pseudocode describing personal agent's actions during the search is shown in Figure 2.

Implicit incorporates the capabilities of having some special agents in the platform. Although each agent encapsulates the ability of contacting the external search engine, e.g. Google, it is also possible to use wrapper agents for transferring the queries to other search engines like Yahoo! or Vivisimo. The Agent Resource Broker (ARB) is the special agent whose main purpose is to provide our agents with links to the services available on other platforms, e.g. wrappers.

The next section presents simulation results obtained using *Implicit*. In this experiment we do not use optional parts of *Implicit* such as wrappers and ARB. Thus, Google is the only external source of the resource links on the platform.

4. EXPERIMENT

In this section we present goal, materials, methods and results of the experiment that was done using the platform. We also define the measures estimating the quality of suggestions produced by SICS.

The aim of the experiment is to understand how the insertion of a new member into the community affects the relevance, in terms of precision and recall, of the links that

Table 1: Scheme of interactions between the system actors within the search session. ***Actor1*** **communicates to** ***Actor2*** **performing the communication act** ***Action*****; Actor1 would like to obtain** ***Target*** **as a result of communication; Actor1 provide** ***Parameters*** **to Actor2; the last column represents protocol or tool within the communication act.**

Actor1	Actor2	Action	Target	Parameters	protocol/tools of communication
user	agent	request	resource-links	keyword	browser, servlets, FIPA Query Interaction Protocol
agent	Google	request	resource-links	keyword	GoogleAPI
Google	agent	inform		resource-links	GoogleAPI
agent	user	inform		resource-links	FIPA Query Interaction Protocol, servlets, browser
agent	agent's SICS	request	resource-links	keyword	java class method call
agent	agent's SICS	request	agent-IDs	keyword	java class method call
agent	DF	request	agent-IDs	——	java class method call
DF	agent	inform	——	agent-IDs	java class method call
agent	agent2	request	resource-links	keyword	FIPA Iterated Contract Net Protocol
agent	agent2	request	agent-IDs	keyword	FIPA Iterated Contract Net Protocol
agent2	agent	inform		resource-links	FIPA Iterated Contract Net Protocol
agent2	agent	inform		agent-IDs	FIPA Iterated Contract Net Protocol
agent	user	inform		resource-links	FIPA Query Interaction Protocol
user	agent	inform		accepted-resource-links	browser, servlets, socket
agent	agent's SICS	inform		accepted-resource-links	java class method call
agent	agent's SICS	inform		accepted-agent-IDs	java class method call
agent	agent's SICS	inform		rejected-resource-links	java class method call
agent	agent's SICS	inform		rejected-agent-IDs	java class method call
agent	agent2	inform		accepted-resource-links	Feedback Protocol
agent	agent2	inform		accepted-agent-IDs	Feedback Protocol
agent	agent2	inform		rejected-resource-links	Feedback Protocol
agent	agent2	inform		rejected-agent-IDs	Feedback Protocol

are produced by SICS. We also want to check the hypothesis that after a certain number of interactions, personal agents will be able to propose links accepted in previous searches.

In our experiment, interaction between agents and users is replaced with interaction between agents and user models that contain user profiles. User profile determines search keywords sequence and acceptance of the results. The results are among the first m links provided by Google for each keyword and the rank of the list is adopted as an identifier. Before the experiment we store the links in a dataset because of the fact that links provided by Google for a certain keyword are reordered very quickly. During simulation we use the dataset instead of contacting Google. User profile is a set of probabilities of choosing a specified link for a specified keyword. The profile is built using n keywords $k_1, k_2, \ldots, k_n$ and determining the probabilities $p(j|k_i)$ of choosing the j-th link, $j \in \{1, \ldots, m\}$ while searching with the i-th keyword. We assume that the user accepts one and only one link during search for the keyword k_i, so $\sum_{j=1}^{m} p(j|k_i) = 1$. The user profile can be seen as a set of association rules with a probability of link acceptance for a given keyword search. In our experiment, the number of keywords n is equal to 10, the number of the links provided by Google m is equal to 10. The user profile is represented in Table 2.

We use the following performance-related notions in order to evaluate the quality of suggestions:

- We call a link **relevant** to a particular keyword if the probability of its acceptance, as specified in the user profile, is greater than some pre-defined relevance threshold.
- **Precision** is the ratio of the number of suggested relevant links to the total number of suggested links, relevant and irrelevant.
- **Recall** is the ratio of the number of proposed relevant links to the total number of relevant links.

We compute recall in a slightly different way. The total number of relevant links is adjusted by adding a number of relevant links proposed by the agents to a number of relevant links presented in the user profile. We do it despite the fact that in reality the links from agents already exist in user profile, because in such a way model of interactions becomes more similar to a real-life situation, where users (and their agents as well) have different collections of links. However, because of this interpretation of recall, the quality of system suggestions is underestimated.

Assuming that all users are members of the same community and have similar interests, the profile for each user is derived from the basic profile given in Table 2 by adding noise. We add noise uniformly distributed in [0.00,...,0.05] to each entry of profile and then renormalize all entries in order to keep the sum of each row equal to 1. According to this procedure we generate 5 different profiles.

From our set of 10 keywords for each agent we generate 25 sequences of 25 keywords by extraction with repetition. Each sequence is used for a search session modelling the query user behavior. We also need to model user acceptance behavior. Given a keyword in the sequence of keywords, accepted result is generated randomly according to the distribution that is specified in the profile. Other links obtained from the agents are marked as rejected.

In a simulation we run 25 search sessions for each agent in the platform. At the end of each session the observation data are deleted. We repeat the search sessions several times in order to control the effect of the order of the keywords and link acceptance. We run 5 simulations for 1,2,3,4,5 agents. With 1 agent in the platform, the agent acts alone without interactions with the others. With 5 agents we have a small community where agents interact with each other. We set the relevance threshold determining link relevance equal to 0.1.

We compute precision and recall of the links proposed by the agents. Line 1 in Figure 4 represents precision of the links that are produced by the personal agent only. The SICS, which is incorporated in the agent, produces these links by analyzing stored observations. Line 2 represents precision of the links proposed by all the agents including

Table 2: Basic profile. **The probabilities of acceptance links for a set of keywords. Links are numbered 1..10.**

						Google rank of the link				
keyword	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
tourism	0	0	0.05	0.4	0.05	0.2	0.1	0.05	0.1	0.05
football	0.05	0	0.1	0.3	0.3	0.1	0.1	0.05	0	0
java	0.35	0.3	0.05	0.05	0.05	0.05	0.05	0.1	0	0
oracle	0.1	0.1	0.45	0.2	0	0.05	0.05	0	0	0.05
weather	0	0.3	0	0	0.5	0	0	0.1	0.1	0
cars	0	0	0.05	0.4	0.05	0.2	0.1	0.05	0.1	0.05
dogs	0.05	0	0.1	0.3	0.3	0.1	0.1	0.05	0	0
music	0.35	0.3	0.05	0.05	0.05	0.05	0.05	0.1	0	0
maps	0.1	0.1	0.45	0.2	0	0.05	0.05	0	0	0.05
games	0	0.3	0	0	0.5	0	0	0.1	0.1	0

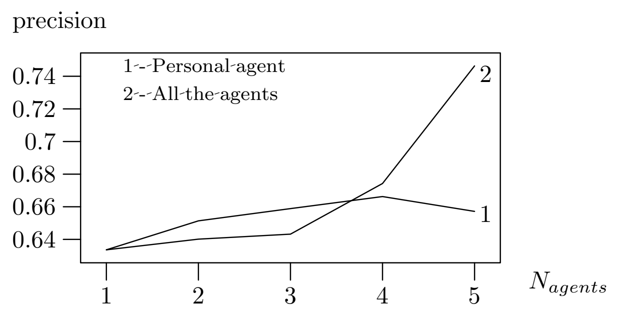

Figure 4: Average precision of 25 simulations with different number of agents.

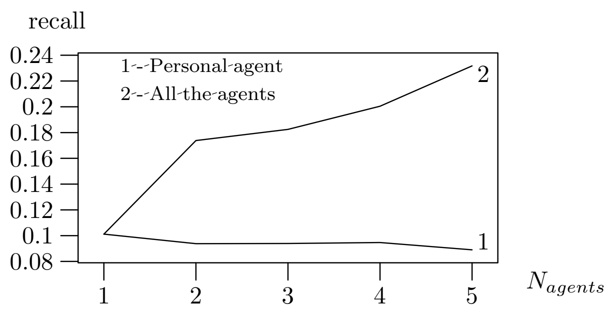

Figure 5: Average recall of 25 simulations with different number of agents.

the personal one. The agents were discovered at the stage of External search or were provided by the DF. In Figure 5 we have analogous curves for recall.

From these figures we can note that the increase of community members causes the increase of the agents' recall. It is probably conditioned by the fact that when we have more agents, we also have more interactions between them. The agents provide each other only one link. So, having growth of the number of links provided by the agents during the search, there is an increase of the percentage of relevant links proposed by the agents and therefore increase of recall. Moreover, recall increase appears without decrease of precision and precision keeps on a rather high level — from 0.63 to 0.75. The value of recall is also rather good and changes from 0.09 to 0.23. We also studied the statistical significance of the difference between agents with the same profile and in different simulations. We performed t-Tests with Bonferroni correction, namely dividing p-value by the number of tests we have performed, in order to control type I error. These tests prove that the average recall for 4 and 5 agents is consistently better ($p < 0.01$) than the average recall of the simulations with smaller number of agents. The results also prove the hypothesis that after a certain number of interactions, agents are able to propose links based on the past user actions.

In other words the obtained results prove that our way of complementing search engine with suggestions, produced as a result of collaboration, makes sense and allows performing web search in a more qualitative way.

We have not run yet experiments for a number of agents bigger than five. Therefore this paper contains only preliminary experimental results. We suppose, though we can not strongly claim that after a number of agents reaches a certain level, the increase of the community members causes only a moderate increment of the performance characteristics.

5. RELATED WORK

In this section we present a brief survey of the related work.

Menczer [19] suggests complementing search engines with online web mining in order to take into account the dynamic structure of the web and to recommend recent web pages which are not yet known by common search engines. To obtain this goal the adaptive population of web search agents united in the multi-agent system emulates user browsing behavior. The system consists of InfoSpiders — agents incorporating neural net inside and analyzing the hyperlinks (and context of the documents corresponding to them) on the currently browsing page in order to propose new documents to the user. So the main goal of this system is the discovery of new information, not yet presented in web search engines, in order to provide more up-to-date service to the user.

Goal-oriented search engine is considered by Liu et. al [18]. Authors suggest not searching by keywords, but by asking normal questions as in everyday life such as "What to do if my pet is sick". The described adaptive system uses a kind of artificially interpreted "common sense", which is stored in the database, in order to produce the answers like "Take it to the veterinarian". Moreover, the system searches the web pages corresponding to the answer — in this case result is homepages of the veterinarians that are the closest to user location.

A multi-agent referral system whose structure is very similar to ours is considered by Yu and Singh [25]. Each user has his/her own personal agent. The agents interact in order to provide the user with answers to his/her question. They are also able to give each other the links to the other agents. There is a complex model of interactions in the system. From agent's point of view the other agents are classified as neighbors and acquaintances and their status in this classification determines the way of contacting them. The system uses ontologies to facilitate knowledge sharing among agents and the ontologies have to be predetermined and shared among all the agents, while we emphasize the implicit support of knowledge by managing documents, links and reference to people. Differently from our system, their agents do not answer all questions but only those are related to their own user interests. The paper is focused more on knowledge (in general) search rather than on web search. Finally, the system is mail-based while *Implicit* is a web based system that adopts FIPA standards and JADE platform.

While we propose using tacit, implicit knowledge accumulated by the group of agents, Turner et. al [23] propose web search agent called FERRET that is able to use an explicit, a priori knowledge in order to improve its search capabilities. FERRET uses such kind of knowledge while elaborating user query and adds context information to his/her query. Further the obtained information is used for the effective search of the scholarly information (only concerning the music) on the web. For this purpose agent interacts with various search engines and content sites. Very simple instance of the pre-defined knowledge is that the user is in a hurry. In this case agent realizes that it is necessary to perform a fast search. One of the main ideas of the paper is specifying the context of the search a priori in order to improve it.

Degemmis et. al [9] present a recommendation system incorporating collaborative filtering and learning user profiles techniques. Thus, this system combines collaborative approach with analyzing web page content. The knowledge about users is represented in user profiles and used within the collaborative filtering algorithm to reduce the time of the recommendation generation.

The collaborative multi-agent web mining system "Collaborative Spiders" is given by Chau et. al [7]. It implements the post-retrieval analysis and implies across-user collaboration in web search. In order to provide the user with recommendations there is a special agent that performs profile matching to find the information potentially interesting to the user. Before the search user has to specify the area of the interest and privacy or publicity of the search. One of the sufficient differences between this system and *Implicit* is that the user should analyze excessive output because he/she has to browse a number of similar already finished search sessions.

Implicit Culture is also related to the notion of social navigation [10]. Both concepts emphasize the social aspect of the interaction between the users even when mediated by artifacts. However, the concept of Implicit Culture is formally defined [5] and it emphasizes the implicit aspects of the interaction.

Implicit is one of the applications of the ideas presented by Blanzieri and Giorgini [3, 5] to the web search area. Preliminary studies related to such an application are presented by Blanzieri et. al [4]. We would also like to mention other uses of these concepts, like [21] in which the concepts of Implicit Culture are applied to support the work of biologists in their laboratories. Adopting this approach the system described in the paper tries to make the knowledge, mainly stored in biologists' notebooks, more useful. There is also work dealing with community of practice that integrates Implicit Culture into community of Java developers at a software development company. Developers while searching information on the company forums receive links from search engine relevant to the subject of interest and also suggestions from other developers' personal agents.

As it appears, there are research studies exploiting ideas similar to those presented in this paper but not dealing with web search. Also there is research that concerns web search but does not use system structure similar to one used in *Implicit*.

6. CONCLUSION AND FUTURE WORK

We described an agent-based recommendation system dealing with the extraction of implicit knowledge from user behavior during web search. The knowledge produced from observations is used in order to suggest links or agents to a group of people and to their personal agents. The main idea is that we do not express this knowledge in explicit form but we use it for improving the quality of further search sessions, including searches performed by new users. Personal agents produce results by asking another personal agent about links and agent IDs. Each agent has the learning capabilities that help to produce results even without interaction. The experience of community members is exploited by means of interactions when the user performs the search already done by someone else. This feature prevents the user from searching "from scratch" and increases the search quality.

The SICS architecture as well as Implicit Culture concepts allow *Implicit* to be a solution to the problem of finding necessary information on the web. One of the main advantages of our approach is represented by the use of both search engine results and suggestions produced by community members. The multi-agent system mimics natural user behavior of asking someone who probably knows the answer. Finally, the process of producing suggestions is completely hidden from the user and therefore does not force him/her to perform additional actions.

There are several directions to improve our system. The composer could take into consideration also balancing between number of acceptances and rejections and it could exploit association rules techniques. The possibility of using association rules mining algorithms for solving recommendation tasks is presented in [17]. In our architecture it is possible to transfer part of the instance-based learning done in the composer module to a rule-based learning in the inductive module. This can be useful in order to improve efficiency and to minimize the storing of the data. Presently, we are conducting some experiments in this direction. Although at the moment user profile contains information only about acceptance and rejection of links obtained from Google, it is possible as future work to build rules for acceptance of links from the other agents. Finally, one of the further directions is to analyze the social relations and interactions between the users.

7. ACKNOWLEDGEMENTS

This research is partially supported by COFIN Project "Integration between learning and peer-to-peer distributed architectures for web search (2003091149_004)". The authors wish to thank Fausto Giunchiglia, Andrea Gastaldello and Claudio Zanoni.

8. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[2] F. Bellifemine and G. Rimassa. Developing multi-agent systems with a fipa-compliant agent framework. *Software - Practice and Experience*, 31(2):103–128, 2001.

[3] E. Blanzieri and P. Giorgini. From collaborative filtering to implicit culture: a general agent-based framework. In *Proceedings of the Workshop on Agents and Recommender Systems*, Barcellona, 2000.

[4] E. Blanzieri, P. Giorgini, F. Giunchiglia, and C. Zanoni. Implicit culture-based personal agents for knowledge management. *Lecture Notes in Artificial Intelligence*, 2926:245–261, 2004.

[5] E. Blanzieri, P. Giorgini, P. Massa, and S. Recla. Implicit culture for multi-agent interaction support. In *CooplS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*, pages 27–39, London, UK, 2001. Springer-Verlag.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[7] M. Chau, D. Zeng, H. Chen, M. Huang, and D. Hendriawan. Design and evaluation of a multi-agent collaborative web mining system. *Decision Support Systems*, 35(1):167–183, 2003.

[8] L. Chen and K. Sycara. Webmate: a personal agent for browsing and searching. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 132–139, New York, NY, USA, 1998. ACM Press.

[9] M. Degemmis, P. Lops, G. Semeraro, M. F. Costabile, O. Licchelli, and S. Guida. A hybrid collaborative recommender system based on user profiles. In *ICEIS: Proceedings of the Sixth International Conference on Enterprise Information Systems*, volume 4, pages 162–169, 2004.

[10] A. Dieberger. Supporting social navigation on the world wide web. *International Journal of Human-Computer Studies*, 46(6):805–825, 1997.

[11] FIPA. foundation for intelligent physical agents. http://www.fipa.org/.

[12] M. Gori and I. Witten. The bubble of web visibility. *Communications of the ACM*, 48(3):115–117, 2005.

[13] Internet systems consortium, inc. internet domain survey. http://www.isc.org/index.pl?/ops/ds/index.php.

[14] iProspect search engine user attitudes survey. http://www.iprospect.com/premiumpdfs/iprospectsurveycomplete.pdf.

[15] H. Ishikawa, M. Ohta, S. Yokoyama, T. Watanabe, and K. Katayama. Active knowledge mining for intelligent web page management. In *Knowledge-Based Intelligent Information and Engineering Systems: 7th International Conference, KES 2003. Proceedings, Part I*, volume 2773 of *Lecture Notes in Computer Science*, pages 975–983, Oxford, UK, 2003. Springer.

[16] H. Lieberman. Letizia: An agent that assists web browsing. In C. S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929, Montreal, Quebec, Canada, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[17] W. Lin, S. A. Alvarez, and C. Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, 6(1):83–105, 2002.

[18] H. Liu, H. Lieberman, and T. Selker. Goose: A goal-oriented search engine with commonsense. In *AH '02: Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 253–263, London, UK, 2002. Springer-Verlag.

[19] F. Menczer. Complementing search engines with online web mining agents. *Decision Support Systems*, 35:195–212, 2002.

[20] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM Press.

[21] M. Sarini, E. Blanzieri, P. Giorgini, and C. Moser. From actions to suggestions: supporting the work of biologists through laboratory notebooks. In *Proceedings of 6th International Conference on the Design of Cooperative Systems (COOP2004)*, pages 131–146, French Riviera, France, 2004. IOSPress.

[22] G. L. Somlo and A. E. Howe. Using web helper agent profiles in query generation. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 812–818, New York, NY, USA, 2003. ACM Press.

[23] R. M. Turner, E. H. Turner, T. A. Wagner, T. J. Wheeler, and N. E. Ogle. Using explicit, a priori contextual knowledge in an intelligent web search agent. In *CONTEXT '01: Proceedings of the Third International and Interdisciplinary Conference on Modeling and Using Context*, pages 343–352, London, UK, 2001. Springer-Verlag.

[24] Y. Z. Wei, L. Moreau, and N. R. Jennings. Recommender systems: a market-based design. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 600–607, New York, NY, USA, 2003. ACM Press.

[25] B. Yu and M. P. Singh. An agent-based approach to knowledge management. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 642–644, New York, NY, USA, 2002. ACM Press.