Adaptive Agents for Information Gathering from Multiple, Distributed Information Sources

Yizhong Fan*
Susan Gauch+

*Motorola, Inc.
+Department of Electrical Engineering and Computer Science
University of Kansas
Lawrence, Kansas 66045
{fanyz@cig.mot.com; sgauch@eecs.ukans.edu}

Abstract

The expansion of information available on the Web has been explosive. The initial approach of collecting all Web pages into a single location and indexing them is limited in its ability to deal with this explosion. Different search engines provide access to different collections, requiring users to access multiple information sources to meet their needs. A scalable approach to information gathering from multiple sources is described which is based on the use of distributed information agents to route queries to the most appropriate sources and fuse information they provide. This paper describes an intelligent, adaptive Web search tool that can not only locate relevant information sources for the user, but also adapt to the frequent changes of the dynamic Web environment. We employ a multiple agent architecture to intelligently categorize and broker queries to remote search engines, learn and continuously update confidence factors for the quality of results provided by each search engine, and automatically detect and adapt to changes in the syntax used by the search engines. Our work is an extension of ProFusion (http://www.profusion.com) [Gauch 96b], a Web meta-search engine developed at the University of Kansas which can currently broker information from nine remote search engines.

1. Introduction

The explosive growth of the World Wide, and the resulting information source overload, has led to a mini-explosion in World Wide Web search engines. Users do not have the time to evaluate multiple search engines to knowledgeably select the best for their uses. Nor do they have the time to submit each query to multiple search engines and wade through the resulting flood of good information, duplicated information, irrelevant information and missing documents. Similar to other meta-search engines such as SavvySearch [Dreilinger 96] and MetaCrawler [Selberg 95], ProFusion sends user queries to multiple underlying search engines in parallel, retrieves and merges the resulting urls. The initial version of ProFusion [Gauch 96a] supported six (now nine) underlying search engines. It categorized incoming queries and routed each query to the best search engines for the identified category based on hard-coded confidence factors for each search engine in each category. ProFusion employs an n-gram-based matching algorithm to identify and remove duplicate urls and creates one relevance-ranked list from the multiple sources, normalizing the weights reported by the search engines, adjusting them with the confidence factors for the search engine, and then merging the results.

While an effective search tool, ProFusion had three major problems: 1) the confidence factors were manually calibrated, a laborious process which was done infrequently, causing the confidence factors to become out of date. Adding new search engines was also difficult due to the need to manually calibrate any new information source; 2) search engines frequently went down and/or became slow and ProFusion did not detect this and route queries to working and/or faster sources of information; 3) the underlying search engines frequently changed the syntax of their results pages (which are parsed by ProFusion) and, less frequently, the syntax of the query string they expected. Updating ProFusion to accommodate these syntax changes required programmer input. To address these issues, a second generation ProFusion was developed which employs adaptive agents to provide higher-quality, more reliable service.

2. Background

2.1 Adaptive Autonomous Agents

Autonomous agent research is currently an extremely active research area in artificial intelligence (AI). Agents represent a shift in emphasis towards "behavior-based AI" as opposed to traditional "knowledge-based AI". As Pattie Maes mentions in [Maes 92], the autonomous agent approach is appropriate for the class of problems that require a system to autonomously fulfill several goals in a dynamic, unpredictable environment. Many of the architectures for autonomous agents that have been

proposed bear characteristics in common. According to Maes, these shared characteristics are: task-oriented modules, task-specific solutions, de-emphasized representations, decentralized control structure, and learning and development.

Autonomous agents must be able to learn in order to improve their performance and adapt to the changes. True adaptive behavior is an inherently dynamic, continuous process. In complex, realistic environments an agent cannot afford to learn every fact that can possibly be learned. Therefore, agents need to first come up with a selection heuristics to decide what is important to learn, and what may be ignored. However, an agent designer needs to decide not only *what* an agent should learn from the environment, but also *how* the agent will learn from the environment. Several learning methods for autonomous agents have been proposed, in particular reinforcement learning [Sutton 91], classifier systems [Holland 86], model builders [Drescher 91] and mixed methods [Sutton 90].

2.2 Selected Agent-Oriented Systems for the Web

Although the area of intelligent agents for the World Wide Web is still in its infancy, like the Web itself it is undergoing rapid change. In this section, we will introduce a few adaptive Web browsing or searching agents that are related to this work. A more complete summary appears in [Haverkamp 98].

Adaptive browsing agents for the WWW help the user locate new information of interest to them. As the user operates a conventional Web browser such as Netscape, the agent tracks the user's behavior and recommends some links for the user to follow. In general, an adaptive browsing agent program is a continuously running process that operates in parallel with the user. It follows links and looks ahead for the information inferred to be of interest to the user, but the user is the one with absolute control over the browsing path [Casasola 97]. Therefore, the agents for browsing are individual assistants which learn what a particular user's interests are. Two examples of such systems are Letizia [Liebermean 95] and, more recently, WebMate [Chen 98].

Adaptive search agents for the WWW are systems which not only search for information of interest to their users, but also continuously change after every search to improve the quality of their search results. Generally, these are implemented as multi-agent systems that consist of information filtering agents and information discovery agents. The information filtering agents are responsible for the personalization of the system and for keeping track of (and adapting to) the interests of the user. The information discovery agents are responsible for information resource handling, adapting to those information resources, finding and fetching the actual information that the user is interested in. Some representative systems are Amalthaea [Moukas 96] and CIG Searchbots [Lesser 98].

Finally, many projects are defining and evaluating agent-based architectures [Jennings 98], with several specializing in fusing information from distributed information sources. These efforts are occurring in the database world [Arens 96] as well as text and/or heterogeneous information sources [Bayardo 97].

3. ProFusion's Multi-Agent Architecture

The new ProFusion multi-agent system consists of four different types of agents, namely, a *dispatch agent*, a *search agent*, a *learning agent*, and *a guarding agen*t. Figure 3-1 shows the control flow and intercommunication between agents in the ProFusion system.

The *dispatch agent* communicates with the user and then dispatches queries to the search agent and the learning agent. The *search agent* interacts with the underlying search engines and is responsible for reporting search results, confidence factors, and time-out values of the underlying search engines to the dispatch agent, as well as invoking the guarding agent when necessary. The *learning agent* is in charge of the learning and development of the underlying search engines, in particular adjusting the knowledge bases (confidence factors). The *guarding agent* is invoked when a search engine is down and it is responsible for preventing the dispatch of future queries to a non-responsive search engine as well as detecting when the search engine is back online.

Our multi-agent architecture demonstrates various desirable agent characteristics [Maes 94] including: task-oriented modules, task-specific solutions, de-emphasized representations, decentralized control structure, and learning and development. The search agent, learning agent, and guarding agent each consists of a set of identical competence modules, each of which is responsible for one of underlying search engines (*task-oriented modules*). These competence modules are self-contained black boxes which handle all the representation, computation, reasoning, and execution that is necessary for its particular search engine (*task-specific solutions*). Although all six competence modules for each of the three agents are implemented using identical code, each uses its own local configuration and knowledge files to achieve its competence. In other words, there is no central representation shared by the several modules. Instead, every task-oriented module represents locally whatever it needs to operate autonomously. The localized representations of different modules are not related (*de-emphasized representations*). Except for the dispatch agent, all of the competence modules of the search agent, learning agent, and guarding agent operate concurrently. None of the modules is "in control" of other modules

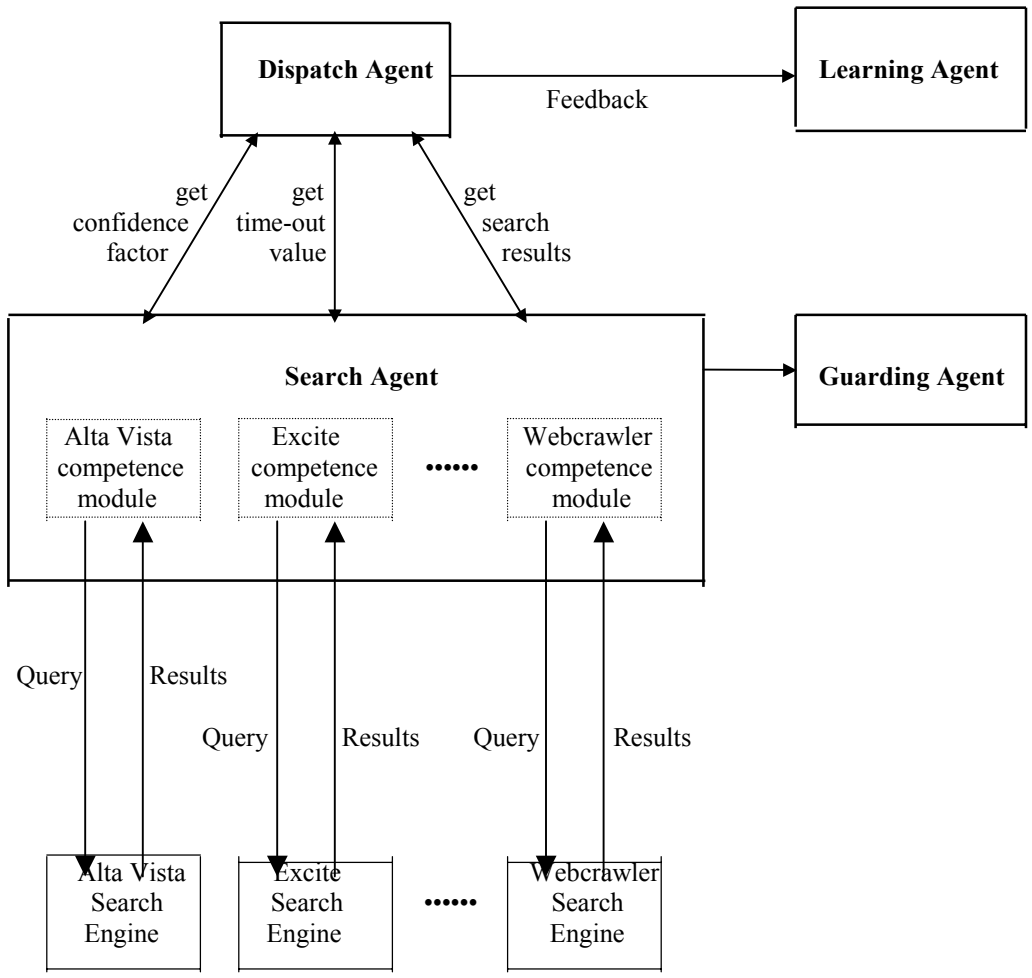

Figure 3-1 Agent Intercommunication and Control Flow Diagram

(*decentralized control structure*). Because of this distributed operation, the new system is able to react flexibly to changes in the environment and make the corresponding adjustments.

4. ProFusion Agents' Adaptation Algorithms

A second major difference between the new multi-agent system and previous ProFusion system is the ability to adapt. Three adaptation algorithms are implemented for the new system to solve the three problems described in Section 1. To review, these are: adapting to changing search engine performance, adapting to changing search engine's response time, and adapting to changing search engine's result formats.

4.1 Adapting to Changing Search Engine Performance

Originally, ProFusion used a hand-built, static knowledge base which did not reflect the dynamic changing performance of each of the underlying search engines. There are two main problems to address with respect to agent learning: what to learn and how to learn.

1. What to learn?

The heuristic we use here is that the search engines' performance can be inferred from user feedback. One fact we can learn is that if a url on the result page is chosen by the user, generally this implies that the search engine which contributed this url performed better on the query than those search engines which contributed urls that were ignored by the user. In other words, the confidence factor for the search engine that contributed the selected url should be higher for the query's category than search engines whose contributions are overlooked.

However, not all links followed by the user at any one time are worth learning. In general, the earlier a link on the result page is selected, the more relevant the link is. According to a study of SavvySearch [Dreilinger 96], on average fewer than 2 links are followed by the user per search. Therefore, to reduce the amount of computation

and allow each user to contribute equally, our learning process learns from only first link followed by each user.

2. How to learn?

We use classifier-based learning where the classifiers are the 13 categories in each search engine's knowledge table. The confidence factor associated with each category is the "strength" of the classifier which represents how well a particular search engine performs on queries in a particular category. We use the rationale that if our confidence factors were perfect, then the top ranked item on the results page would be the first url followed by the user. If, instead, the user follows some lower ranked url, then our confidence factor for the search engine providing the url was too low compared to the confidence factors for the search engines that provided more highly ranked urls. Since the rank order is based on the weight reported by the search engine multiplied by the confidence factor (CF), increasing the CF for the successful search engine will cause future results to be weighted somewhat higher with respect to the unsuccessful search engines. In a sense, ProFusion uses collaborative feedback to continuously calibrate the various search engines performance, much in the same way that collaborative feedback can be used to filter Web pages [Starr 96] or newsgroups [Konstan 97].

The adjustments to the CFs are normalized so that all values remain within [0, 1], which may cause the CFs for the unsuccessful search engines to be decreased. The CFs for search engines which did not contribute any results above the selected url remain unchanged.

4.2 Adapting to Changing Search Engine's Response Time

Another difficulty with the original ProFusion is that it used a fixed time-out value when retrieving results from remote search engines. This algorithm is devoted to solve it. However, different search engines have different normal response times and for a given search engine, the response time varies with the time of day and day of the week. In addition, ProFusion needed to adapt to avoid temporarily unavailable search engines. Two adaptive algorithms were employed:

1. Using a dynamically changing time-out value for each search engine.

Each time a search is performed, the current time-out is used for the corresponding search engine. If the search engine fails to respond within that time, the communication is broken and the main broker doesn't wait for any results. After each search, the current time-out value for the search engine is dynamically re-computed based on the response times for the search engine's previous twenty searches. In this way, the system will always use a time-out value that reflects the current responding speed for that search engine, which ultimately speeds up the overall system's responding rate. This also allows us to provide the option of "Choose Fastest 3" search engines to users who want quick results.

2. Automatically enabling and disabling search engines

If a search engine is detected to be not responding, then the search engine is disabled, but it will be enabled later once it is detected to respond again. To disable an unresponsive search engine, the status of the search engine is detected after each search. If the total number of results returned from the 20 most recent searches are 0, then we assume the corresponding search engine is currently down. To enable a disabled search engine, the status of the search engine is checked by the guarding agent once an hour by running 10 simple queries which are expected to produce search results. If no results are retrieved, the agent will sleep for an hour before another check starts. Otherwise, the search engine will be enabled immediately. Before each search, the status of each selected search engine is checked and disabled search engines will not be used for the search.

4.3 Adapting to New and/or Changing Search Engine's Result Formats

The last problem we addressed with this version is that extraction patterns were hard-coded. A dynamic pattern extractor was built to interpret the result page formats for each search engine. This parser identifies the repeated items on the results page and creates the regular expression that extracts the individual elements within each item (e.g., url, weight, title, summary). The patterns used are extremely general and we can parse the results pages of all our underlying search engines with just two basic patterns. Since the results page patterns are stored in a separate knowledge file, all search agents are implemented using identical code. Thus, search agents can be more easily maintained and new search agents added by updating/writing the knowledge file rather than writing new code.

	Alta Vista	Excite	Infoseek	Lycos	Opentext	Webcrawler
medical-biotech.	1.000	0.515	0.967	0.456	0.602	0.817
computer science	0.909	0.837	1.000	0.162	0.157	0.341

Table 5-1 Confidence Factors on Two Categories

5. Experiments and Results

In order to examine the effectiveness of the adaptive multi-agent system implemented for this thesis, several experiments were run. In particular, we aimed at testing the efficacy of the three adaptation algorithms described in the previous section.

5.1 Evaluating Calibration of Search Engine's Performance

We ran a two-week long uncontrolled experiment designed to test the efficacy of the adaptation algorithm on calibrating search engine performance. Basically, we focused on whether the resulting confidence factors for search engines converge and how fast and how well they converge. In other words, the confidence factors associated with each category for the different search engines should vary less and less, converging on a relatively constant value as more training occurs. Initially, all the confidence factors in the six underlying search engines' knowledge tables were set to be 0.500. After 13 days, the confidence factors associated with 13 categories in the six knowledge tables converged to a fairly stable set of values.

During the experiment, a total of 1165, 1011, and 72 queries were classified into category "music", "recreation-and-entertainment", and "business-and-finance", respectively. It took each of the above categories 48, 79, and 54 classified queries respectively for its associated confidence factors to start to converge. Therefore, we can see that the actual convergence speed for each category is about the same since it took similar numbers of queries for each category's associated confidence factor to converge. Furthermore, Table 5-1 shows typical confidence values for two of the thirteen categories. That is, the confidence factors for three search engines converge to the high end which often range between 0.8 and 1.0, and the confidence factors for the other three search engines converge to the low end, generally between 0.0 and 0.5. However, there are variations in which search engines are at the top and bottom for different categories. For example, for category category "medical-and-biotechnology", the top three search engines are Alta Vista, Infoseek, and WebCrawler, but for "computer science", Infoseek, Alta Vista, and Excite perform the best.

The knowledge tables generated by the experiment were compared with the knowledge tables which were previously built manually by evaluating search results on 39 queries per search engine (three queries per category). Table 5-2 show that the confidence factors obtained by adaptation generally agree with the ones obtained manually. Aside from minor numeric differences, the only major rank-order difference is that Excite performs the best overall in the manually built version while Infoseek takes the lead and Excite drops to the 4th position in adaptation version. This is probably due to the fact that the hand-built knowledge tables resulted from an information retrieval class experiment, and the students who were involved in the experiment used longer queries than most normal ProFusion users do. Excite happens toperform better on long queries because it is a concept-based search engine. Also, when we hand made the knowledge tables, we considered top-10 precision (the number of relevant references in the retrieved set of 10), but our adaptation version is more heavily influenced by the top ranked documents. Nevertheless, we can conclude that the knowledge tables built by adaptation correctly reflect the underlying search engines' performance on different query categories. These knowledge tables are able to change over time if search engines change their relative quality

5.2 Evaluating Tracking of Search Engine's Response Time

We tested the efficacy of the adaptation algorithm on tracking search engine's response times for one week. This study focused on how ProFusion system reacts when the underlying search engine's response time and responding status changes. For each query submission, the underlying search engine's response time for the query and ProFusion's current time-out value for the search engine were kept in a log file, and their averages were calculated thereafter on a weekday and weekend basis. Typical results are shown in Figure 5-1, which clearly demonstrates that the agent version of ProFusion successfully tracks the changing response time during the day. In addition, we were also able to test the effectiveness of the automatic enable and disable search engines feature. During the experiment, ProFusion's guarding agent detected that Opentext was down at 10:05:33 am on May 29, 1997. It then immediately disabled Opentext and didn't enable it until it was back on-line at 9:31:36 am on May 30, 1997. During the time when Opentext was disabled, the guarding agent,

	No. 1	**No. 2**	**No. 3**	**No. 4**	**No. 5**	**No. 6**
hand-built	Excite	Infoseek	Alta Vista	Webcrawler	Opentext	Lycos
adaptation	Infoseek	Alta Vista	Webcrawler	Excite	Lycos	Opentext

Table 5-2 Search Engines' Average Performance Rank Order

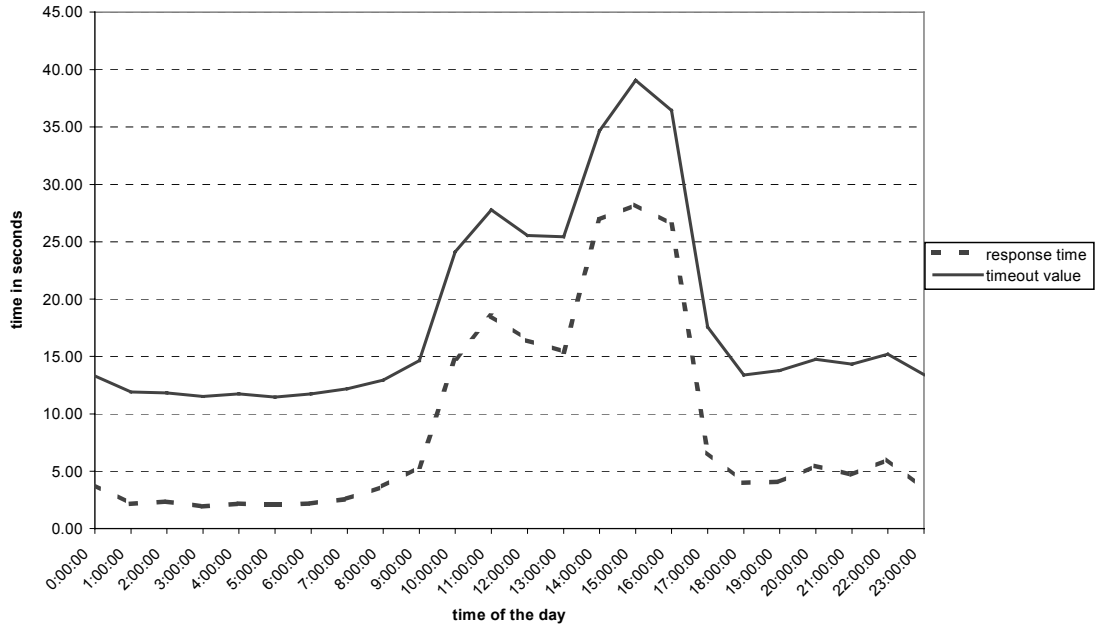

Figure 5-1 Adapting to Alta Vista's Weekday Response Times

periodically ran queries on the actual Opentext home page, but got the same "No server available for the search try again later." message until the morning of May 30th, at which time it re-enabled Opentext. Obviously, this indicates that our system correctly detected and adapted to the search engines' status.

5.3 Evaluating Adaptation to Changing Search Engine Result Formats

We did not pay attention to whether there have actually been any changes in the underlying search engines during the first three months after deploying the agent-based system. However, in this time we have not had to update a search engine result pattern (whereas, with the previous system, one pattern required adjustment every few weeks). However, we did test the parsers to extract different page formats. For example, Infoseek displays a result item as the following:

> Agency in Real Estate
> Based in Winnipeg, Manitoba, Canada.
> 43%
> http://www.pangea.ca/~dtowes/agency.html (Size 5.6K)

To test our parser, we changed the format to:

> Agency in Real Estate, (score: 0.43)
> Based in Winnipeg, Manitoba, Canada.
> http://www.pangea.ca/~dtowes/agency.html
> (Size 5.6K)

and:

> • Agency in Real Estate
> Based in Winnipeg, Manitoba, Canada. [43%]
> http://www.pangea.ca/~dtowes/agency.html
> (Size 5.6K)

In these cases (and others not shown), our parser succeeded in extracting the item. More major changes (e.g., result items with no numeric match score) required manual assistance to update the grammar.

6. Conclusions and Future Work

An adaptive multi-agent architecture for the ProFusion meta-search engine has been implemented and is in daily operation. The original ProFusion was re-designed into a multi-agent system which is easier to extend, maintain, and distribute. In addition, automatic adaptation algorithms were included to the original ProFusion to replace the hard-coded priori knowledge base. With this adaptive multi-agent architecture, the ProFusion system is now more competitive in the dynamic Web environment since it automatically adjusts to changes in its environment. On the other hand, the adaptive agent version of ProFusion we have implemented is still a basic model. Improvements which are left for future work are still needed. These include targeting ProFusion to search domain specific sites rather than general purpose search

engines (see http://sports.profusion.com) for fusion from sports-related sites and http://waldo.rtec.org for fusion from education related sites) and more intelligent post-processing of search queries and results (query expansion and clustering)

References

[Arens 96] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen, "Query Reformulation for Dynamic Information Integration," Journal of Intelligent Information Systems, 6 (2/3), pp. 99-130.

[Bayardo 97] Roberto Bayardo, et al, "InfoSleuth: agent-based semantic integration of information in open and dynamic environments," in Proc. of ACM SIGMOD, May 1997, Tucson, Arizona, pp. 195-206.

[Casasola 97] Edgar Casasola and Susan Gauch, "Intelligent Information Agents for the World Wide Web," Information and Telecommunication Technology Center, Technical Report ITTC-FY97-TR-11100-1, 1997.

[Chen 98] Liren Chen and Katia Sycara, "WebMate: a Personal Agent for WWW Browsing and Searching," Autonomous Agents '98.

[Dreilinger 96] Daniel Dreilinger, "Experiences with Selecting Search Engines using Meta-Search," December, 1996.

[Drescher 91] Gary Drescher, "Made Up Minds: A constructivist Approach to Artificial Intelligence," MIT Press, 1991.

[Gauch 96a] Susan Gauch, Guijun Wang, "Information Fusion with ProFusion," WebNet '96: The First World Conference of the Web Society, San Francisco, CA, October 1996.

[Gauch 96b] Susan Gauch, Guijun Wang, Mario Gomez, "ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines," Journal of Universal Computing, Springer-Verlog, Vol. 2 (9), September 1996.

[Haverkamp 98] Donna Haverkamp and Susan Gauch, "Intelligent Information Agents: Review and Challenges for Distributed Information Sources," Journal of the American Society for Information Science, 49 (4), April 1998, pp. 304-311.

[Holland 86] John Holland, "Escaping Brittleness: the Possibilities of General-Purpose Learning Algorithms applied to Parallel Rule-Based Systems," in Machine Learning, an Artificial Intelligence Approach, Volume II, edited by R.S. Michalski, J.G. Carbonell and T.M. Mitchell, Morgan Kaufmann, 1986.

[Jennings 98] Nicholas R. Jennings and Michael J. Wooldridge (Ed.), Agent Technology: Foundations, Applications, and Markets, Springer-Verlag, 1998.

[Konstan 976] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, John Riedl, "GroupLens: applying collaborative filtering to Usenet news," Communications of the ACM, 40 (3), March 1997, pp. 77-87.

[Lesser 98] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang, "BIG: A Resource-Bounded Information Gathering Agent," To appear in the Proc. of the 15th National Conf. on Artificial Intelligence (AAAI-98).

[Lieberman 95] Henry Lieberman, "Letizia: An Agent That Assists Web Browsing," International Joint Conference on Artificial Intelligence, Montreal, August 1995.

[Maes 92] Pattie Maes, "Modeling Adaptive Autonomous Agents," Artificial Life Journal, edited by C. Langton, Vol. 1, No. 1 & 2, pp. 135-162, MIT Press, 1994.

[Maes 94] Pattie Maes, "Learning Behavior Networks from Experience," In: Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life, edited by F.J. Varela & P. Bourgine, MIT Press/Bradford Books, 1992.

[Moukas 96] Alexandros Moukas, "Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem," Proceedings of the Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology. London, UK, 1996.

[Selberg 95] Erik Selberg, Oren Etzioni, "Multi-Service Search and Comparison Using the MetaCrawler," WWW4 conference, December 1995.

[Starr 96] Brian Starr, Mark Ackerman, Michael Pazzani, "Do-I-Care: A Collaborative Web Agent," Proc. of ACM CHI'96 Conference, Vancouver, BC, April, 1996.

[Sutton 91] Rich Sutton, "Reinforcement Learning Architectures for Animats," in From Animals to Animats, Proceedings of the First International Conference on Simulation of Adaptive Behavior, edited by J.-A. Meyer & S. W. Wilson, MIT Press/Bradford Books, 1991.

[Sutton 90] Rich Sutton, "Integrated Architectures for Learning, Planning and Reacting based on Approximating Dynamic Programming," in Proceedings of the Seventh International Conference in Machine Learning, Austin, Texas, June 1990.