The Evolution of a Practical Agent-based Recommender System

Samhaa El-Beltagy, David De Roure and Wendy Hall

Intelligence, Agents and Multimedia
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
+44 (0)23 8059 2418
{seb,dder,wh}@ecs.soton.ac.uk

Abstract

This paper presents a multi-agent framework that has been developed to assist Web users within organisations in finding relevant documents and identifying colleagues conducting related work. The system exploits the idea that information acquired and created by one user navigating the information space is something that can be utilised to assist other users. The design of the system builds on the practical experiences of an earlier agent-based recommender system that provided navigation assistance based on information about which documents other users had visited. The system presented here has an improved interface and agent architecture, and it significantly extends the functionality. The interface successfully employs bookmarks as a familiar and effective interaction mechanism. Added services within the architecture include handling of multimedia documents and integration with other open hypermedia systems, and the system is also capable of augmenting document content on delivery. The framework is open and extensible to incorporate new functionality.

Keywords: Software agents, Open Hypermedia, information finding, navigation assistance, collaborative filtering.

1 Introduction

The Web, globally and within intranets, is expanding both in the size of the information space and in the number of users of that space: the task of locating relevant information and navigating that space is proving more and more difficult. As a result, tools that can guide users through this chaotic system to find relevant information are becoming rather crucial. To have a system that can dynamically embrace such tools and utilise them immediately and transparently for the benefit of the user is of equal importance. The work presented in this paper investigates ways of bringing together open hypermedia and an agent-based information management and navigation model to fulfil that goal. The basic idea upon which this work builds is that information gained and created by a user navigating the information space can be used to assist other users in their navigation and information finding activities.

As extensibility and openness are among the goals of the presented work, a multi-agent architecture was employed. In this architecture, each agent performs a relatively simple task, but collectively they present the users of the system with powerful functionality in finding and creating information. The goals of the proposed system can be summarised as follows:

- Making maximum use of what users do normally while browsing/searching the Web. Such activities include viewing and bookmarking various documents
- Allowing users to augment documents or aspects of documents, with useful data for their private use or for sharing with other users. This could be a rating or an annotation on a document, or perhaps the creation of various kinds of hypermedia link.

The design of the system we describe is partly motivated by our practical experience with our first agent-based recommender system, MEMOIR (*Managing Enterprise-scale Multimedia using an Open framework for Information Re-use*), which was trialled in a corporate R&D setting (DeRoure et al., 2000). This background is described in section 2, then in section 3 we present the overview and architecture of the present system. Our current work, which develops the bookmarks theme further, is discussed in section 4. The paper concludes with a discussion of related work in section 5 and future work in section 6.

2 Background

2.1 Microcosm and MEMOIR

The system we present draws from experience gained from building a number of systems, among which are Microcosm (Davis et al., 1992) and MEMOIR (DeRoure et al., 1998; Hill et al., 1997). Microcosm belongs to a class of hypermedia systems that predate the Web and which explore the notion of open hypermedia: rather than embedding links in documents, open hypermedia systems are designed to make use of link specifications stored in databases (here called *linkbases*) and managed separately from documents (Davis et al., 1992; Gronbaek and Trigg, 1994). The concept of abstracting links from documents allowed for a great deal of flexibility since it meant adding hypermedia functionality to documents without changing the document format or embedding mark-up information. One type of link introduced by Microcosm is the *generic link*. In the context of text documents, generic links allow a link to be followed from any occurrence of a text string encountered in a given context, consulting the linkbase to determine destination anchors; this technique for *content-based navigation* generalises to multimedia documents, linking from a variety of features. By employing generic links, a great amount of re-authoring is avoided, and link re-use greatly enhanced.

The other system that influenced the presented system is MEMOIR, a recommender system with an agent based architecture (Pikrakis et al., 1998). One of the main goals of the MEMOIR framework was to support researchers working with vast quantities of distributed information in finding both relevant documents and researchers with related interests. The system can be described as a collaborative recommendation system which utilises *trails* made by users (i.e. sets of documents visited) – this notion of trails goes back to the pioneering work of Bush (Bush, 1945). Using this knowledge, the system can recommend documents to other users based on what they find interesting by attempting to locate relevant documents in trails. More importantly, the system uses knowledge about what its users found interesting in order to provide pointers to users that share similar interests.

The MEMOIR system extended Microcosm's notion of first class links to first class trails, and a distributed object-oriented database was used to store these. To facilitate integration in corporate R&D environments, MEMOIR employed an open architecture based on the existing Web infrastructure and used Web proxies as the key components for systems integration. Agents were written in Java and Lisp, and agent communications were embedded in HTTP. The agent communication language used a hierarchical message format based on that of Microcosm and which in subsequent systems has been realised through XML.

2.2 Practical experience

The MEMOIR design phase involved representatives of users, and ultimately the system was trialled in the industrial setting. This experience of working with users raised many issues, including:

1. A trail, in the simple form of a list of documents visited, contains many documents which are not relevant and would therefore be more effective if it also carried information about the usefulness of particular documents. Non-intrusive techniques such as dwell time were considered, and actions such as printing and emailing.
2. MEMOIR had a user interface and was designed for minimal impact on the user interaction (e.g. some notion of logging in) but user feedback showed that the interface and extra interaction could be an obstacle to take-up of the system.
3. While an attraction of the MEMOIR model was that information could be captured non-intrusively, in practice the users had a strong wish to have control of the way in which trails were published to the system.

So although the MEMOIR model seemed to provide added value from data which can be collected automatically (i.e. something for nothing), in practice it was the user interface that imposed a cost. This led us to investigate alternative interfaces in the present system:

- Providing a comprehensive annotation mechanism as an option. This also enabled us to integrate annotations with the hypermedia system, with a view to maximising the value to users;
- Using bookmarks as the primary interface to the system, because bookmarks are already part of the users' normal interaction;

- Giving users control over what gets published.

The centralised database model of MEMOIR, with largely synchronous communication, was felt to be a constraint, particularly with respect to performance and scalability. However, the agent architecture of MEMOIR proved effective: a variety of agents were introduced to the system throughout the project. The system described in this paper draws on this experience and uses a Java-based framework with KQML messaging (Finin et al., 1994), providing a more consistent framework which supports integration through transducers and wrappers; it uses asynchronous communications and employs an advertise-notify model.

3 System Overview and Architecture

Details of the architecture and its implementation can be found in (El-Beltagy et al., 1999). Agents in the presented system can be viewed as information producers and consumers. For example, a user interface agent acquires knowledge about what different users are viewing and with the permission of the user, shares that information with others. It then uses the services offered by agents that utilise this information, to find out about other users' interests or to obtain a recommendation for a document, among other things. Through the use of ontologies, information re-use is made possible.

3.1 The User Interface Agent

Within the presented framework, each user is associated with a personal interface agent that acts on his/her behalf. This agent has a number of important functions:

- It provides the user with an interface whereby he/she can make use of the services available in the system through transparent communication with agents that are needed to accomplish that service.
- It keeps information entered by users about themselves, such as what they consider their interests, their contact information, Web page, etc.
- It provides other agents within the system with information about the user and his/her browsing, bookmarking, rating and annotating activities which can be used by other agents to achieve a number of tasks
- In cases where a user enters a query, the agent is responsible for integrating the responses returned from various agents and presenting them to the user.
- If a service can change the way in which a normal document is presented, the UI agent is responsible for propagating those changes to the user.
- It provides a number of tools to facilitate browsing and bookmarking.

The first time the agent is activated, it registers with a facilitator agent. It then receives all the advertise messages sent by other agents. Advertise messages that fall outside the interests of the agent are simply ignored. The agent activates and de-activates some of its services according to its interaction with other agents.

The user interface agent has been implemented as a Java desktop application. Figure 1 shows a snapshot of the interface offered to a user. The agent application is made up of many components that are essential for its operation. One of the main components of the application is the proxy server which is used to achieve a variety of useful functions. By configuring the Web browser to use that proxy, the user interface agent can monitor the user's browsing activities and report this to agents that are interested in that information. A user can browse the Web in one of two modes: public or private. In a private browsing mode, no information about what the user is viewing is disclosed to any other agents.

For each visited URL, the proxy server creates a URL object that is used by many components of the system. The URL object, contains the title of a page, the date a page was first visited, the date it was last visited, keywords associated with a URL which are extracted using an extraction algorithm, and a list of image URLs contained in the page. The role of the proxy server, as will be detailed in section 3.5, does not end there.

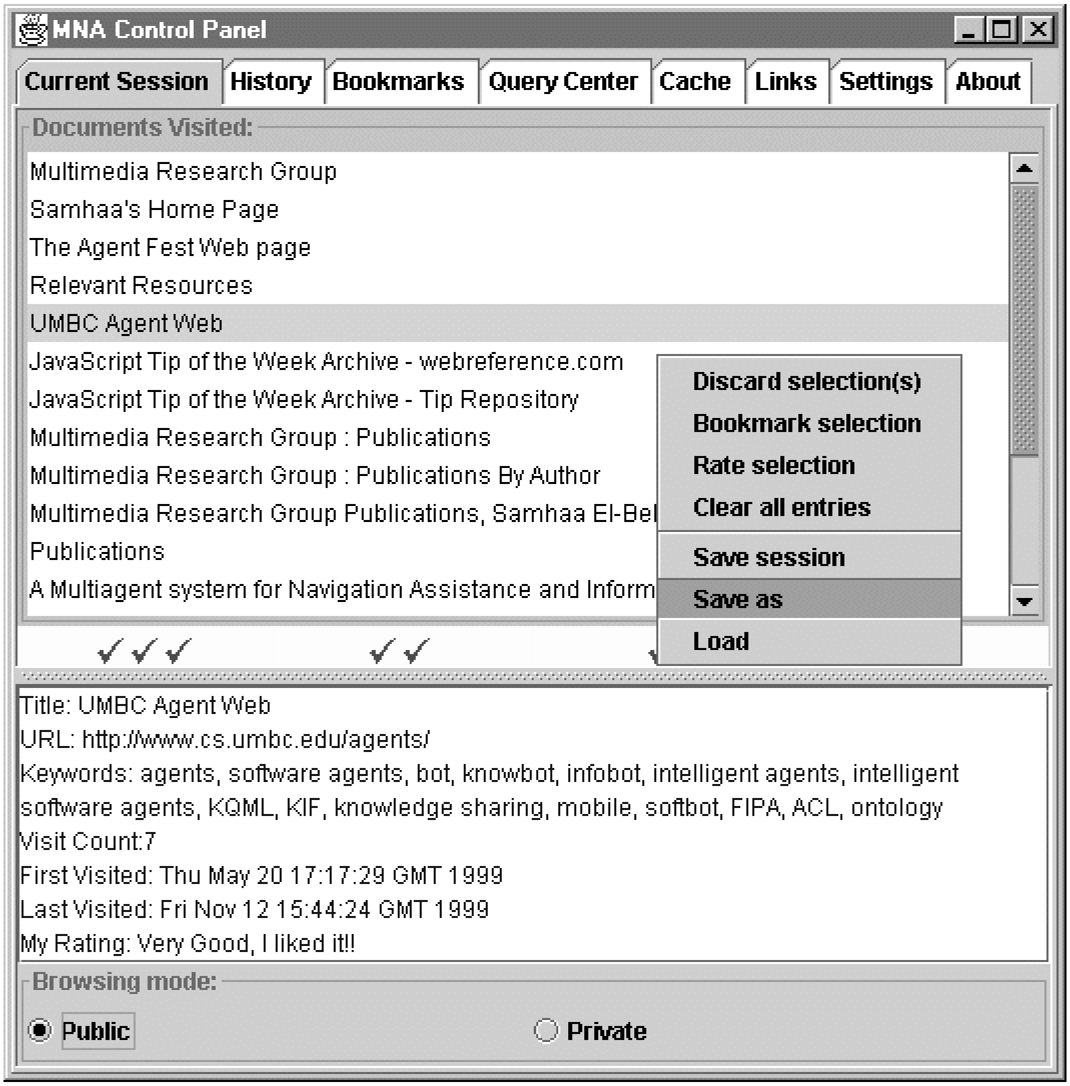

Figure 1: A snapshot of the User Interface Agent

In addition, the agent allows users to save and load sessions where a session is a trail of followed links that the user wants to keep record of rather than bookmark. This allows the user to organize browsing sessions into significant trails they can consult later. The agent also offers users a drag and drop interface for rating documents they have seen. A user interested in rating one or more documents simply selects them and drags them to an icon that represents his/her rating. This information, which is stored locally as well as dispatched to interested agents, is used in a variety of ways. It is known that humans tend to have bad long-term memory, perhaps especially true when it comes to remembering URLs that they have visited and what they thought of them. To prevent users from following links to URLs that they have previously disliked and to point them to links that they have liked, the proxy server uses the locally stored ratings to add symbols beside previously rated links to remind the user of what he/she thought of them. This acts as an incentive for users to rate documents.

An advanced bookmarking facility is also provided. A bookmark is defined in terms of a title, URL, keywords, category, private comment, and a public comment. The bookmark facility allows users to drag and drop one or more URLs from their currently open session into their bookmarks. Duplicate bookmarks are not allowed and the user is warned when any come up. The user can edit the keywords extracted using a keyword extraction algorithm, thus adding keywords that are implicit or removing those that are not really relevant. As in browsing, the user has a public and private option when creating a bookmark. This controls the publishing of bookmarks to interested agents.

This agent also allows users to annotate images and image regions through an annotation tool which gives them the capability of loading images. Images can be loaded directly from a URL, or selected from any of the web pages in a currently open session. The annotations entered by the users are then sent to appropriate agent(s), in this case an image annotation service.

A query interface component whereby the user can access query services provided by the different agents in the system, is among the most important components of the interface agent. A link creation facility is also provided. This facility allows a user to create a generic link and to specify a context for that link. This information is then sent to agents that are interested in it. To enter the user's personal information as well as to allow the user a certain degree of control over the agent and its interactions with the outer world, a setting panel is provided.

3.2 The Organisational Memory agent (OM agent)

This agent serves as an organisational memory for the browsing and bookmarking activities of users within the system. Like all agents within this framework, when this agent comes on line for the first time, it registers with a facilitator and then starts advertising its services. The agent allows other agents to register users and then to publish information about their activities. This particular agent is interested in the browsing, bookmarking, rating and commenting activities. This information is used in a number of ways. By knowing who has seen what, the agent is capable of answering very simple queries such as “Who else has seen the following URL?”. The agent also utilises this information for the purpose of answering more complex queries such as “Recommend URLs related to this document”. Basically the agent can respond to the following queries:

- Who has bookmarked the following URL?
- Find all documents that best satisfy the following keywords
- What is the average group rating for the following URL?
- What have others said about this URL?

The algorithms used to inference the answers to these queries are similar, though not identical to those employed by MEMOIR (Pikrakis et al., 1998). To recommend URLs similar to a given document, all URLs in categories for all users in which this URL appears, are fetched. Ordering of the returned results is based on the duplication of each URL in other categories. None of the URLs are discarded. In case emphasis is put on some keywords, they serve as additional constraints in the inferencing process. To answer who is interested in a given topic, two steps are followed. The first is a straightforward approach where the user interests provided in registering a user, are searched for the topic. It is assumed that the user would have probably supplied this information. The second step involves the consideration of the topic as a keyword and searching the bookmarks for occurrences where this keyword appears more than a given number of times. The results are all then ranked according to maximum occurrence. Finding documents based on keywords is achieved by simply searching through the bookmarks for the URLs that satisfy most of the given keywords.

The knowledge base of this agent has been implemented in Prolog. Prolog has been chosen because it easily supports the required functionality and inferencing capability. The agent itself advertises its ability to be informed about those activities in which it is interested. In KQML terms, it advertises that other agents could insert facts into its virtual knowledge base. In reality, other agents can assert those facts into its real knowledge base.

This agent also performs document modification functions explained in more detail in section 3.5

3.3 The Link Service Agent

The idea of this agent is inspired by the Distributed Link Service (DLS) System (Carr et al., 1995; Carr et al., 1998a) that was in turn inspired by ideas presented in Microcosm (Davis et al., 1992). DLS allows links, mostly generic, from linkbases to be applied to WWW documents. So, the most powerful aspect of the DLS is its ability to insert links into document on the fly, thus bringing the open hypermedia philosophy to the Web. Since the creation of the DLS, there have been a number of other systems that attempted to achieve that very same goal (Gronbaek et al., 1999).

The Link Service Agent is an agent that collaborates with the UI agent to provide services that are very much like the DLS. The UI agent allows different users to create links and to publish them to this agent. Bookmarks are also used to create links. When the agent receives a request for links related to a given word or a group of words, it searches through its database of created links for appropriate matches. The UI agent uses the services of this agent to add links on the fly to documents its user is viewing. Details of how this is accomplished are given in section 3.5.

3.4 The Image Annotator Service

Following the premise that users can benefit from each other's knowledge, an agent has been implemented to capture the user's annotations of image regions assuming that they are provided with a tool that will allow them to create such annotations irrespective of the underlying implementation of such a tool. The agent accepts these annotations in the form of metadata and utilises them in answering queries. Metadata for an image is defined on two different levels: the image metadata level, and the content metadata level. On the first level, descriptors that are usually defined only once for the entire image are specified. These include the image URL, the image and the image creator. In addition, a simple classification for the images could be defined on that level. Identified categories include photographs, paintings, maps and diagrams. Rather than restricting the classification of images under any of these categories, the system allows the dynamic creation of new categories. The second level is defined in terms of a title, keywords and a comment/description. Figure 2 shows a diagram of how this agent is used within the described framework.

The inspiration for this agent lies in Microcosm (Davis et al., 1992) where the idea of linking from parts of media elements to text fragments or other media elements and vice versa is one which was well supported. Emerging Web standards such as SMIL (W3Consortium, 1998) have also recognised the need for this support. However since the current support for this still does not exist directly, a number of tools for locating and viewing parts/segments of images were implemented.

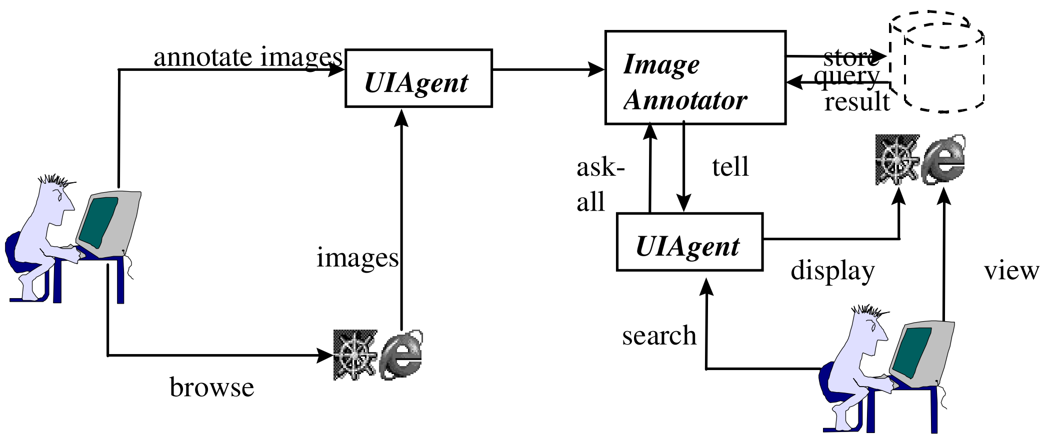

Figure 2: The image annotator as used within the framework

3.5 Agent Interactions

In the previous sections, each of the agents that currently exist within the framework was presented. The interaction between those agents however, was only vaguely outlined. Interaction between the agents exists on two levels. The first level involves sharing of information. The second level involves exchange of services. In this particular framework, two types of services prevail: query answering and document alteration services.

User queries prompt the UI agent to contact appropriate agents. A user query could be very generic entered using keywords in which case agents or wrappers around search resources are consulted. The image annotator and the OM agents both support this mode of querying. In order to demonstrate the flexibility of the framework in dynamically incorporating agents, two wrappers around existing systems were developed to also support this type of querying. The first of these is a wrapper around the Electronics and Computer Science staff and students database at University of Southampton. The second is a wrapper around the Altavista Search engine (Altavista, 1998).

A user query can also be made in relation to his/her current context defined by a given document. "Recommend related documents" is an example of such a query. Requests for comments or average ratings on a given document are yet another example. Figure 3 shows the query interface provided by the UI Agent.

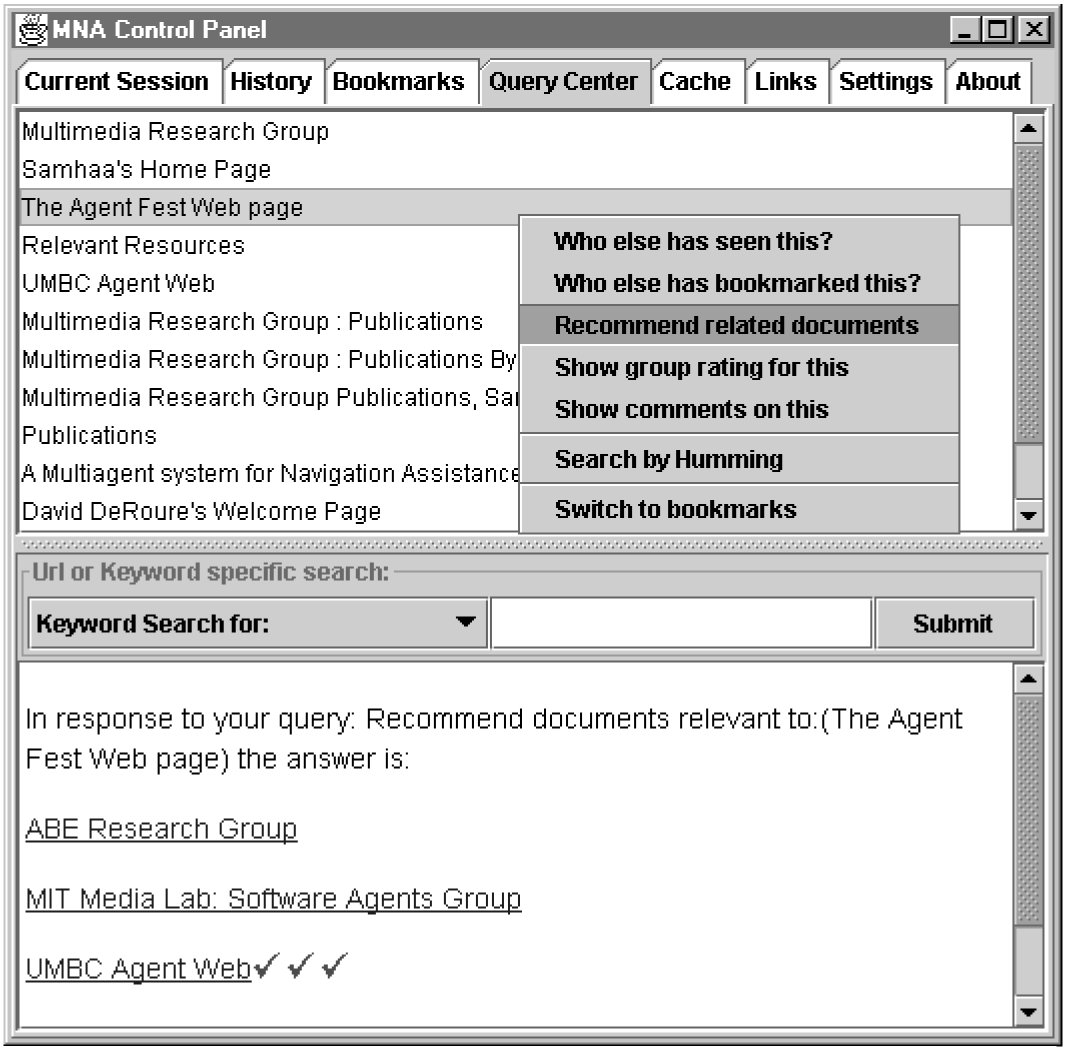

Figure 3: Query interface provided by the UI Agent

Document alteration services are performed by any agent that can somehow change the appearance of a document. The agents that perform such a service are called document modifiers. When document modifiers register with the

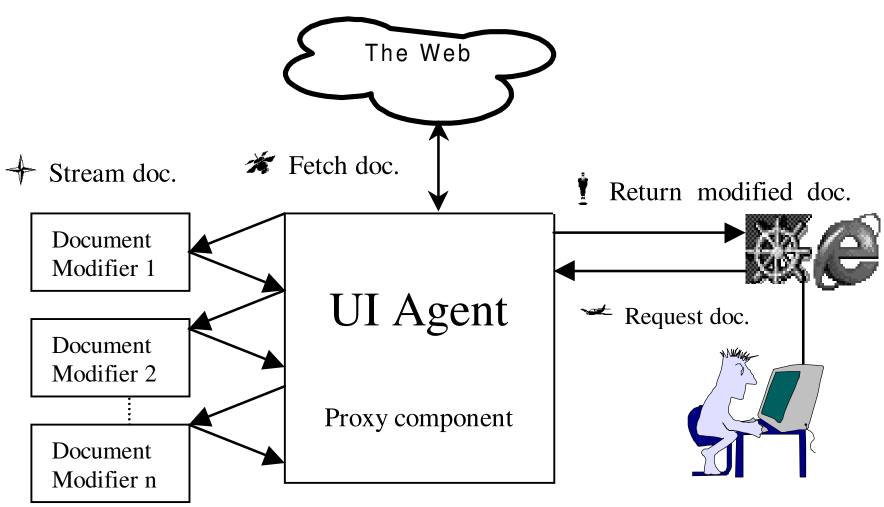

Figure 4: Flow of a document across different agents

system, they have to provide a description of the kind of alteration they are capable of performing. The user interface agent makes the user aware of the existence of such a service and its description, but it is up to the user to choose which of those (if any) to employ. The proxy server component of the user interface agent plays a major part in this process. The way in which the proxy server is built allows for incremental services that influence the display of a requested URL to be added. This process can currently take place in one of two ways according to the user preference and the number of document modifiers that exist in the system. In the first method, the proxy fetches a requested URL, but rather than sending it back to the browser, streams it across the different document modifiers while piping the output of one agent as an input to another. Figure 4 illustrate this technique. This cascading makes the process of document alteration less time consuming, but as the number of document modifiers in the system increases, the response time for displaying a requested URL may not be acceptable. The second method relies on the concept of server push (Netscape, 1999). In this scenario, the proxy server sends the document to the browser as soon as it is fetched. It then sends the content of the URL to the different document modifiers. When the last of the agents finishes, the proxy server "pushes" the final modified document to the browser and closes the connection.

Examples of document alterations that can take place in the system include the addition of links on the fly, the display of personal ratings and/or the average group ratings on a given link whenever it appears in a Web page, and the addition of links to comments made by other users on a given link. All of these alterations serve to guide the user's navigation activities as it allows for information created by other users to be instantly shared. As such, information created by other users is used to recommend a given course of action. Because of the way the system is built, any document modifier can be integrated into the framework at anytime thus extending the ways in which users can personalize the presentation of documents. Currently document modification is limited to *html/text* type of documents. Figure 5 shows how enabling the personal user ratings document modifier effects the final display of a document.

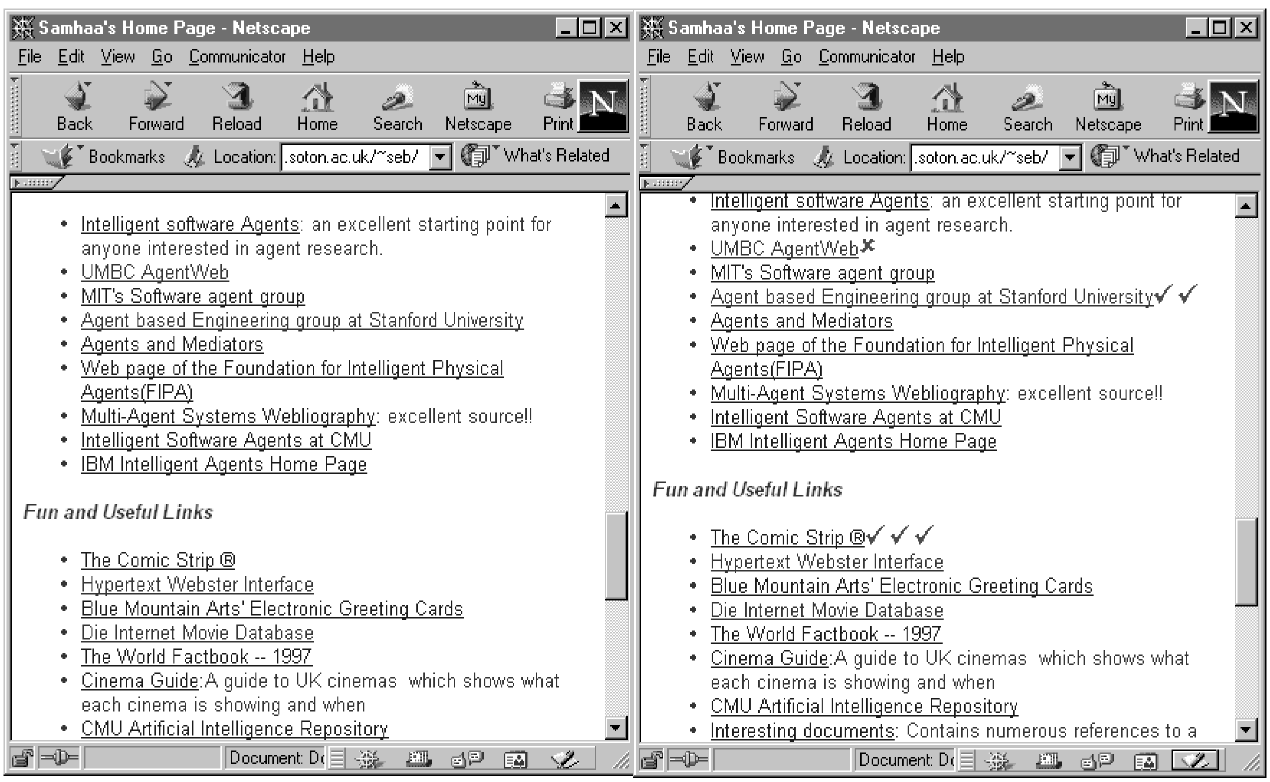

Figure 5: A Web page before and after the Personal Ratings document modifier is activated

4 Towards active bookmarks

The presented agents are currently being extended to provide users with even more powerful recommendation and information finding services. One of the agents undergoing such a change is the UI agent. The agent currently offers users an advanced bookmarking facility, yet the usage of bookmarks has not been exploited to the maximum. We are currently experimenting with the concept of active bookmarks. The concept basically implies that a user can set any of his/her bookmark categories to be active, in which case the user's agent will attempt to expand this bookmark category by finding documents similar to those in that category and placing those in a recommendations subfolder under that category. This allows the agent to model a user's current interests through explicit instructions from the user.

The principal idea is that the UI agent can extract a set of representative keywords from the bookmarks in a given active bookmark category and dispatch those to search engines or other agents in the system that can perform search operations and use a profile of the active category to filter the results. To this end, a profiling component has been developed. Our current implementation is based on the established text classification TF-IDF method (Salton and McGill, 1983). A bookmark category is currently represented as a vector reflecting the aggregate of its contents. In order to build the bookmark category vector, keywords for each of the documents are extracted. Extra weight is given to words in titles and heading, and to keywords entered by the users or extracted from meta-tags. All keywords are then added. The intersection of any keywords in various documents is given higher weight. The aggregate of the documents is then treated as a single document for which TF-IDF vector of length N is calculated. The results returned from the search engines are then filtered using the similarity function. Investigation into ways of combining this with user rating and browsing history is currently underway. This will allow the user interface agent to personalise the process of document retrieval as it will carry out this process based on a user context defined by a bookmark category while taking into consideration the likes and dislikes of the user.

5 Related Work

Recommender systems (Resnick and Varian, 1997) are based on the idea that users often face the problem of having to make choices without sufficient experience and can therefore rely on other people's recommendations. Examples of recommender systems include Webmate (Chen and Sycara, 1998), Letizia (Lieberman, 1995), Alexa (Kahle and Gilliat, 1998), Amalthaea (Moukas and Zacharia, 1997) among others. Recommending a document to a user in these systems is mainly determined by the user's profile acquired by monitoring the user's browsing or keyword search activities. The presented system differs in the approach it has taken to the problem and in which bookmarks shared among users, are used to recommend documents. The agent responsible for recommending documents in this system employs a technique similar to that used by Siteseer (Rucker and Polanco, 1997). The presented system also differs, in the way it has integrated a number of services into a multi-agent system to facilitate information finding.

Document modifiers introduced in this work extended the DLS (Carr et al., 1995; Carr et al., 1998b) model to cater for the dynamic manipulation of content in general rather than just for links. Manipulating content is a powerful concept that has also been addressed in the development of IBM's WBI (Barrett and Maglio, 1998) where intermediaries acted as the document modifiers. The uses cited by the developers of this system include Web advising, targeted advertising, and image distillation. The work presented differs primarily in the way it has addressed document modifiers. By allowing agents to declare the document modification capabilities using a standard protocol, a great deal of flexibility and openness is achieved. As agents join the system, the power of the system expands. The user is always in control of what and how he/she wants to view a document. Personalization of Web document content was only one aspect of the presented framework. Document modifiers that have been discussed in the presented system comply with the notion of recommender systems.

6 Future Work

Rather than having users define the context for which they want a linkbase activated, we are experimenting with the notion of dynamically determining a linkbase context based on document similarity and link structure.

At the moment, all agents in the system are assumed not to behave disruptively, which makes the system vulnerable to attacks. Agents also publish their information to any interested agent that joins the framework and expresses interest in that information. This might allow an agent to have access to unauthorised information. So stricter means for registering an agent in the framework have to be applied. We are considering a security model such as role based access control with delegation, which we believe to be appropriate for agent-based information systems.

There is an open issues as to whether the recommendations derived from confidential information are themselves confidential.

Previous work has addressed the usefulness of adding annotations (LaLiberte and Braverman, 1995), and links (Carr et al., 1995), on the fly. This work however makes use of a multi-agent architecture to bring together recommendation services and document modification services into one integrated architecture that better serves users in their browsing activities.

We have implemented a new Java-based agent framework (the *Southampton Framework for Agents Research* – SoFAR) which has basic support for ontologies and supports mobility. Some of the agents in this paper are being deployed in this framework, and are able to interwork with a variety of agents used in open hypermedia and multimedia research projects.

7 Summary

In this paper, a multi-agent system that supports navigation assistance and information finding was described. The architecture is open and extensible, supporting the dynamic addition and retraction of agents and services. Agents can be added to the architecture at any point in time to alter the content of documents, illustrating the flexibility of the architecture in supporting dynamic modification of document content as well as in personalisation of document presentation. Document modifiers presented in this paper focused on the idea of making information, created as a result of user feedback and link creation activities, dynamically available to others in appropriate contexts so as to assist them in their navigation process.

8 Acknowledgements

The work presented in this paper has been supported by ESPRC grants GR/K73060 and GR/M77086.

9 References

Altavista (1998). The Altavista Search Engine. **http://www.altavista.com**

Barrett, R. and Maglio, P. P. (1998) *Intermediaries: new places for producing and manipulating Web Content*, In Proceedings of *7th International World Wide Web Conference* Elsevier Science, Brisbane, Australia, pp. 500-518.

Bush, V. (1945). As we May Think. *The Atlantic Monthly,* **176**(1), pp. 101-108.

Carr, L. A., De Roure, D. C., Davis, H. C. and Hall, W. (1998a). Implementing an Open Link Service for the World Wide Web. *World Wide Web Journal,* **1**(2).

Carr, L. A., De Roure, D. C., Hall, W. and Hill, G. J. (1995) *The Distributed Link Service: A Tool for Publishers, Authors and Readers*, In Proceedings of *Fourth International World Wide Web Conference: The Web Revolution* O'Reilly & Associates, Boston, Massachusetts, USA, pp. 647-656.

Carr, L. A., Hall, W. and Hitchcock, S. M. (1998b) *Link Services or Agent Services?*, In Proceedings of *Ninth ACM Conference on Hypertext* Pittsburgh, USA.

Chen, L. and Sycara, K. (1998) *Webmate: A Personal Agent for Browsing and Searching*, In Proceedings of *The Second International Conference on Autonomous Agents* , pp. 132-139.

Davis, H. C., Hall, W., Heath, I., Hill, G. J. and Wilkins, R. J. (1992) *Towards an Integrated Information Environment with Open Hypermedia Systems*, In Proceedings of *The Fourth ACM Conference on Hypertext* Milan, Italy, pp. 181-190.

De Roure, D. C., Hall, W., Reich, S., Pikrakis, A., Hill, G. J. and Stairmand, M. (1998) *An Open Framework for Collaborative Distributed Information Management*, In Proceedings of *Seventh International World Wide Web Conference (WWW7)*, Vol. 30 Elsevier, Brisbane, Australia, pp. 624-65.

De Roure, D. C., Hall, W., Reich, S., Pikrakis, A., Hill, G. J. and Stairmand, M. (2000). MEMOIR - An Open Framework for Enhanced Navigation of Distributed Information. To appear in *Information Processing & Management. An International Journal* .

El-Beltagy, S., De Roure, D. and Hall, W. (1999) *A Multiagent system for Navigation Assistance and Information Finding*, In Proceedings of *The Fourth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* The Practical Applications Company Ltd, London, UK.

Finin, T., Fritzson, R., Mckay, D. and McEntire, R. (1994). KQML- A Language and Protocol for Knowledge and Information Exchange, In Technical Report , UMBC, Technical Report CS-94-02.

Gronbaek, K., Sloth, L. and Orbeak, P. (1999) *Webwise: browser and proxy support for open Hypermedia structuring mechanisms on the World Wide Web*, In Proceedings of *8th International World Wide Web Conference* Elsevier Science, Toronto, Canada, pp. 253-267.

Gronbaek, K. and Trigg, R. H. (1994). Design Issues for a Dexter-based hypermedia system. *Communications of the ACM,* **37**(2), pp. 40-49.

Hill, G., Hutchings, G., James, R., Loades, S., Halé, J. and Hatzopulous, M. (1997) *Exploiting Serendipity Amongst Users To Provide Support For Hypertext*, In Proceedings of *Hypertext 97* Southampton, UK, pp. 212-213.

Kahle, B. and Gilliat, B. (1998). Alexa - Navigate The Web Smarter, Faster, Easier, In Technical Report , Alexa Internet, San Francisco, .

LaLiberte, D. and Braverman, A. (1995) *A Protocol for Scalable Group and Public Annotations*, In Proceedings of *Third International World-Wide Web Conference,*, Vol. 27 omputer Networks and ISDN Systems, , pp. 911-918.

Lieberman, H. (1995) *Letizia: An Agent That Assists Web Browsing*, In Proceedings of *The 1995 International Joint Conference on Artificial Intelligence* Montreal, Canada.

Moukas, A. and Zacharia, G. (1997) *Evolving a Multi-agent Information Filtering Solution in Amalthea*, In Proceedings of *Agents' 97* Marina Del Rey, California, USA.

Netscape (1999). An Exploration of Dynamic Documents. **http://www1.netscape.com/assist/net_sites/pushpull.html**

Pikrakis, A., Bitsikas, T., Sfakianakis, S., Hatzopoulos, M., De Roure, D. C., Hall, W., Reich, S., Hill, G. J. and Stairmand, M. (1998) *MEMOIR - Software Agents for Finding Similar Users by Trails*, In Proceedings of *PAAM98 - The Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*(Eds, Nwana, H. S. and Ndumu, D. T.), London, UK, pp. 453-466.

Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM,* **3**(40), pp. 56-58.

Rucker, J. and Polanco, M. J. (1997). Siteseer: personalized navigation for the Web. *Communications of the ACM,* **40**(3), pp. 73-76.

Salton, G. and McGill, M. J. (1983) *Introduction to Modern Information Retrieval,* McGraw Hill, New York.

W3Consortium (1998). Synchronized Multimedia Integration Language (SMIL) 1.0 Specification. **http://www.w3.org/TR/1998/REC-smil-19980615**