Auton Agent Multi-Agent Syst (2012) 24:141–174
DOI 10.1007/s10458-010-9148-z

Implicit: a multi-agent recommendation system for web search

Aliaksandr Birukou · Enrico Blanzieri · Paolo Giorgini

Published online: 20 August 2010

Abstract For people with non-ordinary interests, it is hard to search for information on the Internet because search engines are impersonalized and are more focused on "average" individuals with "standard" preferences. In order to improve web search for a community of people with similar but specific interests, we propose to use the implicit knowledge contained in the search behavior of groups of users. We developed a multi-agent recommendation system called Implicit, which supports web search for groups or communities of people. In Implicit, agents observe behavior of their users to learn about the "culture" of the community with specific interests. They facilitate sharing of knowledge about relevant links within the community by means of recommendations. The agents also recommend contacts, i.e., who in the community is the right person to ask for a specific topic. Experimental evaluation shows that Implicit improves the quality of the web search in terms of precision and recall.

Keywords Implicit Culture · Multi-agent system · Personal agents · Recommendation system · Web search · Collaborative search · Communities

1 Introduction

Internet contains a lot of answers to our everyday questions and search engines are aimed at helping us to find the answers in a set of relevant links. However, results produced by search engines are mostly impersonalized [57] and satisfy needs of "average" users. If interests of a user are specific, the most relevant link might not be among the top 10 shown by conventional search engines. As stated by Gori and Witten [28]

> […]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[…].

A. Birukou (✉) · E. Blanzieri · P. Giorgini
University of Trento, Trento, Italy
e-mail: aliaksandr.birukou@disi.unitn.it

In the literature this problem is addressed using Internet agents, recommendation systems and community-based search. *Internet agents* monitor user browsing behavior, learn preferences and build profiles of users to assist in their web browsing [16,39]. Coalitions of agents are also used for answering queries of single or multiple users [15,44] and specific mechanisms such as auction protocols and reward techniques are applied to implement collaboration among agents [58,59]. In order to personalize recommendations, *recommendation systems* analyze user queries, the content of the visited pages, and/or implicit and explicit indicators of satisfaction in order to extract knowledge about user needs and patterns of behavior. Recommendation systems are usually classified as *content-based* systems, which analyze the content of web pages [16,52,54], *collaborative filtering* systems [30,33,38,41], which produce recommendations based on the similarity of users, and *hybrid* systems that combine the two approaches [4,14,45]. Although groups of users can have common interests or deal with similar problems, Internet agents and recommendation systems usually focus on isolated users. Differently, in the research on community-based web search (e.g., I-SPY [51], Beehive [32], and other systems [2,25]) the focus is on the preferences of the community rather than those of a single user.

In the majority of solutions developed to date, explicit feedback from the user is required. This means that after receiving search results users must evaluate them, e.g., by rating, or ranking. This requires an additional effort from users and, therefore, explicit feedback is often discouraged [49]. Furthermore, sometimes users are inconsistent in the explicit ratings provided [36]. All these suggest that implicit indicators of user interests should be exploited. Moreover, the study by Fox et al. [26] has shown that implicit measures can be a suitable alternative to explicit feedback. All these indicate that systems supporting web search in communities of like-minded users with specific interests are required. Moreover, the systems should use implicit feedback where possible and provide means for sharing search experience with the community members, i.e., the content found relevant by someone should be immediately available for others submitting similar queries. The goal of such systems should be to improve the quality of web search for the community.

In this paper, we present a multi-agent recommendation system called Implicit, which is intended to support the web search of communities of people working together (e.g., a project team, PhD students of the same department, a community of practice, organizational communities). Such communities have specific common interests related to their activities. Even though Web 2.0 provides a lot of tools for representing explicitly such communities (Facebook, LinkedIn, to name a few), these tools not necessarily provide support for web search. Our system is intended to be used in such communities for the purpose of improving their search experience. The system can increase quality, in terms of precision and recall, of search in small communities, supporting collaboration of the community members and sharing experience about using particular web links relevant to their specific interests.

Implicit aims at helping such communities to share their history of searches to recommend links relevant to their interests. This helps to improve community search experience and also to help novel community members to adapt to the new community quicker. Users submit their queries to the system and Implicit suggests specific links and people to contact. To produce recommendations relevant to community's specific interests, the system uses implicit feedback, namely, observed behavior of the members of the community. More specifically, it exploits previous observations about the behavior of other users after they submitted similar queries. Each user has a personal agent that interacts with the personal agents of other users to produce recommendations. The system implements a hybrid recommendation approach, providing users with the suggestions from and about the community members (collaborative recommendations) and with the results obtained from Google (content-based

recommendations). The system allows for the exploitation of social interactions between community members, i.e., by their personal agents, in order to increase the quality of recommendations. Personal agents represent their users in the system, tracking their interests and browsing behavior with respect to using the links and contacts. Thus, Implicit also allows for shifting the burden of the collaboration task, namely, answering queries from other users, from the user to the personal agent of the user. The system is based on the concept of Implicit Culture. Implicit Culture [13] is a generalization of Collaborative Filtering [38], which is a technique of producing personal recommendations using similarities between user ratings. Implicit Culture helps new community members to behave similarly to the other members without the need of expressing explicitly the knowledge of the community.

The following are the main contributions of the paper. The first contribution is architecture and an implemented prototype of a multi-agent system for collaborative web search. The second contribution is the unique recommendation framework that allows for developing recommendation systems by using social interactions. The recommendation framework filters not only links, but also people that make suggestions, thus establishing implicit trust relations between users. Moreover, filtering is based on the activities of people and allows for producing suggestions for an emergent community, discovered by the system "on the fly", even though the system is more intended for the use in an a priori defined community of people. The framework generalizes widely used Collaborative Filtering and allows for filtering not only ratings, but also actions in general. This paper is an extension of the previous work of the authors [8]. With respect to the previous work, the description of the system and the recommendation mechanism has been extended, more information on the evaluation methodology has been presented, and new experimental results have been added.

The paper organized as follows: we start with a discussion of related work in Sect. 2. Section 3 briefly describes the concepts of Implicit Culture and gives the description of Systems for Implicit Culture Support. Section 4 provides an overview of the Implicit system. We go into detail of the internal architecture of the system in Sect. 5 and evaluate the system in Sect. 6. Finally, we discuss the proposed approach in Sect. 7 and conclude the paper in Sect. 8.

2 Related work

In this section, we review the related work. For convenience, we grouped the related work in several areas describing system for recommending contacts, social navigation, community-based search engines, swarm intelligence, agent-based, and other related approaches.

2.1 Recommending contacts

Vignollet et al. in [56] described a recommendation system that adopts the collaborative filtering and social networks analysis techniques. The system recommends contacts instead of contents. The idea behind contact recommendations is that users prefer others' advice to impersonal guidance and also appreciate enriching relationships with others. This system is similar to our work in the sense of taking into account the social aspect of the information search.

A multi-agent referral system MARS has been presented by Yu and Singh [60]. In that system, each user has a personal agent. The agents interact in order to provide users with answers to their questions. Agents are also able to give each other the links to the other agents, which is similar to recommending agent IDs in Implicit. There is a complex model of agent

interactions in MARS. Each agent classifies the other agents 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. The ontologies must be pre-defined and shared among all the agents, while we emphasize the facilitation of implicit knowledge sharing by managing documents, links and reference to people. Differently from our system, the agents in MARS do not answer all questions of other agents, but only those related to the interests of their users. The paper is focused more on general knowledge search rather than on web search. Finally, the system is mail-based while Implicit is a web-based system that adopts FIPA standards and uses the JADE platform.

Another multi-agent referral system described by Singh et al. [50] is similar to Implicit: agents send answers to queries and also give referrals to other agents. The scope of their system is on service provision in general, independently of the kind of service agents provide, while Implicit focuses only on web search. Systems also differ in the way of modeling expertise of agents (or their users): agents in the system of Singh et al. explicitly maintain the expertise of their users and the expertise of neighbors, constantly updating the list of knowledgeable peers, while agents in Implicit do not model expertise of their users and neighbors explicitly, but get this information from the history of past actions. Also, agents in Implicit try to answer each query, not only those related to the expertise of their users (as in the system of Singh et al.).

2.2 Social navigation

Implicit Culture in general and applied for web search in particular is related to the notion of Social Navigation. Initially introduced in [20] and further developed by Dieberger et al. [19] it has been also applied to the problem of the navigation on the Internet [18].

Essentially, Implicit Culture and Social Navigation are very close and have been applied in similar settings. Implicit Culture does not cover all the scope of the Social Navigation problems and it has been formally defined [7] as a relation between groups of agents. The definition emphasizes the transfer of knowledge and behavior between groups and allows one to evaluate whether the Implicit Culture relation arises from some system usage. It can be argued that some effective Social Navigation systems produce such a relation. On the other hand, other Social Navigation systems, for instance, those supporting awareness (e.g., social proxies [24]) do not necessarily produce the Implicit Culture relation. We argue that it is possible to use Implicit Culture as a tool for building, assessing and evaluating Social Navigation systems.

The main problem the Social Navigation approach aims at is guiding people to relevant information, while the Implicit Culture approach aims at achieving the transfer of knowledge and behavior from one group to another. In this perspective, the scope of the Implicit Culture approach seems to be more general than just dealing with the transfer of information between groups. However, when applied in the field of recommendation systems, two approaches look very similar.

2.3 Community-based engines

The Implicit system is also related to community-based search engines, like I-Spy [51], Eurekster,[1] and to social bookmarking services, such as Delicious.[2] However, the Implicit system differs from these systems in several aspects. First, Implicit Culture focuses more

[1] http://www.eurekster.com/. Social search technology.

[2] http://delicious.com/. Social bookmarking.

on an organizational community, rather than on an emergent or online one. Second, it uses collaboration and interactions among agents to improve suggestions. Third, it recommends also agents, therefore establishing implicit trust relationship in the community. Finally, our system can be used to filter and re-rank the results from systems such as Delicious to a specific user community. It would be possible to do this by analyzing the similarity between links from Delicious and links accepted by the community in the past.

2.4 Swarm intelligence

The notion of cultural theory in the Implicit system is related to swarm intelligence theories, in particular, to trail laying, or ant foraging. These theories have been applied in Social Navigation systems [22,55] to guide people such as learners, to relevant information using the data from previous interactions with the system. Unlike these approaches, which suggest that the user follow trails taken by the majority, in the Implicit Culture approach the user's actions are compared with actions of the whole community and not necessarily the most popular ones are suggested. More precisely, taking into account user's past actions, the system can offer actions which are less popular in general, but are common among the part of the community which is the most similar to the user.

2.5 Agent-based systems for improving web search

Menczer [44] 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 achieve this goal the adaptive population of web search agents united in a multi-agent system emulate user browsing behavior. The system consists of InfoSpiders, which are the agents incorporating neural net and analyzing the links and the context of the documents corresponding to the links on the current page in order to propose new documents to the user. 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.

A collaborative multi-agent web mining system called Collaborative Spiders was developed by Chau et. al [15]. The system implements the post-retrieval analysis and enables across-user collaboration in web search. In order to provide a user with recommendations a special agent performs profile matching to find the information potentially interesting to the user. Before the search, the user has to specify the area of the interest and privacy or publicity of the search. Unlike to Implicit, in the Collaborative Spiders system the user should analyze excessive system output because he/she has to browse a number of similar already finished search sessions.

SurfAgent [52] is an information agent that builds a user profile by using user-supplied examples of relevant document. The authors presented and evaluated the mechanism of automatic query generation from the user profile and using the generated queries to provide relevant documents to the user. Such approach of pro-active searching for documents that might be interesting for the user is called the "push" approach. Implicit applies the "pull" approach where recommendations are delivered to the users only when they search. Also, we do not represent information about user searches explicitly, as user profiles. Therefore, query generation from user profiles is not applicable in our system.

AgentSeeker [47] is a multi-agent platform for indexing local and online textual files. It implements FIPA standards and aims at improving search in enterprises. AgentSeeker indexes and retrieves documents using a pre-defined or collaboratively created ontology for

representing the domain knowledge. Implicit and AgentSeeker are similar in the goal of finding information relevant to user's query, but differ in the approaches. Implicit does not deal with crawling and indexing documents, delegating this task to a conventional search engine and augmenting results with recommendations from SICS.

2.6 Other related work

The Implicit system can be used in order to increase quality of search in small communities, supporting collaboration of the community members and sharing experience about using particular web links relevant to their specific interests. In this regard, Implicit is complementary to the work by Geczy et al. [27] who investigated patterns in browsing behavior of a community of knowledge workers.

A recommendation model presented in [57] produces recommendations by using the social network existing between users and modeling the trust relationships with neighbors. The topic of using trust in recommendation systems is deeply investigated in papers by Massa (see, e.g., [43]). Differently to such systems, in Implicit we do not model the social network and trust relationships explicitly. However, trust relations and social ties emerge from interactions between agents. In the conducted simulations we noted that after a certain number of queries, the SICS of each agent mainly contacted only one single agent, who gave the most relevant recommendations in the past.

In the broad sense, the architecture of the Implicit system is similar to peer-to-peer networks, if we imagine that agents can run in different hosts (JADE allows this) and different instances of Implicit system can communicate with each other. Lopes and Botelho [42] present a survey of recent work on the integration of multi-agent systems and peer-to-peer computing for resource coordination. Such coordination includes the resource discovery with web search being just an example of it.

Implicit is one of the applications of the Implicit Culture ideas [7,13] to the web search area. However, we have applied the Implicit Culture ideas also in other domains, for instance, recommendation of web services [11], software patterns [9] and supporting the work of biologists in their laboratories [48]. The detailed description of the recommendation framework was published in [10]. We are currently working on formal definitions of culture and Implicit Culture, and preliminary results were presented in [7,12].

3 Background

This section starts with an overview of the general idea of the Implicit Culture approach that we use to produce recommendations. Then we describe the architecture of a System for Implicit Culture Support (SICS), which implements Implicit Culture ideas in our system.

3.1 Implicit Culture

A group of *agents* working together as a community have and exploit a great amount of knowledge and skills. These skills and knowledge allow agents to behave in an optimal way, i.e. to perform tasks and to solve problems faster. Knowledge can be either *explicit*, which can be formally expressed and transmitted to others through manuals, specifications, regulations, rules or procedures [46], or *implicit*, which is understood without being openly expressed, is unvoiced or unspoken [46], and can be hidden in skills and the experiences of the community members. For instance, someone's experience can be classified as implicit knowledge [21].

When new agents join the community their *actions* are far from optimal due to the absence of skills and knowledge the others have. Therefore they face the problem of acquiring the required knowledge. In other words, in order to act in the community in the optimal way, newcomers should acquire the knowledge that influences the behavior (actions) of the members of the community. We also call this knowledge and behavior the *culture* of the community. In the Implicit Culture Framework [7] the culture of a community is represented as a *cultural theory* consisting of several *if …then* rules.

If it is possible to exert partial control over the environment, then by changing an agent's view about the environment, it is possible to change the set of actions the agent can perform. The change in possible actions can lead to the situation where the agent acts similarly to the way a community member would act. This means that the agent behaves according to the community culture without knowing about the group or 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 group has been defined as *Implicit Culture* [13]. The groups can partially overlap, or even coincide.

Let us map the web search domain to the terms of the Implicit Culture Framework. *Agents* are people searching the web, *actions* are: *requesting* a link specifying a query, *accepting* or *rejecting* the proposed link. A *cultural theory* describing general behavior of the community in our system is

$$\text{request}(a, q) \rightarrow \text{accept}(a, l, q), \tag{1}$$

where a is an agent, q is a query, l is a link. This theory (in this case, just a rule) says that if an agent a searches with a query q then the system should recommend some link l that is likely to be accepted. Such theory is specified a priori and in terms of the Implicit Culture Framework it is called the *domain theory*. More specific *culture* of the community is the set of links accepted by the community for certain queries corresponding to their shared interests. Such theory represents the knowledge about user behavior, and this knowledge is learned by the system from user interactions with the system. An example of a more specific *cultural theory* describing actions of the community could be

$$\forall x \in Group: \quad \text{request}(x, \text{'apartments'}) \rightarrow \text{accept}(x, \text{www.phosphoro.com}, \text{'apartments'}). \tag{2}$$

This theory expresses that for all agents of the group if they search for apartments, they tend to accept the link www.phosphoro.com.

Let us briefly explain the scenario in which this link is non-obvious and relevant. Consider a community of PhD students in Trento who have been living there for quite some time. In the past, they searched for apartments in Trento using the Implicit system, got recommendations about the link www.phosphoro.com and followed this link. All these actions are recorded by the system. This link is relevant to the specific interests of the group, in this case it is assumed that a person searches for the apartments in Trento, Italy and prefers private offers, not those from an agency. Now, let us assume a group of new PhD students arrived in Trento, and they certainly do not know about this link. For instance, in the time of writing this link did not appear among the first 10 results provided by Google for the query 'apartments'. However, this link is of extreme importance for people searching for an apartment in Trento. If Implicit is able to provide the newcomers with this link and they access the desired information, then it is possible to say that new group behave in accordance with the community culture and that the Implicit Culture relation is established.

3.2 Systems for Implicit Culture Support (SICS)

In the Implicit system each agent tries to establish the Implicit Culture relation within the group of agents on the platform. In order to do this, each agent relies on a SICS.

The general architecture of a SICS is shown in Fig. 1 and consists of the following three basic components:

- The *observer*, which collects information about actions performed by the user and stores this information in a database (DB) of observations;
- The *inductive module*, which analyzes stored observations and applies data mining techniques to find patterns of user behavior, i.e. the culture of the community. The discovered patterns are referred to as a *cultural theory*;
- The *composer*, which uses the information collected by the observer and the theory produced by the inductive module in order to produce recommendations to its user or to other agents.

The observer module monitors the actions users perform while interacting with the system. For instance, a query is treated as the request action. It is interpreted by the personal agent both as the request of a relevant resource link and as the request of the ID of an agent which can provide relevant recommendation. Therefore, two observations appear in the DB of observations as the result of the query: `request(user,query,resource-link)` and `request(user,query,agent ID)`. If the user clicks on the recommended link, the link is considered to be accepted and the observation `accept(user, query, resource-link)` is stored in the DB. If the `resource-link` has been suggested by an agent, which could be the user's personal agent or the personal agent of another user, one more observation is stored: `accept(user,query,agent ID)`. When the user starts another search or exits the system, all the recommendations which were proposed to the user

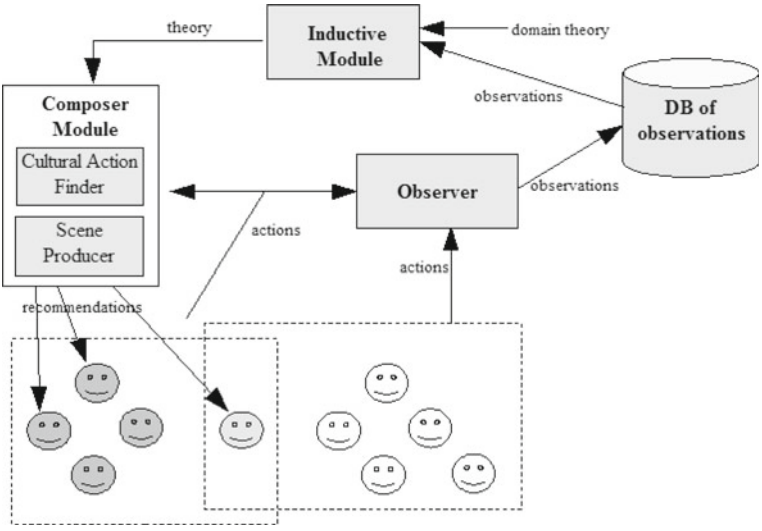


Fig. 1 The architecture of a SICS. The SICS includes three components: the *observer*, which monitors user activities and stores in the DB the observations about the performed actions; the *inductive module*, which discovers patterns of user behavior by analyzing observations; the *composer* module, which produces recommendation using information collected by the observer and induced by the inductive module

but have not been accepted, are treated as rejected. For each rejected link two observations appear: `reject(user,query,resource-link)`, `reject(user,query,agent ID)` (in case the link has been recommended by an agent). It is important to notice that by storing IDs of the agents that provided the accepted or rejected recommendations the system can discover patterns of behavior related to accepting results obtained from a certain agent, thus maintaining implicit trust relationships. At the moment, we do not use any explicit model of trust, but we noticed in the experiments that during intra-agent communication, agents tend to ask certain personal agents (but not others) for advice, therefore forming a kind of social network. The choice of agents to ask is based only on past history of observations, i.e., the agent providing results that are accepted by the user of the agent–requester is more likely to be chosen than the agent providing results that are rejected by the user of the agent–requester.

The inductive module applies data mining techniques in order to extract interesting patterns from the user behavior. In the current version of the system the SICS implements the Apriori algorithm for learning association rules between the actions. This algorithm has been described by Agrawal and Ramakrishnan [1] and it deals with the problem of association rules mining. In our settings, this problem can be briefly formulated in the following way: given a DB of queries and links, it is necessary to find which links are accepted for which queries. Without going into the details of the algorithm (an interested reader can find them in [1]), we can say that mined rules have the form $query \rightarrow link$ and are characterized by confidence and support. The *confidence* of a rule denotes the percentage of cases where the $link$ from the rule (and not other links) was accepted for the $keyword$ from the rule. The *support* denotes the percentage of the actions in the DB which contain this rule. For instance, if there are 100 actions in the DB, 20 of them are $request(*, football)$, meaning that some agents asked for the keyword "football", and seven are $accept$(*, www.fifa.org, football) meaning that in seven cases the link www.fifa.org was accepted for this keyword, then the confidence is $7/20 = 0.35$ while the support is $7/100 = 0.07$. An example of the theory produced by the inductive module is the theory in Eq. 2. Similarly, the problem for discovering which agents are accepted for which keywords can be formulated and addressed. Such problem is related to the problem of finding experts in a specific area of interests.

The composer finds links and agent IDs that are likely to be accepted by the user or the agent performing the search. The internal architecture of the composer module is represented in Fig. 2 and consists of two main modules, the *Cultural Action Finder* (CAF) and the *Scene Producer* (SP). It also includes the *pool* that serves as a buffer for information exchange between the CAF and the SP. The goal of the CAF module is to find *cultural actions*, i.e. the actions that satisfy the right part of the theory (consequent) for the submitted action, corresponding to the left part of the theory (antecedent). The cultural actions are then placed in the pool. The SP module evaluates the actions from the pool. In particular, given the actions the users performed, SP calculates the similarity between the users, and selects the one whose actions are the most similar to the cultural action. The links or IDs from these actions are sent as recommendations to the personal agent who started the search.

The similarity between users is calculated based on their actions, i.e., the more actions two users have in common, the more similarity they have. The similarity between two actions is calculated based on the similarity of names of actions and objects of the actions, i.e., two actions will have similarity one if they have the same names and the same objects (keyword, link) and they will have similarity zero if they have different names. If the actions have the same name but different objects (e.g., different links accepted for the same keyword), the similarity will be 0.5 (the users are similar in the terms of accepting something for a certain keyword, but what they accepted is different). We give an example of how the SICS works in Sect. 5.

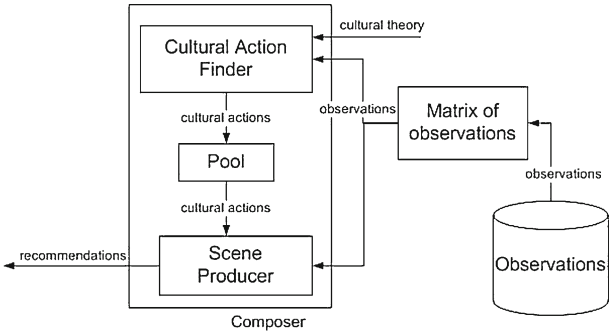


Fig. 2 The internal architecture of the composer module. In the first step, the composer looks through observations to select actions that match the "if" part of the theory. In the second step, it builds the matrix of observations where the columns correspond to links or agent IDs, the rows correspond to users, and the entries contain observations about the actions (accept,reject) performed with the link or agent ID by the agent. In the third step, the *CAF* selects links satisfying the cultural theory to the *pool*. Finally, the *SP* uses the matrix of observations to calculate the similarity between agents and to selects the best link from the pool

The SICS architecture allows the Implicit system to find relevant links and to discover IDs of relevant agents with the same mechanism. The SICS calculates the similarity between the community members in order to produce suggestions. Therefore, it personalizes web search to a certain extent. For the more detailed description of the SICS module, we refer the reader to [7].

4 The Implicit system

In this section, we describe the architecture of the system and the user interface. The details concerning the internal agent architecture and the search process are given in Sect. 5.

4.1 The system architecture

Implicit is a multi-agent recommendation system that aims at improving web search of its users. Figure 3 illustrates the architecture by showing an instance of the system with three personal agents. The system consists of the client part and the server part. A user at the client side accesses the user interface via browser. In the system, there is exactly one personal agent for each user. All personal agents are running on the JADE platform on the server side. The queries submitted by the user are received by the user's personal agent. The personal agent contains behaviors for communication with external information sources, such as Google, and for producing recommendations about the relevant links and agent IDs using the SICS module. The SICS module implementing the Implicit Culture recommendation framework, and, therefore, containing the observer, the inductive and the composer modules, is an essential part of each personal agent. The SICS does not interact with other personal agents directly and only stores observations about actions involving its personal agent. This architectural decision allows for having autonomous agents, so that they can produce recommendations even if there are no other agents in the platform. It also allows for reduce the number of observations stored by the observer of the SICS, therefore decreasing the response time of the SICS. Agents can also produce recommendations using other resources, such as search

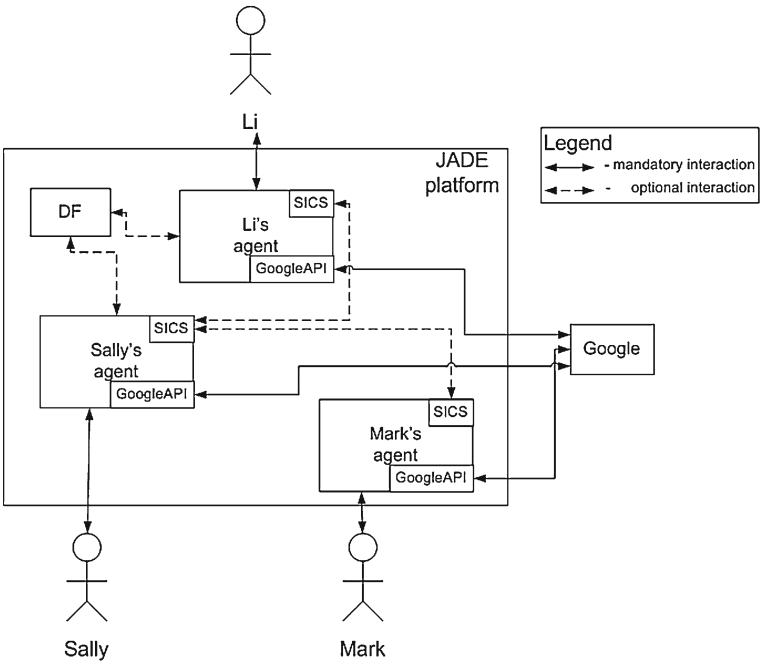


Fig. 3 An example of the system with three agents. *Personal agents* process queries from *users* and interact with each other to share experience of using particular links by their users; the agents produce recommendations by using the *SICS* module; they also use *GoogleAPI* to query the Google search engine. The *DF* agent provides a list of personal agents

trees or bookmark collections of their users. The obtained results appear in the user interface. When producing recommendations, agents aim at finding web pages that members of the community consider relevant to their searches. For this purpose, the agents adopt the Implicit Culture approach, searching for the links that satisfy specific behavioral patterns of the group.

Let us describe how users interact with the system. A user logs into the system, enters a query and receives the results from Google complemented with recommendations produced by the user's personal agent in collaboration with other personal agents. Figure 4 shows the browser window with the list of results. In the bottom part of the window there are the first 10 links obtained from the Google search engine, while in the top part there are several links received as recommendations from the personal agents of the community members. The name of the link provider ("Google" or the name of the community member) appears in the box preceding the link. Whenever user clicks on one of the results, the information about this action is forwarded to the personal agent of the user as *feedback* indicating relevance of the link to the search. After the user exits the system or starts another search, the non-clicked links are marked as rejected.

If it is the first time the user interacts with the system, the SICS will recommend the link expressed by the cultural theory, i.e. corresponding to the Implicit Culture of the community (the links considered relevant for certain keywords). Otherwise, if the user has performed several searches already, the SICS will use the domain theory and the history of observations to find which links were accepted in the past for the entered keyword by similar users.

Fig. 4 Results produced by the system for the entered query. The links recommended by the personal agents of the other users are displayed in the top part of the window. The results from the Google search engine are shown in the bottom part

In the following, we will explain interactions between a user and a personal agent using a running example.

Example Let us consider a user Sally who looks for a website that provides a collection of announcements about apartments available for rent. She logs in the Implicit system and types a query "apartments". The query is processed by the personal agent of Sally. Fist, the personal agent obtains the first 10 results from Google. Figure 4 shows the following Google results: www.apartments.com, www.only-apartments.com, www.rentalinrome.com. Second, the personal agent uses the SICS module to process the query in several steps: searching for links during the internal search and searching for agents to contact during the external search. Searched links and agent IDs should be related to the entered query "apartments". If the agent does not find any agent IDs using the SICS, it contacts the Directory Facilitator (DF) agent (explained in more detail in the next section). Once the personal agent contacted all agents found during the external search or by contacting the DF, it displays all the obtained links in the user browser. In this example the links www.trentinobedandbreakfast.it, www.phosphoro.com and www.apartments.com from Sally and her colleagues, Mark and Li, are displayed. The personal agent stops the search at this point and becomes idle, waiting for the feedback or a new query from Sally and eventually responding to the queries of other personal agents. Let us suppose that Sally clicks on www.phosphoro.com. Her personal agent receives the feedback message about accepting this link. Since the link was suggested by Mark's personal agent, the feedback will be also treated as accepting Mark's personal agent. When Sally exits the system or starts another search, the not followed link, www.apartments.com, and the corresponding agent are marked as rejected.

4.2 Personal search history

Implicit also allows for the quick access to the history of previous user searches. The history is maintained by the personal agent, which accesses it after querying Google and shows the results on the user interface. For instance, in Fig. 4, the link from Sally's personal agent comes from Sally's history of previous searches. Another example of the knowledge available locally is a personal bookmark collection in someone's browser. User personal collection

of bookmarks on Delicious could be an example of user-specific knowledge, which is not available locally, i.e., stored on the Internet.

4.3 Motivation for using agents in the system

The use of agents in the system is motivated by the following: (i) agents assist their users in web search activities, i.e., agents personalize user searches, autonomously interact with other personal agents of the community, and facilitate maintenance of the past search history; (ii) agents provide an interface to different kinds of search, i.e., Google, SICS, without the need of heavy client part of the system; (iii) agents recommend other agents on the platform thus establishing implicit trust relationships in the system; (iv) even if a user is not accessing the system for some time, the personal agent stays there, answers queries from other agents and receives feedback on the proposed results thereby improving its expertise; (v) agents facilitate sharing of information that is usually shared only by word-of-mouth communications; and (vi) finally, in the simulations we conducted to validate the system (see Sect. 6), each agent contained a model of the user in order to simulate users of the system.

4.4 Implementation details

The system has been implemented using JADE (Java Agent DEvelopment framework) [6]. JADE adopts a task-based model of the agent and it is a one of the most powerful tools for the development of FIPA[3]-compliant multi-agent systems. In the current implementation, each agent uses the Google SOAP Search API, but in principle, it is possible to contact any search engine that provides similar API. Alternatively, user queries can be forwarded to other search engines, like Yahoo! or Vivisimo by means of wrappers, implemented by special agents on the platform.[4]

5 Agent architecture and the search process

This section provides more details about the technical description of personal agents, their interactions and the search and recommendation mechanism.

5.1 The architecture of a personal agent

In the following we define basic terms used in JADE and describe the internal architecture of an agent in our system. Figure 5 illustrates the definitions and the architecture.

A *personal agent* is a software agent running on the server side assisting its user in their searches, receiving queries and producing recommendations in response.

A *behavior* is a procedure that implements tasks, or intentions, of an agent [6]. The agent is able to execute a behavior in response to different internal (e.g., calculations finished) and external (e.g., message received) events.

A *behavior scheduler* is an internal agent component that manages the scheduling of behaviors and determines which behavior to run at the moment and what action to perform as a consequence.

An *inbox* is a queue of messages received from the user and from other agents.

[3] http://www.fipa.org/. Foundation for Intelligent Physical Agents (FIPA).

[4] A wrapper agent for the Vivisimo search engine has been developed as a student project at DISI, UNITN.

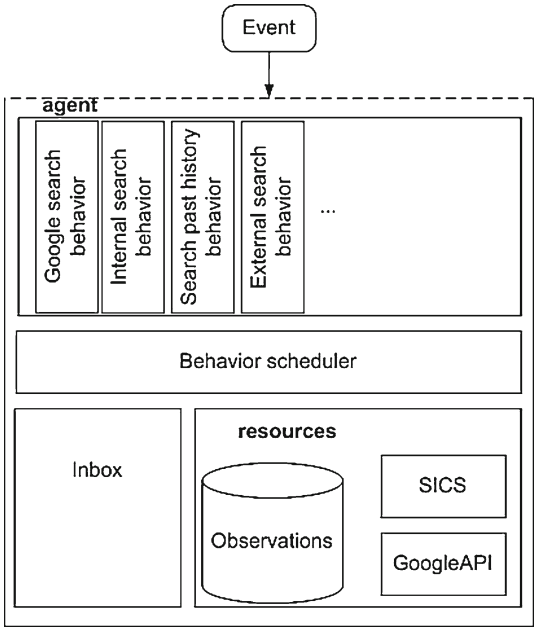


Fig. 5 An internal architecture of the personal agent. A *behavior* of an agent is a task or reactions to an internal or external event. The execution of the behaviors and switching between them is performed by a *behavior scheduler*. An *inbox* contains messages received by the agent. Agent's *resources* include observations, SICS and Google API

To produce recommendations the agent uses its *resources* that include the information available to the agent, e.g., observations about user actions, and specific functionalities such as getting recommendations using the SICS or getting links from Google.

5.2 The search process

Let us describe behaviors and other parts of the agent architecture that participate in the search process in detail. As described in Sect. 4, a query received from the user interface triggers a set of steps executed by the personal agent. The process of producing recommendations that the user finally sees in the browser window consists of several parts, implemented as behaviors. When the agent receives the query message from the interface, it starts three search behaviors that run in the following order: first the *Google search behavior*, then the *Internal search behavior* that includes *Search past history behavior*, and, finally, the *External search behavior*. For brevity, we refer to the sequence of these three behaviors as "the search". The results obtained during all three steps of the search are shown to the user.

The sequence diagram in Fig. 6 illustrates the details of the interactions between the user and the personal agent during the search. During the Google search behavior the agent forwards the query to Google. After receiving the response, the agent shows the obtained links to the user and starts the Internal search behavior. In the internal search the goal of the SICS module is to recommend web links using the information about the past user actions about searches and link acceptance. In case the SICS does not produce any recommendation in this step, the past search history is used to recommend links accepted by the user for similar queries in the past. All the generated links are stored in the list of results and the External search behavior is started. This behavior also uses the SICS, but the goal of the SICS in this case is to find relevant links using external resources, i.e., to propose the IDs of agents to contact. The techniques used within the SICS to recommend links and agents are the same.

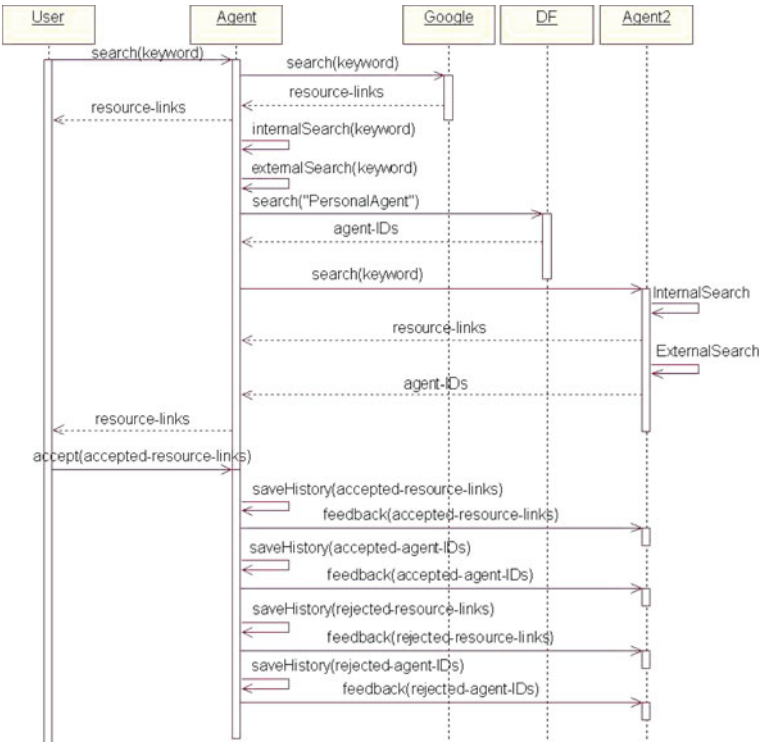


Fig. 6 A sequence diagram of interactions between the user and the personal agent during the search

If there are no suggestions about agent IDs, the agent contacts the DF. According to the FIPA standards, the DF is a mandatory agent that provides yellow pages service on the agent platform. In our system, the DF simply provides the agent with the IDs of other personal agents on the platform. Thus, the use of the SICS module helps to reduce the number of interactions between the agents. Having filled the list of agents to contact, the personal agent starts interaction by sending a query to each agent in the list. When all the agents are contacted, the External search behavior queries new agents that were suggested during the search and so on. When all queries have been answered by the suggested agents, the system adds the obtained links to the list of results and shows all the links from the list to the user.

Currently, the DF provides only IDs of the personal agents. Such IDs are equal to the nicknames of agents' users and, therefore, displayed together with recommendations. In our settings of a small community this should be enough (even on Internet forums people are usually known just by their nicknames). However, in the future, it would be possible to also display, for instance, the top K keywords to which the agent provides best recommendations, so that agents would be known not only by nicknames of their users, e.g., "Sally", but also by the expertise, e.g., "Sally (football, rugby, Java)".

When agents query each other, the agent-responder does not contact Google, because the agent-questioner can do this too. The agent-responder executes the Internal search behavior and the External search behavior to recommend to the agent-questioner links and agents to contact.

5.3 An example of recommendation

In this section, we provide more details on how recommendations are created. For explanations we will use the running example.

Example When the personal agent of Sally receives the query "apartments" and starts the search, an observation is produced by the observer module of the SICS and appears in the DB of observation. The observation includes the name of the requester, and the query, which contains the type of the requested recommendation, i.e. link or agent: `request(Sally,apartments:link)`.

This observation is then sent to the composer module that processes it in several steps. In the first step, the CAF, a submodule of the composer module of the SICS, builds the matrix of observations (Table 1) and matches the request action with the rule of the theory shown in Eq. 1. The action matches the rule, so the right part of the rule, $accept(a, l, q)$, is taken as a cultural action. After substituting the value of the variables a and q with those from the request action, the cultural action $\alpha = accept(Sally, l, apartments : link)$ goes to the pool. The SP, a submodule of the composer module of the SICS, takes the action α from the pool and calculates which agents performed actions most similar to α. For this calculation, the SP uses the matrix of observations. The rows of the matrix contain agent names, and the columns contain links, while the cells contain actions that involve the corresponding agent-link pair, e.g., accept or reject of the link by the agent for a keyword. Since in the matrix of observations in our example (Table 1), Mark's actions are the most similar to Sally's actions, the link www.phosphoro.com is recommended and put in the list of results.

Together with asking the SICS for relevant links, the personal agent of Sally submits another query to the SICS module, requesting agent IDs for the keyword "apartments", and the observation `request(Sally,apartments:agent)` is stored in the DB of observations. Let us suppose that the SICS returns the ID "Li" as the result. The personal agent of Sally contacts the personal agent of Li and gets the link www.apartments.com as a recommendation. This link is put in the list of results and then the results, i.e. www.phosphoro.com and www.apartments.com are displayed in the user browser. The personal agent stops the search at this point and becomes idle, waiting for the feedback or a new query from Sally and eventually responding to the queries of other personal agents. Let us suppose that the users clicks on www.phosphoro.com. The personal agent of the user receives the feedback message that is converted to the actions `accept(Sally, www.phosphoro.com, apartments:link)`, `accept(Sally, Mark, apartments:agent)`. When Sally exits the system or starts another search, the feedback about the not followed link is received about the personal agent and converted to the actions `reject(Sally, www.apartments.com, apartments:link)`, `reject(Sally, Li, apartments:agent)`.

Table 1 A matrix of observations

Agent/link	www.hotel.it	www.phosphoro.com	www.only-apartments.com
Li			accept(apartments)
Mark	accept(hotels), reject(cars)	accept(apartments)	reject(apartments)
Sally	accept(hotels)		

Rows contain users, columns contain links, while cells contain actions performed by the user on the link

6 Experimental evaluation

In this section, we present the results of the pilot study of the system with real users and its evaluation with user models.

6.1 Pilot study with real users

The goal of this pilot study was to see how the system might be accepted by people and whether it provides enough support for sharing experience in a community of practice.

We performed the study in a small company located near Trento, Italy. The company offers courses in different areas of IT. For promoting interaction between course attendees the company provided a forum with topics dedicated to the content of the courses.

For this study people attending the courses related to the Java programming language were selected. Eleven course attendees agreed to participate in the study. When dealing with the system, the participants were focused on a narrower topic of game development in the Java language. The Implicit system was integrated in the company forum, namely, the interface was modified to enable user requests, suggestions from the Implicit system, and feedback on forum messages. Since participants had no prior knowledge of game development in Java, they were similar with respect to their information needs. There were no explicit control group in the experiment, but we performed comparison of the group with the same group in the beginning of the experiment.

Participants used the system in the following manner: whenever they wanted to get an answer to a question, they clicked on the Implicit button in the interface and submitted several keywords as their query to the system. The system ran the general search mechanism described above, forwarding the query to other personal agents and to Google. In addition to this, the query was forwarded to the specific wrapper agent that dealt with the retrieval of the forum messages. This wrapper used the SICS to retrieve messages that were accepted for similar queries previously. The feedback collected previously has been used to rank the results. In this application, first forum messages, then links from the users, and, finally, links from Google were displayed and a result was considered "accepted" if the user clicked on it. Moreover, users were able to express their level of satisfaction with results by means of explicit feedback—ratings in the scale from 1(very bad) to 6(excellent) for the five following categories: (i) quality of message, (ii) quality of the code, (iii) message was useful to the user, (iv) the user would recommend the message to others, and (v) it was relevant to the original question of the user. The users were not required to provide ratings in all five categories. There was also a possibility to rate messages directly when browsing forum threads, without the connection of messages with requests. Messages rated in such a manner were retrieved by the forum wrapper in case the retrieval of messages using the SICS produced no results. This functionality helped us to deal with the cold-start problem [14].

We ran the system for six weeks and measured the number of queries submitted and the number of recommendations accepted. The results containing the number of queries submitted during the study are given in Table 2. As we can see, the users were reluctant to use the system during the first two weeks, while during the second two weeks the number of queries slightly increased and reached the number of 60 queries in the final period.

Table 3 show the how many times one, two, three, and four results were accepted. As can be seen from the table, in most cases users accepted only one result, sometimes two, and

Table 2 The number of requests to the system

Period	Number of requests
Weeks 1 and 2	10
Weeks 3 and 4	18
Weeks 5 and 6	60

Table 3 The number of accepted results

Number of accepted results	Number of requests
1	32
2	16
3	2
4	2

only rarely three or four. For the total of 88 queries,[5] in 52 searches users accepted at least one result from the system, i.e., in 60% of the searches the system provided something of interest.

The obtained results show that the system, after some period, was accepted by the community and able to provide useful recommendations.

Please note that some settings of this study might seem weak (e.g., we have not compared the search behavior of the group that used our system with the group that did not use it; we have not compared the Implicit with just search over company forums), but we decided to report the results anyway, because they show that over time people perceived the value of the system.

6.2 Simulation

Thorough evaluation of the system with real users is hard because it requires a lot of time from the users. Therefore, in this section, we present the goal, materials, methods and results of the simulation that was conducted using the Implicit system. The simulations provide a way for an extensive evaluation of the Implicit system. We also define the measures we used to evaluate the quality of suggestions produced by the SICS.

The goal of the experiment was to understand how the insertion of a new member into a community affects the relevance, in terms of precision and recall, of the links that were produced by SICS. We also wanted to check the hypothesis that after a certain number of interactions personal agents would be able to propose links accepted in previous searches.

The interaction between agents and users was replaced with the interaction between agents and user models. In the following we give a brief description of the user model and we refer the reader to Appendix A for details. The model we used is similar to the one proposed in [3]. Essentially, a user model specifies the sequence of search keywords submitted by the users and determines which links will be accepted for which keyword using the click-through ratio as represented in Table 4. The probabilities of link acceptance were produced by subjective evaluation of the Google results for five keywords and then duplicating these five rows. The duplication was done in order to model correlated keywords and it did not affect the

[5] This number might seem low, but you should consider that people attending the courses were not IT people, some of them hardly seen computer or programmed before courses. Also, the study was conducted several years ago, when the use of search engines was much lighter than nowadays.

Table 4 The probabilities of acceptance of links for a set of keywords

	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
tourism1	0	0	0.05	0.4	0.05	0.2	0.1	0.05	0.1	0.05
football1	0.05	0	0.1	0.3	0.3	0.1	0.1	0.05	0	0
java1	0.35	0.3	0.05	0.05	0.05	0.05	0.05	0.1	0	0
oracle1	0.1	0.1	0.45	0.2	0	0.05	0.05	0	0	0.05
weather1	0	0.3	0	0	0.5	0	0	0.1	0.1	0

Links were different for each keyword and are numbered 1…10

behavior of the system (as compared to 10 truly independent keywords, see more on this in Appendix A). We used 10 keywords and for each keyword 10 first links from Google were considered. In this experiment, we kept queries intact, without splitting them into individual keywords. We did not use the following features of the Implicit system: past search history, wrappers, the inductive module.

We use the following information retrieval measures [5] 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 by the user model, is greater than some pre-defined relevance threshold r.
- *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.
- *F-measure* is a trade-off between precision and recall, calculated as $\frac{2*Precision*Recall}{Precision+Recall}$.

Assuming that all users are members of the same community and have similar interests, the probabilities of accepting links for each user were obtained from the distribution given in Table 4 by adding noise uniformly distributed in [0.00,…,0.05]. The five user models obtained as the result of adding noise are listed in Appendix A.

From the set of 10 keywords shown in the first column of Table 4, for each agent we generated 25 sequences of 25 keywords by extraction with repetition. An example of a sequence generated for one agent for one simulations could be: `weather1`, `oracle1`, `java`, `tourism`, `football`, `weather`, `football`, `java1`, `oracle`, `tourism1`, `football1`, `weather1`, `java`, `java1`, `weather`, `football1`, `tourism`, `oracle1`, `java`, `football`, `java1`, `tourism1`, `football1`, `java`, `football`. Each sequence was used for modeling the user query behavior during a search session.

For each query Google returned all 10 links existing for that keyword in the simulation, while SICS of the user's personal agent provided only one link that was considered the most suitable. Other agents, if queried, also provided only one link each. During a single search zero or one agents (depending on whether agents were recommended by SICS) were contacted by the personal agent, to minimize the amount of interactions on the platform.

The user acceptance behavior was modeled in the following way. Given a keyword in the sequence of keywords, the accepted result was generated randomly according to the distribution that was specified in the user model (Tables 5, 6, 7, 8, 9 in Appendix A). All the links from Google, SICS, and other agents were marked as accepted if they were equal to the accepted result. The links different from the accepted result were marked as rejected.

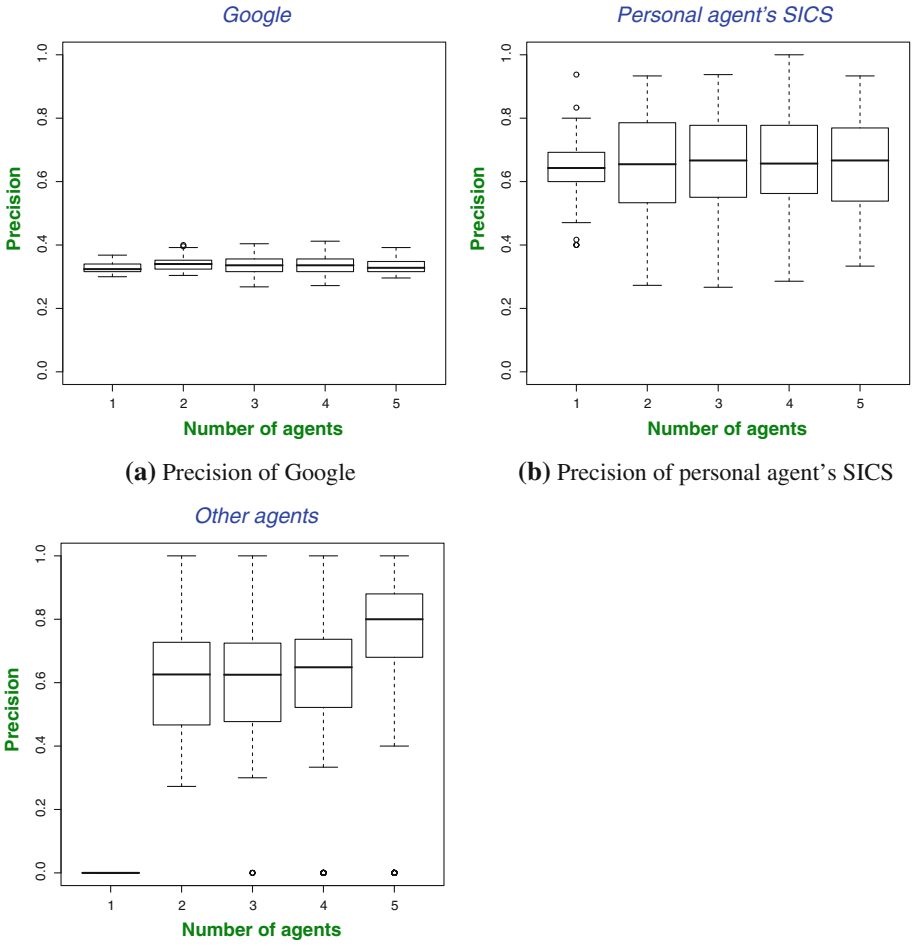
In the simulation each agent performed 25 search sessions. At the end of each session (e.g., after agents all finished the first search session) the observation data were deleted. The reason for performing 25 sessions and not just one is that it allowed us to control the effect of the order of the keywords and link acceptance. We ran five simulations for one, two, three, four, and five agents. With one agent on the platform, the agent acted alone without interactions with the others. Five agents represented a small community where agents interacted with each other. We set the relevance threshold r determining link relevance equal to 0.1. As we report in Table 11 in Appendix A, such relevance threshold for each keyword selects 2 to 5 of the first 10 Google links that are considered relevant, which seems to be close to the our experience in using Google with single keywords. In other words, the selected threshold produces a result comparable with typical user behavior using Google.

Note that the recall was computed without taking into account the search keyword, i.e., for each agent module (Google, SICS, other agents) we summed up the number of proposed relevant links for all keywords and then divided by the sum of the numbers of relevant links for those keywords. For instance, let us consider keywords "java" and "oracle". The number of relevant links for "java" is three, while the number of relevant links for "oracle" is four (Table 11 in Appendix A). Let us assume the keyword "java" repeated in the search session four times (as in the example of the search session above), SICS proposed zero, one, one, and one relevant links for each of the four searches, and the keyword "oracle" appeared only once and the SICS proposed zero relevant links for it. The total number of the relevant links in this example is $(3*4)+(4*1)=16$. The total number of the relevant links proposed by the SICS is $(0+1+1+1)+(0)=3$. In both equations the first parentheses correspond to "java" and the second ones—to "oracle". The recall in this example is $3/16=0.1875$. An alternative approach would be to compute recall for each keyword (e.g., $3/12=0.25$ and $0/4=0$) and then compute the average ($((0.25+0)/2=0.125$), but we do believe that the former approach is more suitable for our task, because we do not want the system to be sensitive on the choice of keywords in the search session.

We computed precision and recall of the links proposed by Google, the SICS of the personal agents and by all other agents. We did not compute the precision and recall of Implicit as a whole because it would be equal to the behavior of Google, since the links coming from SICS and agents are already proposed by Google. However, the experiment allows us to compare the measures for the different components of Implicit. Figure 7a, b, c show, respectively, the precision of Google, the personal agent alone, and all other agents on the platform. The results from the personal agent are the results of the SICS, which was incorporated in the agent. The SICS produced these links by analyzing stored observations. The other agents on the platform were either recommended by SICS during the external search or were provided by the DF. In Fig. 8a and b we have analogous box plots for recall and in Fig. 9a–c—for F-measure. Note that we do not show the recall for Google because it is always 1, since all the links in the experiment come from Google.

In the following we summarize our observations on the results of the experiments.

– From the figures for precision it is possible to notice that the precision of SICS and agents is higher than the precision of Google. On the one hand, this is related to the fact that Google always recommends 10 links, 5–8 of which are irrelevant. On the other hand, it



(a) Precision of Google

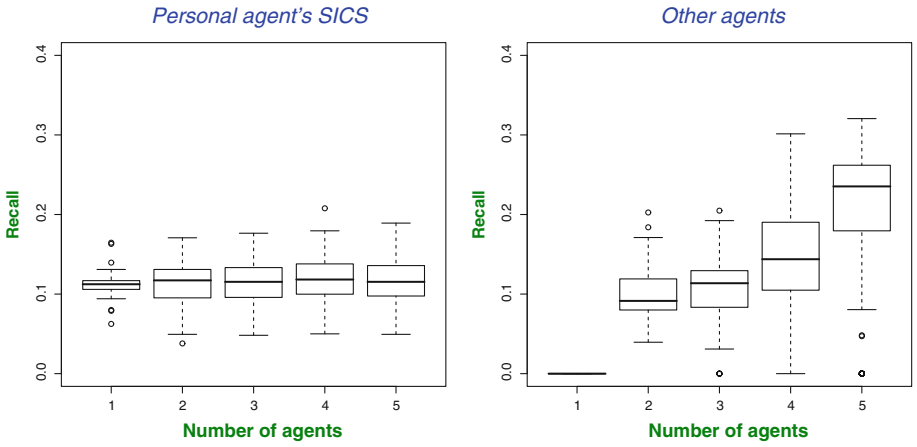
(b) Precision of personal agent's SICS

(c) Precision of other agents

Fig. 7 Boxplots for the precision of Google, the SICS of the personal agent and of other agents in 25 simulations with different number of agents

shows that SICS and agents are good at processing users' feedback on Google results and re-ranking them.

- Precision of agents grows with the number of agents and achieves a surprisingly high median value of 0.8, while the precision of Google and personal agent's SICS does not change with the number of agents.
- The recall of Google is always one, while recall of SICS and agents rarely goes above 0.2. This is easily explained by the fact that Google always recommends all 10 links that exist in our simulation for the search keyword, including all relevant links. Contrary, the suggestions from SICS are limited to one link, so it is not possible to achieve the recall of one. The same applies to the recall of other agents.
- The increase in community members causes the increase in the recall of the suggestions from the agents, while the recall of Google and the personal agent remained at the same



(a) Recall of personal agent's SICS **(b)** Recall of other agents

Fig. 8 Boxplots for the recall of the SICS of the personal agent and of other agents in 25 simulations with different number of agents. The recall of Google is always 1 and, therefore, is not shown

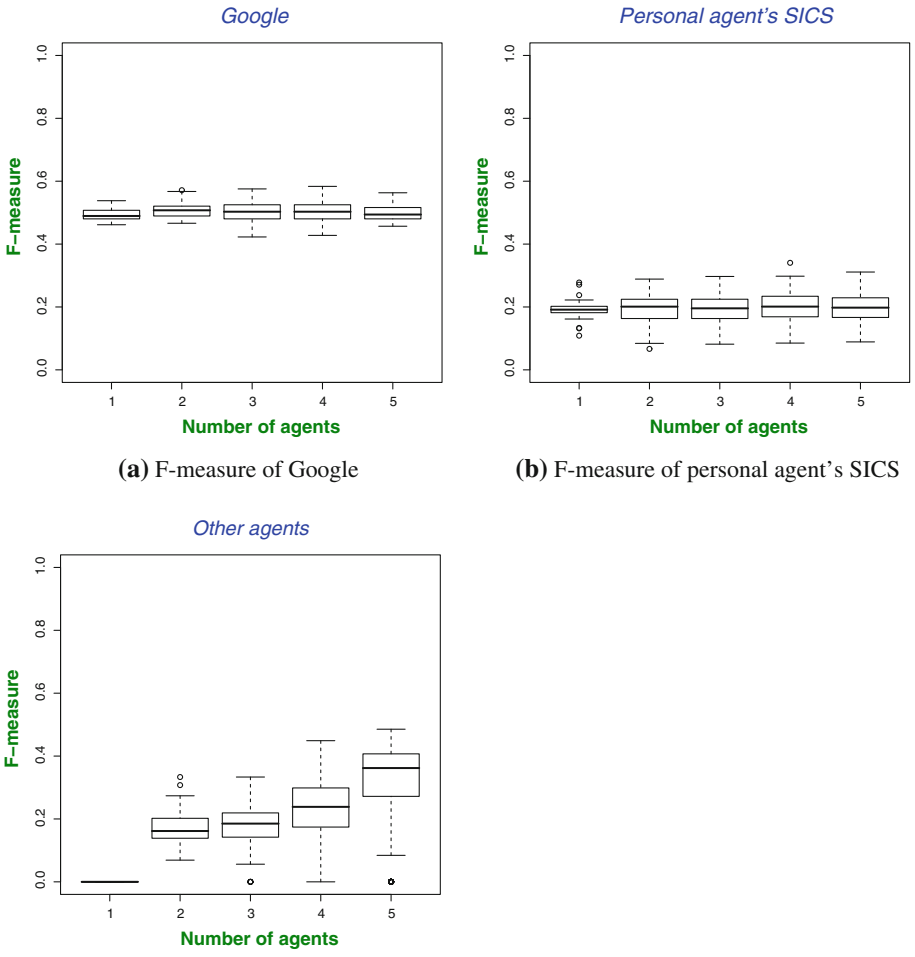
level. It is probably due to the fact that when there are more agents in the system, there are also more interactions between them. The agents provided each other only with one link, thus, when more links were provided by the agents during the search, it led to an increase of the percentage of relevant links proposed by the agents and, therefore, the increase of recall. Moreover, the increase of recall appeared without the decrease of precision which remained at a rather high level with median ranging from 0.61 to 0.8. This result proves the hypothesis that after a certain number of interactions, agents are able to propose links based on the past user actions.

– F-measure behaves similarly to recall in the sense that it grows for other agents as the number of agents in the community grows, while it remains at the same level for Google and the personal agent. The latter is the consequence of the stability of precision and recall for Google and the personal agent.

The obtained results proved that our way of complementing search engine with suggestions that are produced as a result of indirect user collaboration allows for improving the quality of the web search. An important point is that even without extending query with additional keywords and re-ranking links by the system, it is possible to discover which of the Google links the community prefer and to achieve better quality of suggestions than Google.

6.3 Evaluation of the SICS module

In this section, we describe experiments we carried out to evaluate the core of the Implicit system—the SICS module. Some of the experiments (performance evaluation, scalability experiments) have been carried out not with the Implicit system, but with the multi-agent system for recommending patterns, described in [9]. However, we report the results related to the evaluation of the SICS module, which is the same for both systems, therefore, there is no need to re-run these experiments for Implicit.



(a) F-measure of Google

(b) F-measure of personal agent's SICS

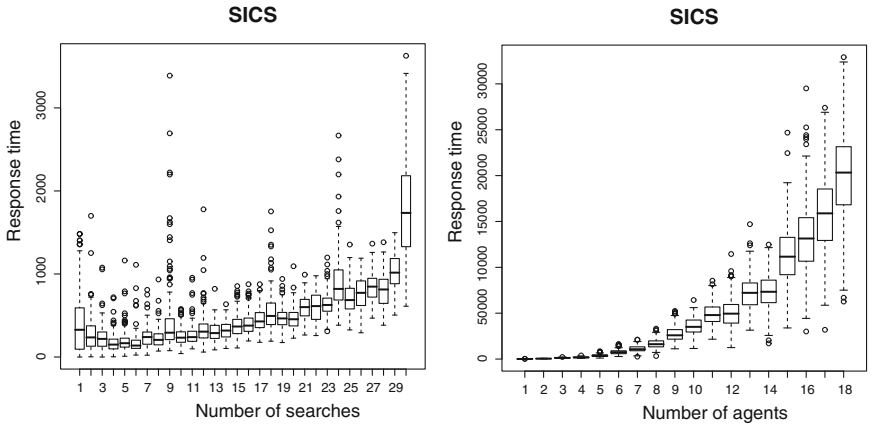
(c) F-measure of other agents

Fig. 9 Boxplots for the F-measure of Google, the SICS of the personal agent and of other agents in 25 simulations with different number of agents

6.3.1 Performance evaluation

We measured the *response time* of the SICS as the time passed from the moment the module received the submitted user query till the moment the results were sent to the user. The response time of the SICS obviously depends on the load of the system, but does not depend on the network load and on the time required for processing the query by the personal agent. Therefore, this time is shorter than the time between the moment when the user actually submits the query and the moment when the user receives results accessing the system in distributed settings.

Figure 10a shows the response time of the SICS in milliseconds. The response time of the SICS depends on the number of searches performed. This is probably explained by the number of observations growing when simulation run contained more searches. The outliers



(a) Response time of the SICS

(b) Dependence of the response time of the SICS on the number of agents

Fig. 10 Performance and scalability of the SICS

for each point in SICS response time correspond to the initialization of the SICS, which takes some time. However, as shown by means, once initialized, the SICS responds faster.

6.3.2 Scalability evaluation

We performed the study on the scalability of the SICS core.

The goal of this experiment was to test how the system response time changes depending on the number of users. In this configuration of the system, there was only one SICS for answering all user queries, not one SICS for each personal agent. We expected the response time to decrease as the number of users increases.

In this experiment we copied the user models several times to simulate the community of u users. Of course, this led to appearance of several chunks of very similar users, but in this experiment the goal was not to focus on the reality of such a community, but merely on the size of the community.

In the experiment we set the number of searches equal to 15 and we ran simulations with the number of users, u, equal to 3, 6, 9, 12, 15, 18, measuring the response time of the system for each query. We replicated the simulation 25 times and averaged the response time.

The results contain the response time of the system as shown in Fig. 10b. As can be expected, the response time grows with the number of agents. The response time of the SICS more resembles quadratic dependence. However, up to nine agents, the response time of SICS is less than or equal to a couple of seconds. Let us consider that nine agents in this experiment do not really correspond to nine users, but, rather, they represent nine users *continuously* and *simultaneously* querying the SICS. This can hardly be seen as a realistic scenario. Thus, we can claim that the system can support more users as long as no more than nine users query our system simultaneously. The number of user that can be served is subject of further investigation. However, this defines a kind of throughput threshold for the system when used with XML files for storing observations. As we described in [10], the use of a relational DB in the system would allow for processing queries faster, and should permit serving more users simultaneously.

6.4 Experiment with the inductive module

In this section, we briefly describe an experiment that was conducted using the inductive module of the system.

The goal of the experiment was twofold: (i) to show that the Apriori algorithm [1] is suitable for learning associations between keywords and links and (ii) to see if there is difference between the specified community preferences and the way people actually select links using the system.

The Apriori algorithm deals with the problem of association rules mining. In the Implicit system, this problem can be briefly formulated in the following way: given a DB of requests and links, to find which links were accepted for which keywords. The mined rules have the form $keyword \rightarrow link$ and are characterized by confidence and support. *Confidence* is the fraction of cases where the $link$ was accepted for the $keyword$, while *support* is the fraction of the actions in the DB which contain the rule. In the experiment, we focused on the confidence of the rules. The application of associated rules in recommendation systems has been previously described by Lin et al. [40].

To conduct the experiment we used the simulator described in the previous section, so the interaction between agents and users was replaced with the interaction between agents and user models. The user models were generated in the same way, using the first five rows of the probabilities shown in Table 4.

From our set of 10 keywords, for each agent we generated 20 sequences of 25 keywords, 20 sequences of 50 keywords, and 20 sequences of 100 keywords by extraction with repetition. Each sequence modeled a user search session and each keyword in the sequence corresponded to one query. For instance, 20 sequences of 50 keywords corresponded to 20 sessions of 50 queries each. User acceptance behavior was modeled as in the previous section, i.e. given a keyword in the sequence, the accepted result was generated randomly according to the distribution specified in the user model; other links were marked as rejected. In the simulation we ran 20 search sessions for each agent, deleting observation data after each session. We performed simulations for 25, 50 and 100 keywords in a search session.

Then we applied the Apriori algorithm on the DBs of observations collected during the simulation. Since our goal was to discover "average" community interests (corresponding to the distribution in Table 4) the Apriori algorithm did not consider agent name when mining rules, only the keyword and the link accepted from the personal agent (the links accepted from Google and from other agents were ignored).

We compared the confidence of the rules learned by the Apriori algorithm with the acceptance rate initially specified in the user model in Table 4. The graph showing the Pearson correlation coefficient between the learned models and the initial model is given in Fig. 11. There is one line for each keyword that show the correlation for the initially specified rules for individual keywords and corresponding learned rules. The line labeled "Overall" correspond to the correlation calculated on complete user profiles (five keywords and 10 links for each keyword). The results show that the algorithm was effective in capturing the preference-driven behavior of the simulated users, since after 25 searches the correlation reaches 0.9 and remains high with minor variations. The variations might indicate that the actual behavior of the community slightly deviates from the defined preferences.

The success reported might be due to the small number of keywords and links we used in the experiment (just five keywords and 10 links for each). Increasing the number of keywords might lead to more noise and less confidence (and surely, less support) in the learned rules. We are planning to evaluate the Inductive module with more than five keywords in the future, while the number of links per keywords seems to be reasonable, since it

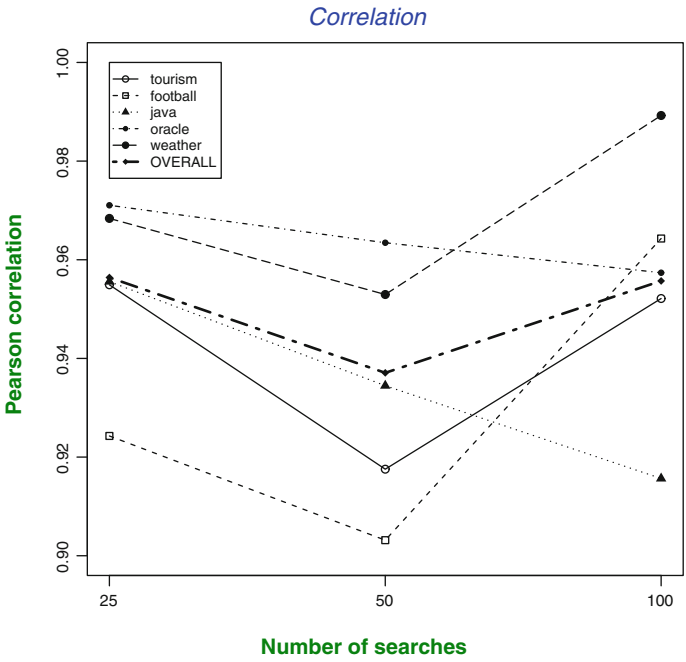


Fig. 11 Pearson correlation between the initial user model and rules learned after 25, 50 and 100 searches

corresponds to the number of links appearing in the first Google page of query results for the most users. Previous studies shown that people rarely go beyond the first page of Google results [17, p. 6].

7 Discussion

7.1 The use of Google as a collection

We are aware that many information retrieval studies use standard collections such as TREC. [6] Such collections permit clear assessments because the information needs and the corresponding number of relevant documents are pre-defined. However, we wanted to make our evaluation as close as possible to the real-world scenario, where people normally use Google for Web search. As noted by Kobayashi and Takeda [35, p. 7] and Gwizdka and Chignell [29, p. 5], in such evaluation (using such a large and dynamic collection, as the Web) it is problematic to calculate recall, because the number of relevant documents on the Web is hard to estimate. In order to address this shortcoming, we considered and estimated relevance of only first 10 results from Google for each keyword in our simulation. This allowed us to fix the number of relevant documents for each query (even though this number slightly varied for each user). This subjectiveness of relevance of query results is in line with the fact that each user in the simulated community might perceive different results as relevant to the query.

[6] http://ir.dcs.gla.ac.uk/test_collections/. Web Research Collections.

7.2 Using noise to generate user models

The goal of adding noise was to have a community of similar users (we call such community single-centered in the sense that they share some behavior with the user represented by the initial profile), who at the same time are also different (in peculiarities of accepting search results). Therefore, by introducing noise we achieved diversity among users. Note that we do not consider communities that have more than one kind of search behavior, i.e. have several sub-communities. Such multi-centered communities are subject of future studies.

7.3 Using implicit interest indicators

In the Implicit, we are not using an explicit model of the user (e.g., where users specify their interests, and other relevant information). Rather, the SICS uses user search history and compares it with the search history of other users in the community in order to provide relevant results. Furthermore, we are not asking users to provide explicit interest indicators, such as ratings or relevance judgments. As we mentioned in the Introduction, this requires an additional effort from users and, therefore, explicit feedback is often discouraged [49]. Moreover, the study by Fox et al. [26] has shown that implicit measures, such as the click-through rate, time spent reading [a web page], actions used to end a search session, can be suitable alternatives to explicit feedback. For instance, Ji et al. [34] propose an algorithm for ranking documents based on the click-through data, while Konig et al. [37] propose an algorithm for predicting the click-through rate of unseen items.

However, there are several issues to be addressed when dealing with implicit user models and implicit interest indicators. For instance, if a user clicks a link, but found it is not relevant and then choose another link, there is currently no way to learn in Implicit that the former link is irrelevant. We believe, and checking this hypotheses might be one of future work directions, that such links will be filtered out by the system in the long run, and only links truly perceived relevant by the community will remain. Another issue is that user might accept a recommendation, but reject it in the future for the same query (e.g., because this link is already known to them). How Implicit deals with such inconsistency is subject of more detailed research, but the SICS should be able to match users with similar behavior w.r.t. specific links, i.e. users who first accepted and then rejected the same link. Also, dedicated user browsing models might be applied to deal with the problem of detection of links that were seen previously [23]. Finally, some links which are marked as rejected in the Implicit because users did not click on them might be actually relevant. However, we would argue that if users consistently prefer other links, then these other links are more relevant than the rejected links, so the system behaves as expected.

7.4 Actions of the cultural theory

In the current implementation of the system we use qualitative predicates such as `accept(user, query, resource-link)`. In case of explicit feedback, it would be possible to use `accept(user, query, resource-link, rating)` for specifying some quantitative value of accept action. However, even in case of implicit feedback which we use in Implicit, it would be possible to consider applying inductive module of the SICS on such qualitative predicates in order to obtain a theory about links accepted and corresponding probability for accept and other predicates. Decision trees could be applied as a learning technique.

7.5 Privacy

Sharing the history of observations with other people in the community via SICS might raise some privacy concerns. For instance, users might not be happy about sharing some particular searches or all searches. The simplest solution is to make people aware that by using the Implicit system they enter a community and share the data about searches, similar to what Google does for Web History. A more sophisticated solution would be to provide "private search" checkbox as in I-Spy [51], or Google Web History, so that the results about this search are not stored. Concerning users who do not want to share search results at all, the system could be able to help their by suggesting recommendations from the community. However, if the observation history of a particular user is not stored, it would not be possible for the SICS to learn from past observations and to improve the quality of the recommendations. To summarize, the Implicit system can provide most benefits in the communities who do not mind sharing their search results (may be with exception of few of them) via the DB of observations. Otherwise, we can think about hiding user identity by anonymizing logs [31,53], but such a solution contradicts the motivation behind the Implicit system, which is helping to improve the search in a community of like-minded people who know each other.

Users can also participate in several communities, and we can imagine that they might have different privacy requirements on sharing their data. For instance, they might want to share full search history with one community and share only anonymized history with another community. This is not currently supported by the Implicit system, but can be added as an extension.

8 Conclusion and future work

We have presented a multi-agent recommendation system for web search, Implicit. The system adopts the Implicit Culture approach for recommending web links and exploits social ties existing within the community. Pilot study with real users proved that the system is working as intended, improving the quality of the web search in a community of people with similar interests. However, a more thorough and rigorous study with real users is required as a part of future work. The simulations we performed allowed us to evaluate different system components, and proved that the multi-agent architecture is viable for building recommendation systems and provides some advantages.

Future work can include analysis of social networks that emerge as the result of intra-agent communication following recommendations from SICS, and in particular, the analysis of patterns of behavior related to accepting results obtained from a certain agent (implicit trust relationships). Another possible direction of future work is dealing with learning about users interests/expertise and maintaining their models over time (e.g., to focus only on the recent interests). Different data mining techniques, such as clustering, could be applied in the inductive module to group similar observations together. For instance, agents can be clustered by interests and past actions of their users. Finally, it would be nice to study how the system deals with bigger communities, i.e. whether it is potentially scalable to the size of the community of Web users, and how it deals with dynamic communities where people enter, quit and their quality of their suggestions change over time.

Acknowledgements This research was partially supported by COFIN Project "Integration between learning and peer-to-peer distributed architectures for web search (2003091149_004)" and by MEnSA (Methodologies for the Engineering of Complex Software Systems: Agent-Based Approach), funded by the Italian government. The authors would like to thank Oscar Menghini for performing real-user evaluation of the system and Andrea Gastaldello for developing the inductive module.

Appendix A: User model

A user model contains the sequence of search keywords submitted by the user and the rules for acceptance of links for the keywords. Such acceptance rules are modeled as a set of probabilities of choosing a link for a keyword. As links we took the first m links provided by Google for each keyword and the rank of the list was adopted as an identifier. Since links provided by Google for a certain keyword change their ranks continuously, before the experiment we stored the links corresponding to the chosen keywords in a data set and replaced querying Google with getting links from this data set. The distribution of probabilities was built by using n keywords $k_1, k_2, \ldots, k_n$ and determining the probabilities $p(j|k_i)$ of choosing the jth link, $j \in \{1, \ldots, m\}$ while searching with the ith keyword. We assume that the user accepts one and only one link during the search for the keyword k_i, so $\sum_{j=1}^{m} p(j|k_i) = 1$. The distribution of probabilities in the user model 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 was equal to 10, the number of the links provided by Google m was equal to 10. The probabilities of link acceptance were produced by subjective evaluation of the Google results for five keywords and then duplicating these five rows. The resulting set of probabilities is shown in Table 4. The duplication was done in order to model correlated keywords. Such duplication did not affect the behavior of the system (as compared to 10 truly independent keywords) because links for each keyword were different, so if a user accepted a link for the keyword *weather* it only affected their further searches for the keyword *weather* and not for *oracle* or *weather1*. Also, since we added noise to rows independently, in actual user profiles (Table 5, 6, 7, 8, 9) rows 1…5 and 6…10 are different.

Table 5 The probabilities of acceptance of links for User 1

Keyword	Google rank of the link									
	1	2	3	4	5	6	7	8	9	10
tourism	0	0.02	0.06	0.3	0.05	0.19	0.11	0.1	0.11	0.06
football	0.03	0.03	0.09	0.23	0.22	0.12	0.12	0.09	0.05	0.02
java	0.24	0.24	0.06	0.05	0.05	0.08	0.08	0.13	0.05	0.02
oracle	0.07	0.09	0.34	0.16	0.01	0.09	0.08	0.06	0.04	0.06
weather	0	0.23	0.03	0.01	0.37	0.05	0.04	0.13	0.12	0.02
tourism1	0.06	0.03	0.09	0.35	0.05	0.15	0.07	0.03	0.07	0.1
football1	0.03	0.03	0.09	0.23	0.22	0.12	0.12	0.09	0.05	0.02
java1	0.24	0.24	0.06	0.05	0.05	0.08	0.08	0.13	0.05	0.02
oracle1	0.07	0.09	0.34	0.16	0.01	0.09	0.08	0.06	0.04	0.06
weather1	0.03	0.24	0.02	0.05	0.36	0.05	0.04	0.09	0.12	0

Links were different for each keyword and are numbered 1…10

Table 6 The probabilities of acceptance of links for User 2

Keyword	Google rank of the link									
	1	2	3	4	5	6	7	8	9	10
tourism	0.02	0.01	0.06	0.29	0.06	0.2	0.12	0.07	0.13	0.04
football	0.05	0.01	0.1	0.22	0.23	0.13	0.12	0.07	0.06	0.01
java	0.26	0.21	0.06	0.05	0.06	0.1	0.09	0.1	0.06	0.01
oracle	0.09	0.07	0.34	0.15	0.03	0.09	0.09	0.04	0.06	0.04
weather	0.02	0.21	0.03	0.01	0.37	0.07	0.05	0.1	0.13	0.01
tourism1	0.06	0.06	0.06	0.33	0.05	0.13	0.07	0.05	0.13	0.06
football1	0.05	0.01	0.1	0.22	0.23	0.13	0.12	0.07	0.06	0.01
java1	0.26	0.21	0.06	0.05	0.06	0.1	0.09	0.1	0.06	0.01
oracle1	0.09	0.07	0.34	0.15	0.03	0.09	0.09	0.04	0.06	0.04
weather1	0.03	0.27	0.02	0.03	0.37	0.01	0.02	0.13	0.07	0.05

Links were different for each keyword and are numbered 1…10

Table 7 The probabilities of acceptance of links for User 3

Keyword	Google rank of the link									
	1	2	3	4	5	6	7	8	9	10
tourism	0.03	0.03	0.08	0.35	0.08	0.18	0.11	0.03	0.07	0.04
football	0.06	0.04	0.11	0.28	0.26	0.11	0.1	0.04	0	0
java	0.28	0.25	0.08	0.09	0.09	0.07	0.07	0.07	0	0
oracle	0.07	0.09	0.36	0.17	0.03	0.09	0.08	0.04	0.01	0.06
weather	0.03	0.25	0.04	0.06	0.41	0.03	0.04	0.07	0.07	0
tourism1	0	0.02	0.07	0.32	0.06	0.19	0.13	0.07	0.08	0.06
football1	0.04	0.02	0.1	0.25	0.24	0.12	0.12	0.07	0.02	0.02
java1	0.28	0.25	0.08	0.09	0.09	0.07	0.07	0.07	0	0
oracle1	0.1	0.11	0.36	0.2	0.05	0.07	0.07	0	0	0.04
weather1	0.01	0.22	0.03	0.03	0.35	0.07	0	0.13	0.1	0.06

Links were different for each keyword and are numbered 1…10

Tables 5, 6, 7, 8, 9 show the five user models we have created from the probability distribution in Table 4 by adding noise uniformly distributed in [0.00,…,0.05]. We have chosen the uniform distribution of noise because we do not prioritize one links w.r.t. others. We added noise to each entry of the table and then renormalized all entries in order to keep the sum of each row equal to one.

Table 10 reports similarity among the initial and generated user models in terms of the Pearson correlation coefficient. Table 11 reports the difference between user models in terms of the number of relevant links.

Table 8 The probabilities of acceptance of links for User 4

Keyword	Google rank of the link									
	1	2	3	4	5	6	7	8	9	10
tourism	0.05	0.05	0.08	0.32	0.06	0.14	0.09	0.05	0.09	0.07
football	0.04	0.03	0.09	0.22	0.29	0.08	0.13	0.05	0.03	0.04
java	0.27	0.26	0.08	0.08	0.07	0.04	0.06	0.07	0.03	0.04
oracle	0.07	0.12	0.34	0.15	0.07	0.03	0.1	0.02	0.02	0.08
weather	0	0.26	0.01	0	0.44	0	0.06	0.09	0.1	0.04
tourism1	0	0.04	0.04	0.3	0.11	0.14	0.14	0.05	0.1	0.08
football1	0.04	0.03	0.09	0.22	0.29	0.08	0.13	0.05	0.03	0.04
java1	0.26	0.26	0.05	0.03	0.11	0.04	0.09	0.09	0.03	0.04
oracle1	0.11	0.13	0.34	0.18	0.03	0.04	0.06	0.01	0.03	0.07
weather1	0.01	0.24	0.06	0.01	0.43	0	0.02	0.11	0.1	0.02

Links were different for each keyword and are numbered 1…10

Table 9 The probabilities of acceptance of links for User 5

Keyword	Google rank of the link									
	1	2	3	4	5	6	7	8	9	10
tourism	0.01	0	0.08	0.33	0.09	0.18	0.08	0.06	0.09	0.08
football	0.04	0	0.13	0.25	0.27	0.11	0.08	0.06	0.02	0.04
java	0.26	0.22	0.09	0.07	0.09	0.07	0.05	0.09	0.02	0.04
oracle	0.1	0.12	0.34	0.15	0.02	0.07	0.07	0.03	0.01	0.09
weather	0.03	0.27	0	0	0.4	0.03	0.03	0.11	0.08	0.05
tourism1	0.05	0.04	0.06	0.28	0.05	0.18	0.12	0.05	0.1	0.07
football1	0.08	0.04	0.1	0.21	0.21	0.12	0.12	0.05	0.03	0.04
java1	0.29	0.27	0.04	0.04	0.06	0.06	0.08	0.1	0.01	0.05
oracle1	0.08	0.07	0.38	0.19	0.05	0.07	0.04	0.03	0.01	0.08
weather1	0.04	0.22	0.07	0	0.36	0.02	0.06	0.13	0.08	0.02

Links were different for each keyword and are numbered 1…10

Table 10 Pearson correlation between user models

User model	User model					
	Original	User 1	User 2	User 3	User 4	User 5
Original	1					
User 1	0.975	1	0	0	0	0
User 2	0.972	0.975	1	0	0	0
User 3	0.98	0.947	0.947	1	0	0
User 4	0.974	0.939	0.938	0.956	1	0
User 5	0.979	0.951	0.947	0.963	0.961	1

Table 11 The number of relevant links to each keyword in the user models

Keyword	User model					
	Original	User 1	User 2	User 3	User 4	User 5
tourism	4	5	4	3	2	2
football	5	4	5	5	3	4
java	3	3	4	2	2	2
oracle	4	2	2	2	4	4
weather	4	4	4	2	3	3
tourism1	4	3	3	3	5	4
football1	5	4	5	5	3	5
java1	3	3	4	2	3	3
oracle1	4	2	2	4	4	2
weather1	4	3	3	4	4	3
total	40	33	36	32	33	32

References

1. Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *VLDB'94: Proceedings of the 20th international conference on very large data bases* (pp. 487–499). San Francisco, CA: Morgan Kaufmann Publishers Inc.
2. Almeida, R., & Almeida, V. (2004). A community-aware search engine. In *Proceedings of the 13th international conference on World Wide Web.*
3. August, K. G., Hansen, M. H., & Shriver, E. (2001). Mobile web searching. *Bell Labs Technical Journal, 6*(2), 84–98.
4. Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM, 40*(3), 66–72.
5. Baldi, P., Frasconi, P., & Smyth, P. (2003). *Modeling the Internet and the Web: Probabilistic methods and algorithms*. London: Wiley.
6. Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing multi-agent systems with jade (Wiley series in agent technology)*. London: Wiley.
7. Birukou, A. (2009). *Implicit Culture Framework for behavior transfer*. Saabrucken: VDM Verlag.
8. Birukou, A., Blanzieri, E., & Giorgini, P. (2005). Implicit: An agent-based recommendation system for web search. In *AAMAS '05: Proceedings of the fourth international joint conference on autonomous agents and multiagent systems* (pp. 618–624). New York: ACM Press.
9. Birukou, A., Blanzieri, E., Giorgini, P., & Weiss, M. (2006). A multi-agent system for choosing software patterns. Technical Report DIT-06-065. University of Trento.
10. Birukou, A., Blanzieri, E., D'Andrea, V., Giorgini, P., Kokash, N., & Modena, A. (2007). Ic-service: A service-oriented approach to the development of recommendation systems. In *Proceedings of ACM symposium on applied computing. Special Track on Web Technologies* (pp. 1683–1688). New York: ACM Press.
11. Birukou, A., Blanzieri, E., D'Andrea, V., Giorgini, P., & Kokash, N. (2007). Improving web service discovery with usage data. *IEEE Software, 24*(6), 47–54.
12. Birukou, A., Blanzieri, E., Giorgini, P., & Giunchiglia, F. (2009). A formal definition of culture. In *Proceedings of the workshop on modeling intercultural collaboration and negotiation (MICON) at IJCAI'09*. Povo: DISI, University of Trento.
13. Blanzieri, E., & Giorgini, P. (2000). From collaborative filtering to Implicit Culture: A general agent-based framework. In *Proceedings of the workshop on agents and recommender systems*, Barcelona.
14. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*(4), 331–370.
15. Chau, M., Zeng, D., Chen, H., Huang, M., & Hendriawan, D. (2003). Design and evaluation of a multi-agent collaborative web mining system. *Decision Support Systems, 35*(1), 167–183.

16. Chen, L., & Sycara, K. (1998). Webmate: A personal agent for browsing and searching. In *AGENTS'98: Proceedings of the second international conference on Autonomous agents* (pp. 132–139) New York: ACM Press.
17. Church, K., Smyth, B., Cotter, P., & Bradley, K. (2007). Mobile information access: A study of emerging search behavior on the mobile Internet. *ACM Transactions on the Web, 1*(1), 4.
18. Dieberger, A. (1997). Supporting social navigation on the world wide web. *International Journal of Human-Computer Studies, 46*, 805–825.
19. Dieberger, A., Dourish, P., Höök, K., Resnick, P., & Wexelblat, A. (2000). Social navigation: Techniques for building more usable systems. *Interactions, 7*(6), 36–45.
20. Dourish, P., & Chalmers, M. (1994). Running out of space: Models of information navigation, short paper. In *HCI'94*, Glasgow, August.
21. Dreyfus, H. L., & Dreyfus, S. E. (2000). *Mind over machine: The power of human intuition and expertise in the era of the computer*. New York: The Free Press.
22. Dron, J. (2005). Control, termites and e-learning. In *Proceedings of IADIS international conference web based communities 2005*, pp. 103–110.
23. Dupret, G. E., & Piwowarski, B. (2008). A user browsing model to predict search engine click data from past observations. In *SIGIR'08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 331–338). New York: ACM.
24. Erickson, T., & Kellogg, W. A. (2003). Designing Information Spaces: The Social Navigation Approach, chapter Social Translucence: Using Minimalist Visualisations of Social Activity to Support Collective Interaction, pages 17–41. Springer Verlag.
25. Farzan, R., & Brusilovsky, P. (2007). Community-based conference navigator. In *Proceedings of sociUM workshop (1st workshop on "Adaptation and personalisation in social systems: Groups, teams, communities") at UM2007*.
26. Fox, S., Karnawat, K., Mydland, M., Dumais, S., & White, T. (2005). Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems, 23*(2), 147–168.
27. Geczy, P., Izumi, N., Akaho, S., & Hasida, K. (2007). Knowledge worker Intranet behaviour and usability. *International Journal of Business Intelligence and Data Mining, 2*(4), 447–470.
28. Gori, M., & Witten, I. (2005). The bubble of web visibility. *Communications of the ACM, 48*(3), 115–117.
29. Gwizdka, J., & Chignell, M. (1999). Towards information retrieval measures for evaluation of Web search engines, unpublished manuscript. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.3212.
30. Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems , 22*(1), 5–53.
31. Hong, Y., He, X., Vaidya, J., Adam, N., & Atluri, V. (2009). Effective anonymization of query logs. In *CIKM'09: Proceeding of the 18th ACM conference on Information and knowledge management* (pp. 1465–1468). New York: ACM.
32. Huberman, B. A., & Kaminsky, M. (1996). Beehive: A system for cooperative filtering and sharing of information. Technical report, Xerox Palo Alto Research Center.
33. Ishikawa, H., Ohta, M., Yokoyama, S., Watanabe, T., & Katayama, K. (2003). 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* (pp. 975–983). Oxford: Springer.
34. Ji, S., Zhou, K., Liao, C., Zheng, Z., Xue, G. R., Chapelle, O., Sun, G., & Zha, H. (2009). Global ranking by exploiting user clicks. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (pp. 35–42). New York: ACM.
35. Kobayashi, M., & Takeda, K. (2000). Information retrieval on the web. *ACM Computing Surveys, 32*(2), 144–173.
36. Kobsa, A. (2001). Tailoring privacy to users' needs. In *UM'01: Proceedings of the 8th international conference on User modeling 2001* (pp. 303–313). London: Springer.
37. König, A. C., Gamon, M., & Wu, Q. (2009). Click-through prediction for news queries. In *SIGIR'09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (pp. 347–354). New York: ACM.
38. Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM, 40*(3), 77–87.
39. Lieberman, H. (1995). Letizia: An agent that assists web browsing. In C. S. Mellish (Ed.), *Proceedings of the fourteenth international joint conference on artificial intelligence (IJCAI-95)* (pp. 924–929). Montreal/San Mateo: Morgan Kaufmann publishers Inc.
40. Lin, W., Alvarez, S. A., & Ruiz, C. (2002). Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery, 6*(1), 83–105.

41. Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing, 7*(1), 76–80.
42. Lopes, A. L., & Botelho, L. M. (2008). Improving multi-agent based resource coordination in peer-to-peer networks. *Journal of Networks, 3*(2), 38.
43. Massa, P., & Avesani, P. (2007). Trust-aware recommender systems. In *RecSys'07: Proceedings of the 2007 ACM conference on Recommender systems* (pp. 17–24). New York: ACM.
44. Menczer, Filippo (2003). Complementing search engines with online web mining agents. *Decision Support System, 35*(2), 195–212.
45. Middleton, S. E., Shadbolt, N. R., & De Roure, D. C. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems, 22*(1), 54–88.
46. Nonaka, I., & Takeuchi, H. (1995). *The knowledge creating company*. New York: Oxford University Press.
47. Passadore, A., Grosso, A., & Boccalatte, A. (2009). Indexing enterprise knowledge bases with agentseeker. In *WOA 2009. 10th National Workshop "From Objects to Agents"*.
48. Sarini, M., Blanzieri, E., Giorgini, P., & Moser, C. (2004). 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)* (pp. 131–146). French Riviera: IOS Press.
49. Schwab, I., & Kobsa, A. (2002). Adaptivity through unobstrusive learning. *KI, 16*(3), 5–9.
50. Singh, M. P., Yu, B., & Venkatraman, M. (2001). Community-based service location. *Communications of the ACM, 44*(4), 49–54.
51. Smyth, B., Balfe, E., Freyne, J., Briggs, P., Coyle, M., & Boydell, O. (2005). Exploiting query repetition and regularity in an adaptive community-based web search engine. *User Modeling and User-Adapted Interaction, 14*(5), 383–423.
52. Somlo, G. L., & Howe, A. E. (2003). Using web helper agent profiles in query generation. In *AAMAS'03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (pp. 812–818). New York: ACM.
53. Sun, X., Wang, H., & Li, J. (2009). Injecting purpose and trust into data anonymisation. In *CIKM'09: Proceeding of the 18th ACM conference on Information and knowledge management* (pp. 1541–1544). New York: ACM.
54. Turner, R., Turner, E., Wagner, T., Wheeler, T., & Ogle, N. (2001). *Using explicit, a priori contextual knowledge in an intelligent web search agent*. In *Modeling and using context* (pp. 343–352). Berlin: Springer.
55. van den Berg, B., van Es, R., Tattersall, C., Janssen, J., Manderveld, J., Brouns, F., Kurvers, H., & Koper, R. (2005). Swarm-based sequencing recommendations in e-learning. In *Proceedings of the 2005 5th international conference on intelligent systems design and applications (ISDA 05)*.
56. Vignollet, L., Plu, M., Marty, J. C., & Agosto, L. (2005). Regulation mechanisms in an open social media using a contact recommender system. In *Proceedings of the second communities and technologies conference* (pp. 419–436).
57. Walter, F., Battiston, S., & Schweitzer, F. (2008). A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems, 16*(1), 57–74.
58. Wei, Y., Jennings, N., Moreau, L., & Hall, W. (2008). User evaluation of a market-based recommender system. *Autonomous Agents and Multi-Agent Systems, 17*(2), 251–269.
59. Wei, Y. Z., Moreau, L., & Jennings, N. R. (2005). A market-based approach to recommender systems. *ACM Transactions on Information Systems, 23*(3), 227–266.
60. Yu, B., & Singh, M. P. (2002). An agent-based approach to knowledge management. In *CIKM'02: Proceedings of the eleventh international conference on Information and knowledge management* (pp. 642–644). New York: ACM Press.