Agent Collaboration as a Resource Discovery Technique

Ellen M. Voorhees
Siemens Corporate Research, Inc.
ellen@scr.siemens.com

Abstract

Asking someone with experience in a particular field is a natural way to learn about that field. For example, students often ask teachers for suggestions for further reading. This paper describes an agent-based information system that explicitly supports such a learning method. Each user is represented in the system by an agent that contains a profile of its user's experiences and scripts that automate frequently-used searching patterns. Profile segments from one agent can be directly imported into another agent, thereby facilitating resource discovery and leveraging the costs associated with script creation.

1 Introduction

The advent of large wide-area networks connecting many diverse collections of data creates the challenge of finding particular data of interest. Given the rate at which new data is added to such an environment, and the lack of consistent descriptions of the contents of the collections, even deciding where to look for the desired information is difficult. This paper describes an information management system that supports a user who must decide how to satisfy an information need by providing the user with the means of examining how other users accomplish their tasks. That is, the system allows users to benefit from the experience gained by other users.

The information management system, called InfoScout, is a prototype system under development at Siemens Corporate Research, Inc. The purpose of the prototype is to investigate ways of providing convenient, timely, and personalized access to information distributed in a large network. The InfoScout system is implemented as a set of agents where each agent represents either a particular user (a *userbot*) or a particular data collection (a *corpusbot*). All of a user's interactions with the system are through his or her userbot, which also acts as a repository for knowledge about the data environment itself that is gained during the interaction.

The next section of the paper gives a general overview of the InfoScout system to provide the appropriate context for a discussion of the information that can be shared among agents. This information, called the user's *profile*, is gathered and maintained by each userbot on behalf of its user. The profile is described in detail in Section 3.

2 System Overview

The current version of the prototype InfoScout system focuses on text retrieval [4] in large networks. The text collections can be any set of documents a provider is willing to allow others to search, such as an encyclopedia or a newswire. The system supports both browsing through the document collections and explicit, topic-oriented searches.

A major goal of the InfoScout system is to provide users with a personalized retrieval tool. To accomplish this, each user is represented in the system by his or her own userbot. Using feedback from the user about the appropriateness of the documents it retrieves, the userbot evolves into an expert on its user, adapting to that user's preferences and idiosyncrasies. Like Etzioni and Segal who view adaptability as a fundamental capability of softbots [1], we believe this personalization is an important step in combating information overload.

Each document collection is similarly represented by a single corpusbot that controls all access to the collection. A corpusbot accepts queries from the userbots, invokes the retrieval engine for each query, and returns the retrieval results back to the userbots. A corpusbot also maintains a list of the names of userbots currently browsing in the collection.

Userbots and corpusbots are implemented as long-lived processes that communicate through messages. Each message is an instantiation of a pre-defined message type. An agent may send a message to any agent whose name it knows. In the current prototype, an agent may also broadcast a message to either the entire set of corpusbots or the entire set of userbots. While this ability to broadcast to all other agents is very useful, it is clearly infeasible in a realistically large network. The InfoScout system can be used in a large a network by restricting broadcasts to some network-defined domain similar to (although finer-grained than) the Internet's division into the `.com`, `.edu`, etc. domains. The Isis Distributed Toolkit [2], which we use as our communication medium, provides support for domain-dependent communication. By enabling userbots to move among domains just as knowbots were envisioned as moving among data resources [3], the InfoScout system achieves the effect of true broadcasts, allowing userbots to find and access resources that are not within the immediate vicinity of their home machines.

3 The Profile

A userbot maintains four kinds of information for its user: specialties, peer names, category descriptions, and scripts. All of the information stored by one userbot can be examined by other userbots,[1] and all but the list of specialties can be imported into other userbots. Importing a profile segment from userbot A into userbot B causes a copy of the segment to be inserted into B's profile. No permanent connection of any kind is maintained between the two segments.

3.1 Specialties

The specialty list is a set of phrases such as "graph theory" or "gourmet cooking" that describes areas in which a user is presumed to have interest and expertise. The system attaches no meaning to these phrases; they are provided as hints to other users about the topics that might be included in this user's profile. A user selects his or her own set of specialties, and can modify the list at any time.

The direct way to search for a user who has knowledge about a particular area is to query the userbots' speciality lists. A message containing the topic of interest is sent to all userbots. A userbot that has the requested topic in its specialty list replies to the message with its name. Using a userbot's name, the user can access that userbot's profile and decide what, if anything, to import.

3.2 Peer Names

The peer names segment of a profile is a cache of the names of userbots that the user wishes to access again. The names can initially be learned in a variety of ways: through searches for experts, by "meeting" other userbots as described below, or through sources external to the InfoScout system.

[1] Being able to examine the complete profile of another user is obviously a massive invasion of privacy that will become even more serious when the agents become better at truly modeling individual preferences. We recognize the problem, and have built in some token restrictions such as being able to mark a script as private, but have concentrated on investigating the power of the learning-from-others paradigm rather than the security and privacy issues.

Indeed, since the names are part of a userbot's profile, a name can also be learned by knowing a userbot that knows it.

3.3 Categories

Categories permit a user to organize the data space according to his or her perspective. The user may group individual text collections into higher level units called categories. For example, an encyclopedia, a dictionary, and an almanac could be grouped into a "Reference" category. Categories may themselves be members of other categories, allowing the user to create a lattice of categories. The category segment of a profile contains the definition of its user's category lattice.

In addition to the definition of the lattice, the segment contains usage statistics for each category. The statistics include such items as the number of times the category was browsed, the number of times it was searched (i.e, the number of times queries were submitted to its component document collections), the number of relevant documents retrieved in those searches, etc. These statistics form a rudimentary characterization of how useful the user has found the category in the past. When a user imports a category description from another user, these statistics are also imported to serve as an initial indication of the category's quality. Of course, since the users will interact differently with the category, the statistics will diverge in the two userbots.

A user browses through the data space by traversing the category lattice. When the user enters a category, the userbot displays the names of the immediate descendents of the category and sample texts from the component text collections. When a text collection (a bottom-level category) is entered, the user *meets* the other userbots currently in that collection. That is, the user is informed of the names of the userbots of the other users also browsing there. This provides a second, more serendipitous method for finding experts in a particular field. Since users with a shared interest are likely to use collections that contain information about that interest, knowing a single data collection that contains appropriate documents can lead to other, perhaps more experienced, users who have the same interest.

3.4 Scripts

Scripts are the main mechanism by which users can customize and extend the functionality of their userbots. Scripts are arbitrary, parameterized subprograms written in a C-like language that contains primitives for interacting with the data environment. The language also permits the script to execute a command at the operating system level, and to transfer execution of the script to another machine. Scripts are implemented as a specific application of technology developed to facilitate generic flow-control problems [5].

The main purpose of scripts is to allow a user to automate recurring data manipulation tasks. For example, a baseball fan could write a script that obtains the previous day's box scores from a newswire, invokes an analysis program to calculate the batting averages of the members of her fantasy league team, and then displays a summary of the team's performance to date. Scripts are a powerful tool, even though creating a script implies all the difficulties involved in writing programs. Since the script language allows commands to be executed at the operating system level, the script has access to every resource or program that is available to an arbitrary user of the current machine. By writing code segments that further manipulate the results of these commands, the user can create custom applications.

A second purpose of scripts is to provide a mechanism for multitasking. In the absence of scripts, the user would have only a single thread of control since all interaction between a user and the InfoScout system occurs through the user's userbot. Once a script is invoked, however, it executes independently of the userbot until completion, at which time a trace of the entire execution is uploaded to the userbot. All explicit searches are performed as executions of a generic query-processing script, and thus occur in parallel with the user's other use of the InfoScout system.

Finally, scripts make it easy for users to share their knowledge. While the original construction of a script may take considerable effort, once created the script benefits the entire user community. Users who import scripts are not only saved the cost of writing their own scripts, but may also discover in the script references to resources that were previously unknown to them.

4 Conclusion

Knowing where to look for information in a large, loosely-coupled data environment such as the Internet is a difficult problem. The InfoScout system supports users in this task by providing access to the experience of other users. A user can initiate a search for specialists in a given area and import the searching strategies used by those specialists. Additionally, a user can browse in a text collection and import the knowledge of other users browsing there.

Knowledge-sharing is facilitated by encapsulating searching strategies in scripts, but scripts offer other benefits as well. By providing a mechanism for automating repetitive search behavior, scripts allow users to customize and extend the functionality of their agents. Scripts also distribute the processing required by a user of the system.

In the prototype InfoScout system, all decisions regarding what to import from whom are made by the user. We are investigating ways to automate these decisions and thus increase the autonomy of the agents.

Acknowledgements

The InfoScout system is an on-going project at Siemens. Narendra Gupta, Ben Johnson-Laird, and Scott Rickard have participated in the system's design and implementation. Russ Greiner read a draft of this paper and suggested improvements. My thanks to them all.

References

[1] Oren Etzioni and Richard Segal. Softbots as testbeds for machine learning. In *Working Notes of the 1992 Spring Symposium on Knowledge Assimilation*, pages 43–50. American Association for Artificial Intelligence, March 1992.

[2] Isis Distributed Systems, Inc. *Isis Distributed Toolkit Version 3.0 User Guide and Reference Manual.* Ithaca, NY, 1992.

[3] Robert E. Kahn and Vinton G. Cerf. The Digital Library Project Volume 1: The world of knowbots (DRAFT). Corporation for National Research Initiatives, March 1988.

[4] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[5] C. Daniel Wolfson, Ellen M. Voorhees, and Maura M. Flatley. Intelligent routers. In *Proceedings of the Ninth International Conference on Distributed Computing Systems*, pages 371–376. IEEE, June 1989.