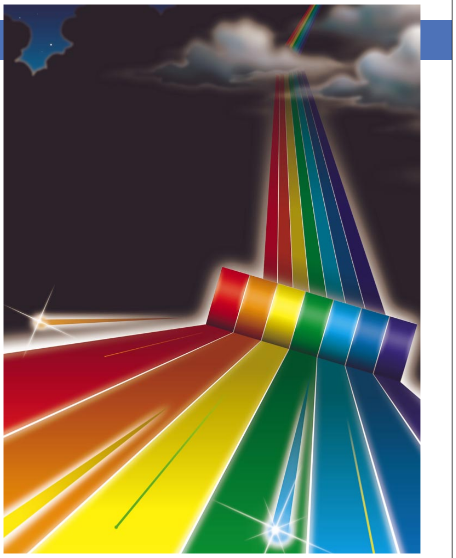

Agents for Information Gathering

José Luis Ambite and Craig A. Knoblock
Information Sciences Institute and Department of Computer Science,
University of Southern California

With the growing number of information sources available through networks, the problem of how to combine distributed, heterogeneous information sources is becoming increasingly critical. The available sources include traditional databases, flat files, knowledge bases, programs, and so forth.

Traditional approaches to building distributed or federated systems do not scale well. Current systems, such as search engines or topic directories on the World Wide Web, provide only limited capabilities for locating, combining, processing, and organizing information.

The solution to this problem is to provide access to the large number of information sources by organizing them into a network of *information agents*. Each agent provides expertise on a specific topic by drawing on relevant information from other information agents. To build such a network, we need an architecture for a single agent that can be instantiated to provide multiple agents. Our design is based on our previous work on the SIMS system,[1,2] an information mediator that provides access to heterogeneous data and knowledge bases. We need to consider several aspects that are critical for any agent-based system: agent organization, knowledge of an agent, communication language and protocol, query processing, and learning capabilities. We briefly discuss each of them in turn (for a more detailed analysis, see "Agents for Information Gathering"[3]).

Agent organization. We expect that agents will be developed to serve the information needs of users in specific domains. More complex agents that deal with wider or deeper areas of knowledge will appear in an evolutionary fashion, driven by the market forces of applications that can benefit from using them. We believe that this bottom-up approach can lead more realistically to the development of *useful* large knowledge bases than can top-down ones such as Cyc.[4]

Similar to the way current information sources are independently constructed, information agents can be developed and maintained separately. Building an appropriate *wrapper* around an existing repository will turn it into a simple information agent. A wrapper is the interface code that will allow it to conform to the conventions of the organization. In general, only one such wrapper needs to be built for any given type of information source (for example, relational database, object-oriented database, and flat file). This greatly simplifies the individual agents, because they only need to handle one underlying language and protocol, making it possible to scale the network into many agents having access to many different types of information sources.

Figure 1 shows an example network of information agents in the Logistics Planning application domain. To perform its task, the top-level agent needs to obtain information on different topics, such as transportation capabilities, weather conditions, and geographic data. The other agents also integrate many sources of information relevant to their domain of expertise. For example, the `Sea_Agent` combines assets data from the `Naval_Agent` (such as ships from different fleets), harbor data from the `Harbor_Agent`, and port data from the `Port_Agent` (such as storage space, cranes in harbors, depth of channels, and so forth—information that comes, in turn, from repositories of different geographical areas). Note that the network must form a directed acyclic graph to prevent queries from looping endlessly. Two agents may draw information from a third one, possibly for different purposes.

The knowledge of an agent. Each information agent is specialized to a single application domain and provides access to the available information sources within that domain. Each agent contains an *ontology* of its domain of expertise—its *domain model*—and models of the other agents that can provide relevant information—its *information source models*. The domain model establishes the terminology for interacting with the agent. Each information source model has two main parts:

- A description of the source contents, including the terms understood by the source, which will be used to communicate with it.
- A description of the relationship between the source concepts and the concepts in the domain model.

The system uses these mappings for transforming a domain-model query into a set of queries to the appropriate information sources. All these models need to be stated in a common language expressive enough to capture all the relevant distinctions found in the sources. In SIMS, we chose a description logic, the Loom knowledge-representation language.[5]

Query processing. A critical capability of an information agent is the ability to flexibly and efficiently retrieve and process data. Query processing requires developing a plan for obtaining the requested data. This includes selecting the information sources to provide the data, the processing operations, the sites where the operations will be performed, and the order in which to perform them.

Some desirable features of the query processor are the ability to execute operations in parallel, to augment and replan queries that fail while executing other queries, and, most interestingly, to gather additional information at runtime to aid the query processing.[6]

Communication language and protocol. The organization of agents needs a *common* communication language and protocol (otherwise, a network of $O(n)$ agents could require as many as $O(n^2)$ bilateral translations. Each agent needs to handle at least a subset of the common protocol and be able to perform a syntactic translation between the common data model and its own data model. In SIMS, we use Loom as the common content language and the Knowledge Query and Manipulation Language[7] as the protocol to organize the dialogue between agents. Queries to an information agent are expressed in terms of its domain model, so there is no need for other agents or a user to know or even be aware of the terms used in the underlying information sources.

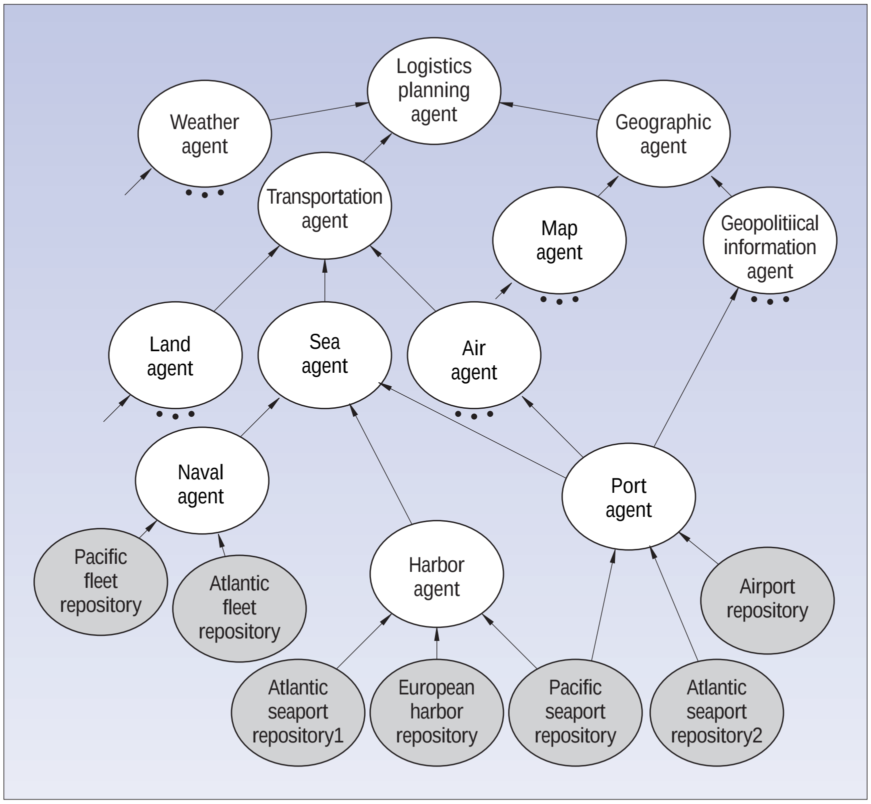

Figure 1. Network of information retrieval agents.

IEEE EXPERT
INTELLIGENT SYSTEMS
& THEIR APPLICATIONS

Learning. An intelligent information agent should be able to improve both its accuracy and performance over time, and deal with the changing environment. In SIMS, we have explored three forms of learning. First, the agents can cache frequently retrieved or difficult-to-retrieve information.[8] Second, an agent can learn about the contents of the information sources to minimize retrieval costs. In particular, an agent can perform semantic query optimization, based on its declarative models and rules learned from the sources, to reformulate a query plan into a cheaper, but semantically equivalent, plan.[9] Finally, an agent can analyze the contents of its information sources to refine its domain model and better reflect the currently available information. Because of the dynamic nature of information and the autonomy of the agents, an agent's *source models* may not accurately represent the *actual contents* of the sources. Thus we need to be able to recognize such disparities and resolve them, automatically, if possible.[10] All these forms of learning improve the efficiency of the system, and the last one also its correctness.

Discussion. The SIMS-based approach has several features we feel are crucial to the success of any information-gathering agent. These include:

- *Modularity*—for representing an information agent and information sources. This is afforded by the separate domain and source models, and the uniform representation and communication languages.
- *Extensibility*—for adding new information agents and information sources. The modular design allows the addition of a new source model without interfering with the mappings of previous sources. An agent can export part or all of its domain model for others to build upon.
- *Flexibility*—for selecting the most appropriate information sources to answer a query. The explicit models allow the agent to dynamically plan for alternative sources when a source or the network goes down. Cached information can be accessed seamlessly.
- *Efficiency*—for minimizing a given query's overall execution time. Building parallel query access plans, using semantic knowledge to optimize the plans, caching retrieved data, and learning about information sources contribute to provide efficient access to large numbers of information sources.
- *Adaptability*—for tracking semantic discrepancies among the domain and source models of an agent, and update them as appropriate.

References

1. Y. Arens et al., "Retrieving and Integrating Data from Multiple Information Sources," *Int'l J. Intelligent and Cooperative Information Systems*, Vol. 2, No. 2, June, 1993, pp. 127–158.

2. Y. Arens, C.A. Knoblock, and W.-M. Shen, "Query Reformulation for Dynamic Information Integration," *J. Intelligent Information Systems, Special Issue on Intelligent Information Integration*, Vol. 6, Nos. 2–3, 1996, pp. 99–130.

3. C.A. Knoblock and J.L. Ambite, "Agents for Information Gathering," in *Software Agents*, J. Bradshaw, ed., AAAI/MIT Press, Menlo Park, Calif., in press.

4. D. Lenat and R.V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley, Reading, Mass., 1990.

5. R. MacGregor, "The Evolving Technology of Classification-Based Knowledge Representation Systems," in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John Sowa, ed., Morgan Kaufmann, San Francisco, 1990, pp. 385–400.

6. C.A. Knoblock, "Building a Planner for Information Gathering: A Report from the Trenches," *Proc. Third Int'l Conf. AI Planning Systems*, AAAI Press, Menlo Park, Calif., 1996, pp. 134–141.

7. T. Finin et al., "KQML as an Agency Communication Language," *Proc. Third Int'l Conf. Information and Knowledge Management*, ACM Press, New York, 1994, pp. 456–463.

8. Y. Arens and C.A. Knoblock, "Intelligent Caching: Selecting, Representing, and Reusing Data in an Information Server," *Proc. Third Int'l Conf. Information and Knowledge Management*, Nat'l Inst. of Standards and Technology, Gaithersburg, Md., 1994, pp. 433–438.

9. C.-N. Hsu and C.A. Knoblock, "Using Inductive Learning to Generate Rules for Semantic Query Optimization," in *Advances in Knowledge Discovery and Data Mining*, G. Piatetsky-Shapiro et al., eds., AAAI Press, 1996, pp. 201–218.

10. J.L. Ambite and C.A. Knoblock, "Reconciling Agent Models," *Proc. Workshop on Intelligent Information Agents*, ACM Press, 1994.

José Luis Ambite is a graduate research assistant at the Information Sciences Institute and a PhD student at Department of Computer Science of the University of Southern California. His research interests include information integration, multiagent systems, planning, and knowledge representation. He received his degree of Electrical Engineer at the Technical University of Madrid (Ingeniero de Telecomunicacíon, ETSIT-UPM), and an MS in computer science from USC. He was also awarded a Fulbright/Ministerio de Educacíon y Ciencia scholarship. Contact him at USC/ISI, 4676 Admiralty Way, Marina del Rey, CA 90292; ambite@isi.edu.

Craig A. Knoblock is a senior research scientist at the University of Southern California's Information Sciences Institute and a research assistant professor in USC's Computer Science Department. His research interests involve developing and applying planning, machine learning, and knowledge representation techniques to the problem of information gathering and integration. He received his BS in computer science from Syracuse University, and his MS and PhD in computer science from Carnegie Mellon. Contact him at USC/ISI, 2676 Admiralty Way, Marina del Rey, CA 90292-6696; knoblock@isi.edu.