SOaP: Social Filtering through Social Agents

Hui Guo, Thomas Kreifelts, Angi Voss

GMD-FIT.CSCW
German National Research Center for Information Technology
Institute for Applied Information Technology
Schloß Birlinghoven
D-53754 Sankt Augustin, Germany
Email: {firstname.lastname}@gmd.de

Abstract

The Web is becoming the premium source of information for a growing number of people. As a result, information overload arises as a problem of extracting useful information. Information gathering on the Web has become a time-consuming work. As an emerging technique for dealing with this problem, collaborative filtering (also known as social filtering) can improve retrieval precision and reduce the amount of time spent. In this paper we propose a social filtering system consisting of various types of agents which mediate between different people, groups and the Web. Agents work on behalf of their clients—users or other agents—under the specified security and/or privacy constraints. They interact with each other and allow people to cluster the URLs, rate and annotate the Web pages, and share the recommendations. Agents could also find people and groups with similar interests, bring people together to form groups and allow them to work within various groups to exploit the collected bookmarks. Eventually, the system could contribute to the social construction of knowledge on the Web.

Introduction

Less than five years old, the concept of collaborative filtering [Shardanand & Maes'95] has already spawned dozens of research prototypes, experimental proprietary systems, and even a few commercially available systems. Our project at the CSCW group of GMD was set up to create an open distributed platform that supports various Web-situated social applications through a wealth of interacting software agents. Ultimately, our agents should support the social construction and evolution of knowledge by communities of people wired on the Web.

By an "agent", we mean an autonomous software process which acts on behalf of a client. We speak of "social agents" when the agents support the social relationship between their clients. Social agents can play several roles:

- As workhorses, they can use the idle time to do repetitive routine work, and thus reduce their clients' overhead and increase the cost/benefit ratio of their service.
- As representatives, they can learn their clients' preferences and changing interests, present them to others while protecting the privacy, thus making personalization feasible.
- As mediators, they can exchange information, match interests, negotiate on behalf of their clients, and bring people with similar interests together so as to facilitate social affiliation.

The first application SOaP built upon the platform is to provide people with "live bookmarks" by combining content-filtering search engines with collaborative filtering techniques from recommender systems [Resnick & Varian'97] to exploit people's assessments of Web pages. It mines the Web for bookmark pages in order to acquire a critical, initial mass for recommendations. To account for motivational factors, it provides groups as contexts where people may feel more compelled to exchange and discuss assessments.

Related to our work are groupware systems that help people to share bookmarks and annotations, Web-based software agents [O'Leary'97], and recommender systems [Resnick'97]. These systems deal with a wide variety of information from the Internet, e.g. URLs, hyperlinks, annotations, queries, bookmark folders, emails, or newsgroup articles. They largely differ in the way that information can be shared. In some annotation sharing systems (JASPER [Davies et al.'95], users can or have to indicate explicitly how to distribute their information. In Answer Garden 2 [Ackerman & McDonald '96], users can specify how to escalate their questions. Thus, the developers claim, the problem of missing context can be reduced because people with a common background can be consulted first. Also, they argue that agents as mediators between persons can side-step certain social barriers. In some others systems, information can be distributed based on implicit criteria: content of information or ratings from human users. Letizia, WebWatcher, JASPER and FAB allow users to specify their interests in terms of keywords, and propose information which matches this description.

The following systems minimize the user's overhead. They extract URLs from sources that exist independently on the Web. PHOAKS [Terveen et al.'97] extracts citations of URLs from newsgroup articles, categorizes them, and recommends more recently or more frequently mentioned ones. While PHOAKS' output is not tuned to a particular user, GAB [Wittenburg et al.'95] and Siteseer [Rucker&Polanco'97] take the user's personal bookmark folders as an implicit declaration of interest. Both compute overlaps with other people's bookmark collections. GAB takes bookmark folders from explicitly selected users, while Siteseer takes bookmarks from a large collection of anonymous users. In contrast to our system they do not interpret the content of the folder titles. Yenta [Foner'97] is planned to consist of decentralized user agents, only. They will analyze the user's outgoing email and try to form clusters. User agents will compare each others' clusters and recommend users with a good overlap. The agents also exchange their acquaintances to find new candidates.

Compared to the systems above, SOaP is unique in its combination of functionality, and it can be easily extended to offer more, such as recommending users with similar interests. Within this system, users have personal workspaces in which they can input queries and get bookmarks back, and are allowed to cluster, annotate and rate these periodically updated bookmarks, as well as form groups to share bookmarks and annotations. The system can match users' interests against each other in terms of different levels of context (query, task, and group), make recommendations accordingly, and make it possible to aggregate knowledge out of user ratings and annotations. The system functionality is basically supported by several types of communicative agents: user agent, task agent, query agent, recommender agent and search agent. These agents have their own knowledge about the user or the environment, and interact with each other through a set of uniform performatives to exchange information in order to achieve their specific tasks. With SOaP, we hope to tackle two important problems, missing context and cold start. Missing context can be supplied by relating URLs to topics and groups. The cold start problems is avoided by starting with published bookmark collections, by using search engines to introduce new material without any personal overhead. In addition, the scalability of SOaP is expected to be guaranteed by distributing required functionality over individual agents which are distributed over networks.

In the following, we give an outline of our system comparing it with others with respect to well-known issues such as incentive and privacy, group context, etc. We also briefly introduce the underlying infrastructure and system status.

Agent-based social filtering for recommendations

In our system, information is filtered by communicative social agents which collect human users' assessments and match users' interests to derive recommendations. With agents, it is possible for users to find relevant bookmarks regarding specific topics, find people with similar interests, find groups with similar topics, and also to form groups for direct cooperations.

User interface and agents

Users interact with the system via dynamically created HTML pages. As shown in figure 1, every user has a personal workspace where all the user's tasks, queries, bookmarks and annotations are kept from unauthenticated access. One root page is linked to the pages of those groups the user belongs to, to a summary page of all groups, and to the pages of the user's tasks. A task page contains the queries issued for the task, the results obtained from the queries or bookmarks dropped by the user, along with their ratings and annotations. There are forms to formulate queries, to import bookmarks and to input annotations. A group page lists the members of the group and is linked to the group's task pages. Task pages of groups are essentially shared user task pages. As a difference, they indicate disagreements between ratings, and the annotations serve as a record of the discussion process in the group. The group summary page (not shown in the figure) summarizes the information about all groups along with their members, hotlists, and tasks. In this system, agents serve three purposes as illustrated in figure 1: they construct and maintain pages of the user interface; they wrap or manage databases which are accessible for retrieval; they perform retrieval subtasks.

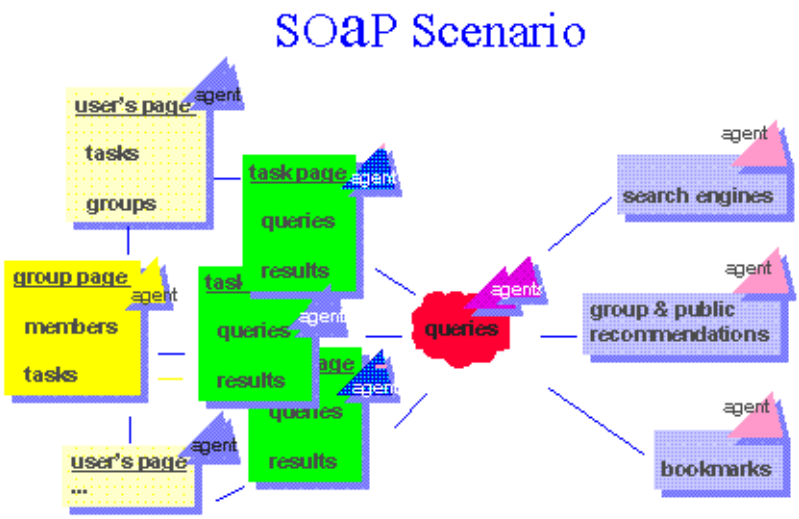

Figure 1: SOaP agents operate between users, groups and the Web.

SOaP Agents

In order to perform the services, agents in our system use knowledge about users, groups of users, the topics that are relevant to a user, the URLs that a user considers relevant to a topic, and a user's assessments of a URL, e.g. his or her ratings and annotations, in the context of a particular group or in connection with a particular topic. According to our design principle, this knowledge should be obtained without effort on the part of the user, or else it should be optional.

Agents in SOaP are specialized with regard to behavior and function. Each type of agent has specialized task to accomplish and plays different roles in the overall system. They interact with each other by exchanging messages of a certain type ("performative"). The interaction between them are conceived of as a conversation. Such conversation patterns may be formalized as finite-state-machines or in a distributed environment as high-level Petri nets following [Kreifelts & v. Martial'90]. The FSM specification of each conversation can be described by conversation tables which specify the state transitions as well as the message to be sent/received in particular states. They may be used to formally verify that the conversations is free of deadlocks even in the presence of message delay and mixed initiative of the conversation partner (not a simple turn-taking protocol) [Woetzel & Kreifelts'89]. The interaction between agents varies in complexity and duration, and principally can be captured by this concept of a conversation, Also it is possible to introduce new ways for agents to interact by specifying new conversation types, e.g. for the interaction between user agents and recommender agents if one wants to migrate certain tasks from task agents into user agents.

Within the system, as shown in figure 1, each registered user has his/her own unique user agent, the user's permanent representative which accepts the user's queries, then initiates task agents which in turn distribute queries to individual query agents. These query agents request services from a search agent—a service wrapping popular search engines like AltaVista, Infoseek—and a recommender agent which implicitly collects all ratings and annotations from users and performs matching and recommending. After receiving recommendations from these services, query agent, task agent and user agent will cooperatively merge, sort, filter, combine, cluster these information by relevance and present tailored results to users. By using this system, users can iteratively input and refine queries, rate the resulting URLs between -2 (very bad) and 2 (very good), make annotations, review recommendations from others who have similar interests, and gradually obtain a valuable bookmark list with respect to a specific task context which groups all queries, bookmarks, ratings, and annotations. This bookmarks list is alive and grows, for the recommender will autonomously push the new recommendations to each registered query agent which presents them to the user accordingly.

Since each user has his/ her own user agent and interacts with this agent for collecting bookmarks relevant to the topics he or she is interested in, this agent can capture the user's changing interests explicitly or implicitly. Also the user's interest is expressed in the form of queries in a specific task context, and the user's relevance judgment of certain bookmarks is made with respect to the current context. The user agent can derive s user profile out of all

these information which can dynamically reflect the user's preference and interest.

In our system, a recommender agent served as a common repository for a society of query agents and task agents. It registers each user's tasks and queries, and stores feedback from users in the forms of ratings and annotations. Compared with a search engine, it applies both content-based and collaborative filtering in order to recommend. The content is filtered by comparing the query with each bookmark which is described and indexed by keywords with their weights. For collaborative filtering, only URLs are selected which have obtained a high combined rating from other users. Recommendations are made based on calculating similarity of stored tasks and results. Task agents register their tasks with the recommender agent, which clusters the tasks (by defining neighbours of each task) and makes ratings and annotations automatically available to those neighbouring task agents within certain cluster. The recommender agent can also only recommend other task agents worthwhile negotiating with, which may be necessary for task agents which interact directly with each other in certain application like buying and selling on the Web.

Group context

To provide a richer and more constrained context for collaborative filtering, we propose a group agent, which store queries, results, ratings and annotations of group members like a recommender agent A group agent allows users to annotate and share bookmarks, matches users' interest and recommends highly-rated bookmarks along with annotations. By cooperating with other information resources like search agents, recommender agents, and other public group agents, a group agent behaves like a space for users to share recommendations and construct personalized views of group bookmark collections. Since users can only join a group by invitation, their identity is visible with regard to ratings or annotations within the group. This prevents non-serious anonymous ratings and ensures better recommendations.

The members of a group form a community; their recommendations will help each other so that providing many and good recommendations (evidenced by other users) can improve their prestige in the group. It seems that cold start is less of a problem in suitably selected groups and social motivations can stimulate substantial personal effort with a group. Last but not least, sharing recommendations within a group allows to detect trends faster than through other communication channels. Trends in groups may be interesting not just for group members, and more generally, some information should be allowed to leak out of a group so that other people may get a chance to join the group. Therefore our system not only allows to use a group's ratings for public recommendations, but allows to suppress selected annotations by specifying what kind of information should be *public / private* to non-members.

Related issues

A well-known problem of recommender system is the cold start problem. A recommender system can produce good recommendations only after it has accumulated a large set of ratings. In our system, a bookmark agent is introduced which collects and exploits publicly available bookmark pages by transforming them into the internal recommendation format for use by the recommender agent.

In order to protect the users' privacy appropriately, each information object which may be transmitted to other users as part of a recommendation has privacy-related access control. A typical example is the annotation. A user can make annotations anonymously, which means the user name won't be visible to whoever might get this annotated recommendation. Privacy control is also defined with respect to context, e.g., if a task is made private, all the queries under this task will be private by default as well.

Also in some recommender systems, there exists a "vote early and often" phenomenon, resulting in recommendations based on faked ratings. Such user actions are detected and inhibited by the user agent in our system: all the highly-rated bookmarks are sent to the public recommender agent only once, and repetitive ratings of the same bookmark will be simply ignored by the user agent. The user agent could also fetch each bookmark page and perform content-based filtering to filter out apparently inserious ratings, however this is not our focus.

The incentive (or cost/benefit) problem is addressed in the system in two ways. First, we keep the rating overhead for users low by interpreting users' actions as implicit votings: our system allows user to discard a URL or to annotate it without rating, and interprets this as negative or positive feedback, respectively. Secondly, our system provides a better quality of information retrieval than public search engines by exploiting human judgements and recommendations, so there is a basic benefit with no cost of rating and annotating. For mutual sharing of costs (and benefits), users are encouraged to rated and annotate, especially in the context of specific user groups.

SOaP implementation

As an application that allows users to assess Web pages and recommend them to other users and groups, SOaP is implemented on top of an open infrastructure. This infrastructure is a distributed platform for interacting software agents which provides a runtime environment with basic functionality, such as a unique agent naming/addressing schema, a message-based agent communication mechanism, fault recovery, persistency of agents, multiple agent accounting and a distributed directory service for agents. This platform is composed of the kernel layer-agent engine that hosts multiple agents and represents the basic runtime environment with kernel services like agent creation, termination, messaging, etc.-and a service layer that implements system services like the naming and alarming services. Both application and infrastructure are implemented in Java. The overall layered architecture is shown in figure 2.

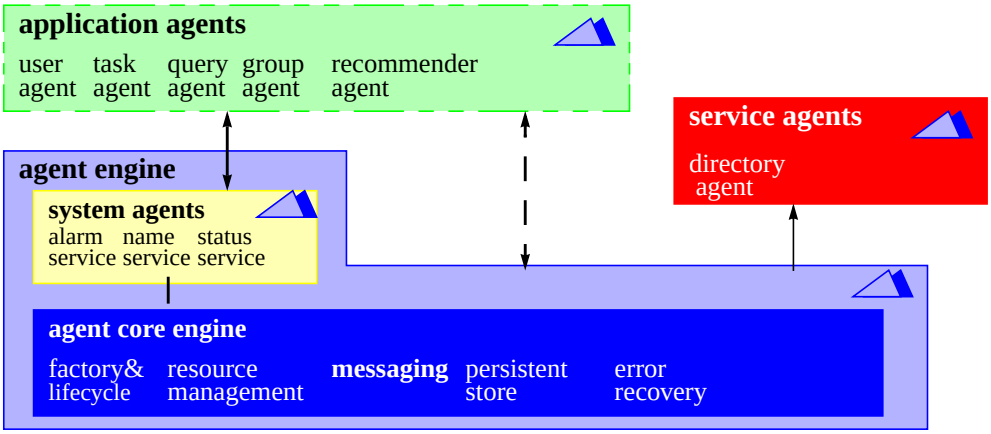

Figure 2: SOaP - the layered agent architecture (one host)

Status and future work

The first prototype released for tests within our research group at GMD in Septemeber'97 included agents which implemented the user interface, tasks and queries, and agents providing information by contacting search engine or collecting recommendations. This prototype is intended for demonstration, exploratory use, and evaluation in cooperation with an industrial partner from the oil business. Prospective users are members of project teams operating in oil field development. Team members are usually spread around the world, and may belong to several teams at the same time. Information retrieval and exchange is central to their work, and it has to occur both in similar areas or using similar techniques.

For the next release, we consider to design agents which provide interface to legacy information system and shared workspace system like BSCA [Bentley et al.'97]. In the context of SOaP, more problems arise such as matching, clustering, use of thesaurus to capture group-specific terminology, and creation of summary queries as operational definitions of tasks, and need further investigation in future research.

References

[Ackerman et al.'96] Ackerman, M.S., McDonald, D. W. "Answer Garden 2: Merging organization with collaborative help," in Proc. CSCW'96 (Cambridge MA, 1996), ACM Press, New York NY, 1996, pp. 97-105.

[Bentley et al.'97] Bentley, R. Appelt, W. Busbach, U., Hinrichs, E., Kerr, D., Sikkel, K., Trevor, J., Woetzel, G. "Basic support for cooperative work on the World Wide Web," Int. J. Human Computer Studies 46('97), 827-846

[Davies et al.'95] Davies, J., Weeks, R., Revett, M. "JAPSER: Communicating information agents for the WWW," in Proc. 4th World Wide Web Conf. (Boston MA, Dec.1995), World Wide Web Journal Vol. 1, 1, O'Reilly, Sebastopol CA, 1995, pp.473-482

[Foner'97] Foner, L. N. "Yenta: A multi-agent, referral-based matchmaking system," in Proc. Agents'97, 1st Int. Conf. on Autonomous Agents (Marina delRey CA, Feb. 1997)
http://lcs.www.media.mit.edu/people/foner/Yenta/overview.html

[Kreifelts & v. Martial'90] Kreifelts, Th., v. Martial, F. "A negotiation framework for autonomous agents," in Y. Demazeau, J.P. Mueller (eds.) Decentralized A.I. Vol 2, Proc. MAMAAW'90 2nd European Workshop on Modelizing Autonomous Agents and Multi-Agent Worlds, North-Holland, Amsterdam, 1990.

[Resnick & Varian'97] Resnick, P., Varian, H.R. "Recommender Systems," Comm. ACM 40, 3 (1997), 56-58

[Rucker et al.'97] Rucker, J., Polanco, M. J. "Siteseer: Personalized navigation for the Web," Comm. ACM 40, 3(1997), 73-75

[Terveen et al.'97] Terveen, L. Hill, W. Amento, B. McDonald, D. "PHOAKS: A system for sharing Recommen dations," Comm. ACM 40, 3(1997), 59-65

[Voss & Kreifelts '97] Voss, A., Kreifelts, Th. "SOaP: Social Agents Providing People with Useful Information," in Proc. Int. ACM SIGGROUP Conf. on supporting group work, ACM, New York NY, 1997, pp-291-298.

[Wittenburg, et al.'95] Wittenburg, K., Das, D., Hill, W., Stead, L. "Group asynchronous browsing on the World Wide Web," in Proc. 4th Int. World Wide Web Conf. (Boston MA, Dec. 1995), World Wide Web Journal Vol. 1, 1, O'Reilly, Sebastopol CA, 1995, pp.51-62.

[Woetzel & Kreifelts'89] Woetzel, G., Kreifelts, Th. "Deadlock freeness and consistency in a conversation system," in B.Pernici, A. A. Verrijn-Stuart (eds.) Office Information Systems: The Design Process, Proc. IFIP WG 8.4 Work. Conf. On Office Information Systems: The Design Process, North-Holland, Amsterdam, 1989, pp. 239-253.