UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

IMPLICIT: A RECOMMENDER SYSTEM THAT USES IMPLICIT KNOWLEDGE TO PRODUCE SUGGESTIONS.

Alexander Birukov, Enrico Blanzieri and Paolo Giorgini

April 2005

Technical Report # DIT-05-038

Implicit: A recommender system that uses implicit knowledge to produce suggestions

Content areas:

The communication between agents (negotiation, coordination, sharing knowledge between agents)
Collaborative filtering, collaborative information retrieval, content filtering or case-based reasoning
The architecture of multi-agent systems (design, methodology and principles)

Abstract

The number of accessible web pages in Internet increases every day and it becomes very difficult to deal with such a huge source of information. There exist several approaches aimed to provide user with high-quality links extracted from the thousands of irrelevant ones. We present a system *Implicit* that combines recommender system and multi-agent system approaches and is intended to be used within community of people with similar interests. It produces suggestions by using implicit knowledge of the members of a community and complementing these suggestions with results from search engines. Agents within the system interact one another and share knowledge in order to increase quality of the recommendations by using similarities in the behavior of different users.

1 Introduction

Although searching the Internet is a day-to-day task of many people, the problem of improving the quality of web search, in particular, providing effective access to information available on-line, is still open. This problem arises as the result of the huge number of pages on the World Wide Web. Because of the vast quantity of information available, it is not a problem to find pages, but it is difficult to discover really relevant and (or) interesting pages among those provided by a search engine. Therefore, web search often results in a rather time-consuming task.

There exist several approaches aimed to solve the stated problem. Search engines are a common and prevailing tool for searching the Web. However, they have several shortcomings. For instance, a query may results in a huge quantity of the pages. Another drawback is a lack of personalization in a sense that sometimes "different users may merit different answers to the same query" [Gori and Witten, 2004]. The first shortcoming could be alleviated by formulating an appropriate query for a search engine. Such a reformulation of the query requires certain intuition and experience from the user. What concerns the lack of personalization, we see the need of supporting the concrete user, and not just responding to the keyword which is context-free and impersonal.

Another solution is the use of Internet agents for assisting the web browsing. In this field, we find personal assistants that collect observations of their users' behavior in order to recommend new, previously unseen web pages that are relevant to users' queries. There exist also multi-agent systems, where personal assistants collaborate one another for improving the quality of the suggestions. The Internet agent approach overcomes the shortcomings of the search engine approach from the personalization point of view. On the other hand, there are other drawbacks like the low number of suggestions generated or even the absence of them in the case of a keyword that has been previously unseen for the personal assistant agent. Sometimes personal assistants require extra efforts from the user, e.g. specifying his/her area of interests or answering additional questions.

Recommender systems can be also considered as tools for the effective access to available information. They can be classified as content-based, collaborative filtering and hybrid systems. Content-based systems produce recommendations by analyzing the content of previously browsed pages and using the obtained information to find pages with similar content. Collaborative filtering systems calculate similarity between the different users and provide the user with the pages that have been selected by the similar users. Hybrid recommender systems exploit both approaches to a certain extent. However, majority of the recommender systems needs user feedback and those systems that collect this feedback in explicit form force user to perform some extra work like rating the items.

In this paper we present *Implicit*, a multi-agent recommender system. It combines Internet agents and a recommender system. *Implicit* uses a search engine in order to obtain a certain number of suggestions for any entered keyword. Personal agents communicate and collaborate in order to produce recommendations more suitable in the context of the current community. So, we complement search engine results with recommendations produced by the agents. This helps to add personalization without decreasing significantly the number of the pages. As in many recommender systems we attempt to learn the user needs from the observations of his/her behavior. One of the uncommon features is that we use universal framework to produce different types of suggestions: links, which are shown to the user, and agents IDs, which are used internally and hints the user's agent who it would be useful to contact. In order to access the information provided by the system, the user does not need to install ad-hoc plugins or a new browser, it is just necessary to register and then load the system homepage. Moreover, we use implicit feedback collection mechanism and no additional work is required from the users.

The rest of the paper is structured as follows. Section 2 describes the *Implicit* system in detail and Section 3 contains some experimental results on the use of our system. Final Sections 4 and 5 reviews related work and concludes the paper, respectively.

2 Structure of the System

In this section we present a detailed description of *Implicit*, a multi-agent recommender system. The system exploits the notion of Implicit Culture [Blanzieri and Giorgini, 2000] in order to produce suggestions by using peculiarities of the community where it works. Each user of the system has dedicated personal agent whose task is to assist the user during his/her search and to provide him/her with the links in response to the entered keyword. For this purpose agents contact a search engine. Agents also produce recommendations by means of the Systems for Implicit Culture Support (SICS) module. This module uses implicit knowledge of the community members to find links that are considered relevant. The framework that produces these links is universal in a sense that it is also used for discovering which agents it would be useful to contact in order to obtain more relevant links. The architecture of the system is represented in Figure 1.

Implicit consists of the client part and the server part. There is an html/php user interface on the client side. On the server side there are Java servlets and multi-agent platform implemented using JADE (Java Agent Development Framework) [Bellifemine *et al.*, 2001]. JADE is a framework for developing multi-agent systems according to FIPA [1] standards. Here we present basic terms used in JADE and in our system.

Personal agent is an agent running on the server side that receives search tasks from its user and then produces recommendations in response to the query. The process of generating suggestions consists of several parts, implemented as behaviors. *Behavior* is a procedure that implements tasks, or intentions, of an agent. The agent is able to execute it in response to different internal and external events. Behaviors are logical activity units that can be composed in various ways to achieve complex execution patterns and that can be concurrently executed. *Scheduler* is an internal agent component that automatically manages the scheduling of behaviors and determines which behavior to run now and what action to perform as a consequence.*Inbox* is a queue of incoming messages (ACL) from the user and from other agents. In order to produce recommendations agent uses its *resources* that consist of *beliefs* and *capabilities*. Agent's beliefs are the information available to the agent (e.g. information on user actions) and the capabilities are particular functionalities used in the behaviors (e.g. the SICS module). The structure of the personal agent is represented in Figure 2.

[1]FIPA. Foundation for Intelligent Physical Agents. `http://www.fipa.org/`.

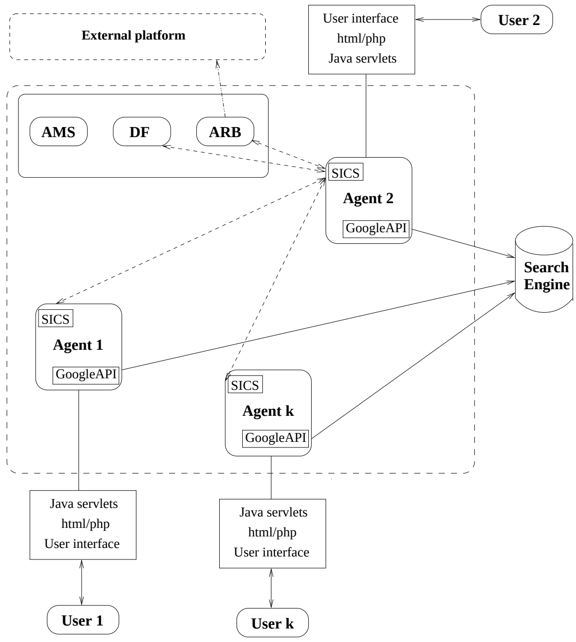

Figure 1: Architecture of the system. *Personal agents* process queries from *users* and interact with each other to exchange links; *Agent Management System (AMS)* controls the platform. It provides agent registration, search, deletion and other services; *Directory Facilitator (DF)* provides agents with other personal agents' IDs. *Agent Resource Broker (ARB)* deals with links to the services available on other platforms.

When an agent receives a query message from the user, it starts search behavior that consists of Google search behavior and Platform search behavior. Platform search behavior comprises Internal search behavior and External search behavior. During Google search behavior the agent process query to Google search engine [Brin and Page, 1998] using Google Web API. As soon as the agent receives the answer, it shows the obtained links to the user and starts Internal search behavior. The SICS module is used during the search in order to produce two kinds of suggestions: a http, ftp or resource link and an ID of the agent to contact. In Internal search the goal of the SICS module is to generate links based on the past user actions. If this step fails, then the SICS tries to create recommendation using internal agent resources such as the *local schema* that will be described below. All the generated links are stored in the memory and External search behavior is started. This behavior also uses the SICS but in this case the goal of the SICS is to propose agents to contact. If there are no suggestions then agent contacts Directory Facilitator. Directory Facilitator (DF) according to FIPA standards is a special agent that provides yellow pages service on the agent platform. Actually, in our case, DF simply provides the agent with the IDs of other personal agents on the platform. Having filled the list of agents to contact, personal agent starts interaction — it sends query to every member of the list. When all the agents are contacted we query new agents that were suggested during the search and so on. When all suggested agents are asked and they answered we show all the obtained links to the user. In the present implementation, the agent

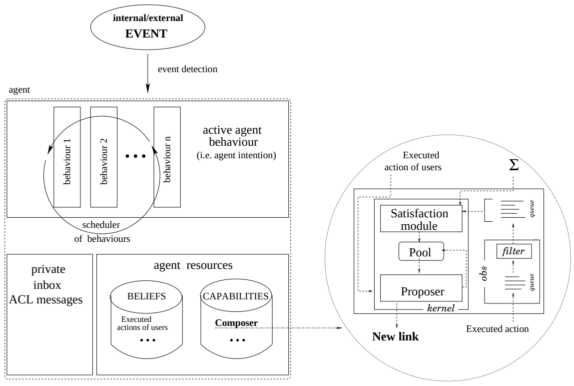

Figure 2: Internal architecture of personal agent

Agent executes *behavior* in response to different internal and external events. *Scheduler* manages execution of the behaviors. ACL messages received from the user or from other agents are stored in *inbox*. *Resources* consisting of *beliefs* (information available to the agent) and *capabilities* (available functionality) are used to produce suggestions. *Satisfaction module* selects links to the *pool* using behavior patterns produced by *inductive module* from the observations on executed actions. *Proposer* selects the best link from the pool.

performs the three types of search in the following order: first Google search, then Internal search and finally, External search. Agents may also query each other, in this case the respondent does not use the capability of contacting a search engine, because the questioner has this capability too. Agent-responder runs Internal search behavior in order to produce, using its own observation history, links that the user of the agent-questioner will probably accept. It also starts External search behavior in order to recommend to the questioner other agents to contact. The techniques used within these two behaviors are the same and are implemented within the SICS module.

The basic architecture for the SICS is shown in Figure3 and consists of the following three basic components: *Observer module* is the part of the SICS that watches and records the actions performed by the user during the use of the system. The next component, *inductive module*, analyzes the stored observations and implements data mining techniques to discover patterns in the user behavior. And, finally, *composer* 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.

The SICS architecture requires the solution of two learning problems. A problem of browsing patterns learning (inductive module) and a problem of prediction of links the user will accept (composer). The inductive module problem is a rather standard learning problem: inducing the behavior patterns of the groups. The problem is not solved yet. The solution of the composer problem exploits the principles of instance-based learning (namely, memory-based or lazy). For more general description of these two problems see [Blanzieri *et al.*, 2004].

The structure of the SICS allows to find out relevant links from the observations and to discover relevant agents using the same mechanism. For producing suggestions the SICS adopts calculation of the similarity between the community

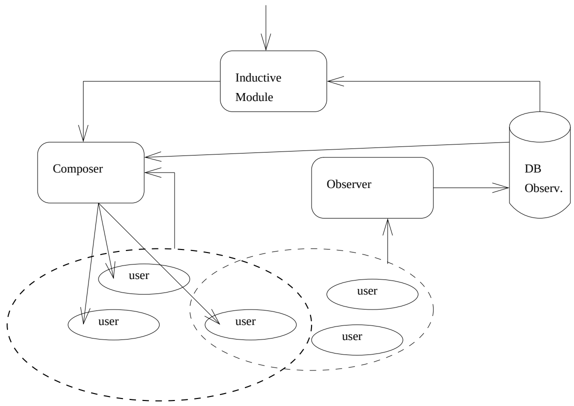

Figure 3: The basic architecture for the Systems for Implicit Culture Support consists of the following three basic components: *observer* that stores in a database (DB) the information about the executed user actions in order to make it available for other components; *inductive module* that analyzes the executed actions in order to discover patterns of user behaviors; *composer* that produces the links to suggest the user

members and therefore personalizes web search of the user to some extent.

Agents use Agent Communication Language (ACL) and standard FIPA protocols for link and agent ID exchange. There is also a feedback protocol for the exchange of information about accepted/rejected links. A feedback from one agent to another is sent as the result of the user browsing behavior. We illustrate the use of communication protocols by the following short example. For instance, a user searches information about *"train timetable"* and asks his/her personal agent, *pagent*. *Pagent* starts Google search, Internal and External searches. Since Google search is finished the user has information about the links (we consider only the first three links for this example) provided by Google: *www.nationalrail.co.uk/planmyjourney*, *www.thetrainline.com* and *www.railtrack.co.uk*. Then Internal search is started in which the SICS module uses data mining techniques to select agents that performed similar actions and then selects the link accepted for the keyword *"train timetable"* by the agent with the highest similarity. During External search behavior the SICS module selects agents that performed similar actions and chooses such an agent that it is likely to propose link that will be accepted by the *pagent*'s user. Let us suppose that SICS suggested the link *www.fs-on-line.it* during Internal search and another agent to contact, *agent1*, during External search. The personal agent sends a request to *agent1* using FIPA Iterated Contract Net Protocol. *Agent1* receives the request from *pagent* and uses its SICS module in order to produce suggestions. Let us consider that Internal search behavior of *agent1* produced the link *www.trenitalia.it* selected from the links accepted by the *agent1*'s user in the past. As a result, *pagent* receives the link *www.trenitalia.it* and shows it to the user. If the user accepts the link *www.trenitalia.it* then *pagent* stores the information that this link is accepted and sends this information (using feedback protocol) to *agent1* because it provided *pagent* with *www.trenitalia.it*. When the user leaves *Implicit* or starts a new search all the not accepted links are considered to be rejected and all the agents involved in the dialog receive the communication. In our example, if the user does not accept *www.trenitalia.it* then *agent1* receives the message that this link is rejected. One of the benefits of our approach is that feedback is collected without any effort from the user, such as giving ratings to the items or specifying his/her interests.

The system uses the *local schema* in order to represent the knowledge about the links that the user accepted in the past. Basically, it is a tree with user interests in the nodes and links in the leaves. The schema is stored in XML. The example of *local schema* one can find in [Blanzieri *et al.*, 2004]. It is not the only source of the locally available knowledge — one can use some other variants such as "yellow pages" reference or his/her own bookmarks.

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

3 Experimental Results

In this section we present the experimental results obtained with the proposed platform. We also define the measures (precision and recall) estimating the quality of the recommendations produced by the 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 produced by the 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 by interaction between agents and models of users, namely sequences of search keywords and results about acceptance. 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. Due to the fact that links provided by Google for a certain keywords are reordered very quickly, before the experiment we store the links in a dataset. During the simulation we used 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 acceptance of a certain link 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 1.

Table 1: 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

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

- Link is considered to be **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 relevant links suggested to the total number of irrelevant and relevant links suggested.
- **Recall** is the ratio of the number of relevant links proposed 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 the agents already exist in the 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, with such an interpretation of recall, the quality of system suggestions is underestimated.

Assuming that all the 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 1 by adding noise. We add noise uniformly distributed in [0.00,...,0.05] to each entry of the profile and then renormalized entries in order to keep the sum of each row equal to 1. Following 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 user query behavior. We also need to model the user acceptance behavior. Given a keyword in the sequence of keywords, accepted result is generated randomly according to the distribution 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 were 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

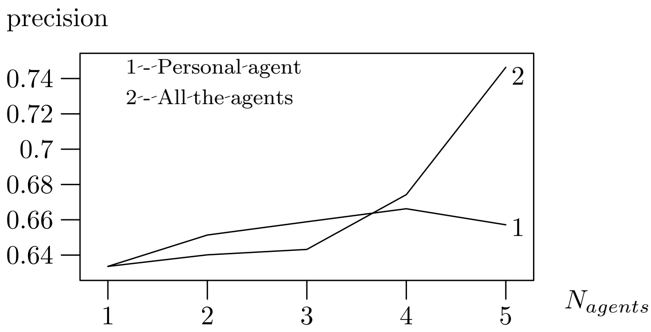

Figure 4: Average precision of 25 simulations with different number of agents.

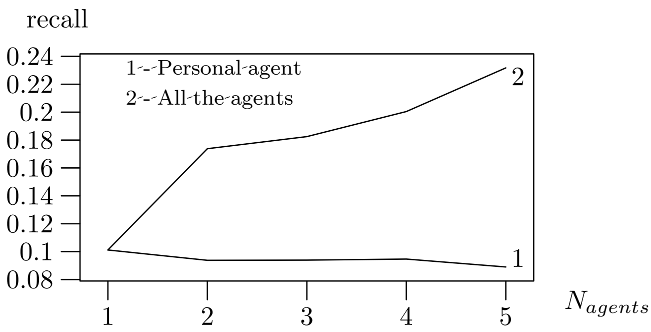

Figure 5: Average recall of 25 simulations with different number of agents.

relevance threshold used to determine the relevance of links equal to 0.1.

We compute precision and recall of the links proposed by the agents. In Figure 4 line 1 represents precision of the links produced by the personal agent only. The SICS module incorporated in the agent produces these links by analyzing stored observations. Line 2 represents precision of the links proposed by all the agents including the personal one. The agents were discovered at the External search stage or 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 with 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 an increase of recall. Moreover, the increase of recall appears without a decrease of precision and the 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 recommendations, produced as a result of collaboration, makes sense and allows performing web search in a more qualitative way.

For the moment we did not run yet any experiment for a number of agents bigger than five. However, we suppose that after a certain number of agents, increasing the number of the community members will cause only a moderate improvement of the performances.

4 Related Work

In this section we briefly discuss the papers related to our work.

A market-based recommender system is presented by Wei et. al [2003]. It is a multi-agent system where agent acts on behalf of its user and sells the slidebar space where recommendations can be displayed. Other agents participate in this auction in order to show their links on this slidebar. The agent-initiator of the auction chooses the most profitable offers and displays them to the user. Providers of the links accepted by the user receive reward. Agents adopt multiple heterogeneous recommendation methods and try to make better suggestions in order to increase their profit.

A multi-agent recommender system is considered by Yu and Singh [2002]. MARS is a referral system for knowledge management that assigns software agent to each user. The agents interact in order to produce answers to the queries of their users. The agents are also able to give each other referrals to other users. There is a complex model of interactions in the system in a sense that it is important from who the query comes — there could be a different set of actions for the different agents. The system uses pre-determined ontologies, shared among all the agents, to facilitate knowledge sharing between them, while we emphasize the implicit support of knowledge by managing documents, links and references to people. Differently from our system, the agents do not answer all questions but only those 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.

Degemmis et. al [2004] present a recommender system that incorporates collaborative filtering techniques and learning user profiles techniques. Thus, this system combines collaborative approach with content-based approach. The knowledge about users is represented in user profiles and used within the collaborative filtering algorithm to reduce the time of the recommendation generation.

A collaborative multi-agent web mining system "Collaborative Spiders" is given by Chau et. al [2003]. There are different types of agents responsible for retrieving web pages, performing post-retrieval analysis, interacting with users, sharing information about user search sessions, performing profile matching and carrying out retrieval and analysis tasks according to a schedule. Before search the user has to specify the area of the interests 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 looking through a number of similar already finished search sessions.

Zhu et. al [2005] present WebICLite - a recommender system that uses behavior models to predict relevant web pages. They conceptualize web browsing as a search for a specific well-defined information need and make assumption that this need can be identified from the pages the user visits and the actions that he/she applies to the pages. Several specific algorithms for identifying information-need-revealing patterns are considered and compared. These algorithms are used in order to turn the inferences about user information needs into the queries for a standard search engine which does the actual retrieval of recommended pages.

Macedo et. al [2003] apply a recommender system approach not only to support user navigation on the Web, but to assist and to augment the natural social process of asking for recommendations from other people. WebMemex is a system that provides recommendations based on the browsing history of the people well-known to the users. In order to obtain the list of such users, a contact list from Yahoo Messenger is used. The system allows the user to keep privacy of web search by hiding his/her browsing for a certain time. The recommendations generated within the system are based on the links between the related documents visited by the users.

5 Conclusion and Future Work

In this paper we have presented an agent-based recommender system that extracts implicit knowledge from user browsing behavior. The knowledge is necessary to suggest links or agents to a group of people and to their personal agents. Personal agents use universal mechanism of producing suggestions about links and agents IDs. Learning capabilities are used by agents to produce results even without interaction. Interactions allow a user to use the already acquired experience of members of his/her community. This increases the quality of the search. The process of collecting feedback and producing recommendations is completely hidden from the user and therefore does not require any kind of extra work from the user.

Implicit can be modified in several ways. It could be enhanced with the capability of analyzing content of visited web pages. In such a way it would combine content-based and collaborative approaches. Classification of the users on “experts” and “novices” could also be implemented in order to take into account information about the author of the recommendation.

References

[Bellifemine *et al.*, 2001] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with a fipa-compliant agent framework. *Software - Practice and Experience*, 31(2):103–128, 2001.

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

[Blanzieri *et al.*, 2004] Enrico Blanzieri, Paolo Giorgini, Fausto Giunchiglia, and Claudio Zanoni. Implicit culture-based personal agents for knowledge management. *Lecture Notes in Artificial Intelligence*, 2926:245–261, 2004.

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

[Chau *et al.*, 2003] Michael Chau, Daniel Zeng, Hsinchun Chen, Michael Huang, and David Hendriawan. Design and evaluation of a multi-agent collaborative web mining system. *Decis. Support Syst.*, 35(1):167–183, 2003.

[Degemmis *et al.*, 2004] Marco Degemmis, Pasquale Lops, Giovanni Semeraro, Maria Francesca Costabile, Oriana Licchelli, and Stefano Guida. A hybrid collaborative recommender system based on user profiles. In *ICEIS (4)*, pages 162–169, 2004.

[Gori and Witten, 2004] Marco Gori and Ian Witten. The bubble of web visibility. *Communications of the ACM*, 2004.

[Macedo *et al.*, 2003] Alessandra Alaniz Macedo, Khai N. Truong, Jose Antonio Camacho-Guerrero, and Maria da Graca Pimentel. Automatically sharing web experiences through a hyperdocument recommender system. In *HYPERTEXT ’03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 48–56, New York, NY, USA, 2003. ACM Press.

[Wei *et al.*, 2003] Yan Zheng Wei, Luc Moreau, and Nicholas 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.

[Yu and Singh, 2002] Bin Yu and Munindar 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.

[Zhu *et al.*, 2005] Tingshao Zhu, Russ Greiner, Gerald Haubl, Bob Price, and Kevin Jewell. Behavior-based recommender systems for web content. In *IUI ’05: Proceedings of the 10th international conference on Intelligent user interfaces. Workshop: Beyond Personalization 2005*, New York, NY, USA, 2005. ACM Press.