Linking in context[*]

Samhaa El-Beltagy, David De Roure, Wendy Hall and Leslie Carr

Intelligence, Agents, Multimedia

Department of Electronics and Computer Science

University of Southampton

Southampton SO17 1BJ, UK

+44 (0)23 8059 2418

{seb,dder,wh,lac}@ecs.soton.ac.uk

Abstract

This paper presents the idea of 'linking in context' as a novel way of offering Web users recommendations for concepts related to what they are browsing. The model presented utilises, extends, and combines ideas from open hypermedia with those from Web assistants and recommender systems to deliver a personalised Web recommendation system. The integration of the various concepts is facilitated by the use of a multi-agent framework. The automatic creation and maintenance of links is an important aspect of the presented work. A technique whereby links are mined from the Web and stored in context in a way that enables link re-use is also outlined.

Keywords: Software agents, open hypermedia, links, information finding, navigation assistance, recommender systems, user profiling, context

1 Introduction

The problem of information overload on the Web is one which has inspired much research on various fronts. On the personal and collaborative level, Web based software agents and recommender systems have been particularly promising in addressing the problem. This class of applications emerged to assist users with their information finding and navigation activities. A

[*] University of Southampton. Department of Electronics & Computer Science, University of Southampton technical report ECSTR-IAM00-6, ISBN: 0854 32 7304

large number of these systems were capable of modifying Web pages on the fly in a way that enabled them to add document recommendations and navigation hints to these pages. Very little work however, has been done to add recommendations to suggest documents relevant to concepts represented by phrases in Web pages based on the context of the page and the interests of a user. The philosophy of abstracting links from Web pages and storing them in linkbases, adopted by open hypermedia systems, offers a way in which to achieve this. However, systems that brought open hypermedia concepts to the Web lacked the necessary intelligence to infer user interests and determine document context.

This paper presents a framework where concepts from open hypermedia and recommender systems are combined within a multi-agent architecture to support the generation and delivery of links in context. Recommendations provided to users are rendered as links to phrases in the context of a document a user is browsing provided that the user is interested in that context. A technique whereby links are mined from Web pages for re-use through contextulized linkbases is also presented.

The presented framework has both collaborative and personal dimensions. Rather than enforce the collaborative aspect of the system as opposed to its personal aspect, the system attempts to utilise all the activities of a single user in a way that would directly benefit that user. However, most of these activities are also utilised for the benefit of other users in the system, but the main idea enforced in the design and implementation of the framework is that while serving themselves firstly, users will still be capable of serving others. The social aspect is but a by-product of users attempting to help themselves within a group setting.
In section 2 relevant background to the research is presented, then in section 3 an overview of how agents interact to establish link and document context, is provided. Sections 4, and 5 provide a detailed description of two agents that are particularly important for the achievement of the stated goals; these are the *Link Extraction and Contextulizer Agent* and the *User Interface Agent* respectively. Finally, section 6, concludes the paper and presents future work.

2 Background

The work presented in this paper draws on research from open hypermedia as well as from concepts from recommender systems. 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). 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 allows for the addition of hypermedia functionality to documents, multimedia or otherwise, without changing the original document's format or embedding mark-up information within it. It also simplifies link maintenance and re-use (Davis et al., 1992). The recently proposed XLink standard (Maler and DeRose, 1999) is increasingly moving towards the open hypermedia approach. Systems such as the *Distributed Link Service* (DLS) (Carr et al., 1995; DeRoure et al., 1996) and Webvise (Gronbaek et al., 1999) were designed in order to bring the open hypermedia philosophy to the WWW community.

The DLS is based on a now classic OHS system called Microcosm (Fountain et al., 1990; Davis et al., 1992). The Microcosm system pioneered the idea of building a hypertext system out of a set of loosely-coupled communicating processes. The processes could be pre-configured so that a particular user of a particular Microcosm application could use specified link databases as befitting the context of their activity. One type of a link introduced in Microcosm is the generic link. Generic links allow a link to be followed from any occurrence of a text string, to destination anchors. By employing generic links, a great amount of re-authoring is avoided, and link re-use is greatly enhanced. Through the employment of proxy technology, the DLS allows links, mostly generic, from linkbases to be applied to WWW documents on the fly.

However, the DLS on the Web suffered from a major limitation. 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, which was implicit in Microcosm, fell on the shoulders of the user. So the major limitation of the system lied in its inability to perform automatic context switching when rendering links.

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 a mechanism that can automate word of mouth. 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 in the system. DLS was deployed in the MEMOIR recommender system (DeRoure et al., 2000), but MEMOIR also failed to explore linking in context as presented here.

3 Establishing Context: an overview

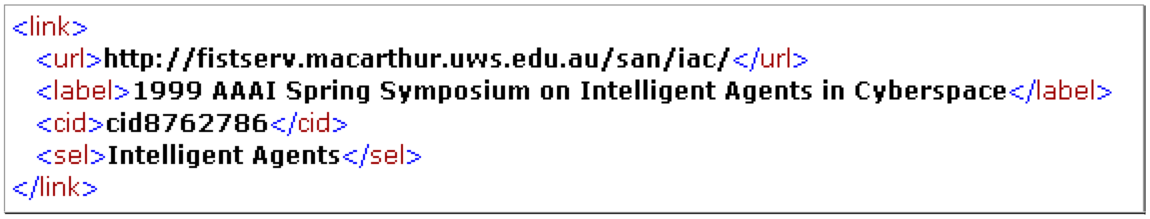
One of the goals of the presented work is to facilitate linking in context as a way of making recommendations. The overall objective is simply to assist users in a group setting with their information finding and navigation activities. To this end, a collaborative multiagent framework that can support agents working towards distributed information management in the context of open hypermedia, was implemented. Details of the architecture and its implementation can be found in (El-Beltagy et al., 1999). 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 achieve some goals (Lesser, 1995). Agents in the implemented system can be viewed as information producers and consumers. For example, a user interface agent acquires knowledge about what different users are viewing and with the permission of the user, shares that information with others. It then uses the services offered by agents that utilise this information, to find out about other users' interests or to obtain a recommendation for a document, among other things.

Within that framework, agents exchange messages using the knowledge query and manipulation language KQML (Finin et al., 1994; Labrou and Finin, 1997). When agents come online for the first time, they register with an agent server using the KQML register performative. They then advertise their services which are broadcast to all registered agents. As soon as an agent registers, it also receives all active advertised messages that were previously sent to it by other registered agents.

Two types of agents are involved in the process of establishing link context: the user interface agent and the link extraction and contextulizer agent, henceforth referred to as *UI Agent* and the *Clinks* (Context Links) agent respectively. Descriptions of other agents in the system, which also contribute to information finding and navigation assistance, can be found in (El-Beltagy et al., 2000; El-Beltagy et al., 1999). A brief description of the agents and the context determination mechanism is given in this section. A detailed description of each of the agents follows in next two sections.

3.1 The Interaction scenario

Within the presented framework, each user is associated with a UI Agent, which acts on his/her behalf. The UI Agent is basically responsible for monitoring a user's various Web related activities and employing those in determining the user interests which it stores in a user profile. Once the agent detects a new user interest, it sends a message to the *CLinks* agent in which it subscribes to the service of being informed of any links related to that interest. User interests are modelled by feature vectors of terms. The content of the KQML message sent to the *CLinks* agent is an XML representation of the vector denoting the user interest for which link subscription is being requested. The *CLinks* agent responds by sending the agent all existing links, also represented by XML and whenever a new link related to that context is added to the *CLinks* knowledge base, it is dispatched to the UI agent for storage in the user's personal linkbase. Figure 1 shows the XML representation of a link as it would be delivered in the content of a KQML message. The *sel* field represents the concept for which the link can be recommended. The next time the user views a Web page that matches any of the contexts in the linkbase, links in that context are rendered to phrases in that document where applicable. Figure 2, shows the interaction between the two agents, details of which are given in the following sections.



```
<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 1: XML representation of a link as used in message exchange between agents

3.2 Determining the 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, one of the goals was implicit capture of user context or document context in order to achieve a specific task. Examples of machine learning techniques that have been used by Web agents include Decision trees, Neural Nets, Bayesian classifiers, Nearest Neighbour and TF-IDF (Mladenic, 1999). TF-IDF (term frequency, inverse document frequency) is in fact an information retrieval technique for representing a document (Salton, 1988). Used in conjunction with a similarity function, it can be employed in learning models. 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 need large data sets for learning. By representing a document through a vector space model computed via TF-IDF, comparing the 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).

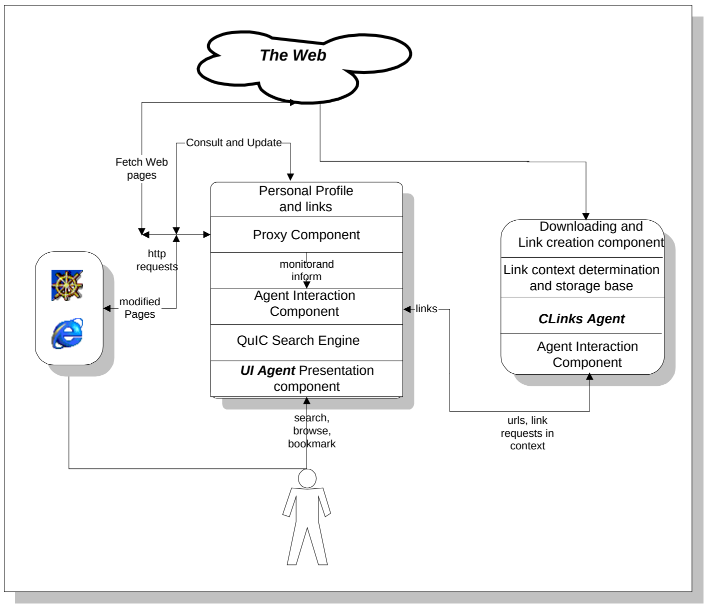

The Web
Consult and Update
Fetch Web pages
Personal Profile and links
Proxy Component
monitorand inform
Agent Interaction Component
QuIC Search Engine
UI Agent Presentation component
http requests
modified Pages
links
Downloading and Link creation component
Link context determination and storage base
CLinks Agent
Agent Interaction Component
search, browse, bookmark
urls, link requests in context

Figure 2: Interaction between the UI agent and the *CLinks* Agent

Using TF-IDF in conjunction with the cosine similarity function seemed well suited for our context determination and user modelling 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 would be calculated by averaging vectors of all documents in a given cluster (Salton, 1988). The employment of the centroid in the user modelling task, is essentially different from its utilisation for the link determination task as will be detailed in the next 2 sections. However, in both cases, it is imperative that only related documents are grouped together. Initial experimentation employed the following algorithm to incoming document vector representations calculated using TF-IDF:

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

Let ***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 $\boldsymbol{c_i}$ calculate Sim($\boldsymbol{c_i}$,, *inDoc*) using the cosine similarity function
2. If maximum similarity across ***C*** is greater then threshold value μ then, add and average weights in inDoc to the vector centroid $\boldsymbol{c_j}$ with greatest similarity,
   Sort the weights for the new vector centroid, trim to maxLength, and store results in $\boldsymbol{c_j}$
3. else, repeat step 2 across ***UD***. If a document d_i is found, then create a new centroid with the resulting vector and remove d_i from ***UD***
4. else, add inDoc to ***UD***

When experimenting with the technique, there were some cases for which it was incapable of distinguishing between documents representing different concepts. For example, a number of documents related to Vannevar Bush[1] where considered similar to some documents related to Kate Bush[2] because some of the highest weighted words were similar. In our application, this mix up was all together unacceptable as Vannevar Bush and Kate Bush represent two different contexts or entities, and they had to be recognised as such. In conjunction with the cosine similarity function, a very simple heuristic was applied and seems to have solved the problem and enabled proper determination of context. So, in order to determine whether two documents were 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

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

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 2 feature vectors being compared if they are to be considered similar.

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

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

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 our current implementation, α is set to 1.5.

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 2 small manually selected document sets contents of which were confused for each other before the addition of the presented heuristics. One set contained 7 documents relating to Vannaver Bush, while the other had 4 which related to Kate Bush. Over the entire data set, 28 clusters were created with 79 documents. Of the 7 documents relating to Vannevar Bush, 5 formed a cluster while the other 2, which were content poor, remained unclustered. All 4 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 approached 100%.

4 The Link Extraction and Contextulizer Agent

The link extraction and contextulizer agent (*CLinks* Agent), is based on the open hypermedia philosophy of abstracting links from documents. 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 expresses links slightly differently from a DLS in its equivalent representation of a linkbase. The agent is capable of compiling contextual information about various Web pages and using those to answer queries in context. A query for this agent could be a request for a set of links for a given context or simply 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)
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 motivation for this agent follows from the fact that very little work has been aimed at providing Web recommendations on a localised level within a document. Margin Notes, a system that was developed at MIT, is one of very few systems which have attempted to achieve this (Rhodes, 2000). Margin notes modifies Web pages as they are being loaded by adding links from the pages to personal files. 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. 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. However Margin notes failed to create links on the level of concepts represented by phrases. This task might seem difficult for the following reasons: automatically creating a link to a phrase might imply that there is a requirement to understand something about the phrase and the document to which it links. Specifically, it raises the question as to how to create links to phrases within a large body of text, when there is little understanding of the text itself. The phrases to which links would be created would usually denote concepts. If links are to be created by the process of extraction from Web pages, then another problem arises: links do not appear as simple links to a given concept. If a link appears in a Web page with the text *'More on Vannevar Bush',* then a link extraction and creation algorithm should be able to understand that the concept to generalise this link for is *Vannevar Bush*.

To address these problems, a simple rule based link extraction algorithm was developed. Even though the algorithm employs simple rules and heuristics, it is effective in extracting links from Web pages in a way that allows them to be applied to other Web pages. The underlying premise is, that if document ***X*** and document ***Y*** appear in context ***Z***, and there is a link to concept ***C*** in document ***X***, then the link can be applied to concept ***C*** in document ***Y***. This applies to all documents in context ***Z***.

One of the advantages of using this technique as opposed to comparing the similarity of a document or sections of a document to other documents, is that enables the addition of links to URLs with little or no text content, but which are still relevant. For example, after modifying a Kate Bush Web page using mined links, one of the links added to the phrase *Red Shoes (*which is a title to an album, and a song*)*, was a URL pointing to a playable midi file of the song.

4.1.1 Creating the links

The *CLinks* agent is responsible for providing links to a group of users. It must therefore create links that fit the interests of these users. It must also obtain the starting points for its links from Web pages that are of reasonably high quality. 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. The best way of achieving both goals is to request notification about Web pages that users have found interesting. In that sense, the agent is dependent on UI agents in the system to convey information correctly about the interests of their users. The process by which they determine such interests is totally transparent to this agent. The agent simply receives the interest in the form of one or more URLs. Once a URL is received, 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. 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 weighted terms). Each link is represented by a *link url* and *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 3.2
6. Pre-process links. This is done as follows: For each extracted link,
   - Check if the link text starts with a URL prefix (http://, ftp://, mailto, etc). If it does, delete
   - Check if the link text is made up entirely of Web stop words (next, back, home, etc), If it does, delete
   - Check if it the link 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 link text into various phrases by considering the text as phrases separated by stop words. For example, for link text *Vannevar Bush and the Memex*, the two phrases *Vannevar Bush*, *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 next fragment or link. If only 1 word appears, and it is in the top *n* features, or if more than 1 appear, then consider this a strong link. Otherwise, consider this a weak link.
9. Create the link. A link is defined by a source URL, a destination URL, the selection it can replace, the text associated with the link (this is the un-fragmented version of the original link text), keywords associated with it (this is used for searching), and the strength of the link. For the example given in step 7, two strong links will be created, one with the selection field *Vannevar Bush* and another with the *Memex.* A link renderer can then use this information to create a link to either concept whenever it encounters it in a similar context, as will is described in the next section.
10. Store the link in the Prolog KB. If the 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.

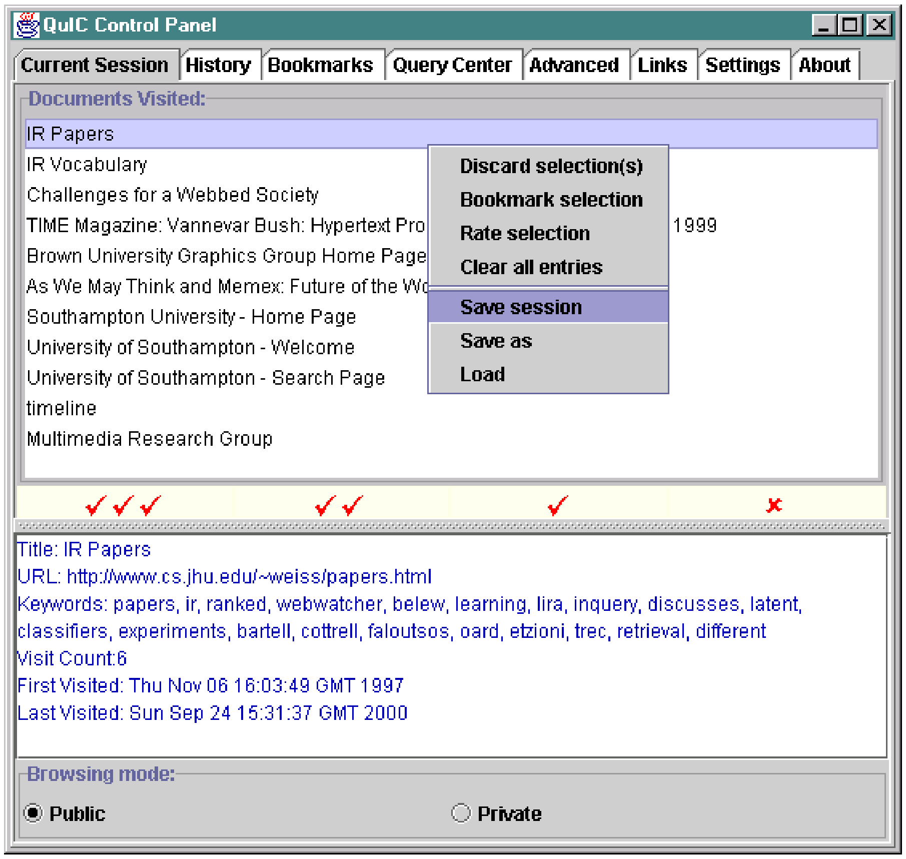
In order to maintain the Prolog KB, the idea of documents, clusters/contexts, and links having a life cycle was introduced. The lifetime of any of these is a function of usage. They are forgotten depending on the frequency of usage and recency of access. The notion of weak and strong links, was introduced to reflect how sure the agent is that a link is relevant to a context.

5 The User Interface Agent

The user interface agent is one of the most complex agents within the system. The agent was implemented as a Java desktop application. Figure 3 shows a snapshot of the interface offered to a user. The basic functions of this agent can be summarised as follows:

- It watches over the user's shoulder and monitors his/her navigation, searching, 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.
- If a service can change the way in which a normal document is presented, the UI agent is responsible for propagating those changes to the user.
- It provides a number of tools to facilitate browsing and bookmarking
- It maintains a personal, context aware linkbase which it populates based on user interests



IR Papers
IR Vocabulary
Challenges for a Webbed Society
TIME Magazine: Vannevar Bush: Hypertext Pro ... 1999
Brown University Graphics Group Home Page
As We May Think and Memex: Future of the Wo
Southampton University - Home Page
University of Southampton - Welcome
University of Southampton - Search Page
timeline
Multimedia Research Group

Title: IR Papers
URL: http://www.cs.jhu.edu/~weiss/papers.html
Keywords: papers, ir, ranked, webwatcher, belew, learning, lira, inquery, discusses, latent, classifiers, experiments, bartell, cottrell, faloutsos, oard, etzioni, trec, retrieval, different
Visit Count:6
First Visited: Thu Nov 06 16:03:49 GMT 1997
Last Visited: Sun Sep 24 15:31:37 GMT 2000

[✓ Public] Private

Figure 3: A snapshot of the User Interface Agent

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 activates and de-activates some of its services according to its interaction with other agents.

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

For each visited URL, the proxy server creates a URL object that is used by many components of the system. The URL object contains the title of a page, the date a page was first visited, the date it was last visited, keywords associated with a URL which are extracted using an extraction algorithm, and a list of image URLs contained in the page. The role of the proxy server 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.

The agent also 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. This 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 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. This acts as an incentive for users to rate documents. The ratings also reflect a user's interest, and thus contribute to the user profile as detailed in the next subsection.

An advanced bookmarking facility is also provided.   A bookmark is defined in terms of a title, URL, keywords, category, private comment, and a public comment. The bookmark facility allows users to drag and drop one or more URLs from their currently open session into their bookmarks. Duplicate bookmarks are not allowed and the user is warned when any come up. The user can edit the keywords extracted using a keyword extraction algorithm, thus adding keywords that are implicit or removing those that are not really relevant. As in browsing, the user has a public and private option when creating a bookmark. This controls the publishing of bookmarks to interested agents. Currently implemented agents use this information for providing recommendations for other users (El-Beltagy et al., 1999; El-Beltagy et al., 2000).

A query interface component, whereby the user can access query services provided by the different agents in the system, is among the most important components of the interface agent. To enter the user's personal information as well as to allow the user a certain degree of control over the agent and its interactions with the outer world, a setting panel is provided.

5.1 Building the user profile

It has been shown 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. 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, and 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 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. However, not all users use bookmarks, and those who do use bookmarks, do not always bookmark everything they like or they find interesting (Abrams et al., 1998). In designing the UI agent, the goal was to make it completely adaptable to user work patterns. As such, the UI agent does not rely on any single activity that a user carries out, nor make any assumptions about the user, but rather utilises all activities that can be used for profiling.

The user profile is meant to reflect user interests, so it is modelled as a set of interests represented by feature vectors of terms. This set in fact can be viewed as the union of two types of sets: an active set, the length of which is constrained by the system designer, and an inactive set, which can be of any length depending on actual user interests. Swapping between active and inactive sets is done using the least recently used algorithm. So in a way, the user interface agent has two types of memory, an active memory and a dormant memory. Each interest represented in either set reflects information about the user's context. Thus the words, *interest* and *context*, will be used interchangeably.

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 inference the interest. Identified activities include: browsing, bookmarking, and rating Web pages, as well as getting feedback on search.

The rules for bookmarking, and rating are straightforward. Any documents 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 just the opening of a Web page does not indicate 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. The UI agent provides the user with a facility to easily rate a Web page as being bad in a way that would remind the user of links he/she disliked and prevent him/her from unknowingly following that link from another Web page. As such recurring visits to a Web page will usually indicate 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 a Web document, the HTML content of the document is parsed into URL objects. 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 3 is applied to add the document to the user profile, which is represented by multiple cluster centroids where each centroid represents a user interest.

5.2 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 than that of a traditional linkbase as implemented in a system such as the DLS. Links in the linkbase are imported from other *CLinks* type of agents based on the interests of a user. Assuming, that the 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 that fit that 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 that Web page has to be downloaded before it can commence. If the display of the Web page is to be blocked by the proxy until the page completely downloads, then the user will only start seeing text in the body of the page depending on the speed of the connection. 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 push or pull will be used to deliver the altered Web page. When implementing the system, the two technologies were employed so as to make the system browser independent.

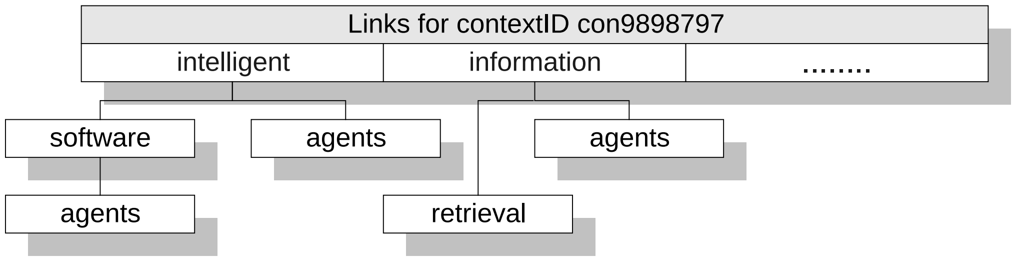
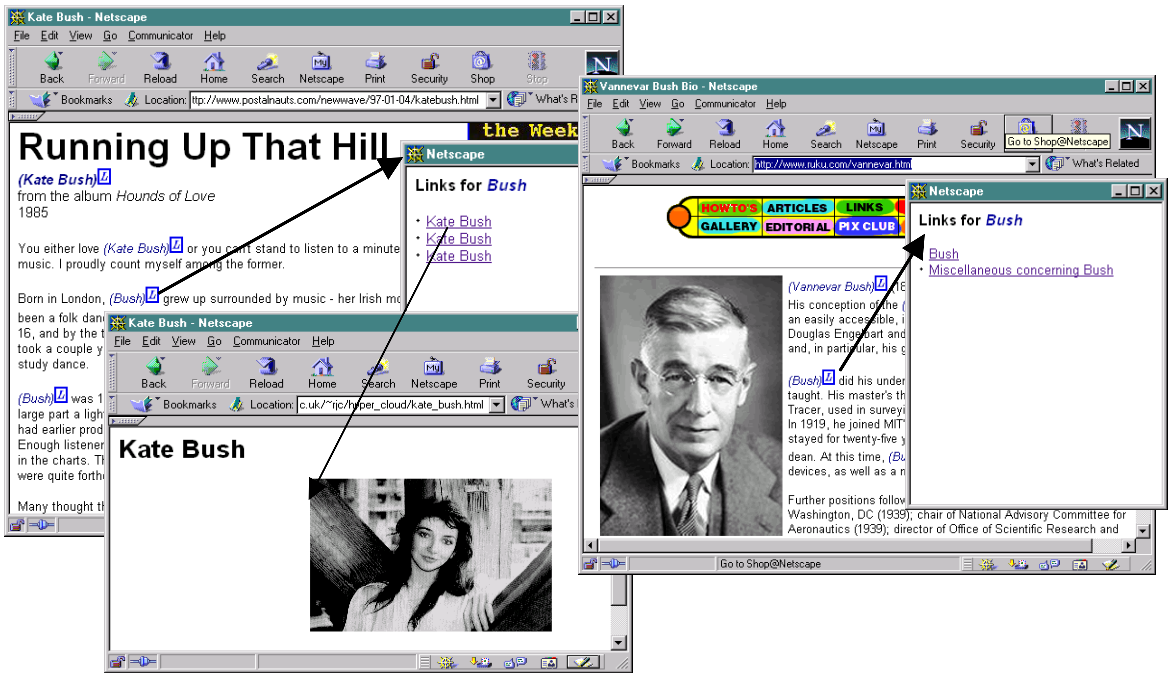


Figure 4: Quick lookup table for link phrases

A number of data structures are employed to facilitate rapid determination of link context. Cluster centroids used to represent user interests are also used to represent link contexts. A table is used to map context identifiers to their weight vector of terms representations. Another table exists which maps every Web page for which a context has been determined, to a context identifier. To speedup the linking, a quick lookup table for every context identifier, is available. Phrases within the lookup table are represented by trees so as to facilitate rapid parsing. Figure 4 shows a simplified diagram of part of that lookup table. Another table is used, to represent the actual links. In the second table, phrases are mapped to 1 or more URLs. After downloading a Web page, if it matches any of the user interests for which links are available, the context identifier mapping of that interest is used to activate the appropriate linkbase. Specifically the context identifier is used to retrieve the lookup table for phrases for which there are links in that context.



Kate Bush - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: ttp://www.postalnauts.com/newwave/97-01-04/katebush.html
What's R
the Week
Netscape
Kate Bush - Netscape
Location: c.uk/~rjc/hyper_cloud/kate_bush.html
Vannevar Bush Bio - Netscape
Location: http://www.ruku.com/vannevar.html
Go to Shop@Netscape
What's Related
HOWTO'S ARTICLES LINKS
GALLERY EDITORIAL PIX CLUB

Running Up That Hill

(Kate Bush)
from the album *Hounds of Love*
1985

You either love (Kate Bush) or you can't stand to listen to a minute
music. I proudly count myself among the former.

Born in London, (Bush) grew up surrounded by music - her Irish mo
been a folk danc
16, and by the t
took a couple y
study dance.

(Bush) was 1
large part a ligh
had earlier prod
Enough listener
in the charts. Th
were quite forth

Many thought th

Links for *Bush*

- Kate Bush
- Kate Bush
- Kate Bush

Kate Bush

(Vannevar Bush) (18
His conception of the
an easily accessible, i
Douglas Engelbart and
and, in particular, his g

(Bush) did his under
taught. His master's th
Tracer, used in surveyi
In 1919, he joined MIT'
stayed for twenty-five y
dean. At this time, (Bu
devices, as well as a n

Further positions follow
Washington, DC (1939); chair of National Advisory Committee for
Aeronautics (1939); director of Office of Scientific Research and

Links for *Bush*

Bush
- Miscellaneous concerning Bush

Figure 5: An example showing how links are rendered 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.

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 into a stack. This procedure continues until either the leaves of the tree are reached, or the look ahead 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 the end of a phrase is reached. 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 which points back to a simple HTTP server implemented by the UI agent. If a user should follow this link, then a query encoded in the link would be used to resolve appropriate links related to that phrase, based on the phrase lookup table. Figure 5, shows how links are rendered and resolved in two different contexts.

6 Conclusions and Future work

We have developed a multiagent web recommender system that has personal as well as collaborative dimensions. The system utilises the activities of various users in a group setting to create and propagate information that can be used by users of the system in their information finding and navigation activities. The novelty of the system comes from the way it has extended a traditional open hypermedia generic link model with the aid of agents, to deliver a personalised Web recommendation system through links in context.

There are a number of areas where ongoing and future work are being aimed. On the ongoing front, more functionality is being added to assist users in retrieving information from within a Web page in the context of that page. The new facilities will allow users to interact with the system and proactively ask for links by selecting text fragments for which the links are desired, or by simply entering a query. We are also currently experimenting with an alternative technique for mining links from Web pages utilising the text body of the document from which links are being extracted. Compared to the presented technique, the new one is likely to yield better results. The following points summarise future directions:

- Getting user feedback on recommended documents and using that in managing the life cycle of links in the *CLinks* knowledge base, and the personalised linkbases, is a feature that needs to be added.
- The representation and presentation of recommended links, is currently rather primitive and thus needs to be improved upon. Presenting recommended links with related meta-data would provide added benefits to the recommendations. The use of XLink and RDF for that task is currently being explored.
- More research is planned for improving concepts related to the life cycle of links and clusters through consideration of the characteristics of human memory

- As discussed in the paper, evaluation was conducted on the level of the algorithm presented. User evaluation experiments are currently underway.

References

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.

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

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

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.

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

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

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. (2000). MEMOIR -- An Open Framework for Enhanced Navigation of Distributed Information. *Information Processing & Management. An International Journal*.

El-Beltagy, S., DeRoure, D. and Hall, W. (1999) *A Multiagent system for Navigation Assistance and Information Finding*, In Proceedings of *The Fourth International Conference on the Practical Applications 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. (2000) *The Evolution of a Practical Agent-based Recommender System*, In Proceedings of *Workshop on Agent-based Recommender Systems, Autonomous Agents 2000*, ACM, Barcelona, Spain.

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.

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.

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

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

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

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

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 Modelling,*(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.

Mladenic, D. (1999). Text-Learning and Related Intelligent Agents: A Survey. *IEEE Intelligent Systems,* **14**(4), pp. 44-54.

Porter, M. (1980). An algorithm for suffix stripping. *Program,* **14**(3), pp. 130 -137.

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.

Salton, G. (1988) *Automatic Text Processing,* Addison-Wesley, Reading.