Information and Telecommunication Technology Center, Technical Report: ITTC-FY97-11100-1.

Intelligent Information Agents for the World Wide Web

EDGAR CASASOLA

SUSAN GAUCH

{casasola, sgauch}@eecs.ukans.edu
Electrical Engineering and Computer Science
University of Kansas

ABSTRACT

This paper provides a snapshot of the current status of Intelligent Information Agents for the World Wide Web (WWW). This area is still in its infancy, but, like the Web itself, is undergoing rapid change. Software agents are being employed in a variety of ways on the Web and elsewhere. However, we focus here on agents that accomplish tasks such as searching, browsing and filtering information available on the WWW. In addition, we discuss KQML and CORBA, enabling technologies that provide a framework for developing future software agents.

INTRODUCTION

The World Wide Web (WWW) contains a complete universe of on-line information. Unfortunately, it has become almost impossible for a casual user to look for specific information without getting lost among huge amounts of mixed data. Some problems are inherent in any information retrieval system. In particular, retrieval failures arise because of the the ambiguity of natural language. Queries are at best imperfect representations of the user's information needs and words chosen by the author are imperfect representations of the information contained in the document. It should come as no surprise that matching query words to document words, which is the heart of any information retrieval system, yields a very imperfect result. However, the distributed nature of the World Wide Web adds new problems to old: documents may be duplicated many times at many different sites; Web pages are added at alarming rates, creating an extremely dynamic information environment; the quality of information contained in Web pages varies greatly; Web pages are deleted or moved frequently, leaving behind dangling references. To navigate the turbulent sea of information that is the World Wide Web, Intelligent Agents are being developed which can act on behalf of their users. These personal electronic assistants can (or soon will) support tasks such as browsing, searching, collecting, comparing and filtering WWW information.

This paper discusses intelligent agents for information gathering from the WWW. We analyze their utility from a user's perspective, as well as their development techniques, learning capabilities, and application domain. We provide a snapshot of the current state of the art in a fast-moving field of research and development. The first section reviews browsing assistants, agents that "look over the shoulder" of the user while he is browsing the Web. Section 2 deals with search systems that exhibit an above average degree of agency or incorporate learning capabilities. Section 3 covers personal search agents that locate, extract and compare information in a non-supervised way. Filtering agents designed to extract and distribute relevant documents from dynamic information sources are mentioned in Section 4. Finally, in Section 5, we discuss *KQML* and *CORBA*, two new technologies that can play an important role in the development of future distributed collaborative agent architectures.

1. BROWSING ASSISTANTS

Browsing assistants for the *WWW* are essentially interface agents that "look over the user's shoulder". While the user operates any conventional Web browser like *Mosaic* or *Netscape*, the agent monitors user activity and recommends some hyperlinks for the user to follow. Their goal is to help the user locate the new information of interest to them. Generally, browsing agents supply suggestions in response to a user request. The agent is a continuously running process, 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. Thus, the agents for browsing are individual assistants, they interact with a single user, and "learn" what this user's interest are. Two examples of this kind of system are *WebWatcher* [Thorsten, Freitag and Mitchell 96] and *Letizia* [Lieberman 95].

WebWatcher requests an initial goal from the user, and the e-mail address to keep track of the user's interests. *WebWatcher* enhances the basic Web browser page with: a menu bar above the page to communicate with the agent, a list of new hyperlinks found to contain the words in the goal, hyperlink recommendations and highlighted hyperlinks. The original prototype was implemented for *Mosaic* users. The actual learning of the system was acquired by logging a user's successful and unsuccessful searches as training data. It suggests an appropriate hyperlink based on the current web page viewed by the user and the user's information goal .

WebWatcher, evaluates the hyperlinks using a utility function based on the Page, the Goal, the User and the Link. The function assigns a value from 0 to 1 to the links, suggesting the best ones. The developers evaluated different learning methods for selecting the recommended links, particularly *tf*idf* with cosine similarity, statistical prediction, and Boolean functions (See [Armstrong, Freitag, Joachims and Mitchell 95] for details and results). The constant on the system was the encoding representation for any given Page, Link and Goal for a specific user, this information was stored as an array of: 200 words taken from the underlined words in a document, 200 sentences words, 100 words from the heading, and up to 30 user goal words.

Letizia is a behavior-based interface agent which doesn't require the user to provide an explicit initial goal. Rather, it attempts to infer the goal from the user's actions. It tracks user behavior and attempts to anticipate items of interest by doing concurrent, autonomous exploration of links from the user's current position. *Letizia* simply suggests a list of hyperlinks ordered by preference, and can give the user a reason for the recommendation upon request.

Letizia doesn't require the user to evaluate the previous searches as successful or unsuccessful, but instead applies heuristics, learning the user's interest through the user's behavior. For example, when a user saves a "bookmark" for a document or follows a link, it assumes that the subject of those pages is of interest. Similarly, when the user skips a link, it assumes disinterest. The subjects are stored as lists of keywords. Using this representation of user interest, it performs a best first search, following links and evaluating against the subjects of interest, eliminating dead end links. The user's previous interests are stored and persist while the user browses over time, and they decay by a factor over time. It also detects the user's search strategy; for example: searching by person or by subject while browsing.

Future research points to the creation of browsing assistants, following some of the ideas espoused by [Lashkari, Metral and Maes 94] on their Collaborative Interface Agents paper. Collaboration between the agents of different users can be used to speed up the learning time, since it requires some learning time for the agent to be useful. Also, collaboration can increase user confidence in the suggestions provided by the agent. The difficulty with this approach is the ability to tell when it is appropriate to generalize interest from one user to others. Nevertheless, a mechanism for communication between different user agents can allow a new agent to get suggestions from other "more experienced ones" based on the actual page, links and goal while it builds its own knowledge and adapts to it user's preferences.

2. SEARCH ASSISTANTS

Search engines are evolving towards more sophisticated systems for locating desired information in the WWW. Some of these systems are incorporating new features and evolving towards more agent-like designs. For example some systems record information about their users and try to manage long term goals for those users, periodically communicating with them to report changes to specified information resources [Gauch, Wang and Gomez 96]. They keep track of the contents of specific information resources, and upon a user's request they can decide a strategy to execute different autonomous and concurrent tasks to solve each of their goals [Dreilinger 96]. Others accept goal specification in ways that facilitate or eliminate from the user the compromise of finding a formal query that completely represent his/her information needs [Krulwich 96]. Also, some systems are incorporating sub-agents as part of their architectures to execute tasks as searching and filtering [Moukas 96].

This section mentions some of the systems like the ones mentioned above. It is important to note that some of these systems completely satisfy the requirements specified for software agents (see [Franklin and Graesser 96]), but their degree of agency is high, and some are evolving towards agent architectures. For example, these systems act on a user's behalf, following autonomous tasks to achieve a goal (sometimes following their own agenda), they are continuously running programs, they act on a restricted environment (the WWW), they sense the environment and their behavior is sensitive to changes as environmental characteristics change.

2.1. Searching by Examples

Generally, search engines require users to enter a query which indicates their information needs. However, the average query submitted to a search engine is just two words long [Croft, Cook and Wilder 95], which makes it difficult for the search engine to identify relevant documents. There are likely to be many relevant documents available which are missed because they do not contain the exact words used in the query. One system that avoids this problem is *InfoFinder* [Krulwich 96].

It asks the users to supply some sample documents, then searches for related documents , returning the URL's of the identified papers by electronic mail. The sample documents provide the agent with information about the frequency of words inside the desired papers, and also patterns or common phrases that can be important to do a better ranking of the final documents. This technique, a variation on relevance feedback, has been shown to dramatically improve search results from single databases [Harman 92].

2.2. Adaptive Systems

Adaptive agents not only search for information, but also continuously change after every search to improve the quality of their results. For example, the most important words in the preferred documents can be used to search for related documents, then those documents can be filtered using words important only in the disliked documents. If there are many resulting papers, the search can be narrowed down by providing a higher rating to the papers with similar patterns or phrases found only in the desired sample papers. Such an agent can behave like the regular assistant who receives some documents from his/her boss and gets the request "Please! Find more information about this!". One prototype that shows the advantages and disadvantages of such systems is *Amalthaea* [Moukas 96], which applies evolutionary systems to information retrieval, providing personalized discovery and filtering of information from the WWW.

Amalthaea is a multi-agent evolutionary system based on genetic algorithms applied to information retrieval [Moukas 96]. Specifically, the agents perform tasks such as: learning the user interests, data discovery of such information, information filtering, and monitoring of periodical (or random) content changes of some specific sources. The system creates two distinct kinds of agents which compete and co-operate: information filtering agents and information discovery agents. A genetic algorithm is used to let the information filtering agents "adapt" to the user's interests, and filter the information they are expecting. The information discovery agents are maintained using a similar technique where the information filtering agents give credit to the ones that are useful and eliminate the ones that are not responding to the user needs. Also, new agents are generated crossing over the characteristics of the existing ones. *Amalthaea* applies information retrieval techniques to rank the resulting documents. The results are then used as feedback to evaluate the agent's effectiveness.

The system was tested using a fixed collection of documents representing the different WWW search engines, and using fixed user interests to check the learning capabilities of the agents. Similar to systems based on relevance feedback, the performance of the agents dramatically improved over time. The main drawback of systems like *Amalthaea* is the need for the user to evaluate the retrieved

documents. Their examples show that it can take about 350 generations, with the user evaluating documents each generation, to train the agents and get a high learning level. In addition, the system is not applicable to large numbers of users since many agents are created for each user. However, they are investigating new approaches to avoid the reliance on user feedback after every generation and developing of shared discovery agents.

2.3. Meta Search Engines

Because of the explosive growth of information in the WWW, many search engines have been developed. However, no single search engine indexes all the available information in the WWW, they each index only different overlapped subsets of documents. "Recent studies indicate the rate of Web expansion and change makes a complete index virtually impossible" [Selberg and Etzioni 95] . Even if complete indexes were possible, the search engines vary in their search algorithms and each would return different results. The problem faced by the user now is how to choose the best search engine for their needs. Learning each interface and repeating the same query with every one can be very time consuming.

Meta-search engines deal with the above problem by providing a single user interface to multiple underlying search engines. They accept a user's query and execute a parallel search, sending the query to different engines and collecting and merging and filtering the results. The main difference among meta-search engines is their techniques for selecting the search engines for a particular query and the post-processing applied to the results. Three examples of meta-search engines are *MetaCrawler* [Selberg and Etzioni 95], *ProFusion* [Gauch and Wang 96], and *SavvySearch*[Dreilinger 96].

MetaCrawler [Selberg and Etzioni 95] accesses six services : *Galaxy, InfoSeek, Lycos, Open Text, WebCrawler, and Yahoo.* It submits the query to all six engines and merges the results. There are two optional post-processing capabilities: 1) the specification of required and non-desired words; and 2) the "expert options", where the user can adjust the ranking of the results by physical location (country or Internet domain). When the first post-processing option is selected, the system fetches the identified documents (not just their URLs) and analyses them to verify the user query and preferences.

ProFusion [Gauch, Wang and Gomez 96] accesses *Alta Vista, Excite, InfoSeek, Lycos, Open Text*. One of the unique characteristics of *ProFusion* is the ability to analyze queries and identify their subjects. Then, based on that analysis, it selects the most appropriate search engines for the query. The actual implementation handles 13 categories, and for every category there is an associated confidence factor associated with every search engine. After executing the parallel search, the system receives

the results from the invoked engines, and eliminates the ones taking more than 60 seconds. Then it ranks and filters out the results, using the ranking value returned by the different engines, and the confidence factor associated with every engine.

SavvySearch [Dreilinger 96] provides a gateway to a large collection of World Wide Web resources (over 30) which vary from general purpose search engines to sites indexing more specific data (e.g., people, software, thesauri). SavvySearch will create a multi-step search plan which selects the most appropriate search engines based on the query and Internet usage. It will optionally partially merge the results, but documents are primarily grouped together by search engine.

2.4. Searching Structured Resources Available On-Line.

Most of the search agents for the Web deal with unstructured information incorporated in HTML documents. However, a few agents have been created to exploit structured resources. These systems act as a gateway between the users and the basic resources, however they require meta-knowledge about the resources they with which interact. One of the best known existing agents capable of handling structured resources is *Softbot* from the University of Washington [Etzioni and Weld 95]. As its name indicates, *Softbot* attempts to be a high-level assistant that shields the user from the Internet. It receives user queries or commands (high-level goals) and defines the sequence of Internet commands to satisfy that goal. It not only accepts information requests but also manipulates objects, monitors events (even events not related with information such as disk utilization, user activity etc.), and enforces constraints (such as types of files and format inside specific directories).

Softbot consists of a task manager, a planner, a model manager and Internet domain models. The Internet domain models contain background information about the Internet (commands, information services, and databases). It incorporates extensive, hand-coded control rules to handle specific tasks. For example, it can combine the *finger* and *ftp* commands to identify a given human name, and transfer some files from his/her home directory. *Softbot* is being tested at the University of Washington. Work is ongoing to provide the agent with extra capabilities such as: learning, incomplete information planning, multi-agent communication techniques, cyber-perception (capability of extracting facts from unstructured and semi-structured documents), and safety. However, it is an example of a new kind of agent, one capable of performing more complex tasks.

3. MATCHMAKING AND COMPARISON AGENTS

Sometimes the information you need is not explicitly available on-line, but rather implicitly available only through data mining. In addition, the information

available changes daily, requiring users to monitor specific resources such as bulletin boards or track subject related sites, summarize and create associations among the contents of the resources found to keep up to date. An agent to monitor specific sites or sources, to collect data, and to use heuristics to imply relations among the data keeping a knowledge base, would increase the utility of an information resource. For example, such a system could offer answers to queries requesting "the cheapest known product", "an expert in some subject", and some questions requiring information collection, interpretation and association.

The basic hypothesis behind the creation of comparison agents is that "although the Web is less structured than we might hope, it is less random than we might fear" [Etzioni 96], so the creation of agents to discover resources and extract information from them is a feasible task. Two agents that will show some of the early research in this field are *ContactFinder* [Krulwich and Burkey 96] and *Shopbot* [Doorenbos, Etzioni and Weld 96].

3.1. Matching Query Subjects Against Human Experts in the Area.

Sometimes the best way to solve a problem is to ask an expert in the area. For specific closed domains, like an enterprise, the electronic mail bulletin boards carry messages from people asking for help relating to some topic, and other messages from experts answering the questions. In such an environment, an agent capable of recommending the name of somebody known, or believed, to have the knowledge to answer specific questions can be of great help. A prototype dealing with this is *ContactFinder* [Krulwich and Burkey 96].

ContactFinder is a research prototype developed to review messages and collect information about experts in some specific areas. The system "remembers" the contents of the messages and the names of the users who deal with specific subjects. It doesn't respond by returning documents or specific resources, but supplies referrals to experts in specific areas. The system has been running on the local developers bulletin boards and "after reading about 5000 messages, it was able to give referrals for 10.1% of the questions, and it has been 86% accurate in its referrals." [Krulwich and Burkey 96]

3.2. Shopping Agents

Two main tasks are involved in the development of an agent capable of comparing information about prices of products: 1) locating the different providers; and 2) extracting and comparing information about specific products from them. Systems like *Shopbot* [Doorenbos, Etzioni and Weld 96] explore the possible applications of agents to Web data mining for commercial purposes.

Shopbot operates in two phases: the first is the off-line learning algorithm which creates vendor descriptions; and the second is the on-line user assistant that helps to shop in real time [Doorenbos, Etzioni and Weld 96]. *Shopbot* uses pattern matching to take advantage of regularities in the merchants' home pages. *Shopbot* creates, in an unsupervised way, vendor descriptions with information about how to interact to specific product merchants, the URL of a page containing a form for a search index, plus a function mapping normal product attributes to fields in that form, and parsing functions to extract data from the returned pages. The system uses a learning mechanism based on fixed queries for dummy products and popular products. The dummy products will retrieve a "Product Not Found" page which is used to learn the no answer response from a new vendor. The popular product queries will be used to identify which kinds of products the new vendor sells and also what the response looks like so that the system can learn to extract desired information (price, product characteristics, etc.).

Preliminary studies showed that "users of *Shopbot* were able to shop four times faster than users relying only on a Web browser" [Etzioni 96] . This is encouraging for future research for the use of agents in this field.

4. FILTERING AGENTS

The search agents mentioned earlier deal primarily with static databases and dynamic queries. Filtering agents work with static queries used to filter information with high quantities of dynamic data. For example, filtering agents can be able to manage incoming information, and can classify, rank, or select them according to some user's requests and preferences. The information filtered may be in the form of HTML documents, newsgroup articles or electronic mail messages.

Filtering agents can filter an incoming list of HTML page references and select or rank them according to a specific subject. *ProFusion* [Gauch, Wang and Gomez 96] is a meta-search engines that can also behave as filtering agents. Users can register a query and receive automatic updates about new URLs retrieved in response to a registered query. Other agents filter all new articles in newsgroups to which the user subscribes. These can be integrated into news reading programs or used to send the filtered articles to the user through e-mail.

SIFT (Stanford Information Filtering Tool) receives user profiles consisting of keyword (which may be weighted or joined together by Boolean operators) through a WWW browser [Yan and Garcia-Molina 95]. Then, this system compares the current newsfeed received at Stanford against all user profiles, and the matching articles are sent to the user via e-mail. The user of *SIFT* can specify the maximum length of the messages, as well as the period of time between reception of messages.

Pefna [Kilander 96] is a system which filters newsgroup articles using representative articles for different categories. It was developed as part of the *IntFilter* project at the University of Stockholm. The user of *Pefna* provides a list of ordered categories and a representative article for each category. The articles are rated by assigning a cosine distance value and comparing them against the sample article for every category. Articles below a certain threshold are then eliminated. When the user launches the news reader, The system presents the surviving unread articles as threads under the name of every category, and the categories are also sorted according to the user's preference. *Pefna* can handle categories for individual users as well as global categories for all users.

Some filtering systems provide "virtual news groups" containing the most interesting articles for a particular user. One example is *NewsWeeder* [Lang 95] which learns from the previous user interests. *NewsWeeder* has an HTML interface that lets the users visit different newsgroups and rate the articles from 1 to 5 as they read them. Based on this information, the system applies a machine learning algorithm to rate new articles and build the virtual newsgroup. *NewsWeeder* also collects all the user's ratings to provide a collaborative filtering news group based on the average score assigned to the individual articles.

5. ENABLING TECHNOLOGIES

A major theme for the next generation of intelligent agents is to provide communication and co-operation among agents to allow them to learn from each other and exchange services. (see [Lashkari, Metral and Maes 94], [Franklin and Graesser 96] , [Lang 95], [Brown 95]). However, new underlying technology is needed to provide the mechanism for the agents to contact each other and to exchange information and knowledge. For this reason, this last section discusses the role of new technologies, *KQML* and *CORBA*, and the benefits they bring to the field.

5.1. The Importance of the Knowledge Sharing Effort and KQML

The Knowledge Query and Manipulation Language (*KQML*) is a new language and protocol for exchanging information and knowledge among intelligent agents. It deals with common pragmatics for effective communication as a primary goal (instead of semantics). In other words, it is a language and a set of protocols that support computer programs as they identify, connect with, and exchange information with other programs.

KQML is a result of the work done by the External Interfaces Group of the ARPA Knowledge Sharing Effort *(KSE)* [Finin and Fritzson 94]. The main objective of the Knowledge Sharing Effort is "to develop conventions facilitating sharing and reuse

of knowledge bases and knowledge based systems. Its goal is to build much bigger and more broadly functional systems than could be achieved working alone". Other working groups of the *KSE* are working with the creation of languages for expressing the content of knowledge-bases (*KIF*), definition of common constructs within families of representing languages (*KL-ONE*), and definition of shared and reusable knowledge bases (known as ontologies or structured vocabularies to interpret the content of messages).

KQML is important because it is concerned with the run-time interaction between systems. It allows any information server to be treated as a knowledge base system, allowing integration of DBMS, Hypertext Systems, server oriented software (finger, mail, HTML servers) and so on. Systems based on *KQML* can be viewed as having two knowledge bases: one representing the agent's information store and the other representing its goals. KQML agents provide meta-data information in the form of performatives (i.e., ask, tell, subscribe, broker, recruit, recommend, and advertise) which describe the meta-data in the messages, specifying their information requirements and capabilities.

One agent architecture based on *KQML* [Finin and Fritzson 94] incorporates two specialized programs**,** "a router" and a "facilitator", and an application program interface known as *KRIL* (**K**QML **R**outer **I**nterface **L**ibrary) as the agent interface mechanism. Using KQML performatives, the *facilitator* acts as a bridge between an information store and external clients. It provides high-level services such as: registering and forwarding of service names, routing of messages based on context, matchmaking between clients and servers (or demand agents and providers), and mediating and translating among agents.

The *router* is a content-independent message routing program associated with a specific agent. It deals with all KQML messages going to and coming from its associated router process. It allows the agent to receive asynchronous incoming messages from independent resources and send messages to other agents. If an agent address is not specified, the router looks for a service agent capable of dealing with that message, or to another router willing to route it (they can not guarantee delivery of all messages). To deliver incomplete messages, routers rely on facilitators. Every agent communicates with its router, using KRIL.

5.2. Importance of CORBA

The Common Object Request Broker Architecture (CORBA) is a specification that allows applications to communicate with each other regardless of their location and developer. It is part of the Object Management Architecture (OMA) defined by the Object Management Group (OMG), a non-profit consortium of over six hundred software vendors and members dedicated to promoting the theory and practice of

object technology for the development of distributed computed systems. OMG's goal is to provide a common architectural framework for object-oriented applications based on widely available interface specifications.

The existence of a distributed uniform space wherein agents could interact would produce an enormous impact on the development of intelligent collaborative agents. CORBA 2.0, the current specification, enables client/server object interaction and interoperability by specifying the allowable interactions between Object Request Brokers (ORBs) from different vendors. The location of agents in this environment would be done by dynamic requests to the local Object Request Broker. Agents are encapsulated within object wrappers through which they can communicate with their local request broker. The agents are able to co-operate with all the other agents connected to their local brokers, regardless of their location in the distributed environment. KQML can use CORBA as a transport layer which provides a uniform space where agents can be found [Chan 96].

6. CONCLUSIONS

The development of agents for information retrieval in the World Wide Web is at this moment an active area of research wherein existing systems are evolving towards more sophisticated and useful tools to explore and extract the gold from this mine of information. Tasks such as browsing, searching, filtering, and manipulating information from the Web may soon be delegated to electronic assistants which are able automatically adjust, understand and manipulating the WWW environment, even if it is not the most friendly environment for them. Additionally, the Web can be the ground where agents can demonstrate their potential real utility. The goal is that these agents will not be just pieces of software, but instead real representatives of their users in the electronic world. They will be the workers in this new information universe, and it is our task to create the best possible workers that our imagination can produce.

REFERENCES

[Armstrong, Freitag, Joachims and Mitchell 95] Robert Armstrong; Dayne Freitag, Thorsten Joachims, and Tom Mitchell. "WebWatcher: A Learning Apprentice for the World Wide Web", in Proceeding of the AAAI Spring Symposium on Information Gathering, Stanford, California, March 1995. URL: http://www.cs.cmu.edu/afs/cs/project/theo-6/web-agent/www/project-home.html

[Chan 96] Francis Chan. "Research on OMG/CORBA". EECS Department, Berkeley. February 1996. URL: http://www.ic.eecs.berleley.edu/~fchan/caddis/corba.html

[Croft, Cook and Wilder 95] Croft, W.B., Cook, R. & Wilder, D. (1995). Providing Government Information on the Internet: Experiences with THOMAS. In *Digital Libraries Conference DL '95*, pp. 19-24.[Doorenbos, Etzioni and Weld 96] Robert B. Doorenbos, Oren Etzioni and Daniel S. Weld. "A Scalable Comparison-Shopping Agent for the World Wide Web", Department of Computer Science and Engineering, University of Washington, Seattle, WA. Technical Report UW-CSE-96-01-03 URL: ftp://june.cs.washington.edu/pub/ai/tr96-01-04.ps.gz

[Dreilinger 96] Daniel Dreilinger. "SavvySearch Home Page". URL: http://www.cs.colostate.edu/ ~dreiling/smartform.html

[Etziony 96] Oren Etziony. "The World-Wide Web: Quagmire or Gold Mine". Communications of the ACM. Vol.39, No.11, November 1996 .pp. 65-68 URL: ftp://ftp.cs.washington.etzioni/softbots/cacm96.ps

[Etzioni and Weld 95] Oren Etzioni and Daniel Weld. "Intelligent Agents on the Internet: Fact, Fiction, and Forecast", IEEE EXPERT, vol. 10, no 1, August 1995 pp. 44-49 URL: ftp://ftp.cs.washington.edu/pub/ai/ieee-expert.ps.Z

[Finin and Fritzson 94] Tim Finin and Richard Fritzson. "KQML as an Agent Communication Language", The Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94), ACM Press, November 1994. URL: http://www.cs.umbc.edu/kqml/papers/kqml-acl-html/root2.html

[Franklin and Graesser 96] Stan Franklin and Art Graesser. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag, 1996. URL: http://www.msci.memphis.edu/~franklin/AgentProg.html

[Gauch and Wang 96] Susan Gauch and Guijun Wang. "Information Fusion with ProFusion", WebNet '96: The First World Conference of the Web Society, San Francisco, CA, October 1996. URL: http://www.designlab.ukans.edu/ProFusion.html

[Gauch, Wang and Gomez 96] Susan Gauch, Guijun Wang, and Mario Gomez. "ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines". Journal of Universal Computer Science, Volume 2, Number 9, Sept. 1996. URL: http://www.designlab.ukans.edu/profusion/

[Harman 92] Donna Harman, "Relevance Feedback Revisited", in Proceedings of the Fifteen Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, June 1992, pp.1-10

[Kilander 96] Fredrik Kilander, “IntFilter Home Page”. K2LAB, Department of Computer Sciences, Stockholm University, Sweden. June 19 1996. URL: http://www.dsv.su.se/~fk/if_Doc/IntFilter.html

[Krulwich 96] Bruce Krulwich. “InfoFinder Internet”, Andersen Consulting’s Center for Strategic Technology Research, 1996. URL: http://www.ac.com/cstar/hsil/agents/framedef_if.html

[Krulwich and Burkey 96] Bruce Krulwich and Chad Burkey. “The ContactFinder Agent: Answering Bulletin Boards Questions and Referrals”, in Proceedings of the AAAI 1996 Conference, Portlan Oregon, 1996. URL: http://www.ac.com/cstar/hsil/agents/framedef_art_cf.html

[Lang 95] Ken Lang. “Newsweeder: Learning to filter netnews. Technical report, School of Computer Science, Carnegie Mellon University, 1995. URL: http://anther.learning.sc.cmu.edu/ml95.ps

[Lashkari, Metral and Maes 94] Yezdi Lashkari, Max Metral and Pattie Maes. “Collaborative Interface Agents”, in Proceedings of the AAAI 1994 Conference, Seattle Washington, August 1994. URL: http://agents.www.media.mit.edu/groups/agents/papers/aaai-ymp/aaai.html

[Lieberman 95] Henry Lieberman, “Letizia: An Agent That Assists Web Browsing”. in Proceedings of the 1995 International Joint Conference on Artificial Intelligence, Montreal, Canada, August 1995. URL: http://lieber.www.media.mit.edu/people/~lieber/Lieberary/Letizia/Letizia.html

[Moukas 96] Alexandros Moukas. “Amalthaea: Information Discovery and Filtering using a Multi-agent Evolving Ecosystem”, Proceedings of the Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology. London, UK, 1996. URL: http://moux.www.media.mit.edu/people/moux/papers/PAAM96/

[Selberg and Etzioni 95] Erick Selberg and Oren Etzioni. “Multi-Service Search and Comparison Using the MetaCrawler”, WWW4 Conference, December 1995. URL: http://www.cs.uashington.edu/research/projects/softbots/papers/metacrawler/www4/html/Overview.html

[Thorsten, Freitag and Mitchell 96] Joachims Thorsten, Dayne Freitag, and Tom Mitchell. “WebWatcher: A tour Guide for the World Wide Web”, CMU-CS-96-xxx, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, September 1996.

[Yan and Garcia-Molina 95] Yan Tak and Hector Garcia-Molina. “Sift - A tool for wide-area information dissemination”. Proceedings of the 1995 USENIX Technical Conference, 1995. pp. 177-186. URL: ftp://db.stanford.edu/pub/sift/sift.ps