University of Southampton

An Agent Based Framework for Navigation Assistance and Information Finding in Context

by

Samhaa Rafee Adel El-Beltagy

A thesis submitted for the degree of

Doctor of Philosophy

in the

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

March, 2001

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE

ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

AN AGENT BASED FRAMEWORK FOR NAVIGATION ASSISTANCE AND INFORMATION FINDING IN CONTEXT

by Samhaa Rafee Adel El-Beltagy

The main objective of this work is to address the problem of information overload within small groups, driven by similar goals in a way that would enable the delivery of personalised and non-intrusive browsing recommendations and hints as well as aid the users in their information finding activities. 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. The presented model utilises, extends, and combines ideas from open hypermedia with those from Web assistants and recommender systems to achieve its goals. The result of this combination is manifested in the idea of 'linking in context' which this work presents as a novel way of offering Web users recommendations for concepts related to what they are browsing. The integration of the various concepts is facilitated by the use of a multiagent framework.

Creating a flexible and open architecture that can accommodate these goals as well as identifying information finding and recommendation building blocks, is one important dimension of this work. Developing a linking model to embrace context on the user and document level, is another.

Table of Contents

List of Figures

List of Tables

To my parents and Khalo Ali, whose mere presence in my life enriches it and gives it meaning and purpose and to Mohammed and Mostafa, the best brothers any sister can ever hope for.

Acknowledgements

First and foremost, I'd like to thank my supervisor Professor Wendy Hall for providing me with the opportunity to work and study under her supervision, for believing in me, and for her incredible ability to make problems disappear. I'd also like to thank Professor David De Roure for his continuous support throughout the years I've been here.

To my friends, I'd like to direct a very special word of thanks. They've made my years of study memorable and enjoyable. First, I'd like to thank my dear friend Emilia Mendes. To the stranger that I was when I first embarked on my studies, her friendly smile and welcoming spirit helped me settle in faster than I thought possible. Thanks to Jatinder Hothi for always seeing the humour in what I said. She has helped me see the brighter sides of what I thought were grim situations. And for company I've always enjoyed, for the smiles and the laughter and the general spirit of joy, I have to thank my dear friends Cora Excelente, Alejandro Valera, Sanghee Kim, Angie (Muan Hong Ng), Stephen Chan, Zhuoan Jiao, Patricia Anthony, Fabiola Lopez, and Marijke Smith. Thanks to Jeff Wood for thought provoking conversations and for excellent company. Thanks to Reem Bassoiuny for putting up with my worst moods and for always willing to be there. Thanks to Manal Ismail, Hoda Waguih, and Khaled Bahnasy for always making me feel remembered.

I don't think there is anyone I know in the IAM group to whom I do not owe a word of thanks. This is one incredible group and I am so glad to have been part of it. I am indeed grateful to the Egyptian Central Lab. for Agricultural Expert Systems for giving me the time off to pursue my studies and for thus making this experience possible.

I'd also like to thank EPSRC for grants GR/K73060 and GR/M77086, which provided the funding for the research carried out in this thesis.

Chapter 1: Introduction

1.1 Motivation

Information is vital resource to individuals and organisations as its timely and accurate location can influence key decisions that affect both. The advent of the Web has revolutionised the way people look for and locate information by providing them with one of the richest sources of information known to date. With the rapid expansion of the Web, information resources are becoming available at an unprecedented volume and speed turning it into one of the largest distributed hypermedia systems known to date, and indeed, the most used. Unlike most hypermedia systems however, the Web lacks the conceptual model which organises and maintains data and link consistency (Nielsen, 1999). There is no doubt that the simple model employed by the Web has had a great impact on its growth. However, this simple model has resulted in the well-known problem of information overload on the Web, by failing to support the massive amount of information that it has assisted in making available. 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 its users is of equal importance. Achieving these goals has come to represent one of the biggest challenges in distributed information management and has led to the undertaking of massive research efforts on various fronts.

The aim of this work is to investigate ways of bringing together open hypermedia and an agent-based information management and navigation model in a way that guides users in their browsing and information finding activities.

1.2 The World Wide Web and Hypermedia

Following the emergence of the World Wide Web and its wide spread use, a strong association between the Web and hypermedia emerged. Researchers in the field of hypermedia would argue that the Web is a hypermedia system which has not fully benefited from research carried out in that mature field which predates the Web by a number of decades (Bieber et al., 1997). Hypermedia evolved from hypertext research which some have argued began as early as 1945 when Vannevar Bush proposed the first automated system for managing information about documents and their relationships through the use of trails (Bush, 1945). It was not before the 1960s however, that serious research in the areas of hypertext and hypermedia started. Influenced by Bush's ideas, Douglas Englebart created the "NLS" system, which is now recognised as the first hypertext system (Englebart and English, 1968). The terms hypertext and hypermedia were coined by Ted Nelson during the second half of the 1960s. When in the late 1980s Tim Berners-Lee created the basis of what is now known as the World Wide Web within CERN's Particle Physics Laboratory in Geneva, Switzerland (CERN, 1998; W3Consortium, 1999) many hypermedia systems were already in operation.

One of the identified goals of hypermedia is "to support (using associative sources) the carrying out of actions which result in the identification of appropriate information (with appropriateness being based on a given set of contextually defined criteria)" (Lowe and Hall, 1999). The current Web model seems to have failed in achieving that goal. This goal is one aspect of the information overload problem that this work sets out to address.

The nature of the Web is one of the reasons the Web has failed in achieving some of the goals of hypermedia. As most traditional hypermedia applications tend to be restricted and often closed, applying ideas from hypermedia to a system as

massive and heterogeneous as the Web is a rather difficult task prompting much research (Bieber et al., 1997).

This work has recognised the potential of applying ideas from *open hypermedia* to the Web as a way of providing personalised Web recommendations within a multiagent framework. Open hypermedia is a concept that has been researched by the hypermedia community for several years and for which a number of systems have been implemented (Davis et al., 1992; Halasz and Schwartz, 1994; Wiil, 1997). In open hypermedia systems (OHS), links are considered first-class citizens. They are managed and stored in special databases called *linkbases*. The idea of abstracting links from documents allows for a great deal of flexibility since it enables the addition of hypermedia functionality to documents, multimedia or otherwise, without changing the format of the original document or embedding mark-up information within it. It also simplifies link maintenance and reuse (Davis et al., 1992). The recently proposed XLink standard (Maler and DeRose, 1999) is increasingly moving towards the open hypermedia approach. Open hypermedia, has been recognised as a powerful technology capable of assisting users in navigating through multimedia information spaces containing both structured and unstructured information resources (DeRoure et al., 2000).

Previous work that has aimed at bringing the open hypermedia philosophy to the Web has suffered from limitations that this work attempts to address. Specifically, the main limitation of previous work lies in its failure to address context on the user and document level. Despite these limitations however, the work has shown that the open hypermedia philosophy can bring added power to the Web navigation model that warrants further research (Grønbæk et al., 1999; Carr et al., 1998a).

1.3 Objectives and Scope

The main objective of this work is to address the problem of information overload within small groups driven by similar goals in a way that would enable the delivery of personalised and non-intrusive browsing recommendations and hints as well as support the users in their information finding activities. 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. Building a flexible and open architecture that can accommodate these goals is one important dimension of this work. Addressing context at the user and document level is another.

Addressing context at the user level implies a level of understanding of the user so as to facilitate personalised information delivery. Documents that a user finds interesting or useful can contribute to an understanding of the user. They can also prompt the process of information delivery. As such an understanding of documents in contexts is also quite important. Because of the implications personalization has in terms of information delivery, it has been recognised as an important aspect of handling the information overload problem (Riecken, 2000). More generally speaking however, in attempting to address the problem of information overload on the Web three major trends highlighting different, though often inter-related, classes of research seem to be predominant:

1. A class of research that aims at introducing new, more advanced, representation models, that can add structure to the existing Web model. Much of the new protocols supported by the WWW consortium, such as XML (Bray et al., 1998), RDF (Lessila and Swick, 1999) and DAML (DARPA, 2000; Rapoza, 2000), and the research surrounding these, fall into this class.
2. A class of research that aims at addressing the problem of information overload on the personal and organisational level in the context of the existing Web representation model. Personal Web agents and assistants as well as Web recommender systems fall into this class.
3. A class that is aiming at automatically introducing structure to the existing Web model by mapping aspects of documents from the existing Web representation model to a more structured one through the employment of intelligent techniques.

The three major trends draw on research from hypermedia, information retrieval, information modelling, artificial intelligence, adaptive hypermedia, and agent technology. The research described in this thesis falls mostly into the second class.

The framework developed to achieve the goals of this work, is a multiagent one. On the personal and collaborative level, agent technology has been recognised as suitable for the tasks of navigation assistance and information finding. A multiagent architecture has the advantage of easily supporting extensibility and openness. The collective behaviour of agents within such an architecture, contributes to its power. The framework that this work presents was built with the following aims in mind:

1. To explore the application of ideas from open hypermedia for the purpose of making recommendations while addressing limitations that previous systems with the same goal have suffered from, particularly focussing on the issue of *context*.
2. To identify building blocks for modelling user activities so as to enable the sharing and reuse of this information across agents or recommender applications in a way that would ultimately be of benefit to the users of the system.
3. To make maximum use of what users do normally while browsing/searching the Web for their personal benefit as well as for the benefit of other users in the system.
4. To show, through user evaluations, that by achieving the first goal and integrating an intelligent model with an open hypermedia one, the system will be capable of acting as a powerful recommendation and information finding tool.

1.4 Contribution

The contribution of this work derives from the way it has approached the problem of assisting users in their information finding and navigation activities. The main contribution comes from the novel way in which this work has extended a traditional open hypermedia generic link model so as to allow for the delivery of

personalised Web recommendations in the form of contextualized links with the aid of software agents. The creation of links and their rendering in context is achieved through the extraction of links from the Web and their re-application in particular contexts. This is an idea that has not been explored before.

The work also presents an architecture that enables various agents with different underlying implementations and techniques to be integrated within the system at any point in time. This facilitates the expansion of recommendation and information finding services offered by the system. Key to achieving this is the identification and modelling of user activities as well as services related to the information finding and recommendation tasks, and representing those through an ontology which can be as simple as a conceptual model. A direct consequence of this is the enhancement of knowledge sharing and reuse across entities in the system, thus making building recommendation systems an easier task.

1.5 Thesis Structure

Chapter 2, Background, provides information about the various areas and concepts of related research as well as a more detailed description of the general problem that the work has attempted to address one aspect of.

Chapter 3, Context, provides further background information focused on the area of *'context'*. In this chapter, a general overview of *'context'* with emphasis on its application in locating information within the Web, is presented.

Chapter 4, Design Considerations and Building Blocks, discusses design issues imposed by the goals of this work as identified in section 1.3, and identifies basic entities that can be used to guide the tasks of information finding and recommendation making.

Chapter 5, Linking in Context, explains the implications and advantages of linking in context. The chapter introduces the idea of 'linking in context' as a feasible outcome of extending the open hypermedia model of a generic link to embrace

'context' and addresses practical questions surrounding the achievement of this idea. Issues addressed include the generation, and propagation of links in a way that would allow their employment in context.

Chapter 6, Design and Implementation, provides an overview of the developed architecture and a number of agents implemented to demonstrate the applicability and utility of concepts proposed by this work.

Chapter 5, The User Interface Agent, presents a demonstrator user interface agent developed to make use of the services offered by other agents in the system for the benefit of its user.

Chapter 8, Evaluation, examines the presented system through a case study aimed at examining the utility of 'linking in context' through an experiment involving a number of users. The chapter also evaluates the presented system through a close comparison with related systems.

Chapter 9, Conclusions and Future work, presents the conclusions derived from the presented work, and offers some future research directions that the author thinks are worth pursuing.

1.6 Declaration

This thesis is based upon work undertaken by the author within a collaborative research environment. Except where it is explicitly stated, what is presented is the original work of the author.

The work presented in this thesis has been supported by ESPRC grants GR/K73060 and GR/M77086 (the Queries in context (QuIC) project).

Chapter 2: Background

The work presented in this thesis draws on current research in a number of areas including agent technology, open hypermedia, intelligent information retrieval, information modelling, and system integration. The purpose of this chapter is to briefly describe these various areas as they form the background for the research carried out. Since the work addresses ways of assisting users with the problem of information overload, the problem is also described.

2.1 Information Overload and the Web

Information overload is a term that is frequently used to refer to the problem created by evolving technologies that enable the flow of information to users at rates and quantities that can easily cause them to become overwhelmed. From emails, faxes and voice mail to advertisements, manuals, and increasing access to more and more TV channels, everyone is touched by the problem in one way or another. The Web is one area where this problem is particularly evident and problematic. With the wealth of content it provides, the Web has become one of the best places to look for information, but because of the very same reason, finding specific information within its vast contents is not so easy and can easily baffle the searcher. A close look at the Web reveals why information overload manifests itself so clearly within it. The Web is a highly unstructured, heterogeneous information repository that is growing at an exponential rate (Baeza-Yates, 1998). Most resources available on the Web lack the semantics that can identify what they represent or contain (Gudivada et al., 1997). Due to lack of

document/system semantics, the prevalent information retrieval model is a simple one primarily based on keyword indexing, which provides the lowest common denominator across most resources. The use of keywords, especially considering the way indexing is performed by search engines, poses a number of problems.

The first relates to resources for which indexing information can not be easily created. Multimedia documents for example, are in fact hidden because indexing can currently only be performed on text based resources. Structured information systems that are continuously being added to the Web are another example. Such resources represent specialised services that are usually abstracted by a set of keywords which get advertised to a search service, and which hide a wealth of content available only through the specialised interfaces provided by these services.

The second problem relates to documents that have been successfully indexed. The lack of structure in the Web makes indexing the most natural way to search for information contained within it, but because of the volume of the data available, this searching model is not the most effective. Even across indexed documents, obtaining the *right* document for a given query can be quite complicated since a single query may map to volumes of Web pages with little or no relationship to each other. For example, a user interested in finding documents on *learning by example* in the domain of *machine learning* may enter the query "learning by example" to a search engine. Such a query will return pointers to documents related to education theory, to others that illustrate how to learn a given subject such as "html" by example, and to a lot of other unrelated pointers and possibly a few relevant URLs. Adding the words "machine learning" would probably improve the relevance of documents returned, but it is also likely to filter out many Web pages that are also relevant to the query, but for which the domain is implied in the content rather than stated. When a query is straightforward and the results returned fall in the appropriate domain, the specific content may not fit what the user had in mind and the quality of returned pages isn't always guaranteed to meet with the searcher's approval. Sifting though numerous links, including some dead

ones, or using one's experience in determining whether or not a link looks promising (if such an experience exists), is often enough the only way to find the required information.

A lot of these problems would disappear if a simple understanding of a query and the existing resources existed. This can be achieved by augmenting existing resources with semantic descriptors, but this is currently not supported by the standard information finding model. This absence of semantics makes it very difficult to obtain a noise free answer to a given query.

The need to adopt a more structured way for representing information on the Web has been recognised by the World Wide Web community. The proposal of metadata encoding standards such as the eXtensible Mark-up Language (XML) (Bray et al., 1998) and the Resource Description Framework (RDF) (Lessila and Swick, 1999) is meant to provide a way through which this can be achieved. The 'semantic web' proposed by Tim Berners-Lee, the creator of the Web, specifically aims to support a more intelligent model for information finding through the use of metadata (Berners-Lee, 1998a). Please refer to section 2.5 for more on metadata and to section 3.5 for more on the semantic web, and the role of structure in finding information on the Web.

As the Web grows in size, the problem can only get worse. It is worth noting that the current problems exist even though only 42% of available Web pages are collectively covered by all search engines (Introna and Nissenbaum, 2000) and despite the fact that most search engines attempt to crawl as many Web pages as possible. This is known as the coverage problem and has simply arisen because of the sheer size of the Web. This is yet another problem, which contributes to the problem of finding information on the Web.

What can be concluded is that solving the information overload problem on the Web is not something that is likely to take place in the near future and attempting to solve that problem on a global scale is out of the scope of any one project.

However, it is rapidly becoming apparent that novel approaches to the problem can yield acceptable results. Among the most promising concepts in addressing overload, is the concept of *context*. Agent technologies have also been recognised as a powerful tool for delivering context and in addressing information management in general. Agent technology and *context* are discussed in depth in the following section, and in the next chapter respectively.

2.2 Agent Technology

The origin of agent research lies in the field of distributed artificial intelligence. However, over the years, different research bodies embraced agent technology in very different ways. This has resulted in much confusion within the research community as to what an agent really is. Intelligence is one of the most controversial attributes of those that have been said to characterise agents. Since the artificial intelligence (AI) community has struggled for years to define intelligence without reaching a conclusion that everyone agrees on, what makes an agent intelligent, is still an ongoing debate and so is the question of whether intelligence is in fact a necessary characteristic of agents. Numerous publications have attempted to clarify the confusion by describing what characteristics agents should have and by offering different agent classifications (Jennings and Wooldridge, 1995; Wooldridge and Jennings, 1995; Nwana, 1996; Franklin and Graesser, 1996; Huhns and Singh, 1997a; Shoham, 1999; Hendler, 1999). Despite these efforts, there still doesn't seem to be much of an agreement on what an agent is, and diverse research activities are taking place under the banner of software agents. A comprehensive survey on that research is out of the scope of this review. However, in the next few subsections a brief summary of major agent characteristics and the use of agent technology in the context of distributed information management, specifically in navigation assistance, resource discovery, information finding and system integration, is presented.

2.2.1 Agent Characteristics

While there is much dispute as to which combination of characteristics an agent should have, there is a set of features that almost every agent system draws from. What follows is a brief description for each of these:

- ***Autonomy***: the ability of an agent to exercise 'exclusive' self control over its actions (Wooldridge and Jennings, 1995; Jennings and Wooldridge, 1995; Belgrave, 1995).
- ***Pro-activeness, purposefulness***: a feature that implies that an agent's actions are not simply controlled by its environment, but are dictated by a behaviour that allows an agent to take 'initiative' and exhibit 'opportunistic', goal directed behaviour (Jennings and Wooldridge, 1995; Franklin and Graesser, 1996).
- ***Communication/Social ability***: the ability of agents to communicate with external entities such as other agents, humans, objects, etc, in order to achieve their own goals or to assist other entities in achieving their goals (Belgrave, 1995; Jennings and Wooldridge, 1995).
- ***Responsiveness, sensibility***: the ability of an agent to perceive its environment and to respond in a 'timely' manner to changes that take place within it (Jennings and Wooldridge, 1995; Franklin and Graesser, 1996).
- ***Persistence***: the ability to maintain a consistent internal state over time (Belgrave, 1995).
- ***Reactivity***: a feature that implies that an agent has no built in complex model of its environment and that its behaviour *emerges* as it reacts to stimuli representing the state of the environment in which it exists (Nwana, 1996).
- ***Mobility***: the ability of an agent to move among different hosts in wide area networks in order to complete its tasks (Jennings and Wooldridge, 1995; Belgrave, 1995; Nwana, 1996)
- ***Adaptability***: the ability of an agent to adjust its behaviour over time, in response to changes in the environment or as a result of newly obtained knowledge. Adaptability is a characteristic that is often associated with learning (Jennings and Wooldridge, 1995; Franklin and Graesser, 1996; Belgrave, 1995).
- ***Believability***: the ability of an agent to exhibit "personality" and an emotional state (Franklin and Graesser, 1996).
- ***Rationality***: the ability of an agent to do the 'right' thing, i.e. to act in order to achieve its goals rather than in a way that prevents its goals from being achieved (Russell and Norvig, 1995).

- ***Honesty***: the feature that indicates that an agent will not knowingly communicate false information (Jennings and Wooldridge, 1995).
- ***Intelligence***: In general, it could be stated that an agent has intelligence if it has some sort of built in learning or reasoning ability or a combination of the two.

2.2.2 Agents and Information Management

The class of agents that perform information related activities has been given the label of "information agents" (Nwana, 1996). The single unifying feature that *information agents* seem to share is simply the fact that they all address information related problems. Other than that, they have quite diverse features. For example, some are mobile, others are static, some have learning capabilities, others have built in mechanisms or rules for dealing with information, some may communicate and co-operate with other agents in order to accomplish their tasks, others act independently and non-socially, etc.

In general, information agents are said to be capable of satisfying one or more of the following tasks (Klusch, 2001):

- **Information acquisition and management**: this entails the retrieval, extraction, analysis, and filtering of data as well as the maintenance of this data. It might also involve the provision of transparent access to one or more information resource.
- **Information synthesis and presentation**: this involves the possible integration of information from various resources, and its presentation to the user.
- **Intelligent user assistance**: this task involves adapting to the users' changing preferences in a way that enables the support of each of the user's particular information needs.

When building an information agent, the characteristics and capabilities chosen for that agent are highly dependent on the nature of the information system it will

target. The nature of information in the system and the degree of structure characterising it thus play a major role in deciding how to build the agent.

A distinction has been made with respect to various sources of information dividing them into three different classes: structured, unstructured, and semistructured. Databases, relational or otherwise, are examples of structured information sources. Examples of unstructured information sources include text files and a large class of Web pages. The Web in fact, is one of the largest repositories of unstructured information. An example of a semistructured information source is a Web page with known fields of content.

The term that has been coined for handling information in distributed repositories is *distributed information management.* Though research in this area still poses many major challenges, the use of software agents has been shown to offer many advantages over more traditional approaches and their effectiveness has been well documented. Two areas in particular, where the use of agent technology has demonstrated such advantages are those of system integration and information finding on the Web (Maes, 1994; Moukas and Zacharia, 1997; Kahle and Gilliat, 1998; Chen and Sycara, 1998; Knoblock and Ambite, 1997; Nodine and Unruh, 1997). Work on system integration seems to have focused more or less on structured or semi-structured information sources (El-Beltagy, 1998), while information retrieval techniques have been used widely to find information in unstructured ones. In both areas of research, devising techniques for finding information is a major part. However, due to the different nature of the resources addressed, the focus and techniques employed are significantly different.

Despite the difference and complexity in their approaches, the focus of most system integration projects using agent technology has been on the logistics of gluing systems together. Systems such as Garlic (Carey et al., 1995), Infomaster (Geddis et al., 1995) and the Information Manifold (Levy et al., 1996) only addressed integration on the level of structured resources. Other projects such as TSIMMIS (Garcia-Molina et al., 1995), Lore (McHugh et al., 1997), SIMS

(Knoblock and Ambite, 1997) and InfoSleuth (Nodine and Unruh, 1997), addressed methods of mapping semistructured sources onto domain models of a different nature through wrappers that are capable of representing their capabilities. Agents in such systems are capable of communicating information about the domain models or the parts of those which they could understand. Discovery of resources is facilitated by the presence of such knowledge. Facilitator and mediator agents in many of these systems, create plans for breaking down complex queries and sending them to appropriate agents. Within some of these systems, agents dedicated to the task of integrating results returned from various agents and passing them on to others also existed.

Agents that were built to target unstructured information resources had an altogether different set of features. This class of agents started to emerge in the wake of the information overload problem and have demonstrated their superiority in navigation assistance, making recommendations, and information finding. Two main features dominate the design of such systems: personalization, and/or the employment of collaborative techniques. Most of the agents that dealt with information finding employed already existing search engines. They have not attempted to change the form in which a query is entered. To refine the queries, these systems employed user profiles acquired over time or thesauri. Alternatively, they employed knowledge of the kind of documents a user liked based on that user's feedback to filter the documents presented to another user with similar interests. Such systems fall under the class of personal Web assistants and information filtering agents, examples of which are: Webmate (Chen and Sycara, 1998), WBI (Barrett et al., 1997), Letizia (Lieberman, 1995), Alexa (Kahle and Gilliat, 1998), Amalthaea (Moukas and Zacharia, 1997) and InfoFinder (Krulwosh and Burkey, 1997). This class of agents is discussed in more details in sections 2.8 and 3.3. A subset of this class of agents has come to be known as Web recommender systems.

2.3 Recommender Systems

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. The range of applications that can be addressed by recommender systems spans any domain where users' opinions can be propagated in a way that can assist other users. Users in a community are recommended items based on what other users with similar interests have found interesting. Voting and rating various items have been two of the most popular ways used as a basis for matching user interests and making recommendations accordingly. The term "Recommender Systems" was coined to extend and generalise concepts introduced by collaborative filtering. Collaborative filtering has similar goals to recommender systems, but the focus is on highlighting or discarding information that is not likely to be relevant to a user's interest or priorities. This goal is subsumed by recommender systems, which in addition focus on highlighting and introducing information that is particularly relevant. Though a large number of Web agents can be classified as recommender systems, recommender systems do not necessarily have to employ agent technology.

The limitations of recommender systems arise from their social aspect, which introduces a number of restrictions. For instance, a large quantity of information has to be rated by many users before a recommender system can become useful. Also, for a new user joining the system, he/she has to rate many items before they can make use of the system. In most systems, providing the user with an incentive to make explicit ratings is lacking. Recommender systems also introduce privacy issues (Resnick and Varian, 1997).

The definition of a recommender system is further extended and generalised in this work. Though the work utilises and recognises the social aspect as an important factor in making recommendations, it also recognises that making use of other techniques, such as content based ones, in conjunction with the collaborative one can enhance the quality of recommendations. So, in the context of this work, a reference to a recommender system will be a reference to any system that can

recommend a course of action or an item of interest regardless of the underlying technique. The work also makes a distinction between two types of recommendations:

- *Recommendations on demand*: which are recommendations made when a user explicitly requests them.
- *Proactive recommendations*: which are recommendations proactively offered to a user based on monitoring their activities.

2.4 Ontologies

Ontologies are key to a number of current research areas. Databases, knowledge engineering, information integration, information retrieval, and agent based systems, are but a few of the fields where ontologies are currently being used (Guarino, 1998). In general, it could be stated that ontologies are required when two or more systems need to communicate or share information or knowledge about the world and what they know about it. In such cases, agreement on a common "vocabulary" is crucial. As such, an ontology was defined by Gruber as "an explicit specification of a conceptualisation" (Gruber, 1993) where a conceptualisation represents a simplified and abstract view of a domain or part of a domain that needs to be represented. By allowing a number of agents or systems operating within a specified domain to share an ontology, knowledge sharing and reuse is made possible since the different agents/systems are capable of "understanding" each other since they use the same "vocabulary". In the context of information systems, or the internet, ontologies are one of the currently identified best ways for capturing semantic relationships among database concepts or document features (Huhns and Singh, 1997b; Luke and Hendler, 1997).

2.5 Metadata

Most of the systems that addressed the problem of finding information within unstructured resources have primarily relied on keywords as a query mechanism rather than attempting to change the form in which a query is entered. The problem with using simple keywords is that they are incapable of conveying any semantics. It is true that user-profiling techniques are used in many cases to improve the relevancy of returned documents. However, the fact that the semantics

of a query is one of the factors that contribute to accurate retrieval of information has until lately, been overlooked.

More recently, the realisation of the limitations posed by a query model based on simple keywords, specifically, the lack of semantics in such a model has prompted the search for alternatives (Guarino et al., 1999; Luke and Hendler, 1997; Lessila, 1998). Augmenting unstructured documents with structured machine readable descriptors or ontologies is one of the solutions suggested to allow for greater flexibility and expressiveness in query formulation and answering (Luke and Hendler, 1997; Lessila, 1998). The idea of using data to describe data is not novel. The concept of metadata, or data about data predates the Web. In fact, its usage is not restricted to electronic media. Over the years, various metadata schemes evolved to address specific problems. For example, the MARC (Machine Readable Catalogue) scheme was specifically developed for libraries and the professionals that work with them. The DIF (Directory Interchange Format) was created for satellite imagery. The Text Encoding Initiative (TEI) proposed a standardised description of electronic text (Miller, 1996). The Dublin core (Dublin Core, 1999) is a popular model for the description of electronic resources.

In order to support the encoding of metadata on the Web, the World Wide Web consortium has introduced the eXtensible Mark-up Language (XML) and the more targeted Resource Description Framework (RDF) as data formats for structured document interchange (Bray et al., 1998) (Lessila and Swick, 1999). RDF extends the XML model for the purpose of describing resources. RDF has been influenced by various Web technologies as well as metadata efforts in the library community (Lessila, 1998) (see section 3.5 for more on RDF).

A few search and navigation assistance systems supporting ontologies through the use of metadata, have emerged. Examples of these systems include: SHOE (Luke and Hendler, 1997; Heflin and Hendler, 2000), COHSE (Carr et al., 2001), and OntoSeek (Guarino et al., 1999).

Though adding structure to Web pages through metadata is likely to have a very positive impact on the problem of information overload, the achievement of that goal is not quite so easy. Publishing documents in a system like the WWW is totally unrestrained and while such an approach provides a feasible and useful method for adding structure to documents and resources, it requires massive re-authoring efforts for documents that have already been published (Marchiori, 1998). Agreement on a universal vocabulary, or coming up with ontologies that cover domains that are present on the Web, is a very involved and problematic area of research (Nwana and Ndumu, 1999). Such ontologies are very likely to emerge rapidly in business related areas because of how crucial they are in such fields and because massive investments will be allocated to create these ontologies, but it is very likely that development of these ontologies in other areas will lag behind. So, despite the usefulness of the proposed approaches and the improvement they are likely to yield, it is unlikely that they will be adopted on the scale of the Web in the short term unless they are used in conjunction with other technologies that would facilitate the automatic augmentation of metadata to already authored documents (see section 3.5).

2.6 Open Hypermedia

The underlying premise of open hypermedia systems (OHS) (Wiil, 1997) is that links are informational entities in their own right and as such they can be stored, and managed independently of documents (Carr et al., 1998). A link is not simply a jump instruction in a document, but it denotes a relationship between a source entity and one or more destination entities. One of the advantages of storing links separately from documents is that this allows link reuse across documents. It also makes the task of maintaining links easier since a change to a link only has to be performed once and is automatically propagated to appropriate documents when rendering the link. Thus in OHS, links are considered first-class citizens. This approach contrasts with the way links are created on the Web using HTML. The current Web approach leads to static hypertext with fragile links and limits the link sources to text documents. To compensate for these limitations, recently proposed standards such as XLink (Maler and DeRose, 1999) and SMIL (W3Consortium,

1998) are increasingly moving towards the open hypermedia approach. Systems that have attempted to bring the open hypermedia philosophy to the WWW such as the *Distributed Link Service* (DLS) (Carr et al., 1995) (which predates the emergence of such standards), Chimera (Anderson, 1997), and *Webvise* (Grønbæk et al., 1999), have demonstrated the utility of this approach.

The DLS is based on a now classic OHS system called Microcosm (Fountain et al., 1990; Davis et al., 1992). The Microcosm system was among the first systems to explore of the idea of building a hypertext system out of a set of loosely coupled communicating processes. The processes can be pre-configured so that a particular user of a particular Microcosm application can use specified link databases, or linkbases as befitting the context of their activity. Like Microcosm, the DLS has a notion of multiple linkbases, and they are user-configurable so the user can select their 'context' from the available sets.

One type of link introduced in Microcosm is the generic link. Generic links allow a link to be followed from any occurrence of a particular piece of content such as a text string, to relevant destination anchors as defined in the linkbase. By employing generic links, a great amount of re-authoring is avoided, and link re-use is greatly enhanced. The DLS allows links, mostly generic, from linkbases to be applied to WWW documents on the fly. Initially, the DLS interface was implemented as an extension to functions offered by common Web browsers, but in order to move to a more neutral, browser and platform independent environment, proxy technology was taken up to deliver links (Carr et al., 1998b).

Webwise basically extended the approach of the DLS by supporting a richer set of structures and tools. It too, employs proxy technology. Despite their usefulness, both this system and the DLS suffer from a number of limitations rising from the way they attempted to adapt linkbases to the Web.

When the concept of linkbases was first introduced in Microcosm, the notion of an application within the boundaries of a well defined domain existed. The system

was open in the sense that documents related to that application could be added at any time irrespective of the medium they were created in, and links would be automatically rendered on those documents. As such, the notion of the generic link was particularly useful because as soon as documents were added to an application, links were automatically available from that document to other documents in the application. When the DLS introduced concepts from open hypermedia to the Web, the responsibility of determining link context fell on the shoulders of the user. So the major limitation of the system lay in its inability to automatically switch between linkbases depending on the context of documents.

2.7 Hyperlinks

A link can simply be defined as an entity that represents a relationship between a source anchor and one or more destination anchors. A number of taxonomies have been proposed to describe various possible link types in hypermedia. In a survey examining the various types of links, the link classification shown in Figure 2.1 was identified (Ashman et al., 1997). Hand made links are simply defined as links that are created by a user. Computed links are those that are automatically created through some computation. These are further divided into two classes. A pre-computed link is generated and stored persistently for use in the future. A dynamic link is created only when the need arises and may or may not be stored persistently after creation. If it is stored, then it becomes a cached link.

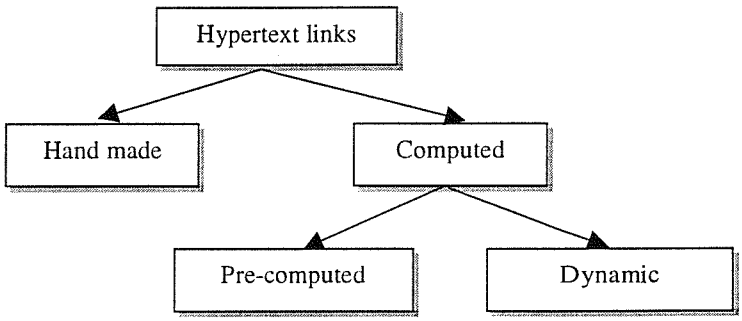

Figure 2.1: A simple classification of link types

The generic link model just described in the previous section, falls under the class of dynamically computed links. Another example of computed links, is that of links retrieved using CGI scripts that take a user supplied parameter invoked as a query to determine appropriate destinations which in turn can be represented as

computational links (Ashman et al., 1997). The functional link model is yet another example. In this model, both end points of a link can be computed using some function and a link computation can be a specification of a role rather than a specification of a single link instance. The functional link model is a flexible one, which allows the representation of any link type (Ashman and Verbyla, 1994).

Another identified link taxonomy is based on the relationship that exists between a link and what it links to. If the relationship is one aimed at the organisation of an information space, the links reflecting this relationship are structural links. If the relationship relates to the content of the information space, then the links embodying this relationship can be associative or referential links (Lowe and Hall, 1999). Associative links embody a relationship between some concept X and another concept Y that are independent while referential links convey a connection between a concept and more details or further explanation for it.

This is by no means a comprehensive overview of link taxonomies and types. More can be found in (DeRose, 1989; and Allan, 1996).

2.8 Examples of Related Agent based Systems and Recommender Systems

Supporting users in their information finding and navigation activities on the Web, has been the focus of much research. Over the past few years, many agent based and recommender systems exploring different ways of tackling this problem have emerged. This section provides an overview of some of the more important systems that fall under this class and which are most related to the work presented in this thesis.

2.8.1 WebMate

WebMate (Chen and Sycara, 1998) is a personal agent that was developed at Carnegie Mellon University (CMU). The goal of WebMate is to assist users in their search activities and to utilise a user profile for compiling a personal newspaper, contents of which, are obtained by crawling a set of predetermined resources. A user's profile is based on a user's browsing activities, which are

monitored through the use of a standalone proxy server. An applet provides the interface between the user and the proxy.

A user profile is represented by a set of feature vectors, each of which denotes a user interest. It is assumed that at any point in time, a user has at most N interests. A viewed document contributes to a user's profile only if the user explicitly states that he/she has liked it. TF-IDF (term frequency-inverse document frequency) is used to obtain a feature vector from each document a user expresses a liking for. The technique is used to calculate the importance of terms appearing in documents. For more details, please refer to section 5.2. Similar interests are combined and represented by a single vector. After the limit of N vectors is reached, vectors are merged based on the maximum similarity between two vectors. Similarity is calculated based on the cosine similarity co-efficient $Sim(d_i, d_j)$ which measures the cosine angle between two vectors d_i and d_j, and returns a number between 0 (totally unrelated) and 1 (identical) reflecting how similar they are (the closer the angle, the more similar they are).

A personal newspaper is compiled in one of two ways. In the first, the user provides a list of URLs he/she wants monitored. WebMate downloads these pages on a regular basis, extracts the links in those pages, and downloads and analyses text documents pointed to by these links in turn. If the similarity of any of these documents and any of the user interests is found to be greater than a give threshold value, then that page is recommended to the user. The second method involves query formulation based on the user's profile. Constructed queries are sent to a number of search engines, and the contents of the returned URLs are analysed and again recommended based on how closely they resemble the user's profile.

In addition, WebMate is capable of assisting users in their search activities through query expansion and relevance feedback.

2.8.2 Fab

Fab (Balabanovic, 1997; Balabanovic and Shoham, 1997) is a recommendation system that was developed at Stanford University. The goal of the system is to provide personalised recommendations through the employment of both collaborative and content based techniques. User profiles are built using content-based analysis, while the similarity between various profiles is used for collaborative recommendations. Representation of documents and user interests is achieved through the use of a vector space model and TF-IDF is employed for document analysis. A user's interest is represented by a single vector, to which a decay function is applied on a daily basis to reflect the user's changing interests. Explicit rating of Web pages is the basis on which the user profile is built. Two types of agents exist in Fab: selection agents and collection agents. Selection agents are responsible for finding pages for a given user while collection agents are responsible for finding pages for a given topic. A small number of topics determined by user interests are distributed among collection agents. Collection agents either search the Web breadth first from pages that make up their profile in an attempt to find documents that best match their topic, or formulate queries and send them to search engines in order to achieve the same goal.

Recommendations in Fab are provided on demand and are offered based on a user profile. When a user requests to view recommendations, he/she is shown the ten highest ranking documents based on his/her profile. By rating these documents, the system obtains feedback by which it refines the user's profile. Fab aims to make use of overlaps between user interests and employs a multiagent architecture to achieve openness.

2.8.3 Syskill and Webert

'Syskill and Webert' (Pazzani et al., 1996) is an example of an early Web assistant that employed user ratings to annotate interesting links and formulate queries. The system, which was developed at the University of California, Irvine, learns a given number of topics by being trained on a set of manually selected representatives of these topics. The focus of the system is on accurate classification of topics.

Training is done off line, and within very constrained domains. User profiles are acquired through user ratings provided on a subset of learnt documents. Web pages are represented by Boolean feature vectors which indicate whether a word is present or not in a given page. The selection of words that are used as features is determined by calculating the expected information gain which provides an indication of how much the presence of a word in a document, can affect its classification. For the classification process a number of machine learning algorithms were experimented with. Naive Bayes proved to be the most accurate. Once a user profile is learned, the system can annotate links in Web pages using icons that indicate whether the system thinks that the user should follow a link or not. Annotation of Web pages is not performed interactively and requires that all links be saved and analysed before the user can actually see any suggestions related to them, which demands a great deal of patience on the part of the user.

2.8.4 Letizia

Letizia (Lieberman, 1997; Lieberman, 1995) is a personal user interface agent that was developed at MIT. The goal of Letizia is to assist users in browsing the Web by recommending documents that are relevant to what the user is viewing and which are likely to be of interest to him/her. Inference of user interests is achieved completely through monitoring the user's browsing activity. Letizia performs a breadth-first search on links starting by those found in a user's current position. Text contents of pages pointed to by these links are analysed using TF-IDF in order to obtain a set of keywords that determine which of these pages are to be recommended to the user. So Letizia basically looks ahead into the contents of Web pages linked to the document a user is browsing to determine whether or not the link is worth following. The depth of a search is constrained by how much a user is interested in a given topic. This is determined through the interaction between the user and the system and how willing the user is to follow recommended URLs. Intially, Letizia starts by searching pages one link away from the user's current document, then if it detects a user interest, it searches pages two links away, and so on. Recommendations are provided in a separate browser window, contents of which are always changing.

Letizia is written in Macintosh Common Lisp. To monitor the user's browsing activity it employs AppleEvents and AppleScript to communicate with Netscape which it also uses as a user interface.

2.8.5 WebWatcher

WebWatcher (Armstrong et al., 1995; Joachims et al., 1997) is a system that was developed at CMU. The WebWatcher system was given the metaphor of a tour guide. In this system, a user is initially asked to enter a set of keywords describing his/her current interest. The agent then monitors the user's browsing activity and the links he/she follows from a given Web page. It also inserts a WebWatcher menu on the top of each page in the form of hyperlinks to WebWatcher functions. Using that menu the user can exit from the system at any point in time. The user can also provide comments, ask for help, ask for statistics relating to the links in the WebPage they are browsing by asking "How many followed each link found in this page?" or ask for similar pages. Based on what the user has specified as his/her interest, the system places icons next to hyperlinks in the Web page a user is browsing, as indictors of the relevancy of these hyperlinks to the user's interests. Various icons denote varying degrees in the strength of a recommendation. Links that are followed and which have not been recommended are used for training the system.

The system is implemented as a proxy like server, but rather than having the user set their browser to a given proxy, the system provides a starting point (a Web page) where links are rendered in a way that points back to the *WebWatcher* server. Every time a link is followed the server is thus able to download it and modify it by re-writing its hyperlinks to point back to itself. This technique is flexible in the sense that a user that reaches a Web page which has WebWatcher functionality, can start using it right away. However, it suffers from the fact that it limits usage of the system to a limited set of pages for which it was built.

2.8.6 MEMOIR

MEMOIR is an early collaborative recommendation system that was developed at the University of Southampton. The MEMOIR system is particularly relevant to the presented work, as one of this work's initial goals was to provide an improved architecture to support similar tasks as those adopted by MEMOIR. It could be stated that the presented work was a natural evolution from MEMOIR (El-Beltagy et al., 2000a). MEMOIR adopted an agent based architecture (DeRoure et al., 1998; 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. MEMOIR's architecture was an open one, based on the existing Web infrastructure and employed proxies as key components. Figure 2.2 provides a diagram of the architecture.

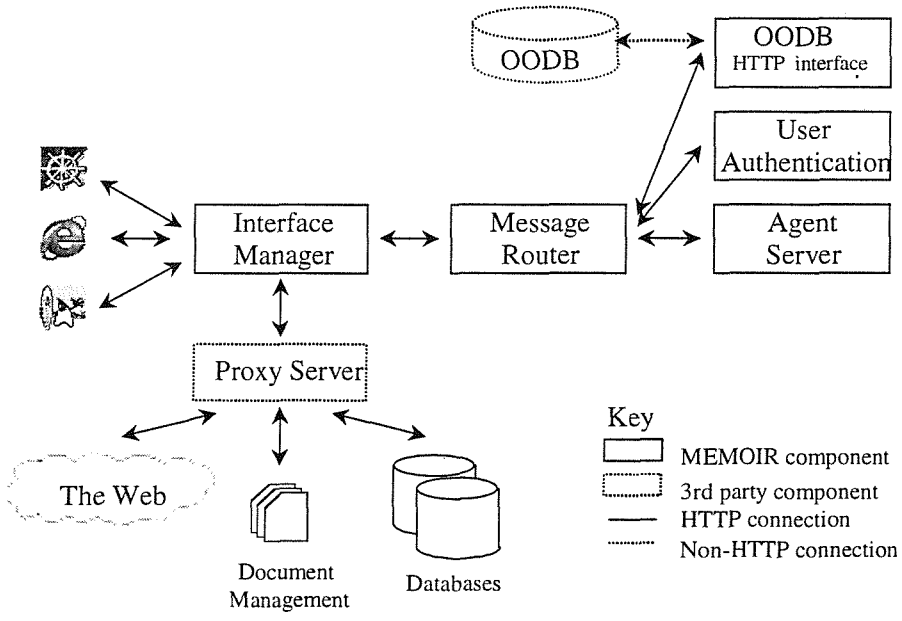

Figure 2.2: The architecture of MEMOIR

Users of the system were provided with the facility of grouping sets of documents they perceived as interesting into trails that were stored in a shared organisation memory. The notion of trails goes back to the pioneering work of Bush (Bush, 1945) where a trail is defined by a set of documents a user employs for the accomplishment of some task. In MEMOIR, a trail is defined as a collection of URLs that a user creates during a browsing session through manual selection from

the entire set of URLs the user has visited. Documents contained in a trail are assumed to be ones the user has found interesting and perceived as related. In MEMOIR, trails were key to some important system functions. By examining trails in which a given document appeared, the system was able to perform document recommendation services. The system used trail overlap to match user interests and to provide other users in the system that share similar interests, with recommendations extracted from the trails.

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. 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.

2.8.7 Margin Notes

Margin Notes is one of the more recent agent systems developed at MIT (Rhodes, 2000). The system is one of very few that have attempted to address context on a localised level within a document. The level of localisation addressed by the system, is that of document sections. One advantage of using such a system is that it is capable of recognising multiple contexts within a single document. The system is said to belong to a class of agents referred to as "Just-in-time information retrieval agents" (JITIR agents). A JITIR agent is defined as one which "proactively retrieves and presents information based on a person's local context in an easily accessible yet non-intrusive manner" (Rhodes and Maes, 2000).

Margin Notes modifies Web pages as they are being loaded by adding hyperlinks from the pages to related personal files. Like a number of other systems, the recommendations offered are based on the comparison and mapping between a document's text content to similar content. But unlike other systems, the recommendation is based on fragments of text in a document, as opposed to recommendations made based on an entire document. The system works by analysing various sections of a document and comparing each section to pre-

indexed text files such as emails and personal notes. Margin notes uses TF-IDF as the basis for document analysis and comparison. A suggestion is made in a pre-designated margin area created by the agent, and is presented in terms of an author, a subject, a date and a file name.

2.8.8 Watson

Watson is an information management assistant that was developed at Northwestern University, Illinois, and is another example of a system that has explicitly addressed context (Budzik, 1999; Budzik, 2000). An information management assistant (IMA) is defined as an entity that is capable of observing users interact with everyday applications and of proactively presenting them with information which is relevant to the context of their tasks. The developers of Watson also use the phrase "just-in-time information" access to describe the functionality of IMAs.

Watson can currently work with Microsoft Word and Microsoft Internet Explorer. The assistant monitors a user while he/she is working on a document or browsing a Web page and proactively suggests related information. Suggestions are made by analysing the content of the document the user is looking at and sending the words identified as important, to one or information retrieval system or search engine. Word importance is calculated based on the frequency of its appearance in a document as well as by employing a set of heuristics (Budzik, 1999). The returned results are clustered and presented to the user in a separate window. Clustering is not performed on the basis of full text analysis of documents, but rather based on the titles and URLs returned by the search engines or information retrieval systems to which the queries were sent.

In addition, Watson provides users with the capability of interacting with the system and of conducting 'queries in context'. This is achieved through the provision of an interface whereby users can enter a query, which Watson refines and contextualizes by adding keywords from the document the user is viewing before sending to a search facility.

2.9 Summary

Information overload within the Web is a challenging problem that has been approached from various perspectives. Information agents, recommender systems, metadata, ontologies, and 'context', highlight major active research areas that touch on this problem. Besides outlining the problem of information overload, this chapter has briefly described most of these areas. In addition, a brief description of open hypermedia and hyperlinks was provided as this work builds on both. Systems that are related to this work and that serve as exemplars of agent based and recommender systems were also discussed.

The next chapter offers a different perspective on the problem of information finding through the examination of the role of *'context'* in the achievement of this task. Some of the concepts and ideas introduced in this chapter are further investigated within that context.

Chapter 3: Context

Context is an important concept that has been examined in many different fields and for various tasks. With the information overload problem becoming more and more pronounced, the importance and urgency of employing context in that area, is becoming apparent. Intuitively, the information overload problem on the Web can be simply described as a direct cause of queries not being answered in context. However, the disciplined employment of context is complicated by the vagueness of the concept and the lack of an agreed upon formal definition for it. The fact that the study and usage of context spans very different research areas such as linguistics, cognitive psychology, artificial intelligence, and information sciences (Akman and Surav, 1996), has not made the problem any easier. The aim of this chapter is to provide a general overview of context with emphasis on its application in locating information within the Web.

3.1 A General Overview

The most general definition for *context* is one that can be found in a dictionary. In the Collins Cobuild English language dictionary one of the definitions given for context is as follows:

- The context of something consists of ideas, situations, events, or information that relate to it and make it possible to understand it fully.
- If something is seen in context or if it is put into context, it is considered with all the factors that are related to it rather than just being considered on its own, so that it can be properly understood.

This generic definition seems to apply to any model of context irrespective of the field in which it has been addressed. The primary difference in approaching context in different fields lies in how each of these fields attempts to identify and address the "ideas, situations, and events" that make it possible to establish context. These factors vary according to the task and domain for which context is needed.

Understanding ideas, situations and events in relation to their context has been a long-standing goal of AI. The work of McCarthy (McCarthy and Buvac, 1994) within logical AI, has inspired most of the research in this area. McCarthy's work focuses on means by which propositions in given contexts can be propagated to other more general contexts using lifting rules. One of the main goals of McCarthy's work on contexts was to establish a formalism for building AI systems with concepts that can transcend contexts that they started out with. Although this goal remains mostly unfulfilled, some advances were made in certain directions. For example, Guha used the ideas of micro-theories, which are theories of limited domains to model contexts on a large scale, and lifting rules to build a modular, common reasoning program called Cyc (Akman and Surav, 1997).

Understanding something that is being said or asked is also related to context in natural language, where determination of context helps eliminate multiple meanings and ambiguities. To fully understand what is being said though, other factors have to be taken into consideration. Examples of such factors include the identity of the speaker, his/her surroundings, what had been said earlier, etc. To describe this sort of context, the term "situational context" or "context of situation" has been coined (Akman and Surav, 1996; Akman and Surav, 1997).

Context-aware applications are an example of an area where context has been successfully employed. At Xerox Parc, where there is much research on mobile devices, the idea of stick e-documents, the electronic equivalent of post-it notes, but with an added dimension, has been explored as a means for creating context aware applications. A typical e-note has a notion of content and context. The content of the note is only triggered in a given context. Examples of contextual

elements include a given static location, proximity to a certain object (e.g. human or computer identifiers), time, imaginary companions, etc. The main idea, is that the e-note contains a set of conditions defining the context and hence the triggering of the content (Brown, 1996). It has been suggested that this approach can be combined with information retrieval to achieve "context-aware retrieval". Simply put, context aware retrieval is a way for allowing users to specify extra conditions for filtering results returned using normal retrieval where the conditions define the context (Brown, 1998).

Though the idea of *context* has been addressed for many tasks from navigation assistance in hypermedia systems (Boy, 1991; Hardman et al., 1993; Mukherjea and Hara, 1997) to security in distributed systems (Ferraiolo et al., 1995), the following subsections will only focus on *context* in relation to the Web. What follows is a closer look into what defines *context* and how it has been approached for guiding the task of information finding.

3.2 Answering Queries in Context: the Problem

To answer a query in context, a contextual mapping function has to exist. Hypothetically, such a function accepts a query and surrounding contextual factors, and maps the query to the resources that can answer it accordingly. Having a full understanding of a query as well as of the resources that can be used to answer it, can significantly help in building such a mapping function. By looking at a complex information system such as the Web, it is easy to see that having a full understanding of its resources is an almost impossible goal. Each page on the Web has a significantly different underlying role, which reflects the purpose for which the information within it is provided and for which it was built. In a way, each Web page is like a complex object which represents an abstraction of a set of real-world things such as a person, a publication, a university, a department, a company, an organisation, an agency, a service, etc. Services represent a class of their own because underneath them there usually lies one or more resources, contents of which are totally invisible except though a specialised interface or wrapper. This intricacy is obscured by the URL abstraction, which conveys

nothing about a resource except its location on the Internet. Moving from such a trivial model to one in which Web resources, and their relationships and interdependencies, are captured is by no means an easy task. Work on metadata and standardised ontologies targets this problem but still has to address a number of challenges (please refer back to sections 2.5 and 2.4). Without such a model, it is difficult to understand Web resources fully.

Understanding a Web query is not always any easier. Defining what constitutes a query is not as simple as it may seem. A query can be any of the following:

1. One or more documents when what the user requires is a set of results related or similar to those documents. Understanding the context and the similarities among the selected query documents directly contributes to the quality of documents retrieved.
2. A text selection in a document for which the user requires related links. In such a case, in addition to the text selection, the search has to be accompanied by some contextual information related to the document in which the text was selected.
3. A set of keywords, in which case the current interest of a user and his/her rating/bookmarking history or any other factor that can contribute to an understanding of his/her context can play a role in retrieving documents.
4. One or more words used in conjunction with a task. The task in that case serves as a constraint on the type of documents returned. An example of specifying a task is requesting a *definition* of a word.
5. A word and a specification of a required concept to retrieve in relation to that word.
6. A constrained request formulated in a formal query language.
7. A natural language request.

This is not an all-inclusive list and any of these queries can be accompanied by additional constraints that can serve to filter results and to further define the context. Such constraints can include the location of the user, the device the user is using, the media type required (html/text, jpeg, pdf, etc), etc. If all factors

surrounding a query are to be considered important, establishing the context of a query can be as complex as establishing context in the area of natural language processing (NLP), where it has been stated that the context "can be the whole world in relation to an utterance act" (Asher and Simpson, 1994). This very characteristic has made the contextual problem a "formidable" one in NLP (Akman and Surav, 1997).

To address the inherent complexity of the problem, many of the developed approaches have focussed on finding a good context approximation mapping function rather than attempting the impossible task of finding a perfect mapping function. Personal agents, novel search services and newly proposed representation models, have all endeavoured to find such a function though each approached the problem from a different angle. These approaches are briefly described in the following sub-sections.

3.3 Personal Web Agents and Context

Over the last few years, a class of systems known as personal Web assistants /agents has emerged to assist users in their information finding and Web navigation activities. Although these agents have various underlying implementations, they generally attempt to capture the user's context, the document context, or both, in one way or the other for the task of information finding or for making recommendations. The process adopted by these agents for achieving their tasks mostly follows one of two approaches: content based, or collaborative based.

In content based approaches, an agent attempts to infer the user's context by analysing the content of documents for which a user has previously displayed a liking/disliking. Agents that follow this approach attempt to personalise the process of information finding by using techniques that vary in complexity in order to extract users' interests based on what they've liked. This is usually done by extracting keywords from a user's browsing history or from documents that the user has expressed a liking for and use these to build a user profile. This class of agents performs well in both information filtering and recommendation tasks. In

the case of information filtering, returned documents are compared to user interests, and those that don't match, are filtered out. In the case of recommendations, documents similar to those that the user has liked are sought using various techniques, and recommended to the user. Recommendation presentation strategies vary significantly from one system to another (please refer back to section 2.8 for more details). Examples of these agents include CMU's WebWatcher (Armstrong et al., 1995), MIT's Letizia (Lieberman, 1995), and Stanford's Learning Information Retrieval Agent (LIRA) (Balabanovic and Shoham, 1995). WebWatcher (described in section 2.8.5) monitors a user's search activities entered in the form of keywords, and learns what the user is interested in by logging successful and unsuccessful searches and by getting feedback on the advice it gives the user. It uses that profile to highlight links that it believes will be of interest to the user. LIRA also uses user feedback in the form of ranks assigned to visited Web pages, and utilises this to search the Web and retrieve documents it believes will be of interest to the user. Rather than accept keywords, Letizia (please see section 2.8.4) infers user interests by monitoring his/her browsing activities, and recommends documents by performing a breadth-first search on Web pages that the user is currently viewing. Amalthaea (Moukas and Zacharia, 1997) also uses user feedback, but employs it to evolve a multiagent ecosystem for information finding. Many other systems belonging to that class of agents have been developed. A review of these agents can be found in (Mladenic, 1999). Agents that are particularly relevant to this work have been described in more detail in the previous chapter. What all these agents have in common is the fact that they use some sort of mechanism to infer user interests in relation to document content. This knowledge defines the context of a query and is used to highlight or filter information that a user receives.

Margin Notes (Rhodes, 2000) and Watson (Budzik, 2000) (described in sections 2.8.7 and 2.8.8 respectively), both follow a slightly different approach. In both systems, the content of a document a user is browsing or working on, define the user's local context. Based on that local context, the systems attempt to predict

documents that a user would be interested in and offer that to the user in a form of suggestions.

For agents that follow a collaborative based approach, also known as social learning/filtering, the underlying assumption and requirement is that a number of users will be employing the agents. Rather than matching users to documents, matching users to users is the focus. The idea is that if two or more users have enough in common in a particular domain to deduce that they have similar interests/likes/tastes, then it is possible recommend to one user, what other user(s) with similar interests have liked. This class of agents falls under the category of recommender systems (see section 2.3). This technique was initially exploited for assisting users in finding news group articles (Resnick et al., 1994), as well as for locating people with similar interests (Foner, 1997; Pikrakis et al., 1998). It was also used for the recommendation of books, movies, and music (Maes, 1994; Shardanand and Maes, 1995). Systems, such as FootPrints (Wexelblat and Maes, 1999), MEMOIR (Pikrakis et al., 1998) and SiteSeer (Rucker and Polanco, 1997), among others, have exploited the approach for information finding and navigation assistance.

Pure collaborative based approaches are best fitted for domains where the content is poor and automating 'word of mouth' is particularly important. When the volume of information is high compared to the probability that users in a given environment will be able to rate that information, then this approach on its own suffers from some obvious limitations. To overcome these limitations, a number of systems have combined both content and collaborative based approaches (Balabanovic and Shoham, 1997; Mladenic, 1999).

In the collaborative based approach, the definition of context is looser than that of the content based approach. The content of documents or objects, for which recommendation is given, is of no importance. Matching of user ratings for the documents defines similarities between users, based on which recommendations are made. So, a user's context is defined by what they have liked irrespective of its

content. If two user contexts are close, then overall context generalisation from one user to another is the basis for recommendation.

3.4 Search Engines and Context

The success of the Web can be attributed to the simple hypertext model that it employs, but with information of every kind imaginable being added to the Web at an unprecedented rate, this model is quickly proving extremely chaotic. The sheer volume of information has motivated search engines to employ indexing techniques that can index many gigabytes of data in the shortest time possible. As a result, most of these engines have resorted to the simplest query model possible. Consequently, the concept of context is lacking from most search engines that exist today. In an attempt to compensate for limitations associated with that simple representation/query model, a number of search engines employing some novel techniques have emerged.

The NECI meta-search engine is one example (Lawrence and Giles, 1998). This engine takes as input a user query expressed as a question and transforms that question into a number of queries that the entered query can translate to. This is done through the use of a limited set of specific expressive forms (SEFs). For example, any query in the form of "What [causes|creates|produces] x?", would be transformed to "x is created", "x is produced", "causes x", "produces x", "makes x", and "creates x". The various representations are then sent to different search engines and returned results are analysed and ranked based on their proximity to the desired phrases (Lawrence and Giles, 1998). Another example of systems that employ context differently is that of QUESCOT in which textual information is analysed and quantified in contexts with the aid of a WordNet database. In this system, a textual context is defined as "a sequence of semantically related items of vocabulary reflecting the exposition of a particular topic and creating an environment where further items are interpreted" (Stairmand, 1997).

Some claim that as the size of the Web grows the keyword search approach will no longer be adequate to address the needs of Web users exactly because of the fact

that this model can not support the idea of context (Theodorakis et al., 1998). The way humans communicate *context* is through natural language and as such it is expected that search engines that can support natural language, will provide a more adequate interface to searching (Theodorakis et al., 1998). Two advanced search engines, START and AskJeeves have addressed the issue of answering natural language queries.

The START system was developed at MIT and can answer questions related to the MIT AI Laboratory, Geography, celebrities and movies. The system is extensible, but requires the use of natural language annotations, which are sentences and phrases describing information segments. It is capable of analysing these annotations and creating a representation structure that natural language queries can be mapped to. Links from these representational structures to information segments, for which they were produced, are also created. (Katz, 1997). Examples of queries that can be answered by START are:

- What country in Africa has the largest population?
- Show me a picture of Bill Clinton?
- When was Harrison Ford born?
- Show me a map of England

Context is determined by the mapping between the question entered by the user and the annotated resources.

AskJeeves (AskJeeves, 1999) is a Web search engine that also accepts natural language queries. Unlike START, it doesn't come back with an answer, but with links to answers to related questions. For example, if you enter the question "How can I overcome insomnia", the system comes back with links to "How can I get a goodnight's sleep" and "How can I deal with the sleep problem insomnia". In addition the system consults popular search engines such as Altavista, InfoSeek, etc. In depth details of how the system works are not available due to the commercial nature of the system, but the developers have stated that a semantic and syntactic network are behind the system's power. The networks are used to

determine word meaning and context within the question and to map that to pre-existing questions and answers. To improve the system's performance, a log of user questions is continuously being analysed and human editors use this analysis to improve the system's knowledge base and semantic networks (Clark, 1999). The AskJeeves system boasts a knowledge base with links to more than 7 million answers based on the most frequently asked questions on the internet (AskJeeves, 1999).

Both START and AskJeeves have resorted to the use of some sort of knowledge for improving the process of retrieving information. Knowledge in both cases can be viewed as a way of better establishing the context of a query.

3.5 The Web Representation Model, and Context

Understanding the concepts that the contents of a document represent and their relationship to other concepts is an element that can greatly enhance the capability of a search facility. This is because the availability of such an understanding would enable reasoning about concepts and entities in a way that would allow for targeted queries and would ultimately provide more directed and relevant answers. As noted before, the fact that the Web employs a simple representation model that can not provide for the definition of document content in any meaningful way, has been an obstacle in the way of achieving better methods for information finding. Recent proposals for a new representation model however, promise to overcome this obstacle. Among these, is the resource description framework (RDF) (Lessila and Swick, 1999).

RDF is the corner stone of the Semantic Web proposed by Tim Berners-Lee, the creator of the Web, as a way of providing machine understandable information (Berners-Lee, 1998a). The framework provides a metadata encoding scheme that can be used for describing Web page content using rich semantics. It thus allows for logical expressions to represent a query. Because of these features, it has been stated that RDF allows for the representation of information in a way that is analogous to a "simple frame system" without defining any reasoning mechanism

that can be used to infer meaning of what it describes (Flammia, 1999). RDF has also been said to be capable of representing entity relationship models (Berners-Lee, 1998b). RDF descriptions can be expressed using the extensible markup language (XML) (Bray et al., 1998). XML name spaces can be used to define the context of concepts that appear in any given document.

Another standard emerging as part of the semantic Web is DAML (the DARPA Agent mark up language) (DARPA, 2000) which is also supported by the World Wide Web Consortium. To make it capable of integrating with other Web technologies, DAML is based on XML and RDF. One of the main goals of DAML is to enable agents to intelligently collect various information pieces from a number of sources and integrate them together to fulfil a given query (Rapoza, 2000).

OntoBroker (Fensel et al., 1998) and SHOE (Guarino et al., 1999; Luke and Hendler, 1997) are examples of two real systems that have employed ontologies on the Web through page mark up. Both systems introduced extensions to HTML that would allow authors to embed ontological descriptions in their documents. Demonstrators for both applications exist for a number of restricted domains.

Approaches that aim at marking up Web pages with rich semantics suffer from a number of limitations. The success of these approaches depends on the creation of standard ontologies or descriptions across most domains, and the degree to which Web page authors adhere to these standards (see section 2.5). The Dublin core (Core, 1999), which is a simple content description model for electronic resources, is an example of one such established standard. Powerful tools are needed to provide authors with an easy way of marking up Web pages. Provided that these tools come into existence, allowing existing documents to become accessible through the new model by converting them to a new standard would require serious re-authoring efforts. Manual re-authoring is probably impossible because of the volume of currently available documents. To address this, an approach has been proposed to automatically add structure to Web pages.

Rather than relying on Web-page authors to describe the contents of their pages, a way of automatically creating computer understandable knowledge bases that reflect the content of the Web was devised. The approach makes use of machine learning techniques to learn to extract information from Web pages based on a set of pre-defined concepts, or a predefined ontology (Craven et al., 1999). One of the limitations associated with this approach is that it requires training using a sufficiently large set of examples. If the ontology being learned can generalise across a very large class of documents, then the use of this method is justified. The degree and requirement for accuracy with respect to learned concepts is still an open research issue.

3.6 Summary

In the previous subsections, context was discussed primarily in relation to the Web and the task of guiding the information finding process. The systems and approaches reviewed are by no means comprehensive. Rather, what was presented, were systems and approaches that represent major trends. The approaches presented varied significantly according to the assumptions they made about the representation and query model. By examining various systems that have addressed context based on the Web's current limited representation model, it can be stated that most of these attempted to focus on particular factors that promise to yield the greatest results in terms of a context approximation. For example, when a query is entered using simple keywords, the most obvious way of determining user context seems to be the use of relevance feedback, or a like/dislike factor, to build a user model. Document context is obtained by analysing document content using methods applied to the words contained in these documents. A similarity metric is applied to map between a query context and/or a user context, to a document context. The complexity of the similarity metric varies across different applications. More sophisticated techniques make use of user profiles, i.e. they explore a user's *context* to further pinpoint and focus the returned *context*. Using machine learning or statistical based techniques, context is mostly determined by

factors that the designer see as indicative and which are acquired and modified dynamically as a user carries out a given activity within the system.

In the logical/symbolic approach, the existence of a well-defined ontological foundation is assumed. The foundation provides symbols and propositions, and the issue addressed there is how to reason about these symbols to determine context. While the first approach is attractive because it does not impose any restrictions on the representation model, it often requires some time to bootstrap, and might not always be able to approximate context to a desired level of accuracy. The symbolic/knowledge based approach, allows for the representation of semantics and thus provides the ability to formulate expressive queries and will generally yield more accurate results. However, it does require a representation model that is currently not supported by the Web, and for its adoption, a considerable amount of document re-authoring will be required.

Though certain factors that contribute to the definition of context, such as the history of users reflecting what they've seen as well as what they've liked/disliked, and a document's context presented by simple keywords, have been examined, other elements that can prove useful for the establishment of context, have been mostly neglected. Examples of these include the task for which a query is being issued (learning, researching, purchasing, etc.), the role of the user, the device the user is using and other document properties (e.g. part of a collection, or metadata such as the name of the author).

In chapter 5, this work's approach to context is described. In the next chapter, some considerations underlying the design of the adopted framework are presented. A number of recommendation and information finding building blocks, are also identified.

Chapter 4: Design Considerations and Building Blocks

4.1 Overview

This chapter outlines the design requirements imposed by the goals of the work first presented in section 1.3, and how they can be fulfilled. The overall aim of the work is to assist users in their navigation and information finding activities using ideas from open hypermedia while explicitly supporting the notion of context. The steps needed to achieve this can be briefly summarised as follows:

1. Develop a multiagent architecture to support agents in a way that enables them to exchange information that can assist each in the achievement of their respective tasks. The justification for using a multiagent framework is provided in section 4.2. Issues related to the creation of such a framework are also discussed in that section.
2. Identify building blocks for modelling user activities so as to promote knowledge exchange between various agents and to enable recommendation and retrieval tasks. The building blocks are identified in section 4.3. A simple model for them is provided in the same section.
3. Extend the open hypermedia model of a generic link to enable 'linking in context'. In this work context is defined on the user and document level through user interests and an abstraction of document context respectively. This is discussed in the next chapter.

4. Design a demonstrator system with a set of agents that can support the ideas put forth by this work. The design and implementation of the individual agents is presented in chapters 6 and 7.

4.2 The Framework

In designing an environment that supports users in their information finding activities within a collaborative setting, one of the main objectives is to provide a flexible, open, and extensible environment where various distributed entities or agents can communicate and collaborate in a robust way. The ability to add and remove agents dynamically within such an environment is also an important requirement. So, the following constraints had to be addressed by the design:

- **Distribution:** The framework should allow for various agents residing across networks of heterogeneous computers to communicate together transparently.
- **Separation:** The functionality of individual agents should be separated from the communication/agent infrastructure in which they operate.
- **Openness:** The framework has to allow for new, possibly heterogeneous entities, to integrate easily into the system as it is expected that the system will evolve over time.
- **Robustness:** Failure of one or more agents within the system should never affect the operation of other agents except in the capacity of the temporary suspension of services these agents provide. Individual agents must also be capable of graceful recovery in case of system failure.

Multiagent systems allow for the incorporation of these requirements while providing other advantages. A multiagent system is composed of a group of agents that are autonomous or semiautonomous and which interact or work together to perform some tasks or to achieve some goals. Multiagent systems have been identified as essential for the successful engineering of complex or large systems. The agents in such systems may either be homogeneous or heterogeneous and they may have common goals or goals that are distinct (Lesser, 1995). The design of individual agents within a multiagent system has the advantage of being independent of the design of other agents as long as each agent abides by an agreed

upon protocol and ontology. This significantly contributes to the breakdown of complexity (Nwana, 1996). Agents are thus viewed as black boxes whose operations are abstracted to the services they provide, and which they announce to a facilitator/broker/matchmaker agent. Agents in this particular system can be viewed as information producers and consumers as will be described in detail later.

Agency in multiagent systems can be defined on a mental or a social level. Mental agency models an agent's activities in terms of mental states described by beliefs, desires, and intentions. Beliefs represent what an agent conceives about the world; desires represent the goal states of an agent, and intentions represent goals an agent has elected to follow. Social agency simply regards agents as communicating entities that exercise some degree of autonomy (Singh, 1998). Mental agency is more grounded in AI and has been referred to as a strong model of agency while weak agency was ascribed to social agency (Wooldridge and Jennings, 1995). To comply with weak agency however, agents also have to exhibit reactivity and pro-activity (section 2.2.1). The model adopted leans more towards social agency.

4.2.1 Achieving Interoperability

The proposed architecture allows for the dynamic addition of heterogeneous agents. This provides flexibility as agents can be implemented in languages that are best suited for their required tasks. As agents have to interact, this gives rise to the question of how to achieve this in a flexible way. Agent communication languages (ACLs) have been recognised as a means for achieving interoperation (Genesereth and Ketchpel, 1994) Standardisation of interoperation protocols remains an active research area within the agent community. A number of issues are involved in the achievement of interoperation. Message format, content and semantics as well as message transport, are but a few. In recent years, two agent communication languages emerged and have been competing to become ***the*** standard ACL. The Knowledge Query and Manipulation Language (KQML) (Finin et al., 1994; Labrou and Finin, 1997) which is an outcome of ARPA's knowledge sharing effort is one, while the Foundation for Intelligent Physical Agents (FIPA) (FIPA, 1996) is another.

KQML (Finin et al., 1994; Labrou and Finin, 1997) is a language that was mainly developed in an attempt to standardise communication between knowledge-based systems so that they could share their knowledge during run time. KQML is suitable for communicating attitudes about information such as querying, requiring, achieving, subscribing and offering, etc, since it provides a context through which agents can view each other's capabilities. Within this context, each agent appears as if it manages a knowledge base (KB) and it is with respect to that KB that communication is based. Such communication may be in the form of queries that question what a KB contains, or requests to add or delete statements from the KB, etc. The implementation of an agent does not however need to be structured as a KB. It could be a simple database system, or a program using special data structures, as long as there exists a wrapper code that translates the representation into a knowledge based abstraction for the benefit of other agents. So it could be stated that each agent manages a virtual knowledge base (VKB).

KQML is a three layered language with the three layers being: content, message and communication. The content layer contains an expression formulated using a language that represents the knowledge to be conveyed. Though the Knowledge Interchange Format (KIF) (Genesereth and Fikes, 1992) has been used frequently in conjunction with KQML to represent content, the language used to formulate the content expression is unimportant to KQML, which makes it indifferent to the content format. The message layer adds a set of features that describe the content. These features include the language that the content is expressed in, the ontology it uses, and the type of speech act represented (query, assertion, etc.). Finally, there is the communication layer which adds a set of low level communication features that include the identity of the sender and receiver and whether communication is meant to be synchronous or asynchronous. These aspects of KQML are captured though KQML messages. KQML messages are called performatives since the messages are intended to perform some action when being sent. Performatives are usually expressed as ASCII strings (Finin et al., 1992). A complete list of reserved performatives is available from (Finin et al., 1994). Transport of KQML messages

has been achieved using various mediums such as SMTP, CORBA, TCP/IP sockets, and HTTP. A typical performative has the following format:

<performative name >
: sender <word>
: receiver <word>
: content <expression>
: language <word>
: ontology <word>
: reply-with <expression>
: in-reply-to <expression>

FIPA is an ACL that has developed significantly over the past couple of years and in which many research efforts are still being invested (FIPA, 1996). Like KQML, FIPA employs speech acts and is also a layered declarative language. However, FIPA attaches more semantics to its speech acts and has developed into a richer and more complex language than KQML. Recently, FIPA has also addressed involved agent-related issues such as mobility. Since neither KQML, nor FIPA have been adopted as the ACL standard, the question of which one to use, is one of preference. When this work was undertaken, FIPA was still in its early stages while KQML was quite widely used. For that reason, KQML has been adopted.

The architecture employs an asynchronous, advertise-notify communication model, where agents advertise their capabilities and are notified of the capabilities of other agents as soon as they register in the system or as soon as other agents advertise their capabilities. The underlying assumption in this work is that agents within the system are benevolent and honest. As such all agents are trusted. Trust and security are active research areas that fall outside the scope of this work.

4.2.2 Transport

TCP/IP sockets were chosen as the transport medium. Although socket based communication imposes the construction of coders, and decoders on the sender and receiver sides respectively, they provide a very flexible mechanism for the exchange of ASCII based strings between heterogeneous agents as they are supported by most programming languages.

4.2.3 Message Exchange

The adopted message exchange model is an asynchronous one. Two types of messaging modes have been identified as important within the system. Persistent messaging, and one-off time-out messaging. The former is used in cases where messages must eventually get through to their required destinations, and as such must not be hindered by temporary communication breakdown. An example of such a case is when a message is sent to register with an agent service. Another example is that of messages sent by agents to advertise their services. The latter is employed in cases where an answer must be received within a given duration of time. An example of this is when a user uses an agent to initiate a request for a search service. If one or more of the agents that carry out this service are down, trying to deliver the message to them when they are up again is of little importance since by then, the user may have lost interest in the original query.

4.2.4 Application Specific Requirements

Although the architecture was designed to be generic, the requirements posed by information finding and navigation were ones it had to address appropriately. The framework is meant to serve users in organisations or groups where it is likely that user interests will overlap and that the experience of one user could be of benefit to others. As such, the framework has both collaborative and personal dimensions. The collaborative aspect derives from attempting to capture information about what users have found useful when carrying out their normal browsing activities. The personal dimension derives from the way the captured information is automatically propagated to users of the system, and which is highly dependent on the individual interests of the users. Captured information can also be used to provide various on-demand services like document recommendations.

The personal dimension can be achieved through the use of personal interface agents that act on behalf of their users. The collaborative dimension could be realised by enabling the various personal agents to provide other agents with information that these agents can use to infer or create information that can be

propagated back to other users through their personal agents, or used to benefit other users through other interfaces.

So, to promote openness and collaboration, services that can be of use to a wide spectrum of users or agents are to be implemented by agents that advertise their capabilities to perform these services in the framework. Each user in the system is to be represented by an agent that would act on the user's behalf and provide him/her with a means of utilising the services of other agents in the system. Such an agent should be capable of monitoring one or more of the user's activities such as the user's browsing, bookmarking, and searching activities and should be capable of representing and maintaining this type of information in its virtual or real knowledge base. The agent can in turn pass this information to other interested agents.

Some of the constraints that this work attempts to address, are summarised as follows:

- Make no assumptions about the needs of the users, but adapt to their needs:
  In real-life dynamic environments, the information needs of users are changing all the time. For certain domains and in certain settings, modelling of information can be very beneficial. However manual information modelling, which is often a time consuming and expensive process, can not catch up with the changing information needs of users with changing interests.
- Place minimal demands on users of the system:
  It has been noted that an agent that requires explicit instructions or the constant provision with information, is an agent that gets in the way of its user more than help him/her (Lieberman, 1997).
- Enable interaction with the system:
  Although users are to be provided with agents that can proactively provide them with recommendation and navigation aids (through interaction with other agents), a user's agent should provide an interface whereby the user can interact with the system and demand recommendations or enter queries.

This could be particularly useful in cases where the user requires information or recommendations that the user's agent is not likely to be able to anticipate, but which the system can still provide.

- Enable non-intrusive proactive recommendations and navigation hints:
  'In context' recommendations and navigation aids have to presented to the user in an accessible, fast, yet non-intrusive way. Otherwise, it is not likely that the system will be used.
- Bootstrap rapidly:
  A common problem with systems that adapt to user interests as well as with collaborative ones, is that it can take a long time for them to obtain enough data to be able to build a profile or make recommendations (Maes, 1994). Since this can easily alienate users, it is important to avoid this pitfall.

4.3 Basic Information Finding and Recommendation Building Blocks

Most applications that have addressed the problem of information finding on a personal level, as well as collaborative recommendation services, have employed a user activity of some type either to build a personal profile or to simply achieve the functionality of recommendation. The various activities that have been utilised in different systems are briefly outlined in the following subsections, as well as the benefits they yield and the disadvantages associated with them. Abstracting a representation for these entities facilitates their re-use across applications. For that reason, following the discussion for each of these entities, a proposed representation that can be used within the system, is given. The proposed representation is a rough and simple one, with many details omitted and is meant only to serve as a guideline.

4.3.1 The User

In the context of this thesis, all work is geared towards serving a set of users in their information finding and navigation activities. So a user is an important entity within the presented system. A user is a person with a number of static attributes presented in the following table:

Attribute Name	Type	Source of Value	Required
FirstName	String	User	Yes
LastName	String	User	Yes
Email	String	User	Yes
GeneralInterests	String*	User	No
Phone_number	Number	User	No
Fax	Number	User	No
UserID	String	System	Yes
User Interests	Interests	System	No

Table 4.1: A user's static attributes

A user also has *interests*, which is a dynamically derived concept. There are many ways of deriving and representing user interests. The system should not be restricted to just one of those. However agents within the system should be able to know whether or not the representation method is compliant with what they want to do with user interests. So, user interests can be represented by:

Attribute Name	Type	Source of Value	Required
UserID	String	System	Yes
DerivationMethod	String	System	Yes
RepresentationType	String	System	Yes
Value	RepresentationType	System	Yes

Table 4.2: User interests

4.3.2 The Document

The types of documents addressed in this work are Web documents. As such the representation will be based on the information that can be easily collected from existing Web documents. The basic attributes associated with a document concept are given in the following table:

Attribute Name	Type	Source of Value	Required
URI	URI	System	Yes

Attribute Name	Type	Source of Value	Required
Title	String	System	No
LastModified	Date	System	No
MimeType	String	System	No
Keywords	String*	User/System	No
Comment	String	User	No

Table 4.3: Document Attributes

URIs are defined as in (Berners-Lee, 1998). URIs are used here for generality as they provide many advantages over URLs an example of which is persistence. However since URIs are not widely supported, URLs are used in the developed components.

4.3.3 User Trails and Browsing History

The utilisation of user 'trails' for information finding is a relatively old idea first proposed by Bush (Bush, 1945). In the context of Web navigation, a user trail is basically the list of URIs a user follows towards the achievement of a given task. A lot of existing systems make use of history information. In its most passive form, building a history can be totally non-intrusive, making it a valuable way to collect information about/from a user without interrupting him/her. However, building the history can also take on an interactive mode where the system augments knowledge to the history. History utilisation is dependant on the task at hand, but generally speaking it has been recognised that history can provide information as to ***who***, ***what***, ***why***, and ***how***, information was browsed (Wexelblat and Maes, 1999).

Who information refers to who has performed a given activity such as looking at a particular URI, and can be used for example, in finding users with similar interests (Pikrakis et al., 1998) and for making recommendations based on similarities between users.

What information is meant to describe what was browsed. The representation of what was browsed can assist in searching for information as well as for modelling user interests. The representation can be trivial if only keywords are used, or complex when a specialised ontology is utilised.

Why information represents the reason for which a document or a set of documents was browsed, i.e. it represents the task for which documents were being viewed. This can also assist in search when the task at hand is important.

How information, outlines steps taken by a user to achieve a given task. This is particularly relevant when the procedure or the order of looking at things is important, such as with a learning task.

Despite their potential usefulness, the ***why*** and ***how*** types of information can not be captured automatically. They have to be explicitly acquired from users using an established ontology to facilitate their utilisation to the fullest.

Other types of information that can be provided by history, include the number of times a particular user has viewed a given document, and the last time a document has been viewed as these can serve as a rough guideline in the user's interest in the contents. A document in a history list can be represented by a ViewedDocument concept attributes of which are given in the following table:

Attribute Name	Type	Source of Value	Required
Doc	Document	System	Yes
FirstSeen	Date	System	Yes
LastSeen	Date	System	Yes
NumOfVisits	Number	System	Yes

Table 4.4: Attributes for a viewed document

The ***who, why, how and what*** types of information represent relationships rather than concepts. Relationships between the various concepts are discussed in section 4.3.8.

4.3.4 Bookmarks

Bookmarking is a powerful and natural facility for a lot of Web users. In a survey conducted in 1996, it was found that 80% of the 6619 participants use bookmarks for locating information (Abrams et al., 1998). Bookmarks can convey similar information as history. History information collected via bookmarking however, is more informative than simple browsing history. Bookmarks fall under the class of user information that can be collected in a non-intrusive manner. Users usually bookmark documents that they have found useful and which they would want to be able to get back to. As such, bookmarking reflects an explicit interest of a user in a document as well as an indication that the user thought highly of the quality of the document, as most users would not want to populate their bookmark list with junk. In a study carried out in 1996, it was found that there are five main criteria that Web users employ when deciding whether to bookmark a Web page: 1. General usefulness, 2. Quality, 3. Personal interest, 4. Frequency of use, 5. Potential future use (Abrams et al., 1998). This makes the use of bookmarks suitable for a wide variety of tasks. They can be used for user profiling as well as for recommending documents in a collaborative environment. For document recommendation, their use can be simply based on the category in which they appear within a collection of bookmarks. A group of bookmarks under a given category indicates that there is a relationship between those bookmarks reflected by the category (Rucker and Polanco, 1997; El-Beltagy et al., 1999a). The content of the bookmarked pages can also form the basis of recommendations (Delgado et al., 1998). Though bookmarks have been used in some recommendation systems (Rucker and Polanco, 1997; El-Beltagy et al., 1999a; Delgado et al., 1998) they have not been as popularly employed as navigation history. The reason for that can be attributed to the fact that bookmarking is usually performed using the specific browser a user is employing and that unlike navigation history which can be monitored easily by proxy like servers, there is no straightforward way for obtaining user bookmarks. Also, bookmarking activities vary significantly across the Web user base (Abrams et al., 1998).

A bookmark concept can be represented by the following attributes:

Attribute Name	Type	Source of Value	Required
Doc	ViewedDocument	System	Yes
Category	String	System	Yes
CreationDate	Date	System	Yes

Table 4.5: Bookmark attributes

4.3.5 Links

Though links are very important information finding aids, they have not been recognised as a recommendation building block, even though every time someone publishes a Web page, any links authored in that page are in fact recommendations made by the author of the page to its viewer. A notable exception to this is the Google search engine, which recognises that every time someone creates a link to a Web page, that author is voting positively for that Web page. Google takes that into consideration when assigning weights to Web pages. Links embedded in pages that get higher votes also get higher weights (Google, 2001). However, Google is a search engine that uses the links in that way to supplement its ranking technique rather than to make recommendations. Making use of authored links to pro-actively offer recommendations is not something that has been explored. The difficulty in re-using links arises from the fact that they are embedded in Web pages in which they were authored and which define their context. MEMOIR (Pikrakis et al., 1998), which embraced DLS (Carr et al., 1998a) like functionality, is the only recommender system known to the author which has attempted to a certain extent, to use links to make recommendations. In MEMOIR this was achieved by allowing users to author links and to share those with other users. However, MEMIOR employed DLS like technology and thus suffered from the same limitations as the DLS. Namely, It had no disciplined way of automatically creating or rendering links in context. It also placed the responsibility of link creation on the users of the system. In this work, links are considered important recommendation entities. A link is presented by the following:

Attribute Name	Type	Source of Value	Required
Source node	URI	User/System	No

Attribute Name	Type	Source of Value	Required
Source Anchor	String	User/System	Yes
Label	String	User/System	Yes
Destination Anchor	URI	User/System	No
ContextID	String	System	No

Table 4.6: Hyperlink attributes

4.3.6 Search History

Searching for information is usually carried out with a specific question in mind and with relation to a specific task. In a group setting, the experience of one user searching for documents that fit a given criteria, can be propagated to other users in the system if enough information is acquired from that user. The documents that satisfied a given query, and the task associated with that query, are examples of information that can stored and reused by other users. They can also be used to model a user's interests. If a query is entered in natural language, then the query itself can be used to mark up documents that satisfied the query and this can also be used to inform other users of the contents. If a task is to be associated with a query, then a task ontology will have to exist.

Assuming the query mode is the simple one currently employed in the Web, a concept *SearchResult* associated with a document retrieved in response to a query for which a user has provided feedback, is represented by the following attributes:

Attribute Name	Type	Source of Value	Required
KeyQuery	String	User	Yes
ResultDoc	URI	System	Yes
Relevant	Boolean	User	Yes
NL Query	String	User	No

Table 4.7: Attributes of a search result

4.3.7 Ratings

Ratings have been a popular way of getting explicit feedback from users about how much they liked some item or document. They have been used for building user profiles, for matching user interests, and for achieving recommendation functionality (Chen and Sycara, 1998; Balabanovic and Shoham, 1997; Pazzani et al., 1996; Resnick and Varian, 1997). The main drawback associated with ratings, is that rating is not part of a user's normal activities and as such, a system that is dependant on ratings for the achievement of its functionality, has to provide a strong incentive for users to make them. A rating can be a simple integer associated with the item or document through a rating relationship.

4.3.8 Relationships and Services

There are a number of straightforward relationships that exist between the outlined concepts. For simplicity, these are represented using a Prolog like syntax:

1. Bookmarked(User, Bookmark).
2. seen(User, Document, Date). (This provides ***who*** information)
3. Rated(User, Document, Rating).
4. Queried(User, KeyWordQuery).
5. ProvidedFeedBack(User, SearchResult).
6. Created(User, HyperLink).
7. ShownInterestFor(User, Document).

The ***how, why,*** and ***what*** types of information can be described by the following relationships

8. ExplainedHowAchieved(User, Task, Trail).

Where a *Trail* is simply a list of URLs, and *Task* is some task represented by a Task Ontology.

9. StatedWhyViewed(User, Document, Reason). (Reason is a justification, from a Justification Ontology).
10. Represents(Document, SomeConcept) (SomeConcept is a concept chosen from a some Representation Ontology)

Relations 8,9, and 10 have not been addressed by the demonstrator system.

There are also sets of obvious services that can be provided based on the outlined relationships. These are:

1. WhoHasSeenThis(+Document, ?ListOfUsers).
2. WhoHasBookmarkedThis(+Document, ?ListOfUsers).
3. RecommendRelatedDocuments(+Document, ?ListOfDocuments).
4. RecommendRelatedDocuments(+AListOfDocuments, ?ListOfDocuments).
5. RetrieveSimilarDocuments(+Document, ?ListOfDocuments).
6. GetAverageRatingFor(+Document, ?Rating).
7. GetCommentsFor(+Document, ?ListOfComments).

4.4 Summary

This chapter outlined the different requirements imposed by the goals of this work as well as discussed how requirements related to the framework, can be achieved. One of the overall requirements entailed the identification of basic entities that can be used to guide the tasks of information finding and recommendation propagation. These entities were discussed and a number of services revolving around them, were briefly outlined.

One of the steps needed to achieve the overall goals of this work is to extend the open hypermedia model of a generic link to enable 'linking in context' as a way of offering recommendations. This idea is discussed in the next chapter.

Chapter 5: Linking in Context

One of the aims of this work is to investigate the use of linking in context as a way of assisting users in their information finding activities. This specifically targets a failing associated with traditional information retrieval models which is attributed to the isolation of these systems from the context in which queries are made (Budzik, 2000). Context is an involved issue, but one conclusion that can be drawn from the review presented on context in chapter 3 is that addressing the context problem itself is very contextual. It depends on the task at hand and the available variables that can be modelled in relation to that task. In the case of 'linking in context' the primary entities involved in this particular task, are those of the user and the document. There are many factors that can affect the context of the user. These include the user's role in an organisation/group/etc, physical location, level of expertise in various topics, browsing history, interests, etc. The context of a document can be defined in many different ways such as by its content, its format (html, pdf, gif, etc), its purpose, the date it was created, the server on which it resides on, its download speed, etc.

If a user's information needs are to be anticipated in advance based on local context and links are to be dynamically added to Web pages that meet a user's interests, then this could lead the user to find information that meets his/her information needs while he/she is browsing. It might also alleviate the need to consult an external search facility. These requirements are very much in line with those of

just-in-time information retrieval agents (Rhodes and Maes, 2000) and just-in-time access systems (Budzik, 2000).

Ideally, most contextual factors affecting the task of 'linking in context' would be taken into consideration. However, this could be an involved and complex task. Following the software engineering premise that a "complex system designed from scratch never works" and that a complex system is easier to evolve from a simple one (Booch, 1994, pg. 13), the idea is to build a simple, and extensible model for linking in context, as well as assess its utility. In our model two factors control linking context: the interests of a user, and the contents of the document within which the links are to be rendered. Assuming that there is a pool of links stored by some agent, then user interests define which links are to be exported for that particular user, and a document's content defines which of these links are to be rendered in that document.

Designing a 'linking in context' model raises a number of questions the most important of which are:

1. How will links be created in the system? Will they be automatically generated, or manually created? This is discussed in section 5.1
2. How will link context be represented? This is discussed in section 5.2
3. How will links be rendered to documents 'in context'? This is discussed in section 5.3
4. Where will links be stored and how will they be propagated to users according to their interests? This is briefly discussed in section 5.3 and in more detail in the next chapter
5. How can the proposed model be generalised? This is discussed in section 5.3.1

This chapter is only meant to provide an overview of issues related to linking in context. Practical and implementation details are covered in the next chapter.

5.1 Creating Hyperlinks

Since the proposed model makes no pre-defined assumptions about the information needs of its users, anticipating and manually authoring hyperlinks, which is in general an expensive and inefficient process, is simply not a feasible option. Instead links have to be automatically generated based on the collective interests of the users of the system.

Generating automatic links is a task that has been investigated by the hypermedia community and is one which has been recognised as challenging since certain relationships between documents can sometimes only be inferred and tagged by humans (Wilkinson and Smeaton, 1999; Allan, 1996). This is particularly true if there is a requirement to automatically create links to concepts represented by phrases. This task implies that there is a requirement to understand something about a phrase, which is to become a source anchor, and the document to which it will link. Specifically, it raises the question as to how to create links for concepts within a large body of text, when there is little understanding of the text itself.

Rather than attempt to automatically generate links from scratch, this work looks into a way of re-using existing ones. The Web is full of a wealth of handmade links that can be used to generate links independently of the documents in which they were authored and re-applied again in contexts similar to those in which they were originally created. The underlying premise is that if document X and document Y appear in context Z, and there is a link related to a concept C in document X, then the same link can be applied to concept C in document Y. This applies to all documents in context Z. So representing context is an important step in the process of extracting and creating links.

The proposed model allows for source anchors, represented by extracted phrases, to have multiple source nodes and multiple destinations. The phrases for which links would be created will usually denote concepts. This gives rise to a problem. Links in Web pages can often have long source anchors, which hide the particular concept to which they relate. For example, if a link appears in a Web page with the

text *'Miscellaneous concerning Vannevar Bush'*, then a link extraction algorithm should be able to understand that the concept to generalise this link for is *Vannevar Bush*. i.e., the source anchor for the link to be created using that link, would be *Vannevar Bush*. In the algorithm implemented to create links and which is presented in the next chapter, this is catered for.

5.2 Representing Context

A number of methods have been developed for using the content of unstructured information resources for the construction of user or filtering models. In these models the capture of user context or document context for the achievement of a specific task is one of the goals. Depending on the specific task at hand, a number of techniques have been employed to build such models or profiles. Examples of techniques employed by Web agents to learn or capture a document or user profile include Decision trees, Neural Nets, Bayesian classifiers, Nearest Neighbour and TF-IDF (Mladenic, 1999).

TF-IDF (term frequency, inverse document frequency) is a very well studied and widely used information retrieval technique (Salton and McGill, 1983). The technique is used to derive weights for terms in a way that would reflect their importance in a given document. TF-IDF is based on the vector space model where a vector is used to represent a document or a query. The cosine angle between different document vectors is a measure of how similar the documents are, and is used as a similarity function. Used in conjunction with a similarity function, TF-IDF can be employed to distinguish between documents. The model has been used successfully for document ranking, document filtering, document clustering, and as the basis for relevance feedback (Baeza-Yates and Ribeiro-Neto, 1999). One of the advantages of using the TF-IDF method is that unlike many machine learning algorithms, it does not require large training data sets in order to distinguish between various documents. By representing a document through a vector space model computed via TF-IDF, comparing a document to other documents or queries is simply achieved through the application of a similarity function. The technique has therefore been employed by a large number of Web assistants examples of

which are: FAB (Balabanovic and Shoham, 1997), WebMate (Chen and Sycara, 1998), and Margin Notes (Rhodes, 2000), all described in chapter 2.

Using TF-IDF in conjunction with the cosine similarity function seemed well suited for our context representation task. The idea was to abstract a context via a cluster centroid built as a result of grouping similar documents together where a cluster centroid serves as a representative of features in a given cluster. Typically, a cluster centroid is calculated by averaging vectors of all documents in a given cluster (Salton and McGill, 1983). TF-IDF is calculated based on the following:

Let N be the total number of documents

Let n_j be the number of documents in which a term k_j occurs

Let $tf_{i,j}$ be the term frequency of term k_j in document d_i

Let $w_{i,j}$ be the weight of term k_j in document d_i

Let idf_j be the inverse document frequency of term k_j

$$idf_j = log_2(\frac{N}{n_j})$$

$$w_{i,j} = tf_{i,j} \times idf_i$$

The weight $w_{i,j}$ provides a measure of the importance of a given word, in a given document. When comparing two vectors, it is important that calculated weights are normalised. A weight $w_{i,j}$ is normalised by applying the following equation:

$$w'_{i,j} = \frac{w_{i,j}}{\sqrt{\sum_{j=1}^{len} (w_{i,j})^2}}$$

len, is the length of the vector representing document d_i

The cosine similarity function measures the similarity between two documents, or between a query and a document, and is calculated by the following equation:

$$\text{Sim}(d_i, d_l) = \frac{\sum_{j=1}^{t} w_{i,j} \times w_{l,j}}{\sqrt{\sum_{j=1}^{t} (w_{i,j})^2 \times \sum_{j=1}^{t} (w_{l,j})^2}}$$

Equation 1: The cosine correlation co-efficent

The result of this equation is a value that ranges from 0 (no similarity) to 1 (identical).

5.2.1 Experimenting with TF-IDF

It was important to test that using TF-IDF to abstract a context representation via cluster centroids would work as anticipated. For that purpose the algorithm was implemented in Java. A dictionary mapping terms to their frequency was constructed using 2559 Web documents covering various, random subjects. The documents were downloaded, any stop words they contained were ignored, and other words were weakly stemmed and stored in the dictionary. Weak stemming was carried out by only employing the first step of the Porter stemming algorithm (Porter, 1980). The resulting dictionary contained approximately fifty thousand terms.

The vector representation referred to in traditional information retrieval books is a theoretical one in which each document in the system is represented by a vector of length t, where t is the total number of terms in the used dictionary. The weight of any word that is in the dictionary and not in the document is set to 0. For the purposes of implementation, this is not a practical representation model. In this work, each document is represented by a feature vector profile of a fixed length t' which is usually a very small fraction of the total number of words in the used dictionary. The profile is populated by capturing a document's highest valued t' weights and their associated terms and discarding all others. When calculating the similarity between two documents, only the terms used in those documents are considered. If a term is present in one document and not the other, then the weight of that term is automatically considered 0. The frequency of a word appearing in a document and not in the dictionary is set to 1. This facilitates a simple and fast implementation model while achieving the goals of the original theoretical representation.

To abstract context representation through document clustering, the following algorithm was applied to incoming document vector representations calculated using TF-IDF:

Let $\boldsymbol{C}$ be the set of available context abstractions (centroids) represented by a feature vector of terms (initially {}), and c_i is an element in $\boldsymbol{C}$, where i $\in$ {1,2,.. $|\boldsymbol{C}|$}

Let $\boldsymbol{UD}$ be the set of un-grouped documents, and d_i be an element in UD, where i $\in$ {1,2,.. $|\boldsymbol{UD}|$}

Let *inDoc* be the feature vector of terms representing an incoming document.

1. For each c_i calculate Sim($c_{i,}$ *inDoc*) using the cosine similarity function
2. If maximum similarity across $\boldsymbol{C}$ is greater than a threshold value μ then, add and average weights in inDoc to the vector centroid c_j with greatest similarity. Sort the weights for the new vector centroid, trim to maxLength (t'), and store results in c_j
3. else, repeat step 2 across $\boldsymbol{UD}$. If a document d_i is found, then create a new centroid with the resulting vector and remove d_i from $\boldsymbol{UD}$
4. else, add inDoc to $\boldsymbol{UD}$

A number of documents including two small document sets relating to two different subject areas that have some overlap in their constituent terms, were used to test the algorithm. The document sets covered topics related to Vannevar Bush[1] and Kate Bush[2]. When testing the algorithm, a number of documents related to Vannevar Bush were considered similar to some documents related to Kate Bush because some of the highest weighted words in both documents were similar. As such, the algorithm was found to be incapable of distinguishing between documents representing different concepts in cases where a few of the highest ranking words are similar. In this work, this mix up was altogether unacceptable as Vannevar Bush and Kate Bush represent two different contexts or entities, and they had to be recognised as such.

---

[1] A visionary and scientist who in 1945 wrote the paper "As We May Think" that inspired hypertext research- http://www.isg.sfu.ca/~duchier/misc/vbush/

To overcome this problem and enable proper determination of context, a very simple heuristic was applied in conjunction with the cosine similarity function. The heuristic was based on the observation that the profiles of documents that are similar have many terms in common, while those that are not actually similar do not share more than two or three terms. So, in order to determine whether two documents were in fact similar, the following rule was applied:

Let v_i be the vector representation for document d_i and t_i be the set of terms in v_i (the same applies if this is a centroid rather than a document)

Let v_j be the vector representation for document d_j and t_j be the set of term in v_j

$$\text{Similar}(d_i, d_j) \Leftrightarrow \quad \text{Sim}(v_i, v_j) > \mu \quad \text{and}$$
$$|t_i \cap t_j| > \beta$$

Where $\text{Sim}(v_i, v_j)$ is the cosine similarity function, μ is the threshold value below which documents can not be similar and β is the minimum number of words that must exist in common between the two feature vectors being compared if they are to be considered similar.

Another modification was applied to the way cluster centroids are constructed. To better capture the context of a group of documents, it is important to emphasise the similarities between them. So, as opposed to averaging the weights for various terms in the construction of a centroid, a boosting factor was introduced. For each term t_k where $t_k \in |t_i \cap t_j|$ and $d_i w_k$ is the weight of term t_k in document d_i, $c_i w_k = \max(d_i w_k, d_j w_k) * \alpha$ where α is the boosting factor. In the current implementation, α is set to 1.5.

5.2.2 Evaluation of the Clustering Algorithm

A data set of 196 documents was used to test the algorithm. Out of these, 185 were randomly selected from a pool of diverse documents. The remaining 11 documents, comprised two small manually selected document sets, the contents of which were confused with each other before the addition of the presented

---

[2] A very successful and popular English vocalist, especially during the 1970s and 80s

heuristics. One set contained seven documents relating to Vannevar Bush, while the other had four, which related to Kate Bush. Over the entire data set, 28 clusters were created with 79 documents. Of the seven documents relating to Vannevar Bush, five formed a cluster while the other two, which were content-poor, remained unclustered. All four Kate Bush documents formed another cluster. Since the algorithm emphasised accuracy, a few documents that could have fitted into some of the clusters were filtered out, but that was in line with our requirements. The clustering accuracy was 98.7% because a single document was mis-clustered. This was measured based on manual examination of the clusters (please see Appendix A). When the single mis-clustered document was examined, it was concluded that it would have been very difficult to distinguish it from other documents with which it was placed without proper understanding of the document. The page was one containing a list of ACM publications and was placed in a cluster in which most documents covered concepts related to metadata. After every ACM publication, a link to its metadata is provided. The fact that this particular page shared in common many other terms with concepts related to metadata, made it very difficult for the algorithm to distinguish it from metadata documents.

5.3 Extending the Open Hypermedia Generic Link Model

In the context of this work, 'linking in context' builds on ideas from open hypermedia, and specifically from the DLS, which as noted in chapter 2 is one of the systems that brought the open hypermedia philosophy to the Web but did not address context appropriately. In the DLS model, the existence of multiple linkbases allows users to switch between linkbases depending on their requirements thus placing the burden of context selection on the users. Our aim was to address this by having a model capable of dynamically switching between linkbases according to context.

One of the contributions of open hypermedia has been the separation of links from documents, thus recognising links as information entities in their own right. This calls for an independent representation of a link that allows for it to be rendered

back into documents. The separation of links from documents allows for a degree of flexibility and control not possible when links are embedded in a document. By abstracting links from documents, rules can be used to control when a link is rendered into a document. For example in the generic link model, a document can be a source node for any given link in an active linkbase if the document contains any text matching the source anchor of the link. It is easy to extend this model to include other rules. For example, another constraint could be applied in which links in a linkbase can only be active for documents residing on a given network domain.

Assuming that there is a set of links divided into linkbases or link sets, then in order to extend the generic link model to explicitly address context, some rule should be applied so that any of the linkbases can only be active in a given context. So, in this particular case the source of a link or a set of links can be thought of as being denoted by a context instead of any specific document or source node. The context is resolved during runtime through a context resolution function. In its most general form, a context resolution function (also referred to as a context resolver) is comprised of one or more rules capable of mapping any document to a context if a suitable context for that document exists. If the context resolution function can resolve that a document belongs to a given context, then a link or a set of links for which this context is identified as a source can be automatically activated. Based on the context representation model chosen for this work, a context abstraction is represented by a weight vector of terms and so the context resolution function is simply the similarity function given in section 5.2 (see Equation 1) which can be used to obtain a match between a document profile and a context abstraction.

Though inspired by the DLS, the proposed model differs from the DLS in a number of ways. The architecture of the DLS has evolved over the years, yet the system has remained essentially service orientated with one or more services providing linking functionality to many users. In the proposed model, one or more link providers exist, but the links providers themselves only provide links rather

than play a role in rendering them. In its most general and simple form, a link provider can simply be a link store. As far as other agents in the system are concerned, they can send it a request for links based on an agreed upon link context specification (e.g. a TF-IDF vector) and the link provider will respond with a set of links. How the link provider obtains the links is totally transparent to other agents. A link provider may (or may not) also offer other agents the ability to subscribe to being notified whenever new links are being added in a given context defined by the link context specification. The rendering component is a separate entity, which through communication with the link provider can obtain links and manage them independently in its own linkbase based on its specific task.

In the demonstrator application the rendering component is the user's personal agent (UI agent), which maintains a private linkbase, controlled by the interests of its user and populated through a subscription service to a link provider. The link provider contains a shared pool of links that are populated through analysis of Web pages that have been found interesting by any user in the system and which trigger link generation as will be detailed in the next chapter. As such, link generation and population on both collaborative and personal levels, involves the users' personal agents which are responsible for monitoring the users' activities.

The generated links are each associated with a link group that is in turn associated with a context abstraction, determined based on the context of documents from which the links were extracted. The link provider also has to automatically maintain those links. Because links are automatically generated, every link has to have a life cycle based on its usability, and so do the contexts in which it appears.

Figure 5.1 shows a simple diagram illustrating how the proposed model extends a DLS like model through this work's chosen demonstrator. Because the model is one in which links can have multiple destinations, the dynamic link renderer should add links in the form of encoded queries which when invoked return a set of suitable destinations based on the context of a given document in which the links are added.

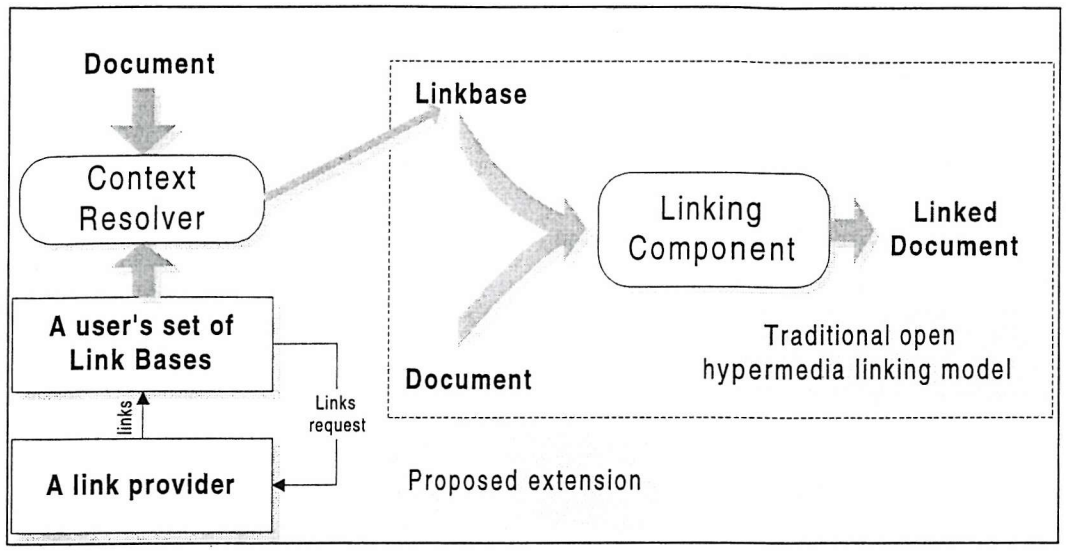

Figure 5.1: An application specific diagram exemplifying the proposed extension to the open hypermedia linking model

The process of rendering links is summarised by the following steps:

1. Determine the context of the document based on a context determination function implemented by a context resolver.
2. Use the context to activate an appropriate linkbase
3. Parse the document and using only knowledge of linkable phrases, add links with encountered phrases as source anchors and encoded queries as link destinations.

5.3.1 Generalising the Model

Though a weight vector of terms has been used to represent context in the proposed model, other factors or representations could be used for the same task. Also, in other applications, a component other than a UI agent may be involved in the process of link rendering and caching using a different strategy for either task. Generalising the linking model presented on an abstract level is quite straightforward. For agents in the system to acquire links to store or process, a link provider should advertise its capability to return links based on one or more contextual factors. Other interested agents in the system should be able to select link providers that fit with their contextual needs.

For rendering links, a link resolver owned by a rendering agent could be thought of as a black box than can take in one or more contextual constraints and using some rules, map that to a pre-established context. In our application the single contextual constraint that the document resolver takes in, as input is a document's TF-IDF vector representation.

The difficulty of actually applying this generalised model is application and implementation dependent. The more contextual factors are taken into consideration, the more sophisticated the context resolver becomes.

5.4 Summary

This chapter addressed questions revolving around the idea of 'linking in context'. Issues addressed included the generation, propagation and utilisation of links in a way that would allow their usage in context. In the next two chapters, the framework and the agents that have been designed and implemented to demonstrate concepts introduced in this chapter as well as in the previous one, are described.

Chapter 6: Design and Implementation

This chapter outlines the design and implementation of a multiagent framework that was developed to support users in their information finding and navigation activities. Within the presented framework, each agent performs a relatively simple task, but collectively, they present the users of the system with a powerful set of tools that can aid in finding and creating information. All developed agents except for the user interface agent, are described in this chapter. The user interface agent plays an instrumental role in utilising the services of other agents for the benefit of its user, but because the description of that agent is quite involved, it is provided separately in the next chapter.

Section 6.1 provides a brief overview of the system and the tools and components developed to provide infrastructure support, while sections 6.2 through 6.5 describe agents developed within the framework.

The work presented in this chapter and the next one has been partly reported to the research community in (El-Beltagy et al., 1999a; El-Beltagy et al., 2000b; El-Beltagy et al., 2000a).

6.1 System Overview and Architecture

Within the presented framework, each user is associated with a personal interface agent that acts on his/her behalf. In addition, a number of agents that support the

user in various tasks exist within this framework. Agents in the presented system can be viewed as information producers and consumers. For example, user interface agents acquire information about what different users are viewing and with the permission of their respective users, share that information with other agents. Any user interface agent in the system can then use the services offered by agents that utilise this information to find out about other users' interests, obtain document recommendations, obtain links, and carry out search, thus enabling the original information to be shared and employed for the benefit of other users. Figure 6.1 shows a simplified diagram of the architecture.

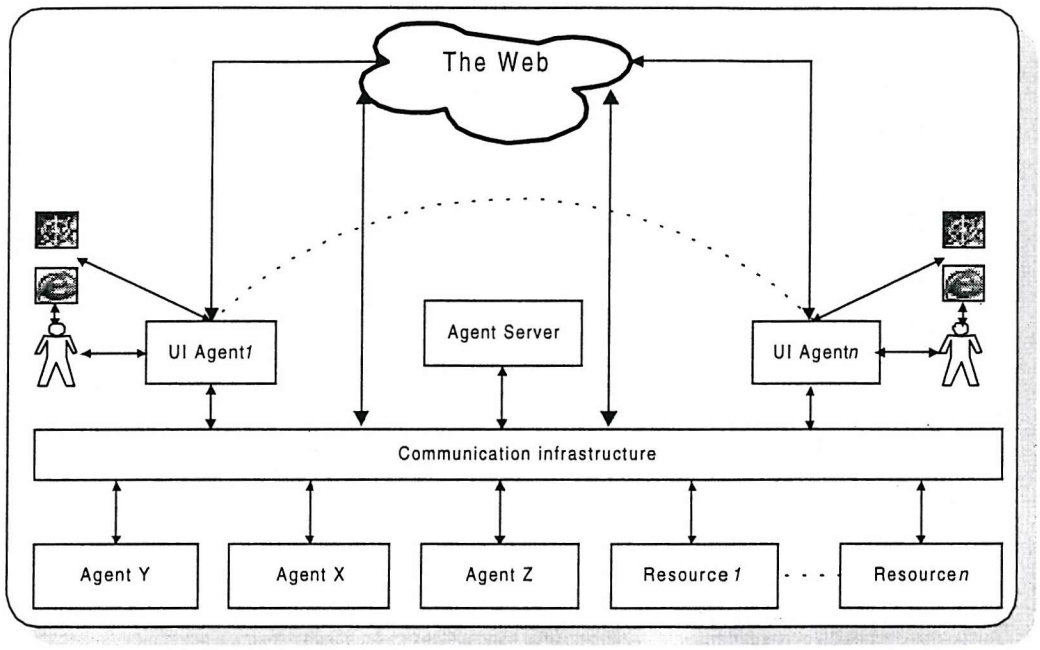

Figure 6.1: A simplified diagram of the architecture

Messages between agents are exchanged in KQML with Prolog or XML acting as the content languages. Communication between agents can be peer-to-peer or can be conducted through a facilitator agent, which is described in section 6.1.2.

6.1.1 Tools

Because of the flexibility Java offers, it has been chosen for the implementation of a set of libraries meant to facilitate the construction of agents and to provide basic

infrastructure support. The main advantages of using Java could be briefly summarised by the following points:

- It is platform independent
- It allows the easy implementation of effective GUIs
- It supports a number of network communication methods. These include low level TCP/IP sockets communication which is the basis of communication within this architecture, as well as higher level remote method invocations, and CORBA, which means that future support for these higher level communication methods can be easily integrated.

If the architecture is to support heterogeneous agents as it is intended to, then all it has to bind the agents to is a protocol rather than to any implementation specific details. It is therefore important to note that the developed tools and libraries were only meant to aid the developers of agents within the framework in implementing Java based agents without restricting the agent development language to Java. Other agents written in any other language could easily integrate with the framework as long as they abide by the communication protocol and the task ontology.

Among the tools developed is a KQML library. The library offers the facility of parsing any string KQML message and converting it into a KQML message object which allows an application easy access to the various fields of the message, examples of which are the sender, the receiver, the content, etc. It provides methods for checking the validity of a message as well as for composing well formed string KQML messages.

One of the design goals was to separate the agent communication layer from agent functionality. To facilitate this, an agent communication layer template was developed. The template provides a basic multithreaded server to handle incoming messages as well as a basic thread, to which messages are passed. In the thread, a method is used to validate the messages and to automatically reply to agents that dispatch invalid ones. Part of the validation process includes checking that the

performative and its content are ones that an agent can handle. In the thread, a method is associated with each performative, so for example, if the incoming performative is tell, then a *'tell'* method is activated with the content of the message passed to that method. A message handler has also been implemented to provide replies received from agents, to objects that requested them. Agent developers can easily extend and modify the template to fit their needs.

For outgoing messages, a flexible and reusable messaging thread was implemented to enable controlled persistent messaging at times when there are communication problems amongst the agents. The messaging thread contains a message queue to which a message is appended upon failure of delivery. The thread sleeps for a given amount of time and when it wakes up, it attempts to deliver all messages in its queue. It then goes to sleep again. Each message is associated with a counter and when message delivery fails for n times, the message is erased from the queue.

6.1.2 The Facilitator

One of the most important components within this framework is the facilitator agent (the agent server) that contributes to the openness and extensibility of the presented architecture and which is independent of any specific application. A facilitator agent within this architecture complies with some of the notions of facilitation as defined by Khedro and Genesereth (Khedro and Genesereth, 1995). Mainly, it supports routing of information, discovery of information, delayed and persistent notification of messages and communication management. When agents come online for the first time, they register with the facilitator agent using the KQML register performative. They then advertise their services to the facilitator, which in turn broadcasts the advertisement to all registered agents. As soon as an agent registers, it also receives all active advertise messages that were previously sent to the facilitator by other registered agents.

The facilitator agent has been implemented in Java. A message to the facilitator agent goes through several stages before being processed. One of the components of the facilitator agent is the message listener, which is a multithreaded Java server

implemented as an extension to the agent communication layer described in the previous subsection. Once a message is received, it is passed on to a KQML message verifier. This component employs the Java KQML library that has been reused across most of the implemented agents. If the verifier fails to validate the message, the appropriate error message is dispatched to the sender. If on the other hand, the message is valid then the facilitator checks to see if the message is addressed to an agent other than itself. If that is the case, it forwards the message to the agent if it is known, otherwise, it replies with an appropriate error message. If the receiver of the message is not specified, then it tries to match the service requested in the content of the message with services advertised by other agents and forwards the message to the appropriate agent accordingly. If the message is addressed to itself, the performative of the message is checked, and a method for that particular performative is activated. If the performative is not supported, a 'sorry' message is dispatched.

Delayed and persistent notification is achieved through the use of a messaging thread.

6.2 The Organisational Memory Agent (OM Agent)

Inspired by the MEMOIR system (Pikrakis et al., 1998), the OM agent serves as a collective memory for the various activities of users within the system examples of which are browsing and bookmarking. In MEMOIR (please refer back to section 2.8.6) trails are key to some important system functions. In this system, bookmarks have replaced trails. For a single user, bookmarks are a powerful facility that can be used to reference documents that a user has found useful and which he/she wants to be able to return to. A collection of bookmarks in a given category denotes a relationship between those bookmarks as indicated by the category. The relationships captured by bookmark categories aren't always easily inferred through content analysis of documents, which makes them a valuable information resource. In a sense, a given bookmark category could be thought of as a trail that has evolved over time rather than from a single browsing session. Bookmarking is a very natural activity for most Web users and unlike the creation of trails that are

often never directly accessible to them again, does not impose an overhead in terms of effort.

6.2.1 Services

Like all agents within this framework, when the OM agent comes on line for the first time it registers with a facilitator and then starts advertising its capabilities. Specifically the agent advertises its interest in being informed about browsing, bookmarking, rating and commenting user activities. The agent allows other agents to register users and then to publish information about their activities. The information it receives is used in a number of ways. By knowing who has seen what, the agent is capable of answering very simple queries in relation to a given URL or document 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, besides the two examples just given, 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 following KQML message, shows one of the advertise messages sent to the facilitator agent (the AgentServer). In this example, the OM agent is advertising its capability to be told that a user denoted by a given email address has seen the given URL.

```
1. (advertise
       :sender om.snoopy
       :receiver AgentServer
       :reply-with ""
       :language KQML
       :ontology kqml-ontology
       :content (tell
                        :sender ""
                        :receiver om.snoopy
                        :reply-with id7889
                        :language Prolog
                        :ontology memoir-ontology
                        :content seen (+URL, +Email) ) )
```

Other ***tell*** messages that the agent is capable of receiving include:

2. ontology: mna, content: person(+Email, +FirstName, +LastName, +URL, +Interests, +Phone, +Fax)
3. ontology: memoir-ontology, content: bookMarked(+Keywords, +Title, +Category, +Url , +Email)
4. ontology: memoir-ontology, content: rated(+Email, +URL, +Rating)
5. ontology: memoir-ontology, content: addAComment(+Email, +URL, +Comm)

The agent also advertises it capability to respond to the KQML ***ask-all*** performative with the following content:

1. ontology: memoir-ontology, content: whoElseHasSeenThis(+URL, +Email, -PersonList)
2. ontology: memoir-ontology, content: whoElseHasBookMarkedThis(+URL, +Email, -PersonList)
3. ontology: memoir-ontology, content: recommendSimilarDocs(+URL, +Email, -UrlList)
4. ontology: memoir-ontology, content: recommendSimilarDocs(+URL, +Keywords, +Email, -UrlList)
5. ontology: memoir-ontology, content: whoIsIntrestedInThis(+topic, +Email, -PersonList)
6. ontology: search-ontology, content: find(+Keywords, - UrlList, 'Prolog')

The agent can respond to an ***ask*** performative with the following content:

1. ontology: memoir-ontology, content: getRatingFor(+URL, -AverageRating, -NoOfVotes)
2. ontology: memoir-ontology, content: getCommentsFor(+Url, -Comments, 'html')

There are a number of Ontologies used across the various agents. Each of these Ontologies defines basic entities and services related to a specific task. These are summarised as follows:

1. A search ontology, containing search constructs.
2. A memoir ontology, containing entities and services that were inspired by the MEMOIR system such as those shown above.
3. A link ontology, containing linking constructs.

6.2.2 Implementation

The algorithms used to infer the answers to incoming 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 in which

this URL appears, are fetched. Ranking of the returned results is based on the duplication of each URL in other categories. None of the URLs are discarded. When emphasis is placed 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.

```
private void askAll(KQMLmessage msg) {

        String answer = null;
        try {
                String service = getService(msg.getContent());
                if (service.equalsIgnoreCase("seenDoc"))
                        answer = parent.seenDoc(msg.getContent());
                else if (service.equalsIgnoreCase("relevantDocs"))
                        answer = parent.relevantDocs(msg.getContent());
                else if (service.equalsIgnoreCase("bookmarked"))
                        answer = parent.bookmarked(msg.getContent());
                else if (service.equalsIgnoreCase("findDocs"))
                        answer = parent.findDocs(msg.getContent());

                if (answer == null) {
                        String m = KQMLmessage.error(parent.name,
                                msg.getSender(), msg.getReplyWithID(), "");
                        sendMsg(m, parent.agentServerHost,
                                parent.agentServerPort);
                        return;
                }
                String m = KQMLmessage.composeMsg(KQMLmessage.TELL,
                        answer, "", msg.getReplyWithID(),
                        MEMOIRontology.PROLOG, "mna",
                       msg.getSender(), parent.name);
                sendMsg(m, parent.agentServerHost,
                        parent.agentServerPort);
                return;
        }catch (Exception e) {
                String m = KQMLmessage.error(parent.name,
                                                msg.getSender(),
                                                msg.getReplyWithID(), "");
                sendMsg(m, parent.agentServerHost,
                        parent.agentServerPort);
                return;
        }

}
```

Figure 6.2: A Java code fragment showing how the OM agent handles messages with an ask-all performative

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 interface of the OM agent with other agents has been implemented in Java. The Java component simply acts as the agent's communication layer. Among other things, it validates any incoming messages before passing their content on to the Prolog component. Figure 6.2 shows a piece of Java code, which handles the *ask-all* performative. Other performatives supported by this agent are handled in a similar manner.

The OM agent advertises its ability to be told about various user activities. In KQML terms, it advertises that other agents could insert facts into its virtual knowledge base through the *tell* performative. In reality, other agents can assert those facts into its real knowledge base. The following facts are examples of possible assertions.

- `person(wh@ecs.soton.ac.uk,'Wendy','Hall','http://www.ecs.soton.ac.uk/~wh', ['Multimedia', 'Open hypermedia', 'Hypertext'], '4060', '9090').`
- `seen('http://www.iam.ecs.soton.ac.uk', wh@ecs.soton.ac.uk).`
- `bookmarked(['agents', 'software agents', 'intelligent agents', 'intelligent software agents', 'kqml', 'kif', 'knowledge sharing', 'mobile', 'softbot', 'fipa', 'acl', 'ontology'],'Software Agents', 'UMBC Agent Web', 'http://www.cs.umbc.edu/agents/', wh@ecs.soton.ac.uk).`

These facts are asserted in a persistent data store to ensure that the information they represent will not be lost in case of failure of any sort. Prolog methods by which queries could be answered and inferenced are also part of the knowledge base. Figure 6.3 shows part of the Prolog code used to answer two queries that can be directed to the OM agent.

```
whoElseHasSeenThis(URL, CurrentUser, Users):-
db_fetch(seenAll(All, URL), _),
      delete(All,CurrentUser,U),
      process(U,Users).

RecommendRelatedDocs(URL, Related) :-
      db_fetch(bookmark(_,_, C,URL,_),_),
      findall(Bookmrk,((db_fetch(bookmark(_,_,C,Bookmrk,_),_),
      not(Bookmrk=URL))),Result),
      order(Result, O_Results),
      remove_duplicates(O_Results,Related),!.
      recommendRelatedDocs (_,[]).
```

Figure 6.3: A Prolog code fragment used for answering two of the queries that can be directed to the OM agent

6.3 The Link Extraction and Contextulizer Agent (QuicLinks Agent)

The primary responsibility of the QuicLinks agent is to generate links based on the information needs of a community of users. The QuicLinks agent, is based on the open hypermedia philosophy of abstracting links from documents. However, there are a number of features that make this agent quite different from agents or applications that have attempted the same goal. For example, link creation and maintenance is an automatic and dynamic process. The agent itself is not a proxy and stores links using a slightly different representation than the DLS so as to account for context. The responsibility of rendering the links is outside the scope of the functionality of this agent and falls on the shoulders of agents that use its services.

The agent is capable of compiling contextual information about Web pages, the URLs of which are passed to it by other agents, and of generating links accordingly. It uses this information to answer queries also presented to it by other agents. A query for this agent could be a request for a set of links in a given context represented by a weight vector of terms or simply a search request represented by a set of keywords. The tasks of this agent can be summarised by the following points:

1. Continuous creation of links in context (this is done via link mining, where link mining is defined as the process of extracting links from Web pages for re-use in similar contexts to that from which they were extracted)
2. Storage and maintenance of links in a knowledge base
3. Provision of various link related services to other agents. These have been identified as follows:
   3.1. the facility of subscribing to link updates in a given context
   3.2. the facility of conducting a query in a given context

The agent is implemented in both Java and Prolog.

6.3.1 Services

The *QuicLinks* agent is responsible for providing links to a group of users. Web pages used to create links have to be of reasonably high quality as well as represent an interest to some of the users within the system. The quality of Web pages is very difficult to determine based only on their content. A Web page might have all the right words, but still be totally useless. A good way of achieving both goals is to request notification about Web pages that users have found interesting. If a user bookmarks a Web page, or rates it highly then it is logical to assume that he/she has found that Web page to be an interesting one. Also, if a user visits a Web page fairly frequently, then again this provides an indication that the user is interested in that page. So like the OM agent, the QuicLinks agent also registers its interest to be notified of a user's bookmarking and rating activities, although for this agent, this information is used in a totally different way. The ***tell*** messages the QuicLinks agent advertises as capable of receiving, are the same as the OM's ***tell*** messages numbers 1, 3, and 4 presented in section 6.2.1, reflecting an interest in the browsing, bookmarking and rating activities respectively.

The agent also advertises it capability to respond to the KQML ***ask*** performative with the following content:

1. ontology: link-ontology, content: getLinkFor(+WeightVector, +CID, -Link)
2. ontology: memoir-ontology, content: getSimilar(+WeightVector, -DocList)

3. ontology: search-ontology, content: find (+Keys, -Result, 'XML')

In addition, this agent advertises is ability to accept subscriptions to message 1, which allows agents to be notified whenever new links are generated for the context represented by the incoming weight vector. The weight vector is encoded in XML and so are the returned links. Figure 6.4 shows how a weight vector is represented in XML while Figure 6.5 shows an example an XML encoded link.

6.3.2 Creating the Links

Once a URL is received and determined to be of interest to a user, it is queued for downloading and processing at a time when the load on the agent is minimal (usually overnight). Links from processed pages are extracted. Those that do not demonstrate a direct relationship to page content are thrown out. Others are analysed and associations between text phrases and those links are created and stored in a Prolog knowledge base (KB).

```
- <WeightFvector>
  - <feature>
      <term>agent</term>
      <weight>1.0</weight>
    </feature>
  - <feature>
      <term>technologi</term>
      <weight>0.5699002</weight>
    </feature>
  - <feature>
      <term>intelligent</term>
      <weight>0.53269964</weight>
    </feature>
  - <feature>
      <term>assistant</term>
      <weight>0.5089797</weight>
    </feature>
  + <feature>
  - <feature>
      <term>application</term>
      <weight>0.48642316</weight>
```

Figure 6.4: Part of an XML representation of a weight vector of terms

```
<link>
  <url>http://fistserv.macarthur.uws.edu.au/san/iac/</url>
  <label>1999 AAAI Spring Symposium on Intelligent Agents in Cyberspace</label>
  <cid>cid8762786</cid>
  <sel>Intelligent Agents</sel>
</link>
```

Figure 6.5: XML representation of a link as used in message exchange between agents

A simple rule based algorithm was developed to infer those associations and to utilise them in a way that would allow for the creation of contextualized generic links. The algorithm followed to carry out these steps is as follows:

For each URL u in the processing queue:

1. Connect to u and use http headers to obtain the last modified date for u, lmd(u)
2. if (lmd(u) > slmd(u)) or notSeen(u), where slmd(u) is the stored last modified date, then continue, otherwise process next *url*
3. download u
4. Parse the html of u in order to extract links in the body and keyword information (a TF-IDF vector of weighed terms). Each link is represented by a *destination anchor* and *source anchor link text.* Keyword information is obtained by ignoring stop words, weak stemming other words, calculating the weights for all words using TF-IDF, and then storing the top l words and their weights in a feature vector, where l is the prefixed vector length.
5. add u to a document cluster based on the algorithm presented in section 5.2, the Prolog implementation of which is shown in Figure 6.6.
6. Pre-process links. This is done as follows: For each extracted link,
   - Check if the source anchor text starts with a URL prefix (http://, ftp://, mailto, etc). If it does, delete
   - Check if the source anchor text is made up entirely of Web stop words (next, back, home, etc), If it is, delete
   - Check if the source anchor text contains any words from the calculated weight vector of terms, if not, then throw the link away. Otherwise, keep.

7. For each remaining link, fragment source anchor text into various phrases by considering the text as phrases separated by stop words. For example, for source anchor text *Vannevar Bush and the Memex*, the two phrases *Vannevar Bush*, and *Memex*, will be extracted.
8. For each extracted text fragment, check if any of the words appear in the feature vector of terms. If none of the words appear, then throw the fragment away and process the next fragment or link. If only one word appears, and it is in the top *n* features, or if more than one appears, then create a link as described in step 9. If one words appears, but it is not in the top n words, then use the complete source anchor text and scan backwards and forwards in the Web page text for a phrase that matches a subset of the link text. For example, if the source anchor text is *"As we may think": an article published in...* , and the keyword found is "think", then by this process the phrase to be used for link creation will be "As we may think", provided it has appeared in the document text.
9. If the link's destination anchor is valid, then create the link by asserting it into the Prolog KB. Validity is determined by verifying that the destination link is not dangling. A link is defined by a source URL, a destination URL, the selection it can replace (the new source anchor), the text label associated with the link (this is the un-fragmented version of the original source anchor), and keywords associated with it (this is used for searching).
10. Store the link in the Prolog KB. If the source document from which the links have been extracted has been assigned a context, then the extracted links are assigned the same context. If not, then they are kept unclassified until the source document is assigned a cluster.

```
cluster([]).
cluster([U1| Rest]) :- cluster(U1, Rest, Rem),
   cluster(Rem).

/*Cluster 1st doc and get remaining Docs, and then recursively call
cluster again with remaining docs. U is a Url, V is its feature
vector, Inter is the intersection, ClustSum is the cluster summary,
BMatch is the Best Match*/

cluster(U, R, R) :- db_fetch(profile(U, V),_),
getMaxAcrossClusts(V, CID, Max, Inter),
      Max > maxVal, !, db_fetch(cluster(CID, ClustSum),_),
      addVectors(V, ClustSum,Inter, Result),
      addClustMapping(U, CID),
      trimSummary(CID, Result, R2),
      updateClustSum(CID, R2),
      updateLinkInfo(U,CID), !.

cluster(U, Others, Rem ) :- db_fetch(profile(U, V),_),
      computeMaxP(V, Others, BMatch,0, Max, Inter),
      Max > maxVal, !, db_fetch(profile(BMatch, Pro),_),
      addVectors(V, Pro ,Inter, Result),
      getNewConID(CID),
      addCluster(CID, Result),
      addClustMapping(U, CID),
      addClustMapping(BMatch, CID),
      updateLinkInfo(U,CID),
      updateLinkInfo(BMatch,CID),
      delete(Others, BMatch,Rem), !.

cluster(U, Others, Rem ) :- delete(Others, U, Rem).
addPage(U, no) :- getClusterMapping(U, _),!.
addPage(U, no) :- not(db_fetch(profile(U, _),_)),!.
addPage(U, CID)   :- getUnClustered(Urls),
                        cluster(U, Urls, R, CID),
                        updateUnclust(U, Urls, R, CID),!.
```

Figure 6.6: A code fragment showing part of the Prolog implementation of the clustering algorithm

One of the advantages of using this algorithm is that it enables the generation of links to URLs of documents with little or no text content, but which are still relevant in a given context. For example, after adding links to a *Kate Bush* Web page using mined links, one of the destination anchors associated with the phrase *Red Shoes* (which is a title of a song) was a URL pointing to a playable midi file of the song.

In order to maintain the Prolog KB, the idea of documents, clusters/contexts, and links having a life cycle was introduced. The lifetime of any of these is a function of usage. They are "forgotten" depending on the frequency of usage and recency of access. Although these ideas have been introduced on a conceptual level, they have not been actually been implemented.

This is particularly important since some users can introduce noise into the system in form of poor and noisy documents. For that noise to result in the creation of links it has to be clustered with similar content. In the unlikely case that it does get clustered with similar content of a noisy nature, some one will have to express an interest in that sort of content before noisy links could be propagated to them, the probability of which is quite low. Because the lifetime of links is dependent on usability, noisy links will eventually be removed from the link KB.

6.4 The Image Annotation Service

Searching for multimedia information embedded in WWW pages is still an area where much research is being conducted. However, little attention has been directed towards implementing tools for locating and viewing parts/segments of multimedia. Specifically, the limitations of image representation on the Web could be summarised as follows:

1. The representation of multimedia fragments is currently not well supported.
2. There are no means by which authors can specify links to multimedia fragments.
3. Augmenting multimedia fragments, or even entire multimedia elements, with metadata is not well supported.

Since one of the main aims of this work is to assist users in finding information multimedia or otherwise, these limitations were addressed (El-Beltagy et al., 1999b). The inspiration for the *image annotation service* 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 was well supported. This idea

was further explored in the MAVIS system (Multimedia Architecture for Video, Image, and Sound) (Lewis et al., 1996). Emerging Web standards such as SMIL (W3Consortium, 1998) have also recognised the need for this support. However since direct support for this still does not exist, a number of tools for locating and viewing parts/segments of images were implemented.

In order to enable representation of image fragments, an image viewer applet capable of displaying an image with a highlighted region was implemented. The applet takes in as parameters an image's URL and the co-ordinates of the fragment that needs to be highlighted within the image. This tool indirectly supports linking to image fragments. A script can be easily developed to automatically generate HTML on the fly, with the applet embedded in such as way as to represent image fragment. So, in order to link to an image fragment, the destination of the link would be a pointer to that script with the necessary parameters encoded.

Augmenting multimedia entities with metadata can greatly enhance the ability to find these entities through a search facility thus making information that would otherwise be inaccessible, available. For example, if a map is annotated for regions that represent cities, then searching for a city will not only yield a pointer to the map, but a pointer to the city on that map.

Following the premise that users can benefit from each other's knowledge, this 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. The agent was mainly implemented as a proof of concept as to how the architecture can embrace a more structured model of information acquisition.

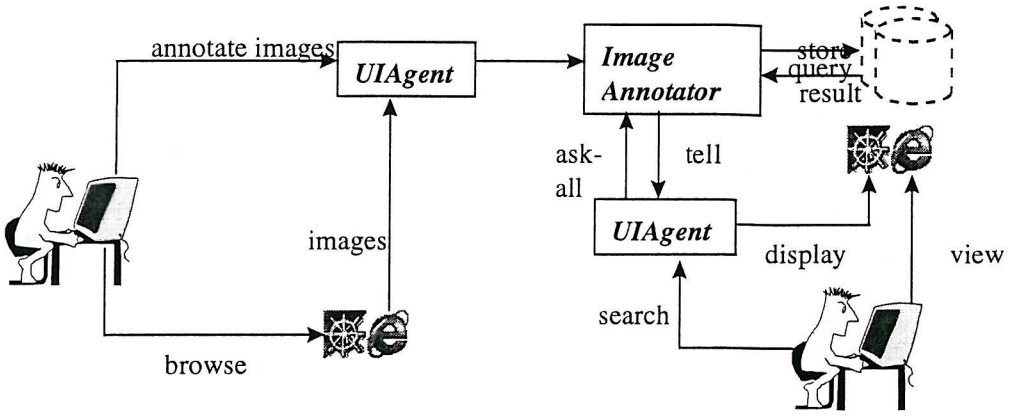

Figure 6.7: The image annotator as used within the framework

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 can be defined at 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 6.7 shows a diagram of how this agent is used within the described framework.

The KQML messages that this agent advertises itself as being capable of understanding are:

- Performative: tell, ontology: image-ontology, language Prolog, content: imageRegion(+URL, +ImageTitle, +Creator, +RegionTitle, +Keywords, +Category, +x1, +y1, +x2, +y2)
- Performative: ask-all, ontology search-ontology, language Prolog, content: find(+Keywords, - UrlList, 'Prolog')

6.5 Search Wrappers

As a proof of concept of the flexibility of the framework in dynamically incorporating agents, three wrappers around existing search systems were developed. The first of these, is a wrapper around the Electronics and Computer

Science staff and students database. The second and third are wrappers around the Altavista (Altavista, 1998) and Google (Google, 1999) search engines respectively. All three wrappers are implemented in Java and accept the KQML ask-all performative, with the ontology: search-ontology, language Prolog, and content: find(+Keywords, -UrlList, 'Prolog'). The Altavista and Google search engine also accept find(+Keywords, -UrlList, 'XML'). Figure 6.8 and Figure 6.9 show the two different result representations.

```
- <results>
  - <result>
      <url>http://mtu.www.media.mit.edu/people/mtu/agents.html</url>
      <title>Commercial Software Agents</title>
      <summary>Commercial Software Agents |Programming | Survey results | Related
        References. Commercial Software Agents . Presentation in zipped PowerPoint
        4.0...</summary>
    </result>
  - <result>
      <url>http://www.ca.sandia.gov/~carmen/hicss.html</url>
      <title>HICSS-32 Mini-Track on Software Agents</title>
      <summary>HICSS-32 Mini-Track on Software Agents. Part of the Software Track of
        HICSS-32. 32nd Hawaii International Conference on System Sciences Maui,
        Hawaii...</summary>
    </result>
```

Figure 6.8: An example of a search result expressed in XML

```
[['http://mtu.www.media.mit.edu/people/mtu/agents.html', 'Commercial
Software Agents', 'Commercial Software Agents |Programming | Survey
results | Related References. Commercial Software Agents . Presentation
in zipped PowerPoint 4.0...'],
['http://www.ca.sandia.gov/~carmen/hicss.html', 'HICSS-32 Mini-Track on
Software Agents', 'HICSS-32 Mini-Track on Software Agents. Part of the
Software Track of HICSS-32. 32nd Hawaii International Conference on
System Sciences Maui, Hawaii...',.
```

Figure 6.9: An example of a search result expressed in Prolog

6.6 Summary

This chapter presented a number of agents that are capable of providing various information finding related services. The OM agent and the QuicLinks agent have

demonstrated how two different agents can use the same information to carry out different tasks. Other agents can be easily introduced in the framework to utilise the same information in order to achieve similar or totally different tasks.

The next chapter presents the user interface agent and shows how through communication with the agents described in this chapter, it can assist its user in various information finding related tasks.

Chapter 7: The User Interface Agent

This chapter presents the user interface agent (UI agent), which is one of the most important and complex agents within the developed system. In the presented framework, each user is associated with such an agent, which acts on his/her behalf. The design of the framework allows for the existence of any agent capable of providing only a subset of the functionality offered by the agent described in this chapter. However, in developing this agent, the aim was to demonstrate as much functionality as possible.

7.1 Overview

In designing the demonstrator UI agent, the goal was to build a relatively intelligent agent that can be of assistance to its user even if for some reason it fails to communicate with other agents in the system. The agent however would achieve optimum performance though such communication. The basic functions of this agent can be summarised as follows:

- It watches over the user's shoulder and monitors his/her navigation, searching, rating, and bookmarking activities. It utilises these activities for building a user profile as well as for other tasks.
- 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 their tasks, provided the user is carrying out these activities in public mode.
- 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.
- It provides a number of tools to facilitate browsing and bookmarking.
- It maintains a personal, context aware linkbase which it populates based on user interests
- It modifies Web pages on the fly so as to provide the user with navigation hints and information finding aids

The first time the agent is activated, it registers with the agent server. It then receives all the advertised messages sent by other agents. Advertised messages that fall outside the interests of the agent are simply ignored. The agent is basically interested in all the performatives introduced by agents discussed in the previous chapter. The agent activates and de-activates some of its services according to its interaction with other agents.

7.2 Implementation

The UI agent has been implemented as a Java desktop application. 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 agent. Figure 7.1 shows a snapshot of the interface offered to a user. As shown in the figure, the

agent keeps track of where the user has been and displays that information to the user.

For each URL visited, 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 a keyword extraction algorithm, and a list of image URLs contained in the page. The role of the proxy server does not end there. Whenever the user interface agent needs to add navigation hints or recommendations to documents on the fly, the proxy propagates those changes to the user's browser.

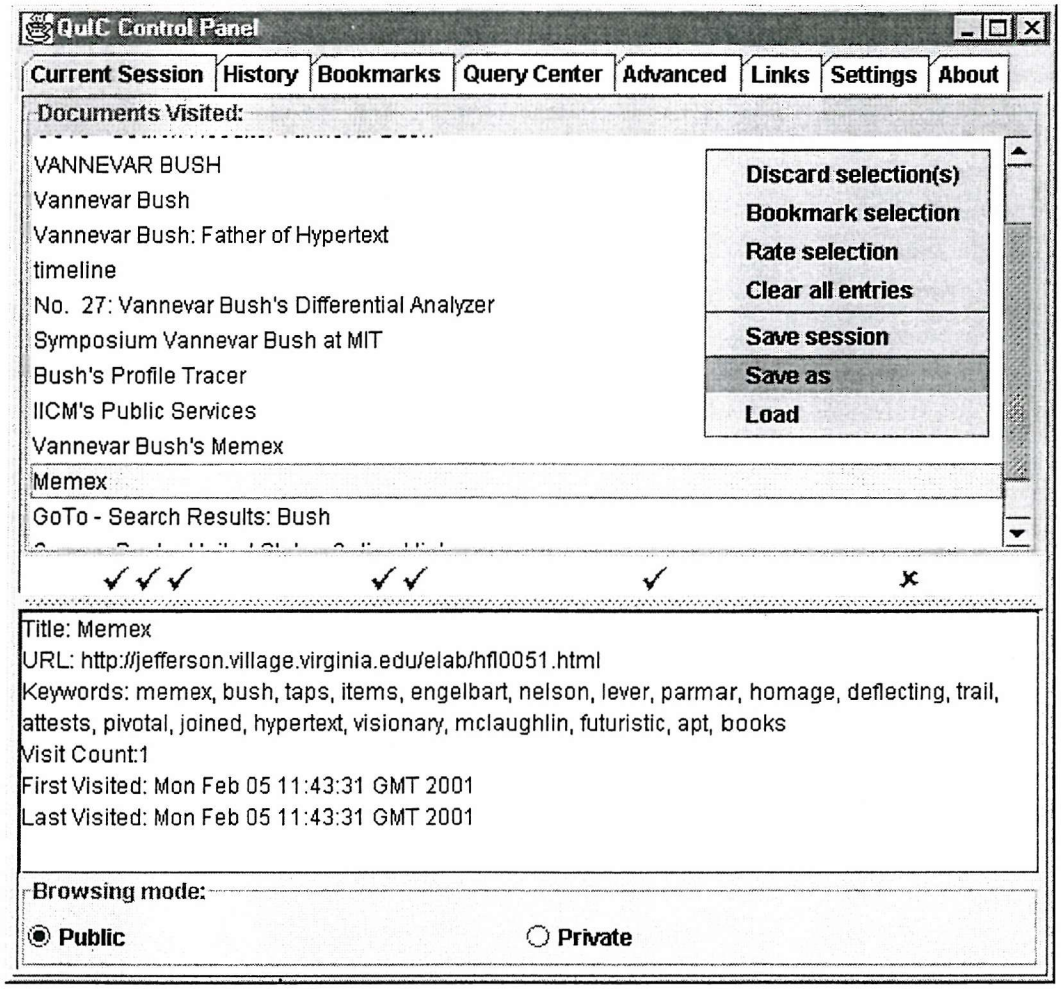

Figure 7.1: A snapshot of one of the User Interface Agent's panels

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 a record of rather than bookmark. This allows the user to organize browsing sessions into significant trails they can consult later. It 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. Alternatively they can rate a document through a small menu that is added by the proxy and displayed at the bottom of any Web page they are browsing. The rating information, which is stored locally as well as dispatched to interested agents, is used in a variety of ways. There is evidence to suggest that humans tend to have poor long-term memory for detailed information such as URLs. In a study conducted to understand how people search the Web, users were given searches to carry out over a period of two days. It was found that some of the users followed dead end paths on the second day that they already had pursued on the previous day (Maglio and Barret, 1997). To prevent users from following links to URLs that they have previously found not useful, and to remind them of links that they have liked, the UI agent uses the locally stored ratings to add symbols that reflect what a user has thought of a URL whenever it appears as a link in a document the user is browsing.

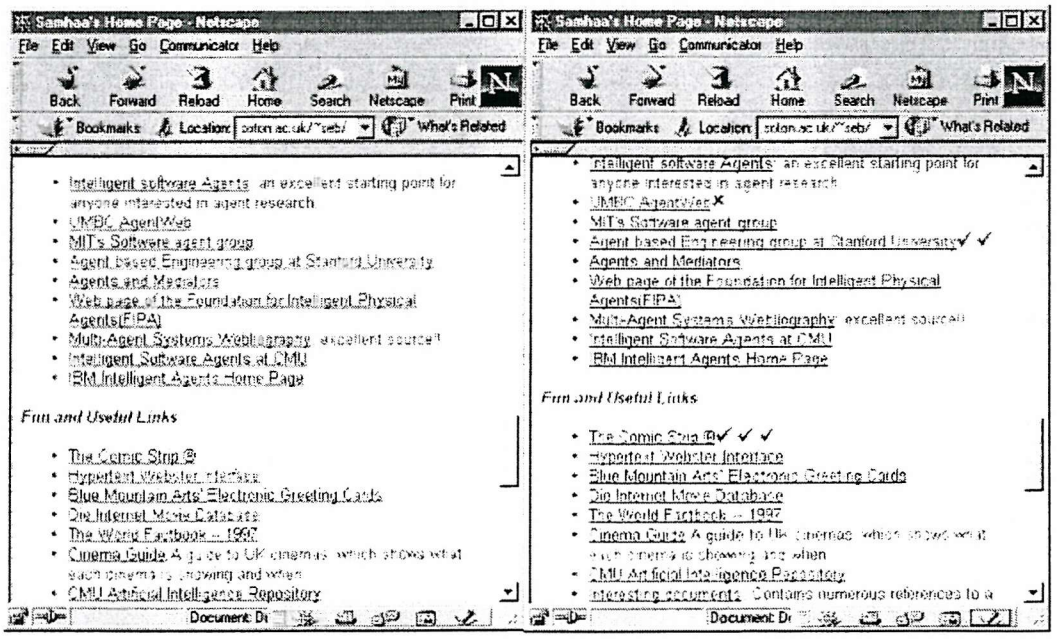

Figure 7.2: A Web page before and after ratings are added

Figure 7.2 provides an example of a Web page that has been modified based on a user's ratings. Because rating documents in that way can have a direct and visible impact on a user's browsing activity, it can act as an incentive for users to rate documents. A user's rating also reflects his/her interest in a Web page, and thus contributes to the user profile as detailed in section 7.3.

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. A utility was built to allow users to import all or a subset of their bookmarks from any standard Netscape bookmark file. As in browsing, the user has a public and private option when creating a bookmark. This controls the publishing of bookmarks to interested agents. Currently implemented agents use this information for providing recommendations to other users.

The UI agent also provides its user with an interface whereby he/she can annotate images and image regions through an annotation tool. The tool gives the users the option of loading image either directly from a URL, or through selection of any existing ones in web pages available in the currently open session. The annotations entered by the users are then sent to appropriate agent(s), in this case an image annotation service. Figure 7.3 shows a snapshot of the interface to that tool.

A query interface component through which the user can access query services provided by the different agents in the system, is among the most important components of the interface agent. This is discussed in more detail in section 7.5. A link creation facility is also provided. This facility allows a user to create generic links. 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 settings panel is provided.

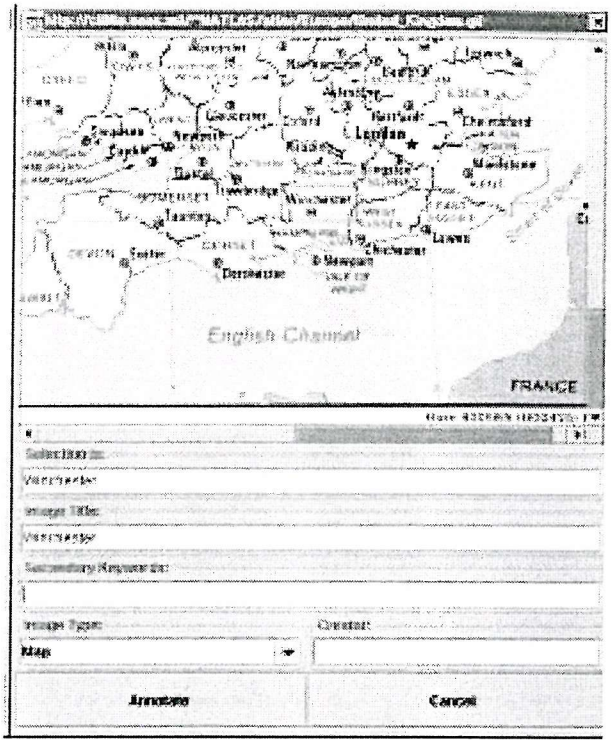

Figure 7.3: A snapshot of the image annotation utility

Another component of the UI agent is an interface to the user's default browser. The interface allows the UI agent to invoke the display of any URL in the browser irrespective of the platform on which the agent is running. This allows the user to double click on any of the locations in his/her bookmarks or current session, and for that location to be displayed in his/her browser. It also allows the user to follow HTML links displayed in the agent's panels as shown in Figure 7.9 for example, and for the link to be displayed in the browser.

7.3 Building the user profile

7.3.1 Constraints

It is important that the UI agent understands what the interests of the user are so that it can populate the user's personal linkbase with links that can assist that user in finding information in the context of their interests or aid the user in formulating a query. One of the problems with 'learning' a user profile is that the process often takes some considerable time. It has been observed that users can be put off from using an agent if it requires a long learning time before the user can reap benefits from it (Lieberman, 1997). The presented system tries to avoid this in two ways. The first is by trying to make the system immediately useful, irrespective of whether a profile has been built or not. The second is to attempt to bootstrap the profile as fast as possible so that the user can make maximum use of functions activated by the agent's understanding the user's interests. For the achievement of the latter goal, the use of bookmarks seems well suited. Bookmarks are quite effective for building a user profile since they explicitly embrace documents that a user has shown an interest in. When a user starts using the system he/she can import their bookmarks thus enabling an almost immediate way of building the profile.

However, it is important to point out that not all users use bookmarks (Abrams et al., 1998), and those who do use bookmarks, don't always bookmark everything they like or they find interesting (Rucker and Polanco, 1997). In designing the UI agent, the goal was to make it completely adaptable to a user's work patterns. As such the UI agent does not rely on any single user activity nor does it make any assumptions about the user in building a profile. Instead, it utilises all activities that can be used for building the profile. Identified activities include browsing, bookmarking, and rating Web pages as well as getting feedback on search.

7.3.2 Representation

The user profile is meant to reflect user interests. The adopted model for the profile is one in which each of the user's interests is represented by a feature vector of terms. A user will usually have many interests, so the user profile is represented by

a set of feature vectors of terms. As will be detailed in the following sections, the profile is frequently searched and the size of the profile affects the time required to carry out the search. For that reason, the set of user interests was split into two types of sets: an active set, the size of which is constrained by the system designer, and an inactive set, which can be of any size depending on actual user interests. The overall interests of a user can be viewed as the union of the two sets and each of the sets can be viewed as the agent's memory of the user's interests. So in a way, the user interface agent has two types of memory, an active memory and a dormant memory. Swapping between active and inactive sets is performed using the least recently used algorithm. Each interest represented in either set reflects information about the user's context. Thus the words, *interest* and *context*, will be used interchangeably.

7.3.3 Implementation

The user profile is built using documents that a user finds interesting. Rules for determining what the user has found interesting vary according to the type of activity that is used to infer the interest. The rules for bookmarking and rating are straightforward. Any document that the user bookmarks or rates with a value of very good or excellent are assumed to be interesting. The same applies for relevance feedback on search, if a document is marked as relevant to a user's search, then it is added to the profile. Browsing is not that straightforward because the act of opening a Web page does not necessarily indicate an interest in that page. Rather, it indicates an interest in what might be in the page. Whether the user liked the Web page or not, is not instantly possible to deduce unless the user decides to rate it or bookmark it. As described before, the UI agent provides the user with a facility to easily rate a Web page as being bad in a way that would remind him/her of links he/she disliked and prevent him/her from unknowingly following that link from another Web page. As such it is logical to assume that recurring visits to a Web page that has not been badly rated will usually indicate an interest in that page. The number of visits that a user must make to a page before it is considered interesting, is user configurable.

Whenever a user opens an html/text Web document, the html content of the document is parsed into a URL object. Keyword information is obtained by ignoring stop words, and weak stemming other words by using only the first step of the Porter stemming algorithm (Porter, 1980). A counter is used to keep track of the occurrences of each word in the document, and a weight value is assigned to words according to their position in the document text. Words appearing in the title or headings get a higher weight than other words. The feature vector of fixed length l is then calculated for the document by sorting and keeping terms with the highest l weights. Assuming that the system has determined that a given Web page is of interest to a user, then the algorithm presented in section 5.2 is applied to add the document to the user profile, which is represented by multiple cluster centroids where each centroid represents a user interest.

7.4 Rendering the Links

One of the components of a UI agent is a personal context aware linkbase. Representation of links in this type of linkbase is quite different from that of a traditional linkbase as implemented in a system such as the DLS. A context aware linkbase is a linkbase where links in a given context are grouped together. So the context aware linkbase could in fact be thought of as multiple linkbases partitioned by context. The context of each group is represented by a feature vector of terms which is used to obtain the links from the link extraction agent. Specifically, links in the linkbase are imported from link extraction agents based on the interests of a user, each of which is represented by a feature vector of terms used to group retrieved links together. Once a new user interest is detected, the UI agent sends a subscription message to the link extraction agent for links related to that specific interest which is represented by the feature vector of terms. The link extraction agent then sends the UI agent all available links for that interest by matching the input vector to its available cluster summaries. As soon as new links become available, they too get dispatched to the subscribed UI agent. Figure 7.4 shows the overall scenario for linking in context.

Assuming that the UI agent's context aware linkbase is populated with links that are of interest to the user, then the UI agent is responsible for rendering those links to Web pages which fit that context.

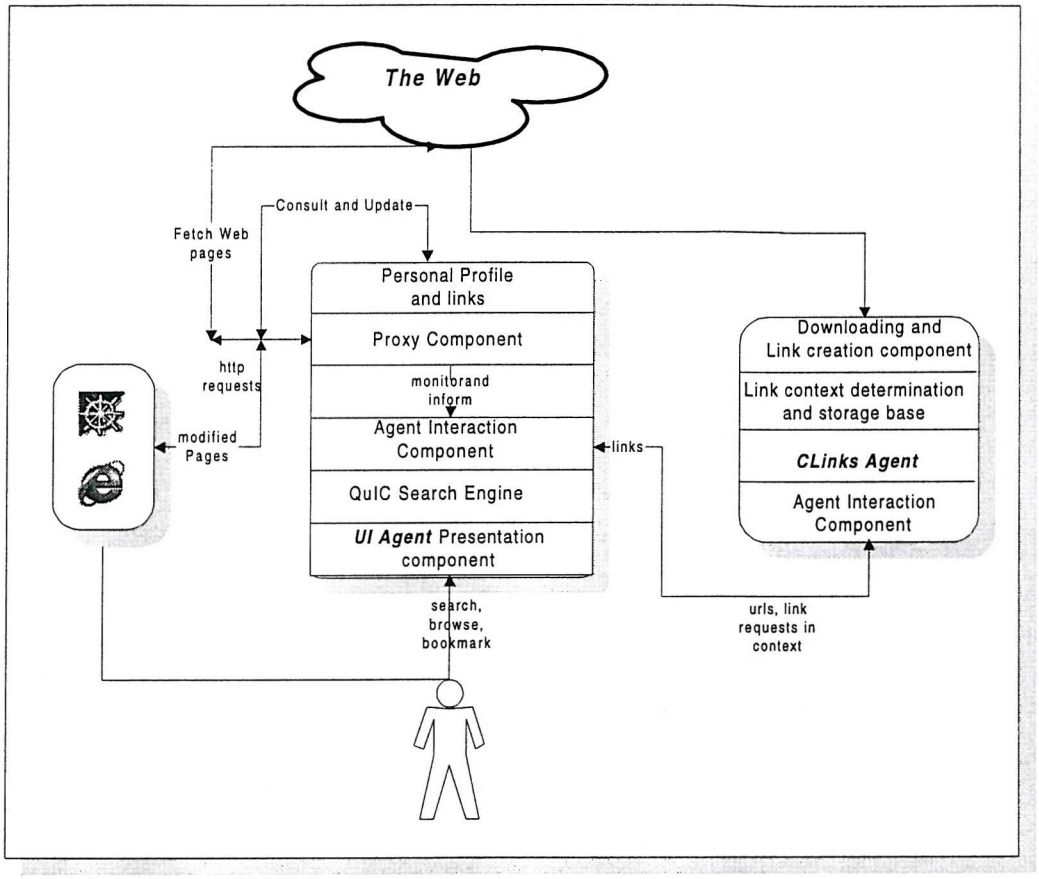

Figure 7.4: The interaction scenario for linking in context

The process of rendering links on the fly has to be done as quickly as possible. To render a link in context, the context of the Web page that is being viewed has to be determined first. Though this process is not a lengthy one, in cases where a user is viewing a Web page for the first time, the Web page has to be downloaded before the context determination process can commence. If the display of the Web page is to be blocked by the proxy until the page completely downloads, then the speed at which the page appears will be dependant on the speed of the connection between the agent and the server from which the document is requested. In some cases this might not be acceptable. As a result, during the design phase, it was decided that the display of pages would take place in an asynchronous way, and that a

technology such as server push or client pull would be used to deliver the altered Web page. In server push, the connection between the browser and the server is kept open after a Web page is downloaded until the server pushes new content, which replaces the current one. In client pull, the connection is closed after a Web page is downloaded and then requested again from the server after a specified time had passed (Netscape, 1999). When implementing the system, the two technologies were employed so as to make the system browser independent.

A number of data structures are employed to facilitate rapid determination of link context. As noted before, cluster centroids are used to represent link contexts. A table is used to map context identifiers to cluster centroids, each of which is represented by a weight vector of terms. When a Web page is downloaded for the first time, it is compared to the various existing cluster centroids (representing user interests for which links exist), and in case a match is found, a context identifier is returned. The mapping between Web pages for which a context has been determined, is stored together with a context identifier in another table for rapid context lookup the next time the same Web page is viewed. The context identifier is used to activate a group of links, which collectively can be thought of as a linkbase in their own right. This is done by retrieving a quick lookup table using the context identifier and passing this to the rendering component.

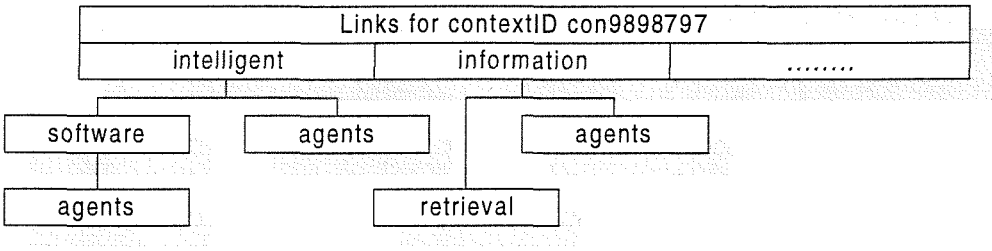

Figure 7.5: Quick lookup table for link phrases

Following an automatically rendered link activates a set of possible destinations from the source anchor of the link as opposed to opening a specific Web page. So the only information required for the process of rendering the links is the knowledge of source anchors represented by phrases. Phrases within the lookup table are represented by trees so as to facilitate rapid parsing. Figure 2 shows a

simplified diagram of part of that lookup table. Basically, once the context for a document has been determined, words in a document are scanned and compared to the roots of trees in the lookup table based on the context identifier given to that document.

If a word is found that matches a root, the next word is compared to nodes in the next level of the tree. If it matches, then it is placed onto a stack. This procedure continues until either the leaves of the tree are reached, or the lookahead word does not match any of the nodes in the level at which computation is taking place. In the latter case, words in the stack are popped one by one until a term, which qualifies as an "end of a phrase", is reached or the stack becomes empty. A word would qualify as an end of phrase if a boolean flag, indicating whether one or more URLs are associated with it, is set. At this stage a link is created from that text phrase to a dynamically generated link in which the context identifier and the phrase are encoded as a query and which points back to a simple HTTP server implemented by the UI agent. Figure 7.6 and Figure 7.7 provide examples of generated destination URLs.

```
http://localhost:9090/?links:Bush;con981371585841
```

Figure 7.6: A link generated for Bush in the G. W. Bush context

```
http://localhost:9090/?links:Bush;con980936717330
http://localhost:9090/?links:Vannevar+Bush;con98093671730
```

Figure 7.7: Two links generated for Bush, and Vannevar+Bush respectively in the Vannevar Bush context

Within the UI agent, another table is used to represent the actual links. In the second table, phrases are mapped to one or more URLs. When a user follows a dynamically added link, the context identifier encoded in the URL is used to activate the appropriate table while the query encoded in the link is used to retrieve

appropriate links related to that phrase through a rapid search process. Figure 7.8, shows how links are rendered and resolved in two different contexts. In the two documents, different links are suggested for "Bush" based on the context. An icon appears besides linked phrases to distinguish recommended links from original links in the page. The links shown were created as part of the experiment described in the evaluation chapter (chapter 8).

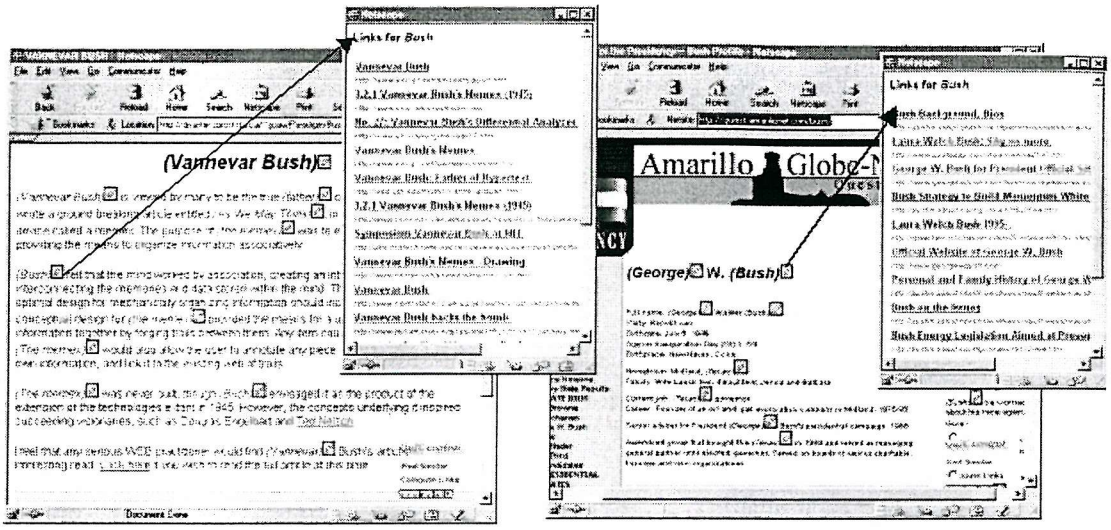

Figure 7.8: An example showing how links are rendered in two different contexts.

7.5 Answering Queries

Besides proactively adding navigation and information finding aids, the UI agent makes it possible for the user to interactively enter queries. User queries prompt the UI agent to contact appropriate agents in order to obtain answers. A user query could be a generic one entered using keywords in which case searchable resources are consulted. The image annotation service, the OM agent, and the search wrappers described in the previous chapter, all support this mode of querying. As soon as a searchable resource comes on-line (any agent that can support the ask-all performative with content find(+Keys, -Results, +Required_result_language), and ontology search-ontology) the UI agent can immediately contact it in order to answer a user's query. Alternatively, the query could be a specific one made in relation to a document.

In the case of keyword queries, the UI agent can help the user refine his/her query based on their profile.

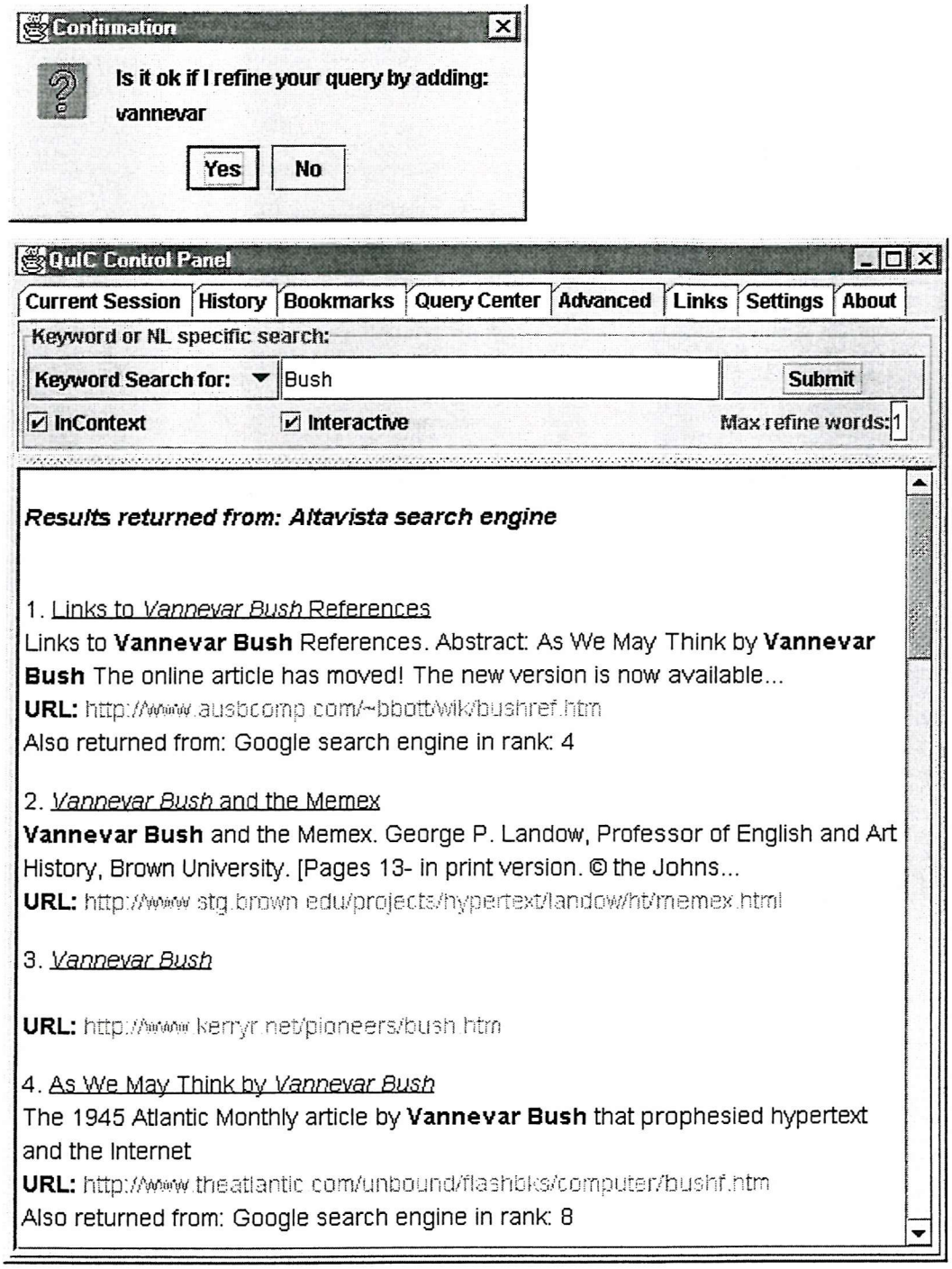

Figure 7.9: A snapshot of the keyword search interface offered to users.

Basically, the UI agent attempts to match keywords entered by the user to various interests or contexts present in their profile. If a match is found, it can automatically refine the query for the user or if the user specifies an interactive mode, it can suggest those refinements before carrying them out. If two or more interests match the user's query, the agent currently uses the most recent interest. Figure 7.9 shows the results returned when the user accepts the suggestion of the UI agent for refining his/her query. The dialog in the top box shows the agent's suggestion. The number of words the agent can suggest is user configurable as shown in the interface. The user can also turn the 'in context' option off or on at any time. When more than one resource is consulted, the interface shows the results returned from each, displaying the results from the agent which provides the fastest answer first. When other answers are received, the display is refreshed and these answers are appended to the bottom of the list with their source clearly indicated. In addition, a cross reference between the results from the various agents is created. This is exemplified by results number 1 and 4 shown in Figure 7.9 where results returned from the Altavista search engine are shown to have also been returned by the Google search engine. The rank assigned to the results by the Google engine is also stated.

Future modifications to the system should allow the user to edit the agent's suggestions. In case an entered query matches with more than one of the user's interests, they should also enable the agent to inform the user of the various contexts in which his/her search word(s) appear thus allowing him/her to actually select the appropriate context for their query.

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. Each of these services is associated with a KQML message. For the answering of these queries, the UI agent contacts agents, which have advertised the associated KQML messages. Figure 7.10 shows the query interface provided by the UI Agent for that mode of querying. In addition to enabling users in making

queries regarding what they have just browsed, the interface also allows users to make similar queries with respect to their bookmarks. By selecting the 'switch to bookmarks' option, their bookmarks are displayed and they can select any of them and invoke the same queries as shown in Figure 7.10. Alternatively, they can enter a URL by hand and request similar services.

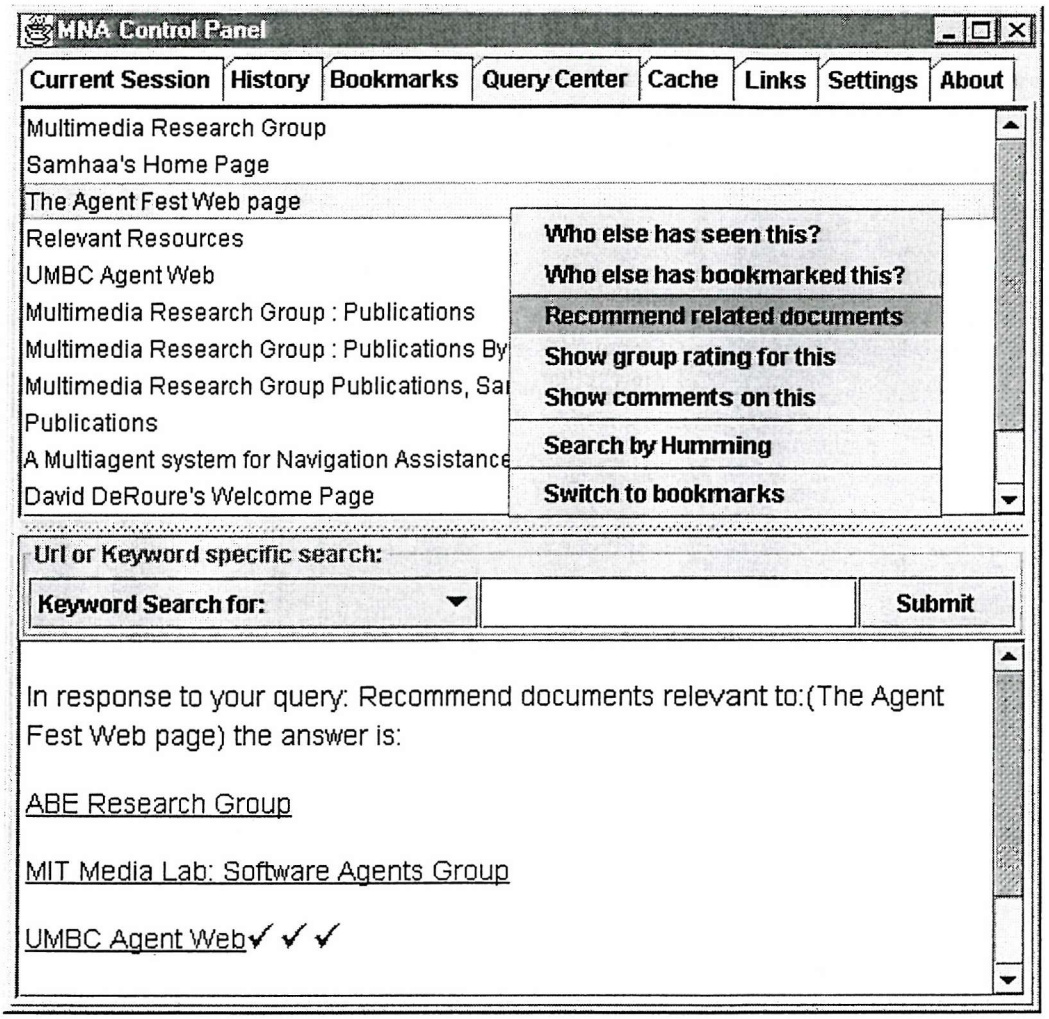

Figure 7.10: A snapshot of the document related query interface

7.6 A Discussion of Document Alteration

Initially, the UI agent was designed so that it can make use of document alteration services. Any agent that can somehow change the appearance of a document advertises its ability to perform a document alteration service through an *'achieve'* performative. Agents that can perform such a service are called document

modifiers. When document modifiers register with the system, they 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 was built allowed for incremental services that influence the display of a requested URL to be added. The process is allowed to 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 the requested URL, but rather than sending it back to the browser, it streams it across the different document modifiers while piping the output of one agent as an input to another. Figure 7.11 illustrates this technique. This cascading makes the process of document alteration less time consuming, but as the number of document modifiers in the system increased, it was observed that 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 contents 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.

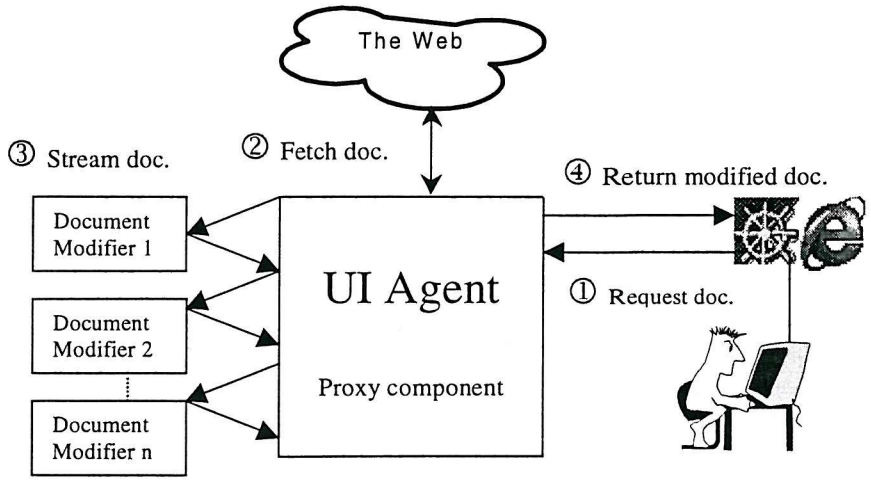

Figure 7.11: Flow of a document across different agents

Examples of document alterations that were identified include the addition of user authored links on the fly, as well as displaying the average group ratings and comments related to given links whenever they appeared in a Web page. These alterations were recognised as ones that could serve to guide the user's navigation activities as they allow for information created by other users to be instantly shared.

Later implementations of the framework and the UI agent eliminated document modification services since they were found to pose a number of problems. For example, any agent that does not function properly could ultimately ruin the display of a given Web page. Streaming documents was also found to be fragile process in which any breakdown in communication between the involved agents could cause a document to display improperly or not at all.

Irrespective of the implementation details, the idea of making information created by users of the system instantly available to other users through a document alteration service is one that warrants mention as well as further research into how to make it feasible.

7.7. Summary

In this chapter, the developed user interface agent was described. The way in which it capitalises on the services offered by other agents in the system to assist the user in his/her information finding activities 'in context' was also outlined. Among the services introduced to assist users in finding information 'in context', is the 'linking in context' facility provided by this agent through adding links on the fly. In the next chapter, the effect of this service in aiding users in their information finding activities is examined. The presented system is also compared to other related systems.

Chapter 8: Evaluation

The purpose of this chapter is to look closely into the work presented in this thesis and evaluate what it has to offer. One of the novel ideas presented by this work is the notion of automatically generating links based on what users in the system have seen and liked, and applying the generated links to documents on the fly based on document context and the individual interests of a user. Our hypothesis is that this can actually lead users to benefit from what other users have seen by pointing them to related documents of reasonable quality. To substantiate this hypothesis, a study was conducted to examine the effect of adding such links on information finding tasks. The study is described in section 8.1.

A close examination of the work is also presented in the form of a comparison between the implemented prototype, and other systems that have similar goals. By comparing the presented system to other implemented systems, it is possible to draw conclusions as to the advantages and disadvantages of the presented system which also serves to evaluate the system. This comparison is presented in section 8.2.

8.1 A Case Study: The Effect of Adding Links on the Fly on Information Finding

To study the implications of dynamically creating and adding links in their proper contexts on the process of locating information, a controlled experiment was set-up. The experiment was a two-phase one in which a set of ten users was randomly divided into two groups. Each group was assigned one of two unrelated

information finding tasks. The two tasks revolved around the private life of Vannevar Bush and relatively basic technical questions about CORBA[3] respectively. Table 8.1 and Table 8.2, provide the questions making up each of the tasks while Appendix B provides the actual questionnaires given to the participants. To demonstrate that the system is capable of distinguishing between two different contexts that can be confused with each other, the Vannevar Bush task also contained three questions on the private life of George W. Bush[4].

Participants in the experiment were all members of the IAM research group who all have a background in computer science but who have varying individual research directions. Users in each of the experimental groups were selected randomly.

1) What were the names of the parents of Vannevar Bush?
2-a) What was the name of Bush's first invention? (hint, it was invented in 1913)
2-b) What did it do?
2-c) Find a URL for a photo of the invention
3) Name three positions that Vannevar Bush held
4) What year was Bush married and what was his wife's name?
5-a) How much did the Differential Analyzer weigh ?
5-b) How many vacuum tubes did it contain?
6) Find three URLs to Web pages describing the Memex
7-a) Where was George W. Bush born?
7-b) What is name of George W. Bush's wife?
7-c) Find two URLs related to her

Table 8.1: Questions users were asked about V. Bush and G. W Bush

[3] Common Object Request Broker Architecture

[4] The current president of the United States of America

Before conducting the experiment the users were asked to indicate how familiar they were with the subjects in question. Table 8.3, in which U*i* refers to user *i*, summarises their responses.

1) What is CORBA?

2) What is an ORB?

3) Locate two introductory CORBA tutorials (pdf or ps formats are acceptable).

4) What do the following terms stand for in CORBA: BOA, COS, GIOP, PIDL, SSI

5) Find a Web page that explains why CORBA is better than COM

Table 8.2: Questions users were asked about CORBA

	Vannevar Bush	CORBA
U1	Vaguely familiar	Unfamiliar
U2	Vaguely familiar	Vaguely familiar
U3	Vaguely familiar	Vaguely familiar
U4	Unfamiliar	Unfamiliar
U5	Vaguely familiar	Vaguely familiar
U6	Unfamiliar	Vaguely familiar
U7	Familiar	Vaguely familiar
U8	Unfamiliar	Vaguely familiar
U9	Vaguely familiar	Unfamiliar
U10	Unfamiliar	Vaguely familiar

Table 8.3: A table showing the familiarity of different users with areas surrounding the two evaluation tasks

8.1.1 The Experiment

In the first phase, participants in the experiment were asked to carry out their task by finding information using any of the standard available search engines. Users one through five carried out the Bush task, while users six through ten carried out the CORBA task. Initially, no time limit was placed on the answering of the questions, but one in each task proved harder than anticipated (the first in the Bush task and the fifth and last in the CORBA task). As a result a time limit of 20 minutes for the Bush question and 13 minutes for the CORBA question, was placed. For the Bush task, two users were incapable of finding an answer for

question one within the time limits while for the CORBA task, only user four was able to find an answer for question five. The time it took to answer each question, the number of links traversed to find the answer, and the number of queries entered to reach an appropriate answer, were all logged. Charts conveying this information can found in Figure 8.1, Figure 8.2, Figure 8.3 and Figure 8.4.

Users were asked to highly rate pages from which they found answers using a menu that was added by a proxy and which appeared on every page. URLs from which users found the answers were sent to the Link extraction agent (QuicLinks agent) which clustered the URLs and generated links accordingly. The number of URLs used amounted to 39. Of these, 15 were related to Vannevar Bush, 8 to George W. Bush, and 16 to CORBA. For these URLs, four different document clusters were created. Cluster one contained 13 of the Vannevar Bush related documents. Cluster two contained all the George W. Bush related documents. Cluster three contained two Vannevar Bush documents discussing the Memex at length rather than containing personal information about V. Bush. Cluster four contained 14 of the CORBA related URLs. Appendix C, contains the created clusters. The remaining two CORBA related URLs remained unclustered because their contents didn't match with any of the other documents strongly enough. Examples are given below.

Examples of the source anchors created in the context of Vannevar Bush:

Memex, Vannevar Bush, Vannevar, Bush, MIT, timeline, As We May Think, Profile Tracer, Differential Analyzer, 1945.

Examples of the source anchors created in the context of George W. Bush:

George, Bush, President, Laura Welch Bush, Laura, Welch, Governor Bush, Texas, photo gallery, Presidency, News

Examples of the source anchors created in the context of CORBA:

Distributed Applications, CORBA, OMG, ORB, Distributed Objects, Objects, IDL Interfaces, IDL, Object References, CORBA Architecture

In the second phase, the tasks were swapped from one group to the other so those who carried out the CORBA task were asked to carry out the Bush one and vice

versa. In this phase, the participants were asked to accomplish their tasks using a search engine in conjunction with the linking in context utility provided through a UI agent, which they were encouraged to use. In this phase all users were assigned an imaginary user's UI agent, which was made aware in advance that its user is interested in the three areas surrounding the tasks. The agent was made aware of these interests by invoking a browsing pattern that would reflect such an interest. The same time limits established in the first phase were placed on the users. All users however, were capable of finding the answers to the questions well before the limits were reached.

In both phases, all participants carried out their tasks on the same machine so as to make sure that the computational environment was not a factor in the results obtained. Users however, were asked to use the search engine that they felt most comfortable with, as we didn't want lack of familiarity with any specific search engine to affect their performance. For both tasks, users U1, U2, U4, U5, U7 and U10 used Google, U3 used Yahoo, U6 used Altavista while users U9, and U8 used a different engine for each task. Despite the fact that various search engines were used, the trend observed when using the context links seems independent of the search engines used. This is confirmed by the analysis of the query logs where the number of queries entered by users was lower for those using the linking in context utility (please refer to Figure 8.3 and Figure 8.4 for query statistics).

8.1.2 Results

The average time taken to complete the CORBA task in the first phase was approximately 34 minutes while the average time to complete the Bush task was 30 minutes. These figures exclude the time taken to actually write down the answer to any particular question. The average time taken to complete the CORBA task in the second phase was approximately 13 minutes while the average time it took to complete the Bush task was 16 minutes.

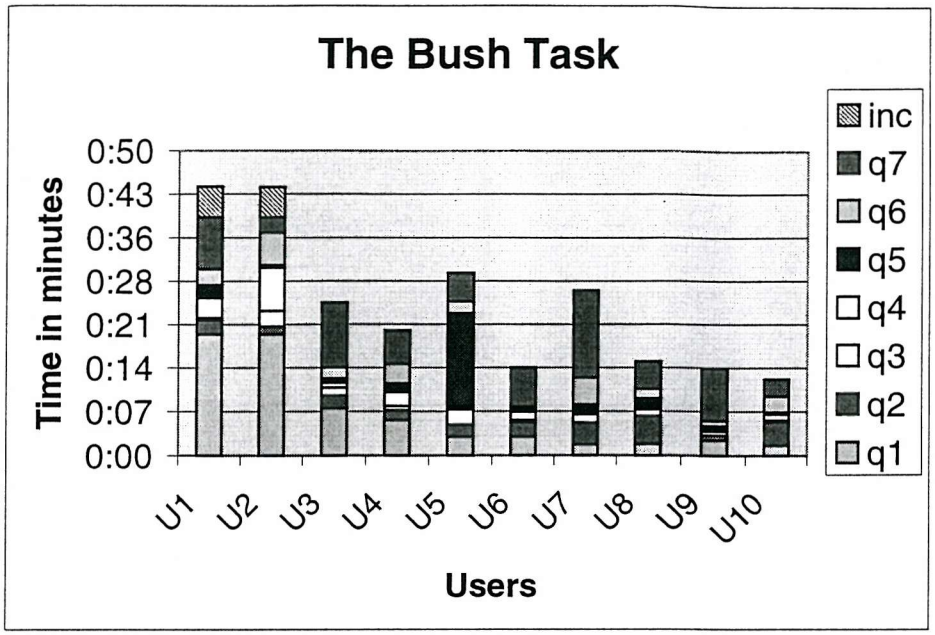

Figure 8.1: Chart showing the time taken by each of the participants to complete the Bush task. Users 1 through 5 used a standard search engine only to carry out their task, while users 6 to 10 used the linking utility in conjunction with a search engine.

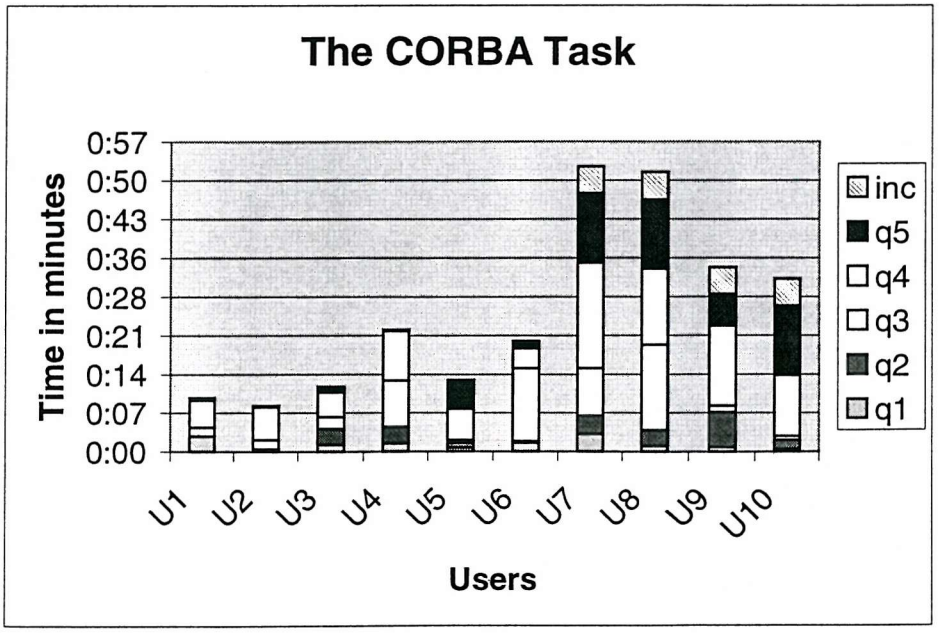

Figure 8.2: Chart showing the time taken by each of the participants to complete the CORBA task. Users 6 through 10 used a standard search engine only to carry out their task, while users 1 to 5 used the linking utility in conjunction with a search engine.

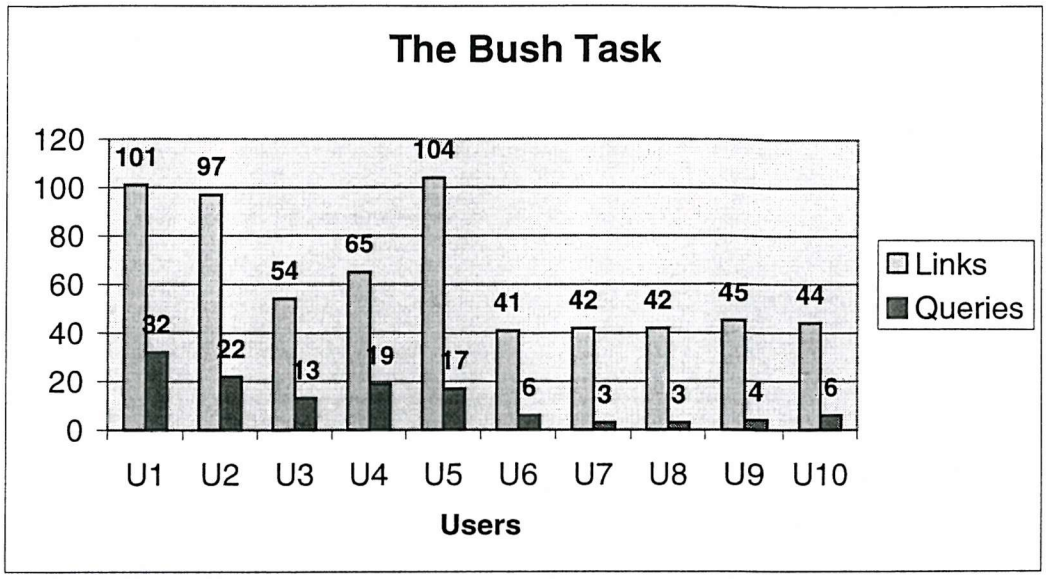

Figure 8.3: Chart showing the number of links traversed and queries entered by each of the users in order to complete the Bush task

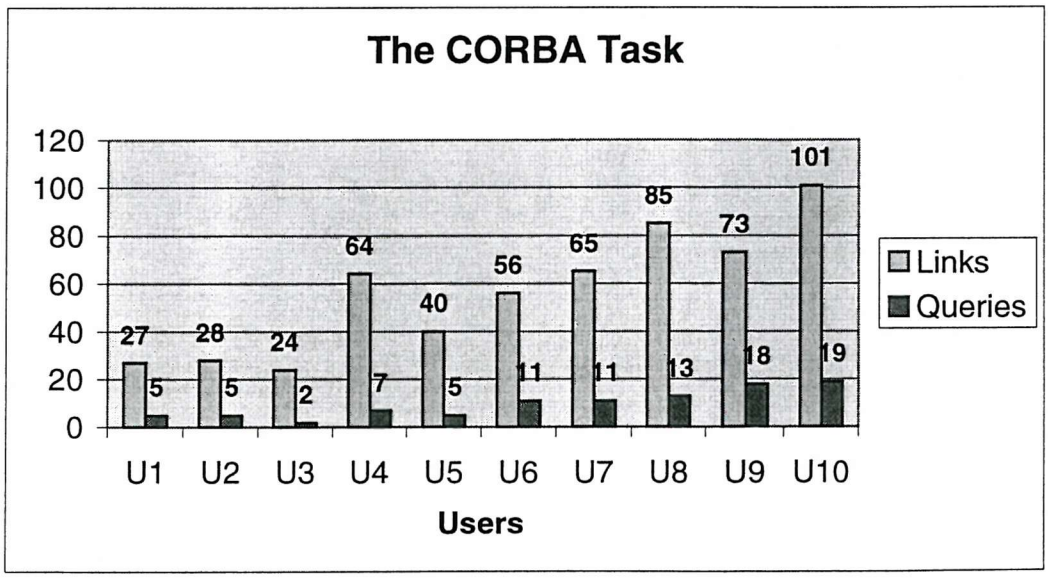

Figure 8.4: Chart showing the number of links traversed and queries entered by each of the users in order to complete the CORBA task

So on average, users using the linking facility completed their task in 38% of the time used by users employing a search engine only for the CORBA task and in 53% of the time taken by users using a search engine only for the Bush task. In

addition, all users using the linking facility were able to complete their task, which is not true of users using a search engine only. Figure 8.1 and Figure 8.2, where *qi* denotes question *i* and ***inc*** is an indicator that the task was not completed, show the time it took each user to complete a given question in each of the tasks. When using the context links, the time taken by users carrying out the CORBA task was significantly lower than the time taken by their counterparts using only a search engine. This is true for all users except user U4 who felt more comfortable with a search engine and was reluctant to look closely at the offered link sets. Though not as significantly low as in the CORBA task, the time taken by users employing the linking in context facility to complete the Bush task was also less than that taken by their counterparts using a search engine only. The only exception to this was user U7 who later indicated that she didn't find the purpose and meaning of the links intuitive.

8.1.3 User Feedback

This case study provided an opportunity to obtain user feedback on the linking facility. All participants in the experiment, were first time users. User seven (U7) was the first to carry out her task using the linking in context utility. Not enough information was provided to the user about what the links would actually do which seems to have confused the user. At the end of the experiment, the user indicated that understanding the purpose of the links was not intuitive. This is not surprising since most Web users are accustomed to the click and jump Web link model. However, it made it obvious that to make maximum use of the offered links, their purpose and what invoking them actually means, has to be clarified to users of the system. Subsequent users were told that invoking a dynamically added link is like initiating a search for the phrase for which the link is being offered in the context of the document in which it was added. They were also told that what they would get back is a list of links related to the concept represented by the phrase and which they can examine and follow based on their needs. This helped subsequent users in choosing which link sets to activate, and encouraged them to use the linking facility in a more effective way.

Because link labels that serve as a brief description of any particular link have been created automatically through extraction of the text the original author of the link attached to it, the users found some of the labels elusive. For example, a number of links related to Vannevar Bush, were simply labelled *Vannevar Bush*. There was no explicit indication as to which of the pages would actually contain the information required to achieve a given task which made the links less valuable in that particular context. One of the users indicated that a summarisation of the content of a provided link would have been useful. In the case of CORBA, the links created were of a much more informative nature. However there were too many links created which made going through them a bit of a task. For that same reason some of the users missed some obvious links which if followed, would have made their task easier. Figure 8.5 shows the set of links offered to users upon activating a link attached to the term *CORBA*.

- nice CORBA information site about the Java 2 ORB
- CORBA
- CORBA Architecture.
- Introduction CORBA ORB Architecture
- CORBA: Quick Recap
- CORBA is a standard for ORBs.
- Real-time CORBA standard
- Patterns and Performance of Real-time CORBA
- Why is CORBA important in a networked environment?
- org.omg.CORBA
- Real-time CORBA implementation
- CORBA FAQ
- What is CORBA?
- Introduction to CORBA
- Why is CORBA better than COM?
- A Brief Tutorial on CORBA
- All CORBA Clients
- The CORBA Architecture
- CORBA Components
- An Overview of CORBA COSS Event Services
- How does CORBA work?
- CORBA GIOP
- DevDirectory : Java : Articles and Tutorials : CORBA
- CORBA and Java Tutorial: Digging Deeper
- CORBA ORB
- Implementing Multi-threaded CORBA Applications with ACE and TAO
- Examples of Writing CORBA Applications
- CORBA Objects are Described by IDL Interfaces
- CORBA Acronyms [UPDATED!]
- Using Real-time CORBA Effectively
- CORBA Services
- CORBA IDL and the IDL/Java Mapping
- CORBA Overall Architecture.
- Implementing a CORBA Client
- Software Development with CORBA
- Introduction to Distributed Object Programming with CORBA
- CORBA as a Standard for Distributed Objects
- CORBA Tutorial Material
- CORBA Products
- CORBA (Common Object Request Broker Architecture) ORB
- CORBA 2.0 Overview
- Free CORBA page

Figure 8.5: Link labels generated for source anchor *CORBA*

Because Web pages are rendered using push or pull technology, a flicker occurs when the links are rendered. Three of the users found this flicker effect distracting. One user also indicated that the way link icons are added to Web pages interrupts his/her flow of reading. Consequently, investigation into other less disruptive ways of rendering links should be carried out. Despite this limitation however, nine out of the ten users stated that they found the added links more useful than distracting and eight said that they would definitely use the system if it were available to them.

8.1.4 Observations and Conclusions

When observing users carry out their search task, it was obvious that different users have different search strategies. It was also clear that while one search strategy would pay off for a given task, it would not for another.

For example, the two users who were incapable of completing the Bush question asking for the names of the parents Vannevar Bush tried long query combinations which besides *Vannevar Bush* included keywords such as *parents*, *father*, *mother*, and *family*. None of these were good choices because none of the existing Web pages had these words. The two Web pages found to contain this information referred to the names of Bush's parents by stating that he was "born to Perry and Emma Bush in Everette, Massachusetts". The users who were able to find this information using a search engine just entered *Vannevar Bush* as a query. The number of Web pages returned for this generic query was quite high, but the users that exhibited patience and followed enough links, eventually found the answer. This is one case where a general query was effective in leading to an answer. On the other hand, the only user able to answer the question "find a Web page explaining why CORBA is better than COM", was the only user who tried a very specific query by entering most of the keywords in that question. The others went for more generic queries and ended up giving up on the task all together. The same user also found the answer to question four in the CORBA task very quickly by entering most of the terms for which an expansion was required (see Table 8.2 for

the actual question), in one query. One of the results of the query was a CORBA acronym page in which the user was able to locate the answer. The same question proved to be one the most time consuming for other users.

Flexibility in terms of rapidly being able to switch from one search strategy to another is probably the best way of searching the Web. Yet, once a user selects a path, no matter what that is, it will probably take some time to discover whether it is a dead end or not. The advantage of offering links in context is that the user does not even have to think of a query to enter. A number of users using the linking in context utility were able to name the parents of Vannevar Bush by activating the *Bush* link and following a link to *Events in the life of Vannevar Bush* (a timeline) from the set of offered links. Even though the timeline URL was sent to the QuicLinks agent for addition because users in the first phase found it useful when answering other questions in the Bush task, users who saw this link were able to recognise it as a potential candidate for answering that particular question. Question four in the CORBA task, which as mentioned before was one of the most time consuming questions for users using a search engine only, was an easy question to answer using the provided links as one of the links provided for CORBA was *CORBA Acronyms*. The users of search engines did not think of adding the word acronym in the keywords, yet when the link was offered to users of the linking in context utility, it was readily recognised as a page where the answer for that particular question could be found.

So, by dynamically adding links the cognitive overhead required in performing a search, is reduced. The probability that users will get links unrelated to the document from which the link is followed is very low. In a way, activating a link allows users to see associations the original authors of the links have made with respect to the phrase for which a link is available. Offered destinations allow users to find information that might not be so easily located via a search engine.

8.2 Comparison with Related Systems

The purpose of this section is to compare the presented framework with systems that share some common features and goals while highlighting the similarities and differences between them. The system is compared with agent based and recommender systems presented in section 2.8 as well as with hypermedia systems that have addressed dynamic link generation.

8.2.1 Agent Based And Recommender Systems

Because the presented framework embraces various forms of information finding and navigation functionalities supported through a number of agents, it is hard to compare it as a whole to any individual system. In the next few subsections however, an attempt is made to make such a comparison. If a reviewed agent is particularly relevant to one of the implemented agents, a mention of that agent is made. At the end of the section, Table 8.4 summarises the various capabilities of the different systems.

8.2.1.1 WebMate

The user interface agent within the presented system utilises a profiling approach similar to that employed by WebMate. However, there are a number of differences that meet the specific requirements of the UI agent, which are distinct from those of WebMate. While the goal of having a user profile in WebMate is to compile an automatic newspaper by crawling news resources, the main aim of a user profile in the presented system is to enable the UI agent to request appropriate links from link provider agents. These links are stored and rendered in documents that match their context.

Rendering links in context requires a level of matching accuracy not achieved in WebMate because of the limit placed on the number of user interests as well as the fact that the algorithm employed by WebMate will confuse similar yet unrelated contexts such as those of Kate Bush and Vannevar Bush. To keep user interests at the specified number in WebMate, every time the maximum number of interests is reached, interest vectors are merged based on the pair wise maximum similarity between the vectors and which is computed across all interests. The lack of

accuracy comes from a threshold value not being employed in the process of merging interest vectors, which means that even if the maximum similarity between two vectors turns out to be of a very small value, the vectors would still be merged. Also, because WebMate does not employ the heuristics employed by the UI agent, it can not distinguish between documents that are different, but which share the same set of highly rated keywords. Though this might have been acceptable for the purposes of WebMate, it is not at all acceptable for the achievement of the aims intended for the UI agent.

Another difference between the two agents is that in WebMate the focus is on building a model that implicitly captures the domains users are interested in through explicit rating. In the presented system the focus was on building a model that implicitly captures user context through monitoring of various activities, one of which is explicit rating.

WebMate relies on searching and crawling for obtaining documents and does not have any support for collaboration. Adding links or navigation aids of any form, is not something that was addressed in WebMate.

8.2.1.2 Fab

The main similarity between the presented system and Fab is that they both employ a multiagent architecture and TF-IDF as well as aim to present users with information that is relevant to them. The two systems also employ both content-based and collaborative based methods. However, there are many differences between Fab and the presented system. The main features of Fab that make it different from the presented one, are summarised as follows:

- In Fab, user profiling is achieved solely through ratings of documents.
- A new system user has to rate a large amount of documents before the system can be of use to him/her.
- A user's interest is represented through a single vector as opposed to the multiple interest representation employed by the presented system.

- Recommendations are provided on demand and are obtained based on pages retrieved through sending queries to standard Web engines.
- The idea of context is not supported

8.2.1.3 WebWatcher

WebWatcher resembles the presented system in a number of aspects. For example, it makes use of the collective browsing activity of users in the system for the aid of other users. It also adds annotations to Web pages on the fly using its proxy-like server. However WebWatcher uses the information it collects from users in a totally different way than the presented system, and utilises different learning techniques to establish its recommendations. The system does not generate links, but rather emphasises the importance of links in the Web page a user is browsing based on its understanding of the user's interests, which are explicitly provided rather than learned. Links to the WebWatcher system have to be hard coded in the page that acts as a starting point for the system and the document set that can be used in the learning process is restricted to links reachable from that point. This means that the recommendations made by WebWatcher are limited to a narrow set of Web pages.

8.2.1.4 Letizia

The main similarity between Letizia and the presented system is that both attempt to non-intrusively make use of users' activities to provide them with document suggestions. Unlike the presented system however, Letizia just makes use of one of the user's activities, which is browsing. Letizia has no support for collaboration and recommendations are based solely on exploring links present in a document a user is browsing. The suggestions made by the system are presented in a window that is separate from the user's browsing window. By using a separate window to display recommendations, the system adds to the cognitive overhead of the user as it requires him/her to constantly monitor the window for new recommendations. The creator of the system acknowledges that the interface might not necessarily fit the cognitive style of all users (Lieberman, 1997). By simply embedding indicators (in the form of links) to possible recommendations in a Web page the user is browsing, the presented system has overcome this problem without overwhelming

the user with the presentation of recommendations that he/she might not really be interested in. Unlike the presented system, Letizia is both browser and platform dependent.

8.2.1.5 Syskill and Webert

Compared to other systems and to the presented one, *'Syskill and Webert'* suffers from some major limitations. The following highlight the most important of these:

- A user profile is static and the system requires a lot of ratings before it becomes usable.
- No incentive is provided to encourage users to rate documents and there is no guarantee that rating a document, which is not a natural activity for the user, would later result in obtaining a recommendation.
- The system only covers a limited number of predetermined domains for which large data sets are required.
- Annotation of Web pages is not performed interactively and requires that all links be saved and analysed before the user can actually see them which demands a great deal of patience on the part of the user.

8.2.1.6 MEMOIR

The main similarity between MEMOIR and the presented system is that they both support the notion of a collaborative, multiagent system. Also, in the presented system bookmarks are used in a way that is quite similar to the way trails were used in MEMOIR. The presented work however, differs significantly from MEMOIR in a number of ways relating to both its design and functionality:

- In MEMOIR, the user interface was a non-persistent entity implemented as an applet. In our system, a user interface is also an intelligent user agent which is persistent and maintains the user's context between browser sessions and offers a variety of services to the user based on his/her previous browsing experience and what he/she have expressed a liking/disliking for.
- In MEMOIR, explicit user profiling was not supported.
- In MEMOIR, a proxy server was used as a central component accessed by all user interface agents to monitor what users were seeing. In our system, each

user agent has a proxy component which serves the same function, but which only publishes what the user identifies as public and performs other functions specific to that user. Private viewing of documents is thus achieved.

- MEMOIR employed a messaging syntax that was closely related to Microcosm's message syntax (Fountain et al., 1990). In our system, KQML messages are employed. This has contributed to greater flexibility in the agent architecture.
- In MEMOIR the addition of an agent required human intervention, but within our framework it can be totally dynamic.
- The goal of MEMOIR was to basically monitor the user's browsing activities and to identify their interests. In this system, this is only one of the goals. Assisting the users in browsing and locating information within other information stores, is another.
- In MEMOIR trails where identified as first class citizens. Within our system, bookmarks have replaced trails.
- In MEMOIR, an object oriented database was used as the organisational memory. In our system, a Prolog knowledge base was used for this purpose.
- Many services, such as making use of user document ratings as well as linking in context, were not available in the MEMOIR framework

8.2.1.7 Siteseer

Siteseer (Rucker and Polanco, 1997) is a pure collaborative Web recommender system. It employs user's bookmarks for the achievement of the recommendation process by utilising the organisation of bookmarks within folders to infer relevancy to other pages. The infrastructure for collecting bookmarks and delivering recommendations has not been described at any length in the literature presenting the system. The system has similar goals to the MEMOIR system, but uses bookmarks instead of trails for their achievement. In that respect, it is similar to the organisational memory agent presented in section 6.2. In Siteseer the overlap between the bookmarks of the users of the system is used to group them and to

make the recommendations accordingly (Mladenic, 1999). Again, this makes the employed algorithm quite similar to that employed by the OM agent.

8.2.1.8 Margin Notes

Margin Notes is one of the most relevant systems to the presented work as it too addresses local context. Like the presented system, Margin Notes uses TF-IDF to analyse document content. Both systems aim to present users with localised recommendations with respect to documents that they are viewing. While Margin Notes presents this in the form of a list of documents related to sections in a document, the presented system does this in the form of links attached to phrases. In one way, the presented system achieves a higher level of contextualization by offering information on the level of phrases as opposed to document sections. On another level, it fails to target the particular context of a given section as addressed by Margin Notes.

Another difference between the presented system and Margin Notes is that Margin Notes does not take into account user history or user interests. It attempts to add suggestions to any document that the user is viewing via centralised service. This means that users can get suggestions to things that they are really not interested in. It also increases the computation overhead and can result in longer times for the display of the suggestions.

Suggestions are limited to pre-indexed static text archives. In the presented system the information repository is continuously changing and has a collaborative aspect to it. Margin Notes doesn't place enough constraints on the document matching process which means that on occasion it is prone to get the context wrong.

8.2.1.9 Watson

Like Margin Notes and the presented system, Watson addresses context and attempts to utilise an understanding of what a user is looking at, to recommend documents. The following features of Watson make it different than the presented systems:

- Watson is not a collaborative system.

- In analysing documents, Watson uses frequency information in conjunction with a set of heuristics that use the location, style, and font of a word among other things, to assign term weights.
- To make recommendations, Watson consults information retrieval systems or search engines.
- Watson displays its document suggestions in a separate window thus imposing the same cognitive overheads on the user as Letizia.
- In Watson, recommendations made are on the level of an entire document and are thus not very localised.
- Watson is application and platform dependent.

8.2.1.10 Summary

Table 8.4, summarises the various capabilities of the different systems. Although the presented framework shares some characteristics with a number of the Web recommender systems outlined, it differs significantly in the way it has approached the task of assisting users and offering recommendations to them. A number of these systems could actually be easily built using the framework provided on this thesis. By abstracting user-related activities from any specific implementation constraints, and representing them in a standardised way, effective re-use of these activities is possible across a wide spectrum of applications. Though processing of this information can be exported back to the producer of the information (the UI agent in the implemented system) through advertised services, it can also be of use to other agents that join the framework. The framework thus provides a way whereby multiple techniques and methods for recommending documents and assisting users in their information finding and navigation activities can be plugged in. This contrasts to the static nature of other systems.

In addition, the idea of extracting links from Web pages and re-applying them in context as a way of recommending documents is a novel one that has not been explored in any of the outlined systems. This way of recommending documents provides targeted recommendations specific to certain concepts in a document. Again this contrasts with providing recommendations to documents similar to

those a user is browsing, simply highlighting links that might be of interest, or just compiling a list of pointers to diverse documents that fit a user's interest.

Except for Letizia, Margin Notes and Watson most of the outlined systems were intrusive in the way they interacted with users. In these systems, users are required to rate documents if the system was to be of any use to them. Though MEMOIR did not use a rating model for making its recommendations, it required users to create trails. Rating and creating trails are not natural activities for Web users, and there is evidence to suggest that users will not carry out such tasks even if they are of benefit to them, unless presented with a strong incentive. The possibility of being provided with some good recommendations on the long run, is usually not incentive enough (Resnick and Varian, 1997). Rather than enforce a single activity for determining user interests, the presented work has utilised normal user activities, such as browsing, bookmarking and searching, for that purpose. Ratings were also used, but an incentive for rating documents was provided in the form of reminders of what the user has thought of a link, rendered on the fly.

Except for MEMOIR and Fab, none of the system attempted to provide users with an integrated and open environment for carrying out various information finding related activities

8.2.2 Hypermedia Systems

The presented work addresses issues that have been subject of research in the hypermedia community. Of particular importance is the issue of dynamic link generation. Dynamic link generation has been addressed by a number of systems, though each of these systems addressed them from a different perspective. The most important of these, are briefly outlined in the following sub-sections.

	WebMate	Fab	Margin Notes	WebWatcher	Letizia	Syskill and Webert	Watson	MEMOIR	Presented system
Uses proxy	Yes	No	Yes	Yes (proxy-like server)	No	No	No	Yes	Yes
Proxy usage	Monitoring	N/A	Monitoring and page annotation	Page annotation and learning	NA	NA	NA	Monitoring and page annotation	Monitoring and page annotation
Multiagent	No	Yes	No	No	No	No	No	Yes	Yes
Dynamic user profiling	Yes	Yes	No	No	Yes	No	No	No	Yes
Profile based on	Document Ratings	Document Ratings	NA	NA	Browsing activity	Document Ratings	NA	NA	Document Ratings, browsing, bookmarking and searching
Basis for Learning/Matching method	TF-IDF	TF-IDF	TF-IDF	Winnow, Wordstat, TF-IDF	TF-IDF	Naive Bayes	Frequency info + heuristics	Trail overlap	TF-IDF, Bookmark overlap
Recommended documents acquired from/through	Crawling/ Searching news resources	Web searching, breadth first crawling	Personal files and notes	Learning by feedback provided while browsing	Breadth first search from a Web page	Manually from the Web	Search engines	User trails	User's bookmarks, rated documents, trails and relevant search results
Content based	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Collaborative	No	Yes	No	Yes	No	No	No	Yes	Yes
Fixed Domain	No	No	No	Yes (bounded by a starting point)		Yes	No	No	No
Recommendation presentation	Compiled newspaper	On-demand, as top 10 URLs to match user's profile	Suggestions made next to document sections	Highlighting of relevant links through icons	A list of related documents presented in a separate window	Off-line annotation of documents	A list of related documents presented in a separate window	On-demand, list of documents related to a particular URL	Addition of link markers related to user links and to particular concepts, to a Web page. Also supports on-demand interaction mode

Table 8.4: Summary showing the similarites and differences between the various system

8.2.2.1 VOIR

The main motivation behind building the VOIR system (Golovchinsky, 1997a; Golovchinsky, 1997b) is to assist users in navigating huge collections of text. In the VOIR system, the user specifies a topic by entering a set of keywords and gets back a collection of articles that match the query with some of the query terms added as anchors in the returned articles. When a user follows an added link, terms from the anchor are added to the original query, which defines the context of the link and which causes a new set of articles to be displayed. In this system, the query highlights what the user is interested in and is used to generate links. Links are resolved during runtime through query expansion

The VOIR system differs from the presented one in many ways. For example, while the presented system is capable of generating links represented by phrases, VOIR's links are represented by single terms. The VOIR system's link anchors are resolved during run-time, while the anchors in our work are determined during the link creation; a process that employs manually created links. The presented system also has a set of features that are not supported by VOIR. For example:

- It can add links to Web pages it has never seen before.
- It builds on the normal WWW browsing interface.
- The use of a personal agent means that user interests are determined automatically, and links are added pro-actively.
- The document set used to create the links is dynamically obtained and continuously updated, and links are added and removed all the time.

It is not clear how the VOIR system will perform if its document collection includes articles from two different contexts, which could be confused with each other.

8.2.2.2 The Link Apprentice

Bernstein's Link Apprentice (Bernstein, 1990) is another example of work that has addressed automatic link generation. The system uses simple pattern matching and string comparison algorithms to generate links. However, the Link Apprentice is liable to generate erroneous links if two unrelated documents representing different

concepts are expressed in similar words, something, which our work has addressed. As a result, the Link Apprentice system has been recognised as suitable for assisting hypertext authors in their link creation activities rather than actually dynamically creating links. There are a number of other systems that are capable of generating link suggestions to hypertext authors (Lelu and Francois, 1992; Robertson et al., 1994).

8.2.2.3 Link Generation and Hypermedia

Link generation is an important issue in hypermedia, and has been addressed by a number of systems. For example, another dimension to link generation is presented through work that describes ways of gathering hypertext documents, linking them, and annotating them by descriptions of inferred link types (Allan, 1996). There is also an overlap between the goals of link adaptation as addressed by this work, and adaptive hypermedia (DeBra, 1999). However, to the knowledge of the author, the approach presented here for link generation and presentation has not been addressed by any other systems, adaptive, or otherwise.

8.3 Summary

The aim of this chapter was to closely examine the presented system. This has been done through a study of the implications of adding context links on the fly for users, and by comparing the implemented system to related systems. In the next chapter, the overall conclusions derived from the presented work are discussed. Future directions for the work are also proposed.

Chapter 9: Conclusions and Future work

9.1 Summary and Conclusions

This thesis presented an agent-based framework for navigation assistance and information finding in context. The main objective of the work was to address the problem of information overload within small groups driven by similar goals in a way that would enable the delivery of personalised and non-intrusive browsing recommendations and hints, as well as support the users in their information finding activities. To achieve this objective, a number of goals were identified in section 1.3.

One of these goals was to identify building blocks for modelling user activities so as to enable the sharing and reuse of information across agents or recommender applications in a way that would ultimately be of benefit to users of the system. Through an examination of related work presented in sections 2.8 and 8.2 it is clear that most Web assistants and recommender systems have focused on techniques for achieving a given information finding or recommendation task. The knowledge units that these systems used, especially those created by a user, examples of which are the user's browsing history, or ratings, remained embedded in the system and offer very little possibility for sharing and reuse. The presented work took a different approach to building collaborative recommendation and information

finding services. The work has argued the case for moving from an embedded representation of user activities to a representation that would allow its dissemination to interested parties that can utilise it and process it for the ultimate benefit of the users of the systems. Key to this is the identification of the user activities that need to be shared and which constitute recommendation building blocks. The achievement of this aim was described in chapter 4. Among other things, this allows for ease of building recommender services, or services that center around user activities in general. It also allows the framework as a whole to evolve and expand in ways that might have not been foreseen by the developer(s) of such a system. This claim was substantiated in chapter 6 where various user activities were used by the OM agent to achieve a set of recommendation tasks, while the same activities were used by the QuicLinks agent in its link creation task. In a similar manner, other agents can join the framework and may use all or a subset of these activities for achieving similar or different tasks. If the agents advertise their services in an ontology the implemented UI agent can understand and is interested in, then their services can instantly become available to the user.

Another related goal was to attempt to make maximum use of what users normally do. In a way, this goal was realised by modelling user activities as this allows various agents to use this information in different ways, thus maximising their benefits. The idea enforced in the design of the framework is that while serving themselves firstly, users will also be indirectly assisting others. The social aspect of the framework is simply a by-product of users attempting to help themselves within a group setting.

Exploring the application of ideas from open hypermedia for the purpose of making 'in context' recommendations and examining the effectiveness of this approach through user evaluations, is an important dimension to this work. To realise this goal, the generic link model was extended to that of a contexualized generic link model discussed in chapter 5. The links offered in context can be thought of as recommendations offered by the system to the user. The recommendations are made as a result of a UI agent detecting that a user is

interested in a given topic and conveying that information to a link provider agent, the linkbase of which is built as a result of other users recommending certain Web pages. By recommending a Web page, users indirectly recommend links embedded within that Web page, which are extracted and used to generate dynamic links. So the final recommendations made are based on links extracted from Web pages that a user has recommended through the act of bookmarking, rating, etc. Based on preliminary user evaluations, it is clear that links created as a result of employing information found useful by one user navigating the information space, can guide other users to information sources that can otherwise be difficult to locate. The whole idea of linking in context contributes to users finding information related to concepts found in Web pages they are viewing. Invoking a dynamically added link is like initiating a search for the linked phrase among links created as a result of the experience of other users, which are less likely to contain dangling links and are quite likely to be of high quality. This way of recommending documents provides targeted recommendations specific to certain concepts in a document.

One of the advantages of using this system is that it is not restricted to any particular domain and can adapt rapidly to its users' interests. A by-product of trying to achieve this was the innovation of a way to dynamically generate links. In general, link generation in the context of the Web is not an area where much work has been done because of the scale and openness of such a system. Extracting links from Web pages and reusing them in context is a novel idea that, despite the simplicity of the algorithm offered to achieve it, has proved its usefulness.

The presented ideas were brought together through the use of a flexible multiagent architecture. The architecture is open and extensible, supporting the dynamic addition and retraction of agents and services.

The following points highlight some of the advantages of this framework, which haven't already been mentioned:

- Since an agent's functionality was designed to be totally independent of its communication layer to enable an agent to be reused in another architecture,

this layer can by simply replaced with one that is compatible with that architecture. The importance of this flexibility was demonstrated when the need to integrate the QuicLinks agent with the Southampton Framework for Agent Research (SoFAR) (Moreau et al., 2000) arose. The task of writing a different communication layer for that agent was completed in less than three days as part of a three day event that took place within the IAM group (The AgentFest).

- Allowing users to import their bookmarks when they start using the system, enables both the UI agent and other agents such as the OM agent and the QuicLinks agent, to bootstrap rapidly.
- Using the users' ratings to assist them in navigating the Web is also a useful feature especially when it warns users off from re-visiting pages they have already expressed a disliking for. It also provides users with an incentive to rate pages.
- In order to avoid an intrusive interface, small link icons are added to linked phrases to indicate subtly that there are links available for those phrases. Small icons are also used to remind users of what they've previously thought of a given link.

The framework has thus achieved the goals it has set out to address. Personalised and non-intrusive browsing recommendations and hints are realised through the use of links imported based on a user's interests and the use of reminders based on how a user has rated a given URL and which are added proactively to Web pages by UI agents. The users can also interact with the UI agent to assist them with their information finding activities and to request recommendations using a whole set of information related functions for which the UI agent utilises the services of other agents in the system to achieve. The system can be made even more powerful by addressing a number of other issues. These are discussed in the following section.

9.2 Future work

9.2.1 Future System Enhancements

There are a number of features that the author thinks will make the presented system more useful. These are briefly described by the following points:

- **Get feedback on recommended links** such that links that do not meet with the general approval of the users are deleted from the system, and links that do not meet with the specific approval of a particular user, are deleted from his/her linkbase.
- **Allow users to control the contents of their own linkbases**, so that they can immediately delete or add links in a specified context.
- **Employ active bookmarks**. The functionality of the UI agent can be easily extended to provide users with even more powerful recommendations and information finding services through the idea of *active bookmarks*. The UI agent currently offers users an advanced bookmarking facility, yet the usage of bookmarks has not been exploited to the maximum. The principal idea of *active bookmarks* is to allow a user to set any of his/her bookmark categories to be active. Upon doing so, the user's UI agent would 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. The infrastructure for implementing this functionality already exists. The functionality can be achieved by extracting a set of representative keywords from bookmarks in the active bookmark category and dispatching those to search engines or other agents in the system that can perform search operations. Filtering of the results would then be performed based on a profile of the active category.
- **Re-introduce document modifiers** described in section 7.6 while avoiding the limitations also described in that section.
- **Address security**. 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 and as such raises security and privacy issues. Security among agents is an area of research in its own right. However it is an important issue that needs to be addressed within the presented framework.

- **Use the architecture to compare different recommendation techniques.** Since the architecture allows for the ease of building recommender services and services that center around user activities, it can serve as a test bed for comparing various recommendation methods. This is an aspect that has not yet been explored.
- **Look into ways of achieving higher levels of contextualization.** Contextualization in the presented system is achieved by adding links for phrases in a document on the fly, depending on the context of the document. The same links will be provided for a given phrase no matter where it appears in the document. Investigating ways of being more sensitive to the location of a phrase and providing links accordingly can make the links more useful. This could possibly be achieved by tightening clustering rules, which will lead to the creation of more clusters representing more specialised contexts. The more specialised clusters of contexts can then be mapped to document sections. This might result in the provision of different links for the same phrase depending to where it appears in the text.
- **Investigate collaboration among QuicLinks agents.** It is assumed that the presented system would be set up for a small group of users driven by similar goals. Within a single organisation, many such groups of users could exist simultaneously. It would be interesting to explore the use of different QuicLinks agents in a way that would allow one to export links from another, thus capitalising on the experience of a group with established interests.
- **Look into other ways of extracting phrases.** The heuristics presented in chapter 7 for extracting phrases from manually authored links so as to generate links, are very simple ones. Investigating the use of more advanced methods and possibly using natural language techniques for this might yield better results.

- **Investigate means of improving the linking interface** based on user comments presented in chapter 8. This would entail looking into ways of eliminating the flicker that occurs when rendering links in documents as well as ways of organising the suggested links, when there are too many of them.
- **Carry out usability experiments** to evaluate the user interface as a whole.
- **Compare the presented technique for clustering, with other techniques** so as to make sure that what is employed is an optimal technique.

9.2.2 Future Research Directions

There are two ideas that the author believes warrant further long term research and investigation. The first relates to extending the presented *context* model while the second addresses the author's vision as to how the system could scale.

Context is an interesting and exciting area of research, the importance of which is only likely to increase over time. This work has addressed context on the user and document level using a technique that relies heavily on analysing the content of documents for the achievement of both. Using the content of documents as the basis for modelling or determining context seems to work quite when the pages encountered are content rich. However, it is often the case that there will be a failure to use and reference relevant documents because the text content of these documents has failed to capture their context. There are also other cases, when having more knowledge about documents and users, could provide users with better recommendations. Looking into ways of addressing this would make the system more powerful. Investigating the incorporation of a metadata model that would provide a higher level of context information, and require a higher level of reasoning about the contextual factors is one aspect that would assist in this. In general, extending the presented model to take more contextual factors into account would be an interesting future research direction.

To achieve this, the author envisions a system where communities of agents can evolve over time and by exhibiting opportunistic behaviour, make use of the most effective way of answering a query, or making a recommendation at a given point

in time. Agents are very well suited for the achievement of this task. Collectively, agents with different levels of expertise and query answering capabilities can negotiate an answer that best responds to a user's query under a given circumstance, taking advantage of all factors/capabilities that are known to influence the context at that point in time. A system like this needs to address scalability issues, as should the presented one.

The presented system allows for the distribution of users and of information and as long as the system is used by a small group of people, it will be able to cope. However, as the number of users in the system increases, so will the amount of information being created and the system will start to suffer from scalability problems. One way to deal with this issue would be to replicate the agents. Another, would be to dynamically divide users into a number of smaller groups with associated agents, based on their common interests. Replicated agents should be automatically bootstrapped with information that is associated with its users and their interests. This raises interesting research issues as to how to manage the process of dynamic group and agent creation. It also raises the question as to how agents, such as QuicLinks, serving different groups can collaborate with each other so as to be of greatest use to their users.

Appendix A: Result of Testing the Clustering Algorithm

This appendix contains the clusters created as a result of testing the clustering algorithm described in chapter 5. For more information on the experiments, please refer back to section 5.2.2.

Each created cluster is shown in a table, and so is part of its cluster summary in which only the top 25 words of the actual centroid summary appear.

Cluster 1

Cluster Summary: cosm, hypermedia, user, link, application, information, resource, filter, integrate, document, open, message, linkbase, viewer, environment, allow, service, hypertext, architecture, access, figure, distribut, cooperative, introduction, discovery

IRDS and Microcosm
http://www.mmrg.ecs.soton.ac.uk/publications/papers/cstr93/93-18.html#RTFToC9

Introduction
http://www.mmrg.ecs.soton.ac.uk/publications/papers/other95/coop-cstr/cstr-report/node1.html#SECTION00010000000000000000

An Open Framework for Integrating Widely Distributed Hypermedia Resources
http://www.mmrg.ecs.soton.ac.uk/publications/papers/TNG/papers/icmcs96.htm

Cluster 2

Cluster Summary: query, cach, cache, cost, mediator, optimization, snapshot, refresh, database, answer, statistic, stale, distribut, warehouse, scheme, data, dss, enrollment, course, source, decision, enhanc, saharia, environment, differential

ENHANCING DATABASE PERFORMANCE IN A DSS ENVIRONMENT VIA QUERY CACHING
http://hsb.baylor.edu/ramsower/ais.ac.97/papers/babad.htm

Query Caching and Optimization in Distributed Mediator Systems
http://www.cs.umd.edu//projects/hermes/publications/abstracts/cost.html

Cluster 3

Cluster Summary: knowledge, stanford, ksl, ontologi, representation, engineer, base, share, laboratory, paper, ontology, gruber, ontolingua, compositional, ps, kqml, abstract, ontological, shade, defense, language, interlingua, theme, seminar, jump

Stanford Knowledge Systems Laboratory
http://www-ksl.stanford.edu/

Knowledge Sharing Papers
http://ksl-web.stanford.edu/knowledge-sharing/papers/README.html#kif

Cluster 4

Cluster Summary: hci, resource, human, interaction, research, book, center, usability, oversite, michigan, si, job, collection, social, information, jodi, hcrl, crew, collaboratory, journal, informatic, miscellany, interface, university, practitioner

HCI OverSite
http://www-personal.umich.edu/%7Ebing/oversite/hci.html

HCI Resources
http://www.lib.umich.edu/libhome/ILSL.lib/hci.html

Cluster 5

Cluster Summary: electronic, transaction, submit, material, ieee, manuscript, society, title, accept, form, compression, author, guide, information, file, chief, copyright, diskette, acceptable, figure, ftp, format, editor, prefer, guideline

Submitting Materials in Electronic Form
http://www.computer.org/author/transguide/elecsubs.htm

Author Information Guide
http://www.computer.org/author/transguide/

Cluster 6

Cluster Summary: bibliomania, mirror, vist, ibrary, etwork, relocate, library, network, home, thank, bookmark, old, longer, page, main, exist, update, site

BIBLIOMANIA, The Network Library: Home Page
http://www.mk.net/~dt/Bibliomania/Fiction/dickens/greatexp/index.html

BIBLIOMANIA, The Network Library: Home Page
http://www.mk.net/~dt/Bibliomania/

Cluster 7

Cluster Summary: java, tip, javaworld, applet, article, mitchell, class, code, word, john, object, ll, application, chuck, month, jdk, create, javascript, program, bytecode, security, trick, column, summary, developer

DevEdge Online Articles - Tips and Tricks
http://developer.netscape.com/docs/articles/tips.html

How-Tos and Tutorials - JavaWorld Topical Index
http://www.javaworld.com/javaworld/common/jw-ti-tutorials.html

Cluster 8

Cluster Summary: ibm, agent, intelligent, grosof, raise, watson, alphawork, research, project, rule, conflict, paper, wbi, benjamin, abe, webby, dvi, almaden, rc, postscript, pdf, abstract, format, commonrule, colin

IBM Intelligent Agents
http://www.networking.ibm.com/wbi/wbisoft.htm#upgrade

Intelligent Agents Project at IBM T.J. Watson Research
http://www.research.ibm.com/iagents/

Papers -- FORMER Intelligent Agents Project at IBM T.J. Watson Research
http://www.research.ibm.com/iagents/publications.html

Cluster 9

Cluster Summary: sicstu, prolog, logic, sic, program, language, emulator, predicate, constraint, swedish, wam, interface, commercial, finite, implementation, spexception, intro, sp, restore, introduction, throw, sppredicate, jasper, chapter, permission

SICStus Prolog 3
http://www.sics.se/isl/sicstus2.html#Order

SICStus Prolog - Introduction
http://www.jcu.edu.au/docs/sicstus/sicstus_1.html

The World Wide Web Virtual Library: Logic Programming
http://www.comlab.ox.ac.uk/archive/logic-prog.html#Prolog

Class jasper.SICStus
http://www.sics.se/isl/sicstus/jasper/jasper.SICStus.html

Cluster 10

Cluster Summary: prolog, logic, amzi, solari, linux, window, program, command, artificial, bp, webl, lang, expert, isbn, stori, adventure, tar, gz, repository, explorer, freeware, customer, server, intelligenceproduct, emulator]

Topic: lang/prolog/
http://www.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/lang/prolog/0.html

B-Prolog
http://www.cad.mse.kyutech.ac.jp/people/zhou/bprolog.html

Amzi! Prolog+Logic Server, Logic Explorer, WebLS, Adventure in Prolog tutorial, Expert Systems in Prolog, Freeware, Articles, Source Code, Consulting
http://www.amzi.com/

Cluster 11

Cluster Summary: finger, plan, log, file, request, read, gateway, program, brett, planedit, aol, automagic, login, logout, pipe, chang, quote, loop, validateit, hostname, username, write, optionally, false, value

Automagic plan changing and finger logging program
http://www.twisted-helices.com/computing/plan/plan.html#installation

Brett's Web to Finger Gateway
http://www.rickman.com/finger.html

Cluster 12

Cluster Summary: net, semantic, knowledge, engine, search, stenmark, dick, represent, representation, concept, searchenhancer, volvo, relationship, ai, augment, agent, precision, frame, object, prototype, information, management, retrieval, research,filter

Semantic Nets, Frames Objects
http://www.cs.curtin.edu.au/~roger/thesis_html/node15.html

Searching Intranet Using Knowledge Representation (AI, semantic net, search engine, augmented search)
http://w3.informatik.gu.se/~dixi/publ/snet-a.htm

Dick Stenmark - Research interests
http://w3.adb.gu.se/~dixi/research.htm

Cluster 13

Cluster Summary: agent, intelligent, software, intelligence, autonomou, multi, artificial, aaai, symposium, mobile, mae, ai, pattie, proceed, distribut, vol, sycara, workshop, conference, pp, california, pdf, icma, bot, pc

Distributed Artificial Intelligence and Multi-Agent Systems
http://www.informatik.hu-berlin.de/~lukowsky/agents.html

Agent Technologies
http://www.insead.fr/CALT/Encyclopedia/ComputerSciences/Agents/

IEEE Expert Intelligent Agents Article
http://cdr.Stanford.EDU/NextLink/Expert.html

PC AI - Intelligent Agents
http://www.primenet.com/~terry/New_Home_Page/ai_info/intelligent_agents.html

Carnegie Mellon University Robotics Institute - Intelligent Software Agents
http://www.cs.cmu.edu/~softagents/publications.html

MIT Media Lab: Software Agents: Publications
http://agents.www.media.mit.edu/groups/agents/publications/

IC Interview: Pattie Maes
http://computer.org/intern et/v1n4/maes.htm

Cluster 14

Cluster Summary: ontology, knowledge, engineer, formal, model, ontological, representation, cnr, group, theme, document, project, vector, understand, conceptual, pospelov, situational, compartment, set, rule, integration, aggregation, enterprise, natural, finn

Ontology, conceptual modeling, and knowledge engineering
http://www.ladseb.pd.cnr.it/infor/Ontology/ontology.html

Ontology Projects World-Wide
http://www.kr.org/top/projects.html

Chapter 4: Ontology based document understanding:
http://www.acsa2000.net/notion_l.html

Cluster 15

Cluster Summary: recipe, arabic, arab, bean, sauce, east, arabicnew, middle, bread, soup, patti, finely, tablespoon, chop, kibbeh, pound, yogurt, teaspoon, tahini, pepper, salad, meat, overnight, fry, soak

Arabic Recipes and Arab Recipes from Middle East
http://www.arabicnews.com/ansub/Daily/Recipes/Falafel.html

Arabic Recipes and Arab Recipes from Middle East
http://www.arabicnews.com/ansub/Daily/Recipes/RecipesDB.html

Cluster 16

Cluster Summary: swing, article, java, api, persistence, tm, graphic, connection, th, timer, click, dynamically, action, listener, ve, feel, button, xml, jtabbedpane, cardpanel, cardlayout, jdk, toolbar, use, panel

The Swing Connection
http://java.sun.com/products/jfc/swingdoc-current/index.html

The Swing Connection
http://java.sun.com/products/jfc/tsc/index.html

Cluster 17

Cluster Summary: hypermedia, hypertext, object, database, architecture, metadata, space, structure, model, kb, topic, museum, semantic, br, text, ako, newcomer, approach, consortium, mentor, partof, demonstrator, artefact, poster, doctoral

topics
http://www.kom.e-technik.tu-darmstadt.de/~ht99/html/topics.html

A Semantic Database Approach to Knowledge-Based Hypermedia Systems
http://www.comp.glam.ac.uk/pages/research/hypermedia/papr4.htm

ACM Digital Library: Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space structure in hypermedia systems
http://www.acm.org/pubs/contents/proceedings/hypertext/276627/index.html

Cluster 18

Cluster Summary: test, queendom, personality, cyberia, junkie, love, health, psychological, career, webpsych, quizcard, poll, mind, emotional, answer, relationship, credibility, iq, mindgame, seriously, mental, choice, admission, instrument

Tests, Tests, Tests...
http://www.queendom.com/tests.html

Body-Mind QueenDom
http://www.queendom.com/test_col.html

Cluster 19

Cluster Summary: xml, wdvl, javascript, data, html, dhtml, resource, java, structur, perl, cgi, tutorial, introduction, domain, encyclopedia, free, internet, design, middle, commerce, asp, com, developer, icon, br

WDVL: XML: Structuring Data for the Web: An Introduction
http://WDVL.Internet.com/Authoring/Languages/XML/Intro/look_like.html

WDVL: XML: Structuring Data for the Web: An Introduction
http://WDVL.Internet.com/Authoring/Languages/XML/Intro/benefits.html

Cluster 20

Cluster Summary: metadata, data, content, engine, multimedia, identification, semantic, mood, dublin, dc, process, core, automatic, identify, object, rel, dynamic, element, schema, purl, href, meta, identifi, letterform, miller

Metadata for the masses
http://www.ariadne.ac.uk/issue5/metadata-masses/#expert

Automatic and Dynamic Identification of Metadata in Multimedia
http://computer.org/conferen/meta96/adams/paper.html

Cluster 21

Cluster Summary: logicweb, logic, program, lp, prolog, html, url, page, predicate, fact, code, display, structur, mirror, relate, isa, clp, lw, internet, goal, link, module, mosaic, hyperlink, concept

Internet and WWW programming using LP and CLP
http://www.clip.dia.fi.upm.es/lpnet/index.html

Logic Programming with the World-Wide Web
http://www.cs.unc.edu/~barman/HT96/P14/lpwww.html

http://www.cs.mu.oz.au/~swloke/papers/two-level/two-level.html
http://www.cs.mu.oz.au/~swloke/papers/two-level/two-level.html

Cluster 22

Cluster Summary: southampton, bu, university, service, link, transport, uni, information, travel, uk, website, airline, unilink, highfield, bj, accredit, airport, aim, soton, logistic, laid, area, yacht, charter, alphabetical

Southampton University - Community Information: Travel
http://www.soton.ac.uk/services/local/travel.html

Southampton University - Uni-Link Bus Service
http://www.soton.ac.uk/~unilink/

Cluster 23

Cluster Summary: hypertext, literary, robert, hypermedia, darmstadt, award, conference, blustein, introductory, keynote, newcomer, ht, sigweb, humaniti, paper, nelson, cailliau, diversity, workshop, hm, reference, reception, poster, chair, registration

Home
http://www.kom.e-technik.tu-darmstadt.de/~ht99/

Introductory Hypertext References (J. Blustein)
http://www.csd.uwo.ca/~jamie/.Refs/ht-refs.html

Cluster 24

Cluster Summary: fitness, pm, spofitcard, swimfitcard, level, workout, member, room, sport, weight, studio, aerobic, road, induction, circuit, suite, free, thur, choreography, burgess, wed, tue, session, class, card

fit
http://www.soton.ac.uk/~sportrec/fit.html#C21

fit
http://www.soton.ac.uk/~sportrec/fit.html#C2

Cluster 25

Cluster Summary: agent, intelligent, group, mellon, carnegie, matchmaker, university, project, wooldridge, multi, kqml, language, stanford, interface, cooperative, research, multiagent, coordination, shade, maxia, mediator, plant, eil, robotic ,kif

Intelligent Agents: Theory and Practice
http://www.ai.univie.ac.at/~paolo/lva/vu-sa/html/ker95/ker95-html.html

Other agent projects and related information
http://www.lkn.e-technik.tu-muenchen.de/~bq/agents/agents.html

Carnegie Mellon University Robotics Institute - Intelligent Software Agents
http://www.cs.cmu.edu/~softagents/

Agents and Mediators
http://www-ksl.stanford.edu/knowledge-sharing/agents.html

Multiagent Systems Research at EIL
http://www.ie.utoronto.ca/EIL/ABS-page/ABS-intro.html

Cluster 26

Cluster Summary: agent, ontologi, multi, ontology, knowledge, kaw, unipv, paper, abstract, ksl, interaction, protocol, sabina, ipvaim, security, ma, postscript, intelligent, pavia, giordano, architecture, network, sumpy, user, social

Using Ontologies in Multi-Agent Systems - KAW'96
http://ksi.cpsc.ucalgary.ca/KAW/KAW96/falasconi/index.html

http://www.cs.umbc.edu/agents/papers/papers.shtml
http://www.cs.umbc.edu/agents/papers/papers.shtml

Cluster 27

Cluster Summary: bush, vannevar, memex, hypertext, president, think, trail, link, dead, father, hyper, associative, inevitable, code, book, immediately, item, tap, hopefully, form, average, future, article, duchier, mit

As We May Think
http://www.isg.sfu.ca/~duchier/misc/vbush/

Vannevar Bush: Father of Hypertext
http://web.utk.edu/~toto1/vannevarbush.html

HTML Explained... Vannevar Bush
http://www.base22.com/hypertext/bush.html

Vannevar Bush's As We May Think
http://www.kerryr.net/pioneers/memex2.htm

As We May Think and Memex: URL's, hypertext links and Web pages
http://www.home.gil.com.au/~bredshaw/memex2.htm

Vannevar Bush Bio
http://www.ruku.com/vannevar.html

Cluster 28

Cluster Summary: kate, bush, cloudbust, song, album, dreamer, breath, babooshka, army, word, hound, home, hill, music, female, vocal, celestial, voice, spaw, violin, scent, deliu, age, haunt, ethereal

Never For Ever - Kate Bush
http://reality.sgi.com/btd/kate-bush/fnfe.html

Kate Bush searchable Lyrics and MIDI files
Http://reality.sgi.com/btd/kate-bush/

Cloudbusting: Kate Bush In Her Own Words
Http://www.cogsci.ed.ac.uk/~rjc/hyper_cloud/cloudbusting.html

Kate Bush at Celestial Voices
Http://www.loobie.com/kbush.htm

Appendix B: Questionnaires used for the Evaluation Experiment

Questionnaire 1

Please answer the following questions using your favourite search engine, which is: _____

You do not need to answer the questions in order. You have to answer these questions using tools on the Web even if you already know the answers.

The questions you are about to answer are related to the life of Vannevar Bush. How familiar are you with the subject in question? (a) Unfamiliar (b) vaguely familiar (c) familiar (b) expert

1) What were the names of the parents of Vannevar Bush?

_____

Time taken to complete question:
Number of links followed:

2-a) What was the name of Bush's first invention? (hint, it was invented in 1913)

_____

2-b) What did it do?

_____

_____

2-c) Find a URL for a photo of the invention:

_____

Time taken to complete question:
Number of links followed:

3) Name three positions that Vannevar Bush held:

1. _____

2. _____

3. _____

Time taken to complete question:
Number of links followed:

4) What year was Bush married and what was his wife's name?

_____

Time taken to complete question:
Number of links followed:

5-a) How much did the Differential Analyzer weigh ?

_____

5-b) How many vacuum tubes did it contain?

_____

Time taken to complete question:
Number of links followed:

6) Find three URLs to Web pages describing the Memex:

1. _____

2. _____

3. _____

Time taken to complete question:
Number of links followed:

7-a) Where was George W. Bush born?

_____

7-b) What is name of George W. Bush's wife?

_____

7-c) Find two URLs related to her:

1. _____

2. _____

Time taken to complete question:
Number of links followed:

Questionnaire 2

Please answer the following questions using a your favourite search engine, which is:

_____

You do not need to answer the questions in order. You have to answer these questions using tools on the Web even if you already know the answers

The questions you are about to answer are related to CORBA. How familiar are you with the subject in question? (a) Unfamiliar (b) vaguely familiar (c) familiar (b) expert

1) What is CORBA?

_____

_____

Time taken to complete question:
Number of links followed:

2) What is an ORB?

_____

_____

Time taken to complete question:
Number of links followed:

3) Locate two introductory CORBA tutorials that can be easily printed out in one go (pdf or ps formats are acceptable).

1) _____

2) _____

Time taken to complete question:
Number of links followed:

4) What do the following terms stand for in CORBA:
BOA : _____

COS : _____

GIOP : _____

PIDL : _____

SSI : _____

Time taken to complete question:
Number of links followed:

5) Find a Web page that explains why CORBA is better than COM

_____

Time taken to complete question:
Number of links followed:

Questionnaire 3

Please answer the following questions using a search engine and the facilities provided to you by the QuIC user interface agent. These will be described to you before you start your task. Please ignore greyed fields.

Search engine to be used:_____

The questions you are about to answer are related to the life of Vannevar Bush. How familiar are you with the subject in question?
(a) Unfamiliar (b) vaguely familiar (c) familiar (b) expert

1) What were the names of the parents of Vannevar Bush?

_____

Time taken to complete question:
Number of links followed:

2-a) What was the name of Bush's first invention? (hint, it was invented in 1913)

_____

2-b) What did it do?

_____

_____

2-c) Find a URL for a photo of the invention:

_____

Time taken to complete question:
Number of links followed:

3) Name three positions that Vannevar Bush held:

1. _____

2. _____

3. _____

Time taken to complete question:
Number of links followed:

4) What year was Bush married and what was his wife's name?

_____

Time taken to complete question:
Number of links followed:

5-a) How much did the Differential Analyzer weigh ?

_____

5-b) How many vacuum tubes did it contain?

_____

Time taken to complete question:
Number of links followed:

6) Find three URLs to Web pages describing the Memex:

1. _____

2. _____

3. _____

Time taken to complete question:
Number of links followed:

7-a) Where was George W. Bush born?

_____

7-b) What is name of George W. Bush's wife?

_____

7-c) Find two URLs related to her:

1. _____

2. _____

Time taken to complete question:
Number of links followed:

Post task questionnaire

1. Did you find the ways in which links were rendered within documents
(a) More useful than distracting
(b) more distracting than useful
(c) Neither

2. Would you use this system if it were available to you?

_____

3. Please write down any comments that you have

_____

_____

_____

_____

Questionnaire 4

Please answer the following questions using a search engine and the facilities provided to you by the QuIC user interface agent. These will be described to you before you start your task.
Search engine to be used:_____

The questions you are about to answer are related to CORBA. How familiar are you with the subject in question?
(a) Unfamiliar (b) vaguely familiar (c) familiar (b) expert

1) What is CORBA?

_____

_____

Time taken to complete question:
Number of links followed:

2) What is an ORB?

_____

_____

Time taken to complete question:
Number of links followed:

3) Locate two introductory CORBA tutorials that can be easily printed out in one go (pdf or ps formats are acceptable).

1) _____

2) _____

Time taken to complete question:
Number of links followed:

4) What do the following terms stand for in CORBA:
BOA : _____

COS : _____

GIOP : _____

PIDL : _____

SSI : _____

Time taken to complete question:
Number of links followed:

5) Find a Web page that explains why CORBA is better than COM

_____

Time taken to complete question:
Number of links followed:

Post task questionnaire

1. Did you find the ways in which links were rendered within documents
(a) More useful than distracting
(b) more distracting than useful
(c) Neither

2. Would you use this system if it were available to you?

_____

3. Please write down any comments that you have

_____

_____

_____

_____

Appendix C: Clusters Created during the Evaluation Experiment

This appendix includes the set of clusters created as a result of carrying out the evaluation experiment described in chapter 8. Each cluster is shown in a table, and so is part of its cluster summary in which only the top 25 words of the actual cluster summary, appear.

Cluster 1
Cluster Summary: bush, vannevar, president, memex, mit, tuft, roosevelt, engineer, invention, differential, analyzer, machine, national, college, electrical, war, scientific, appoint, navy, rockefeller, bomb, medal, chairman, institute, tracer
Symposium Vannevar Bush at MIT http://alto.histech.rwth-aachen.de/www/quellen/bush/photos.htm
Vannevar Bush http://jefferson.village.virginia.edu/elab/hfl0034.html
Timeline http://www.cs.brown.edu/research/graphics/html/info/timeline.html
Vannevar Bush's Memex http://www.kerryr.net/pioneers/memex.htm
3.2.1 Vannevar Bush's Memex (1945) http://www.iicm.edu/liberation/library/reports/rp_feedback/n39/n41/n42? hyperwave=action%3dplain.action
Bush's Profile Tracer http://www.sinc.sunysb.edu/Class/his352/vbtracer.htm
Vannevar Bush backs the bomb http://www.bullatomsci.org/issues/1992/d92/d92.zachary.html
No. 27: Vannevar Bush's Differential Analyzer http://www.uh.edu/engines/epi27.htm
WP: Vannevar Bush http://english.ttu.edu/kairos/5.2/reviews/carter/bush.html
Vannevar Bush http://www.northstar.k12.ak.us/schools/ryn/projects/inventors/bush/bush.html

Vannevar Bush's Memex - Drawing
http://www.kerryr.net/pioneers/memex_pic.htm

3.2.1 Vannevar Bush's Memex (1945)
http://www.iicm.edu/liberation/library/reports/rp_feedback/n39/n41/n42

Vannevar Bush: Father of Hypertext
http://web.utk.edu/~toto1/vannevarbush.html

Cluster 2

Cluster Summary: bush, texa, governor, laura, george, welch, president, lady, presidential, election, tx, father, republican, childhood, austin, midland, born, taxe, tax, oil, education, px, mr, margin, Texan

George W. Bush
http://www.infoplease.com/spot/georgewbush1.html

George W. Bush for President Official Site: 1946 - Present
http://www.georgewbush.com/bushcheney/georgelaura/photoalbum/one.html

The First Lady
http://www.governor.state.tx.us/FirstLady/biography.html

Laura Welch Bush 1995-
http://www.twu.edu/twu/exhibits/firstladies/b1/b1.html

Personal and Family History of George W. Bush
http://austin.about.com/citiestowns/southwestus/austin/library/weekly/aa010900b.htm

The First Lady
http://www.governor.state.tx.us/FirstLady/

Quest for the Presidency -- Bush Profile
http://quest.amarillonet.com/bush/

Laura Welch Bush: Shy no more
http://www.usatoday.com/news/e98/e2147.htm

Cluster 3

Cluster Summary: memex, bush, record, trail, photocell, vannevar, item, lever, tap, device, thought, bow, form, dot, insert, film, machine, picture, photography, card, electrical, accordance, arithmetical, tube, selection

Vannevar Bush and the Memex
http://calliope.jhu.edu/press/books/landow/memex.html

As We May Think by Vannevar Bush
http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm

Cluster 4

Cluster Summary: corba, object, orb, idl, omg, interface, distribut, server, request, skeleton, stub, tutorial, client, protocol, implementation, iiop, java, dii, giop, boa, poa, api, acronym, remote, broker

CORBA and Java Tutorial: Digging Deeper
http://www.randomwalk.com/~sbarber/owuk98/ShorterJavaCORBADeeperOWUK98/index.htm

Introduction to CORBA
http://developer.java.sun.com/developer/onlineTraining/corba/corba.html

DevDirectory : Java : Articles and Tutorials : CORBA
http://www.devdaily.com/Dir/Java/Articles_and_Tutorials/CORBA/

CORBA Acronyms [UPDATED!]
http://arya.ncst.ernet.in/~shridhar/corba-faq/corba-acronyms.html

CORBA
http://holly.colostate.edu/~mabrake/

CORBA Tutorial Material
http://www.cs.wustl.edu/~schmidt/tutorials-corba.html

CORBA Acronyms [UPDATED!]
http://www.aurora-tech.com/corba-faq/corba-acronyms.html

What is CORBA?
http://corba.ebi.ac.uk/intro.html

CORBA ORB
http://jiio16.jic.bbsrc.ac.uk/BrassicaDB/Seminar/ACEDB_2000/acedb_2000/sld004.htm

A Brief Tutorial on CORBA
http://www.sunderland.ac.uk/~ts0jti/corba-web/corba1/tuto.html

The CORBA Architecture
http://www.cs.aston.ac.uk/support/tutorials/java/tutorial/idl/intro/corba.html

CORBA Tutorial Abstract
http://info.gdb.org/~letovsky/omg.html

CORBA ORB Architecture
http://goesser.mit.edu/CAPAM/memos/96-7/node6.html

CORBA
http://www.cs.umbc.edu/~thurston/cbatop.htm

CORBA GIOP
http://www-cse.ucsd.edu/users/flavio/cse225/slides/tsld037.htm

Bibliography

Abrams, D., Baecker, R. and Chignell, M. (1998) *Information archiving with bookmarks: personal Web space construction and organization*, In Proceedings of *Human factors in computing systems*, ACM, Los Angeles, CA, USA, pp. 41-48.

Akman, V. and Surav, M. (1996). Steps toward formalizing context. *AI Magazine,* **17**(3), pp. 55-72.

Akman, V. and Surav, M. (1997). The Use of Situation Theory in Context Modelling. *Computational Intelligence,* **12**(4), pp. 427-438.

Allan, J. (1996) *Automatic hypertext link typing*, In Proceedings of *the seventh ACM conference on Hypertext*, ACM, Bethesda, MD USA, pp. 42-52.

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

Anderson, K. M. (1997) *Integrating open hypermedia systems with the World Wide Web*, In Proceedings of *the eighth ACM conference on Hypertext*, ACM, Southampton, United Kingdom, pp. 157-166.

Armstrong, R., Freitag, D., Joachims, T. and Mitchell, T. (1995) *WebWatcher: A Learning Apprentice for the World Wide Web*, In Proceedings of *AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, AAAI Press, Stanford, USA.

Asher, R. E. and Simpson, J. M. Y. (Eds.) (1994) *The encyclopaedia of language and linguistics,* Pergamon Press, Oxford.

Ashman, H., Garrido, A. and Oinas-Kukkonen, H. (1997) *Hand-Made and Computed Links, Precomputed and Dynamic Links*, In Proceedings of

Hypertext - Information Retrieval - Multimedia (HIM97), Dortmund, Germany, pp. 191-208.

Ashman, H. L. and Verbyla, J. L. M. (1994) *Dynamic Link Management Via the Functional Model of the Link*, In Proceedings of *Basque International Workshop on Information Technology.*

AskJeeves (1999). AskJeeves. **http://www.askjeeves.com**

Baeza-Yates, R. and Ribeiro-Neto, B. (1999) *Modern Information Retrieval,* Addison-Wesley/ACM Press.

Baeza-Yates, R. A. (1998) *Searching the Web: Challenges and Partial Solutions*, In Proceedings of *String processing and Information Retrieval*, IEEE, Santa Cruz del La Sierra, Bolivia.

Balabanovic, M. (1997) *An Adaptive Web Page Recommendation Service*, In Proceedings of *First International Conference on Autonomous Agents*, Marina Del Ray, CA, USA.

Balabanovic, M. and Shoham, Y. (1995) *Learning information retrieval agents: Experiments with automated web browsing*, In Proceedings of *AAAI Spring Symposium Series.*

Balabanovic, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM,* **40**(3), pp. 66-72.

Barrett, R., Maglio, P. P. and Kellem, D. C. (1997) *WBI a Confederation of Agents that Personalize the Web*, In Proceedings of *Autonomous Agents '97*, pp. 496-499.

Belgrave, M. (1995). The Unified Agent Architecture: A White Paper. **http://www.ee.mcgill.ca/~belmarc/uaa_paper.html**

Berners-Lee, T., Fielding, R., Irvine, U.C. and Masinter, L. (1998). Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396 **http://www.ietf.org/rfc/rfc2396.txt**

Berners-Lee, T. (1998a). Semantic Web Road map. **http://www.w3.org/DesignIssues/Semantic.html**

Berners-Lee, T. (1998b). Web design issues; What a semantic can represent. **http://www.w3.org/DesignIssues/RDFnot.html**

Bernstein, M. (1990) *An Apprentice That Discovers Hypertext Links*, In Proceedings of *ECHT'90*, INRIA, France, pp. 121-223.

Bieber, M., Vitali, F., Ashman, H., Balasubramanian, V. and Kukkonen, H. O.-5. (1997). Fourth Generation Hypermedia: Some Missing Links for the World Wide Web. *International Journal of Human-Computer Studies: Special Issue on "HCI and the Web"*, **47**(1), pp. 31-65.

Booch, G. (1994) *Object Oriented Analysis and Design,* The Benjamin/Cummings Publishing Company, Redwood City, California.

Boy, G. A. (1991) *Indexing Hypertext Documents in Context*, In Proceedings of *Hypertext '91*, pp. 51-61.

Bray, T., Paoli, J. and Sperberg-McQueen, C. M. (1998). Extensible Markup Language (XML) 1.0, In Technical Report, World Wide Web Consortium , http://www.w3.org/TR/1998/REC-xml-19980210.

Brown, P. J. (1996) *The Stick-e Document: a Framework for Creating Context-aware Applications*, In Proceedings of *Electronic Publishing*, Vol. 8 , Palo Alto, CA, USA, pp. 259-272.

Brown, P. J. (1998). Triggering Information by Context. *Personal Technologies,* **2**(1), pp. 18-27.

Budzik, J. (2000) *User Interactions with Everyday Applications as Context for Just-in-time information Access*, In Proceedings of *Intelligent User Interfaces (IUI)*, ACM, New Orleans, LA USA, pp. 44-51.

Budzik, J. a. H., K. (1999) *Watson: Anticipating and Contextualizing Information Needs*, In Proceedings of *Sixty-second Annual Meeting of the American Society for Information Science*, Medford, NJ, USA, pp. 44-51.

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

Carey, M. J., Haas, L. M., Schwarz, P. M., Arya, M., Cody, W. F., Fagin, R., et al. (1995) *Towards Heterogeneous Multimedia Information Systems: The Garlic Approach*, In Proceedings of , the Fifth International Workshop on Research Issues in Data Engineering (RIDE): Distributed Object Management.

Carr, L., Bechhofer, S., Goble, C. and Hall, W. (2001) *Conceptual Linking: Ontology-based Open Hypermedia*, In Proceedings of *the 10th International WWW Conference (in press)*, Hong Kong.

Carr, L. A., DeRoure, D. C., Davis, H. C. and Hall, W. (1998). Implementing an Open Link Service for the World Wide Web. *World Wide Web Journal,* **1**(2), pp. 61-71.

Carr, L. A., DeRoure, 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.

CERN (1998). Hypertext and WWW information. **http://weboffice-old.web.cern.ch/WebOffice-Old/www0/Welcome.html**

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

Clark, D. (1999). Natural language, relevancy ranking, and common sense. *IEEE Intelligent Systems,* **14**(4), pp. 17-19.

Core, T. D. (1999). Dublin Core Metadata Initiative. **http://purl.oclc.org/dc/**

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., et al. (1999). Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence.*

DARPA (2000). DAML. **http://www.daml.org/**

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.

DeBra, P. (1999) *AHAM: A Dexter-based Reference Model for Adaptive Hypermedia*, In Proceedings of *Hypertext'99 Conference*, ACM, Darmstadt, Germany, pp. 147-156.

Delgado, J., Ishii, N. and Ura, T. (1998) *Intelligent Collaborative Information Retrieval*, in H. Coelho (Ed.): (1998) Progress in Artificial Intelligence -

IBERAMIA'98. Springer-Verlag, Lecture Notes in Artificial Intelligence Series No. 1484.

DeRose, S. J. (1989) *Expanding the Notion of Links*, In Proceedings of *Hypertext*, ACM, Pittsburgh, PA USA, pp. 249-257.

DeRoure, D., Carr, L., Hall, W. and Hill, G. (1996) *A Distributed Hypermedia Link Service*, In Proceedings of *The Third International Workshop on Services in Distributed and Networked Environments (SDNE'96)*, Macao, pp. 156-161.

DeRoure, 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. 624-625.

DeRoure, 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. *Information Processing & Management. An International Journal.*

El-Beltagy, S. (1998). Approaches to System Integration For Distributed Information Management, In Technical Report, University of Southampton, Southampton, UK , ECSTR MM98/7.

El-Beltagy, S., DeRoure, D. and Hall, W. (1999a) *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, pp. 281-295.

El-Beltagy, S., DeRoure, D. and Hall, W. (2000a) *The Evolution of a Practical Agent-based Recommender System*, In Proceedings of *Workshop on Agent-based Recommender Systems, Autonomous Agents 2000*, ACM, Barcelona, Spain.

El-Beltagy, S., DeRoure, D. and Hall, W. (2000b) *Where Do I Go From Here: A Multiagent System To Support Collaborative Web Browsing*, In Proceedings of *the Ninth International World Wide Web Conference(Poster Proceedings)*, Amsterdam, the Netherlands, pp. 40-41.

El-Beltagy, S., Hall, W. and Roure, D. D. (1999b). Tools for Facilitating Linking to and Searching for Image Regions on The Web, In Technical Report, University of Southampton, Southampton, UK , ECSTR M99/1.

Englebart, D. and English, W. K. (1968) *A Research Center for Augmenting Human Intellect*, In Proceedings of *AFIPS*, Vol. 33, Fall Joint Computer Conference.

Fensel, D., Decker, S., Erdmann, M. and Studer, R. (1998) *Ontobroker: The Very High Idea*, In Proceedings of *11th International Flairs Conference (FLAIRS-98)*, Sanibal Island, Florida.

Ferraiolo, D., Cugini, J. and Kuhn, R. (1995) *Role-Based Access Control (RBAC): Features and Motivations*, In Proceedings of *the Annual Computer Security Applications Conference*, IEEE Computer Society Press.

Finin, T., Fritzson, R. and Mckay, D. (1992) *A Language and Protocol to Support Intelligent Agent Interoperability*, In Proceedings of *Fourth National Symposium on Concurrent Engineering, CE & CALS*, Washington, USA.

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.

FIPA (1996). Foundation for Intelligent Physical Agents. **http://www.fipa.org/**

Flammia, G. (1999). The skinny on metadata. *IEEE Intelligent Systems,* **14**(4), pp. 20-22.

Foner, L. N. (1997) *A Multi-Agent Referral System for Matchmaking*, In Proceedings of *The First International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology*, London, UK.

Fountain, M. A., Hall, W., Heath, I. and Davis, H. C. (1990) *MICROCOSM: An Open Model for Hypermedia with Dynamic Linking*, In Proceedings of *Proceedings of Hypertext: Concepts, Systems, and Applications (ECHT'90)*, (Eds, Rizk, A. and Andre, J.), Cambridge University Press, pp. 298-311.

Franklin, S. and Graesser, A. (1996) *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*, In Proceedings of *Third International*

Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag.

Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. and Widom, J. (1995) *Integrating and Accessing Heterogeneous Information Sources in TSIMMIS*, In Proceedings of *AAAI Symposium on Information Gathering*, Stanford, California, USA, pp. 61-64.

Geddis, D. F., Geneserth, M. R., Keller, A. M. and Singh, N. P. (1995) *Infomaster: A Virtual Information System*, In Proceedings of *the Intelligent Information Agents Workshop at SIKM'95*.

Genesereth, M. and Fikes, R. (1992). Knowledge interchange format, version 3.0 reference manual, In Technical Report, Computer Science Department, Stanford University , Logic-92-1.

Genesereth, M. R. and Ketchpel, S. P. (1994). Software Agents. *Communications of the ACM,* **37**(7), pp. 48-53.

Golovchinsky, G. (1997a) *Queries? Links? Is there a difference?*, In Proceedings of *CHI'97*, ACM, Atlanta, GA USA, pp. 407-414.

Golovchinsky, G. (1997b) *What the query told the link: the integration of hypertext and information retrieval*, In Proceedings of *The eighth ACM conference on Hypertext*, ACM, Southampton, United Kingdom, pp. 67-74.

Google (1999). The Google Search Engine. **http://www.google.com**

Google (2001). Google Search Technology. **http://www.google.com/technology/**

Grønbæk, 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.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition,* **5**(2), pp. 199-220.

Guarino, N. (1998) *Formal Ontology and Information Systems*, In Proceedings of *FOIS'98*, (Ed, Guarino, N.), IOS Press, Trento, Italy.

Guarino, N., Masolo, C. and Vetere, G. (1999). OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems,* **14**(3), pp. 70-80.

Gudivada, V. N., Raghavan, V. V., I., G. W. and Kasanagottu, R. (1997). Information Retrieval on the World Wide Web. *IEEE Internet Computing,* **1**(5), pp. 58-68.

Halasz, F. and Schwartz, M. (1994). The Dexter Hypertext Reference Model. *Communications of the ACM,* **37**(2), pp. 30-39.

Hardman, L., Bulterman, D. C. A. and Rossum, G. v. (1993) *Links in Hypermedia: the Requirement for Context*, In Proceedings of *Hypertext '93*, pp. 183-191.

Heflin, J. and Hendler, J. (2000) *Searching the Web with SHOE*, In Proceedings of *AAAI-2000 Workshop on AI for Web Search.*

Hendler, J. (1999). Making Sense out of Agents. *IEEE Intelligent Systems,* **14**(2), pp. 32-37.

Huhns, M. N. and Singh, M. P. (1997a). The Agent Test. *IEEE Internet Computing,* **1**(5), pp. 78-79.

Huhns, M. N. and Singh, M. P. (1997b). Ontologies For Agents. *IEEE Internet Computing,* **1**(6), pp. 81-83.

Introna, L. and Nissenbaum, H. (2000). Defining the Web: The Politics of Search Engines. *IEEE Computer,* **33**(1), pp. 54-62.

Jennings, N. R. and Wooldridge, M. (1995). Applying Agent Technology. *Applied Artificial Intelligence,* **9**(4), pp. 357-369.

Joachims, T., Frietag, D. and Mitchell, T. (1997) *WebWatcher: A Tour Guide for the World Wide Web*, In Proceedings of *IJCAI97.*

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

Katz, B. (1997) *From Sentence Processing to Information Access on the World Wide Web*, In Proceedings of *AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, Stanford University, Stanford CA.

Khedro, T. and Genesereth, M. R. (1995) *Facilitators: A Networked Computing Infrastructure for Distributed Software Interoperation*, In Proceedings of *Workshop on AI in Distributed Information Networks, IJCAI-95*, Montreal, Quebec.

Klusch, M. (2001). Information Agent Technology for the Internet: A Survey. *Journal on Data and Knowledge Engineering, Special Issue on Intelligent Information Integration,* **36**(3).

Knoblock, C. A. and Ambite, J. L. (1997) In *Software Agents*(Ed, Bradshaw, J.) AAAI/MIT Press, Menlo Park, CA, USA.

Krulwosh, B. and Burkey, C. (1997). The InfoFinder Agent: Learning User Interests through Heuristic Phrase Extraction. *IEEE Expert/Intelligent Systems & Their Applications,* **12**(5), pp. 22-27.

Labrou, Y. and Finin, T. (1997). A Proposal for a new KQML Specification, In Technical Report, UMBC.

Lawrence, S. and Giles, C. L. (1998). Context and Page Analysis for Improved Web Search. *IEEE Internet Computing,* **2**(4), pp. 38-46.

Lelu, A. and Francois, C. (1992) *Hypertext paradigm in the field of information retrieval: a neural approach*, In Proceedings of *ACM European Conference on Hypertext ECHT'92*, ACM, Milan, Italy, pp. 112-121.

Lesser, V. (1995). Multiagent Systems: An Emerging Subdiscipline of AI. *ACM Computing Surveys,* **27**(3), pp. 340-342.

Lessila, O. (1998). Web Metadata: A Matter of Semantics. *IEEE Internet Computing,* **2**(4), pp. 30-37.

Lessila, O. and Swick, R. R. (1999). Resource Description Framework(RDF) Model and Syntax Specification, In Technical Report, World Wide Web Consortium , http://www.w3.org/TR/PR-rdf-syntax/.

Levy, A. Y., Rajaraman, A. and Ordille, J. J. (1996) *Querying Heterogeneous Information Sources Using Source Descriptions*, In Proceedings of *the 22nd International Conference on Very Large Databases (VLDB-96)*, Bombay, India.

Lewis, P. H., Davis, H. C., Griffiths, S., Hall, W. and Wilkins, R. J. (1996) *Media-based Navigation with Generic Links*, In Proceedings of *the Seventh ACM conference on Hypertext*, ACM, Washington, USA, pp. 215-223.

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

Lieberman, H. (1997) *Autonomous Interface Agents*, In Proceedings of *the ACM Conference on Computers and Human Interface, CHI-97*, Atlanta, Georgia, USA.

Lowe, D. and Hall, W. (1999) ***Hypermedia and the Web***, John Wiley and Sons.

Luke, S. and Hendler, J. (1997). Web Agents that Work. *IEEE Multimedia,* **4**(3), pp. 76-80.

Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM,* **37**(7), pp. 31-40.

Maglio, P. P. and Barret, R. (1997) *How to build Modelling Agents to Support Web Searchers*, In Proceedings of *The sixth international conference on User Modeling*, (Eds, Jameson, A., Paris, C. and Tassso, C.), Springer.

Maler, E. and DeRose, S. J. (1999). XML Linking Language (XLink), In Technical Report, World Wide Web Consortium , http://www.w3.org/TR/1998/WD-xlink-19980303.

Marchiori, M. (1998) *The Limits of Web Metadata, and Beyond*, In Proceedings of *the Seventh International World Wide Web Conference (WWW7)*, Brisbane, Australia.

McCarthy, J. and Buvac, S. (1994). Formalizing Context (Expanded Notes), In Technical Report, Computer Science Department, Stanford University , Stanford, CA , STAN-CS-TN-94-13.

McHugh, J., Abiteboul, S., Goldman, R., Quass, D. and Widom, J. (1997). Lore: A Database Management System for Semistructured Data. *SIGMOD Record,* **26**(3), pp. 54-66.

Miller, P. (1996). Metadata for the masses. **http://www.ariadne.ac.uk/issue5/metadata-masses/**

Mladenic, D. (1999). Text-Learning and Related Intelligent Agents: A Survey. *IEEE Intelligent Systems,* **14**(4), pp. 44-54.

Moreau, L., Gibbins, N., DeRoure, D., El-Beltagy, S., Hall, W., Hughes, G., et al. (2000) *SoFAR with DIM Agents: An Agent Framework for Distributed Information Management*, In Proceedings of *the Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, Manchester, UK.

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.

Mukherjea, S. and Hara, Y. (1997) *Focus+Context of World-Wide Web Nodes*, In Proceedings of *Hypertext 97*, Southampton, UK.

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

Nielsen, J. (1999). User Interface Directions For the Web. *Communications of the ACM,* **42**(1), pp. 65-72.

Nodine, M. and Unruh, A. (1997). Facilitating Open Communication in Agent Systems: the InfoSleuth Infrastructure, In Technical Report, MCC , MCC-INSL-056-97.

Nwana, H. S. (1996). Software agents: an overview. *The knowledge Engineering Review,* **11**(3), pp. 205-244.

Nwana, H. S. and Ndumu, D. T. (1999). A Perspective on Software Agents Research. *The Knowledge Engineering Review,* **14**(2), pp. 1-18.

Pazzani, M., J., M. and Billsus, D. (1996) *Syskill & Webert: Identifying interesting web sites*, In Proceedings of *National Conference on Artificial Intelligence*, Portland, OR.

Pikrakis, A., Bitsikas, T., Sfakianakis, S., Hatzopoulos, M., DeRoure, D. C., Hall, W., et al. (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.

Porter, M. (1980). An algorithm for suffix stripping. *Program,* **14**(3), pp. 130 -137.

Rapoza, J. (2000). DAML could take search to a new level. *PC Week,* , pp. http://www.zdnet.com/eweek/stories/general/0,11011,2432538,00.html.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. (1994) *GroupLens: an open architecture for collaborative filtering of netnews*, In Proceedings of *the conference on Computer supported cooperative work*, Chapel Hill, United States, pp. 175-186.

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

Rhodes, B. J. (2000) *Margin Notes: Building a Contextually Aware Associative Memory*, In Proceedings of *Intelligent User Interfaces (IUI '00)*, ACM, New Orleans, LA USA, pp. 219-224.

Rhodes, B. J. and Maes, P. (2000). Just-in-time infromation retrieval agents. *IBM Systems Journal,* **39**(3/4), pp. 685-704.

Riecken, D. (2000). Personlized Views of Personlization. *Communications of the ACM (special issue on personalization),* **43**(9), pp. 27-28.

Robertson, J., Merkus, E. and Ginige, A. (1994) *The Hypermedia Authoring Research Toolkit (HART)*, In Proceedings of *ECHT'94*, ACM, Edinburgh, Scotland, UK, pp. 177-185.

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

Russell, S. J. and Norvig, P. (1995) *Artificial Intelligence: A Modern Approach,* Prentice Hall, Englewood Cliffs, NJ.

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

Shardanand, U. and Maes, P. (1995) *Social Information Filtering: Algorithms for Automating "Word of Mouth"*, In Proceedings of *Conference on Human factors in computing systems*, ACM, Denver, CO USA, pp. 210-217.

Shoham, Y. (1999). What we talk about when we talk about software agents. *IEEE Intelligent Systems,* **14**(2), pp. 28-31.

Singh, M. P. (1998). Agent Communication Languages: Rethinking the Principles. *IEEE Computer,* **31**(12), pp. 40-47.

Stairmand, M. A. (1997) *Textual Context Analysis for Information Retrieval*, In Proceedings of *SIGIR*, ACM, Philadelphia, USA, pp. 140-147.

Theodorakis, M., Analyti, A., Constantopoulos, P. and Spyratos, N. (1998) *Context in Information Bases*, In Proceedings of *The Third International Conference of Cooperative Information Systems*, New York, USA.

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

W3Consortium (1999). Tim Berners-Lee.

http://www.w3.org/People/all#timbl%40w3.org

Wexelblat, A. and Maes, P. (1999) *Footprints: history-rich tools for information foraging*, In Proceedings of *CHI 99 conference on Human factors in computing systems: the CHI is the limit*, ACM Press, Pittsburgh, PA USA, pp. 270-277.

Wiil, U. K. (1997). Open Hypermedia: Systems, Interoperability and Standards. *Journal of Digital information, Special Issue on Open Hypermedia,* **1**(2).

Wilkinson, R. and Smeaton, A. F. (1999). Automatic Link Generation. *ACM Computing Surveys,* **31**(4).

Wooldridge, M. and Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review,* **10**(2).