Linking in Context

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

Intelligence, Agents, Multimedia
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
E-mail: samhaa@acm.org, {wh,dder,lac}@ecs.soton.ac.uk

ABSTRACT
This paper explores the idea of dynamically adding multi-destination links to Web pages, based on the context of the pages and users, as a way of assisting Web users in their information finding and navigation activities. The work does not make any preconceived assumptions about the information needs of its users. Instead it presents a method for generating links by adapting to the information needs of a community of users and for utilizing these in assisting users within this community based on their individual needs. The implementation of this work is carried out within a multi-agent framework where concepts from open hypermedia are extended and exploited. In this paper, the entities involved in the process of generating and using 'context links' as well as the techniques they employ to achieve their tasks, are described. The result of an experiment carried out to investigate the implications of linking in context on information finding, is also provided.

KEYWORDS: Links, link generation, dynamic linking, open hypermedia, information finding, navigation assistance, software agents, context.

INTRODUCTION
During the past few years, it has become apparent that there is an urgent need for tools that can guide Web users in their information finding and navigation activities. This paper explores the idea of dynamically creating links and employing them in context via an open link service in a way that would serve that goal.

One of the failings of traditional information retrieval models is attributed to the isolation of queries from the context in which they occur [4]. Context is an involved issue that has been addressed in many fields as discussed more fully in [10]. One of the elements that contribute to a user's local context is the Web page that the user is browsing which might contain concepts the user wants to know more about [27].

If a user's information needs are to be anticipated in advance and links are to be dynamically added to Web pages that meet a user's interest, then this could lead the user to find information that meets his/her information needs while they are browsing without having to consult an external search facility. This is the motivation behind this work. As such, within this work context is defined by user interests and the document within which the links are to be rendered. The 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.

Since the work makes no pre-defined assumptions about the information needs of its users, a way of automatically generating useful links that can serve a community of users is presented. The implementation of this idea is presented as part of the QuIC (Queries in Context) system where concepts from open hypermedia are extended and exploited within a multi-agent architecture [12]. The framework is meant to serve users in organizations 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 presented work 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 in order to assist other users in the system, in this case by triggering link generation. 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. Though the QuIC framework addresses the dissemination of different kinds of information that can help users in their information finding activities, the focus of this paper is on information created and propagated in the form of contextualized links.

BACKGROUND
Open hypermedia is an area that has been researched by the hypermedia community for several years and for which a number of systems have been implemented [7,18,26]. 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 [7]. The recently proposed XLink standard [20] is increasingly moving towards the open hypermedia approach. Systems such as the *Distributed Link Service* (DLS) [6,5], Webvise [17], and Chimera [2] were designed in order to bring the open hypermedia philosophy to the Web.

The DLS is based the Microcosm OHS system [14,7] which was among the first systems to explore the idea of building a hypertext system out of a set of loosely-coupled communicating processes. The processes could be pre-configured so that users of Microcosm applications could use the linkbases most appropriate to the context of their activity. 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. 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 as a result of it being a natural extension to Microcosm. 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.

SYSTEM OVERVIEW

One of the goals of the presented work is to address the limitation of switching between linkbases according to the context of documents. Another is to enable dynamic creation of links to populate linkbases. Manual authoring of links is an expensive and inefficient process. 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. So, in the context of this work, there are three steps involved in the process of linking in context:

1. The creation of links in context (via link extraction)
2. The propagation of links to users based on user interests (which within this work define user context)
3. The rendering of links based on document context

Within the developed multi-agent architecture, two types of agents are responsible for carrying out these tasks: link extraction and contextualizer agents, and user interface agents. Details of the architecture and agents used to implement the work described in this paper can be found in [13,12]. A user interface agent (UI agent) is associated with every user in the system and it is the component through which the user can interact. The agent is implemented in Java and is capable of monitoring a user's browsing, bookmarking, and searching activities, and of building a user profile reflecting user interests, accordingly. With the permission of the user, URLs that he/she has found interesting are communicated to a number of agents such as the link extraction and contextualizer agent which uses this information to download, and analyze Web pages and to create links accordingly. One of the components of the UI agent is a personalized context aware linkbase, which is populated

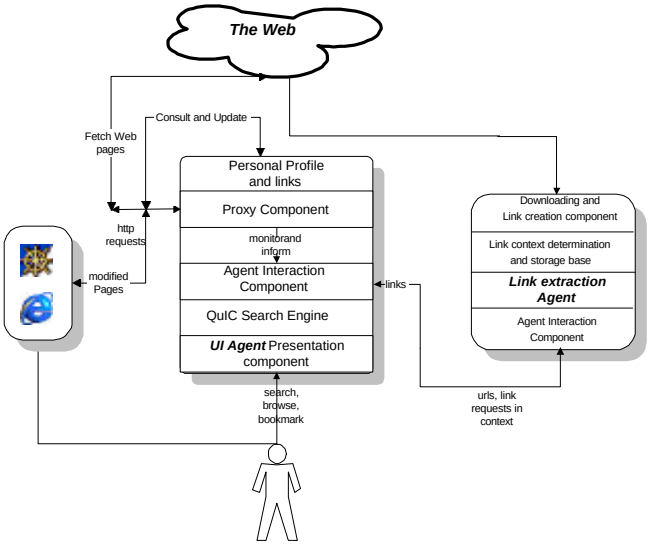

Figure 1: The interaction scenario for linking in context

by links through various subscriptions to the link extraction agent based on user interests.

The overall scenario for achieving linking in context is illustrated by Figure 1. Various user interface agents in the system communicate URLs their users have found interesting to the link extraction agent (e.g. when a user bookmarks a document), which uses those to generate links and store them in its knowledge base. Whenever a UI agent detects that its user has developed a new interest, it asks the link extraction agent to provide it with links related to that interest and to send it any new links related to that interest as soon as they become available. The UI agent stores the retrieved links in its context aware linkbase. Whenever the user requests a Web page, the proxy component of the UI agent intercepts the HTTP

request, downloads the page, attempts to locate links for it in its context aware linkbase and if found, to add them dynamically to that Web page. The steps involved, are described in more details in the following sections.

CREATING THE LINKS

The task of creating the links is assigned to a link extraction and contextualizer agent which employs 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 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, described in the next section) or simply a search request represented by a set of keywords. The tasks of this agent can be summarized by the following points:

1. Continuous creation of links in context
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

Generating automatic links is a challenging task since certain relationships between documents can sometimes only be inferred and tagged by humans [27,1]. This is particularly true if there is a requirement to create links automatically to concepts represented by phrases. This task implies that there is a requirement to understand something about the 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 to concepts within a large body of text, when there is little understanding of the text itself.

To overcome this difficulty we make use of already authored links through the use of a simple rule based algorithm developed to utilize those in the creation of generic links in context. 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***. The 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 would usually denote concepts. This gives rise to another 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 generalize this link for is *Vannevar Bush*. i.e., the source anchor for the link to be created using that link, would be *Vannevar Bush*. The algorithm takes this into account. The first step towards applying this algorithm is to determine the context of documents from which links are to be extracted.

Determining the Context

TF-IDF (term frequency, inverse document frequency) is an information retrieval technique based on a vector space model where a vector is used to represent a document or a query[25]. The technique is used to calculate the importance of terms appearing in documents. Using TF-IDF in conjunction with the cosine similarity function seemed well suited for our context determination task. The cosine similarity function $Sim(d_i, d_j)$ measures the cosine angle between two documents represented by 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). The aim is 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 [24,25]. 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 (centroids) represented by a feature vector of terms (initially {}), and $\boldsymbol{c_i}$ 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 $\boldsymbol{c_i}$ calculate $Sim(\boldsymbol{c_i}, \textit{inDoc})$ using the cosine similarity function
2. If maximum similarity across $\boldsymbol{C}$ is greater than a threshold value m 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 $\boldsymbol{UD}$. 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 $\boldsymbol{UD}$

When experimenting with the technique, we found that 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] were 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 recognized as such. In conjunction with the cosine similarity function, a very simple heuristic was applied to solve the problem and enable 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

$$\text{Similar}(d_i, d_j) \Leftrightarrow \text{Sim}(v_i, v_j) > m \quad \textbf{\textit{and}} \quad |t_i \cap t_j| > b$$

Where $\text{Sim}(v_i, v_j)$ is the cosine similarity function, m is the threshold value below which documents can not be similar and b 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.

A further modification was applied to the way cluster centroids are constructed. To better capture the context of a group of documents, it is important to emphasize 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, $c_i w_k = \max(d_i w_k, d_j w_k) * a$ where a is the boosting factor. In our current implementation, a 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 with 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 emphasized 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 over this data set was 98.7 %. This was measured based on manual examination of clusters.

[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/

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

Extracting the links

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 (the UI agent monitors the user's browsing, bookmarking, and searching activities as a way of determining those). In that sense, the agent is dependent on UI agents in the system to correctly convey information about the interests of their users. The process by which they determine such interests is totally transparent to the link extraction 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 analyzed and associations between text phrases and those links are created and stored in a Prolog knowledge base (KB). 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 the previous section
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 it 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*, *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 1 word appears, and it is in the top *n* features, or if more than 1 appear, 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.

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.

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.

Rendering the links

One of the components of a UI agent in our framework 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 divided 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 which is used to group retrieved links. Details on how the user profile is built and used for this task can be found in [11]. 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 a 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 available cluster summaries. As soon as new links become available, they too get dispatched to the subscribed agent. 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 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, 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 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 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 a new content, which replace 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 [21]. 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. 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. Following an automatically rendered link activates a set

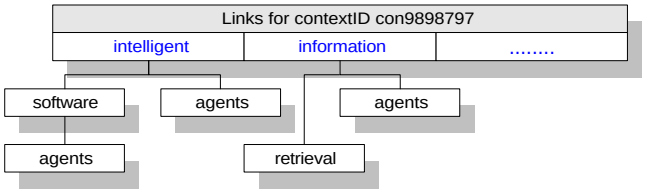

Figure 2: Quick lookup tree for link phrases

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. The following are examples of generated destination URLs:

A link generated for *Bush* in the G. W. Bush context

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

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

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

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 3, shows how links are rendered and resolved in two different contexts. The links shown were created as part of the experiment described in the evaluation section.

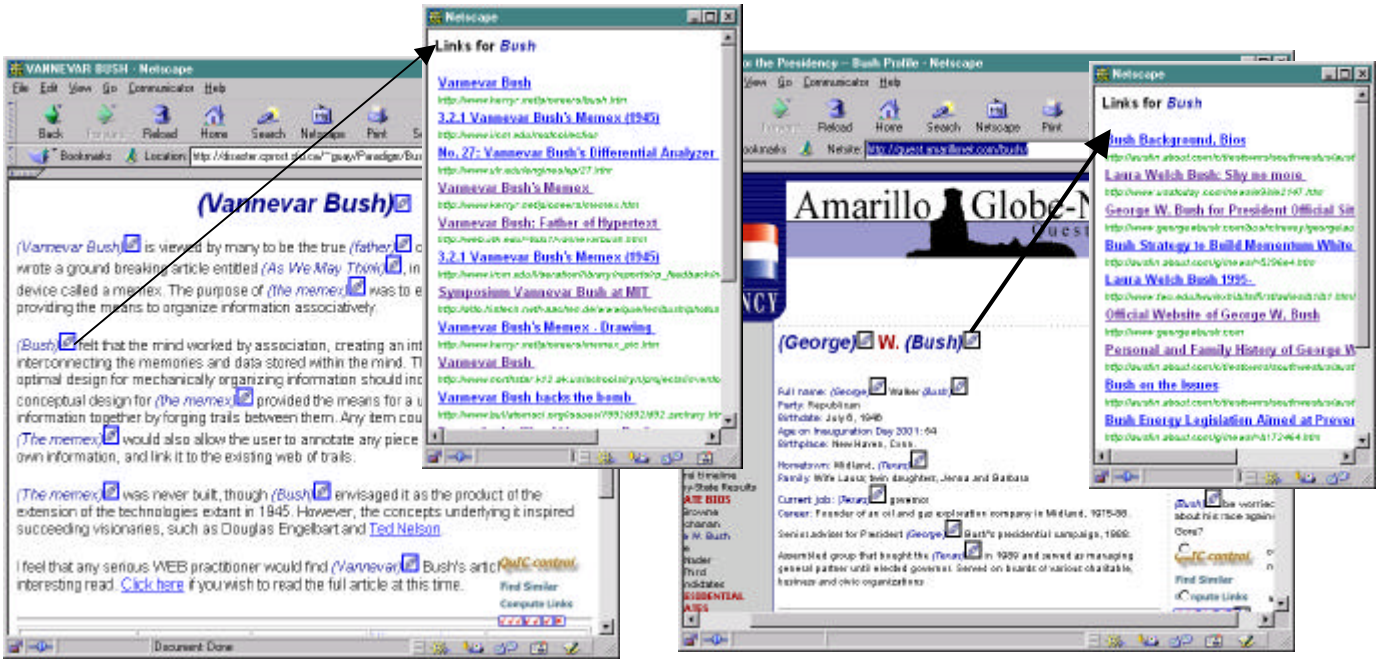

Figure 3: An example showing how links are rendered in two different contexts. In the two documents, different links are suggested for "*Bush*" based on the context. An icon appears besides linked phrases to distinguish recommended links from original links in the page.

RELATED WORK

Dynamic link generation has been addressed by a number of systems, though each of these systems addressed them from a different perspective. The VOIR system [15,16] is

one example. The main motivation behind VOIR 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. Another example of work that has addressed automatic link generation is Bernstein's link apprentice, which uses simple pattern matching and string comparison algorithms to generate links [3]. However, the link apprentice is liable to generate erroneous links if two unrelated documents representing different concepts are expressed in similar words, something which the presented system (QuIC) has addressed. As a result, the link apprentice system has been recognized 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 [19,23]. 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 [1]. There is an overlap between the goals of link adaptation as addressed by this work, and adaptive hypermedia[8]. However, to the knowledge of the authors, the approach presented here for link generation and presentation has not been addressed by any other systems, adaptive, or otherwise. For example, while QuIC is capable of generating reliable links represented by phrases, most other systems generate links represented by single terms. QuIC can add links to Web pages it has never seen before. It also 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.

Related agent based and recommender systems are discussed in [11]. Particularly relevant are the MEMOIR [9], and Margin Notes[22] systems. MEMOIR employed DLS technology and as a result suffered from the same limitations as the DLS. Though Margin Notes addressed local context for suggesting documents, it did not address linking at all.

EVALUATION

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 seven 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. Tables 2 and 3, provide the questions making up each of the tasks. 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. The users were asked to indicate how familiar they were with the subjects in question. Table 1, in which U*i* refers to user *i*, summarizes their responses.

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

Table 1: A table showing the familiarity of different users with areas surrounding the two evaluation tasks

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 2: The questions users were asked about V. Bush and G. W Bush

In the first phase, participants in the experiment were asked to carry out their task by finding information using one of the standard available search engines. Users 1 through 3 were assigned the Bush task, while users 4 through 7 were assigned 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 was placed on these

[3] Common Object Request Broker Architecture

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

particular questions. For the Bush task, only user 3 was able to find an answer for question 1 and for the CORBA task, only user 4 was able to find an answer for question 5. 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.

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 3: The questions users were asked about CORBA

URLs from which users found the answers were sent to the Link extraction agent. 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. 4 different document clusters were created. Cluster one contained 13 of the Vannevar Bush related documents. Cluster two contained all 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. The remaining two CORBA related URLs remained unclustered because their contents did not match with any of the other documents strongly enough.

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 QuIC linking in context utility (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 the user is interested in the three areas surrounding the tasks. The same time limits 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, and U5 used Google, U3 used Yahoo, U4 used Altavista while users U6, and U7 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 (with the exception of U5).

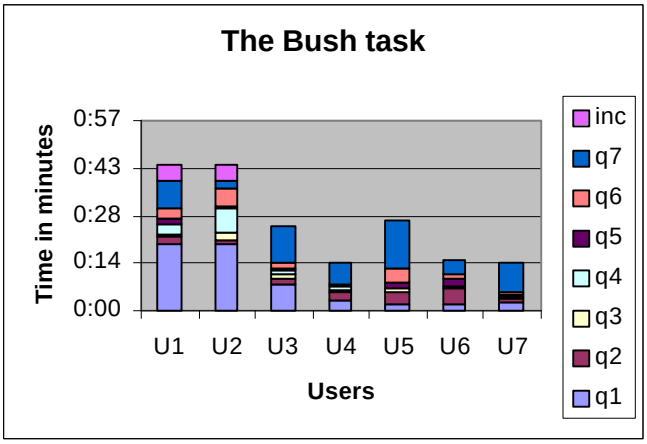

Figure 4: Chart showing the time taken by each of the participants to complete the Bush task. Users 1 through 3, used a standard search engine only to carry out their task, while users 4 to 7 used the linking utility in conjunction with a search engine

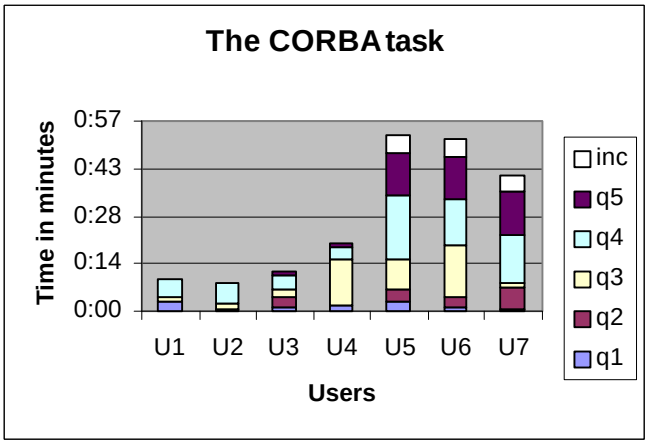

Figure 5: Chart showing the time taken by each of the participants to complete the CORBA task. Users 4 through 7, used a standard search engine only to carry out their task, while users 1 to 3 used the linking utility in conjunction with a search engine

The average time of the completion of the CORBA task in the first phase was approximately 36 minutes while the average time to complete the Bush task was 34 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 10 minutes and 13 seconds while the average time taken to complete the Bush task was 19 minutes and four seconds. So on average, users using the linking facility completed their task in 28% of the time taken by users employing a search engine only for the CORBA task and in 55% 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. Figures 4, and 5 show the time it took each user to complete a given question in each of the tasks, where *qi* denotes question *i* and ***inc***, is an indicator that the task was not completed. When using the context links, the time taken by all users carrying out the CORBA task was significantly lower than the time taken by their counterparts using only a search engine. This was the same for the Bush task except in the case of one user (U5), but even then, the user's task completion time was comparable to the quickest person using a search engine only.

While watching users carry out their search activities, it was obvious that different users have different search strategies and that no one strategy is always effective across all tasks. Flexibility in terms of rapidly being able to switch from one search strategy to another, is probably the best way for 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 one or not. The advantage of offering context links is that the user does not even have to think of what query to enter. In our experiment, some of the more challenging and time consuming questions, were easily answered by users of the context links as they readily recognized an association between the questions that were put to them and some of the links being offered.

CONCLUSION AND FUTURE WORK

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. This work has addressed this through the use of software agents that employ small collections of documents gathered by monitoring the activities of a group of users and detecting which documents they have found useful. Both the document collection, and the links contained within them are used to create the links. Because these documents are obtained as a result of users expressing a strong interest in them, through bookmarking for example, the quality of their content is likely to be high. The fact that the generated links will be only be propagated to users who have displayed an interest in similar content, means that if some noisy content has been added, it will not be used and will eventually be removed from the link knowledge base in which usability is one of the factors that control the lifetime of the links.

Based on the 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 and which are less likely to contain dangling links and are quite likely to be of high quality. Small link icons are added to linked phrases to indicate subtly that there are links available for those phrases. Six out of the seven users of the system indicated that they found the way in which links were rendered more useful than distracting. 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.

There are a number of areas where ongoing and future work is being aimed. 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. More extensive user evaluations are also underway. The following points summarize future directions:

- Allowing users to control the contents of their own linkbases, so that they can immediately delete or add links in a specified context
- Getting user feedback on recommended documents and using that in managing the life cycle of links in the link extraction agent's knowledge base, and the personal linkbases. In general, more research is planned for improving concepts related to the life cycle of links and clusters through consideration of the characteristics of human memory
- Investigating the use of natural language techniques in link phrase extraction
- Experimenting with ways of achieving a more localized level of contextualization.
- Improving the representation and presentation of suggested links

ACKNOWLEDGMENTS

The work presented in this paper has been supported by the QuIC project, ESPRC grant GR/M77086.

REFERENCES

1. Allan, J. Automatic hypertext link typing, in Proc. of the seventh ACM conference on Hypertext, ACM, Bethesda, MD USA, pp. 42-52, 1996.

2. Anderson, K. M. Integrating open hypermedia systems with the World Wide Web, in Proc. of the eighth ACM conference on Hypertext ACM, Southampton, United Kingdom, pp. 157-166, 1997.

3. Bernstein, M. An Apprentice That Discovers Hypertext Links, in Proc. of ECHT'90 , INRIA, France, pp. 121-223, 1990.

4. Budzik, J. User Interactions with Everyday

Applications as Context for Just-in-time information Access, in Proc. of Intelligent User Interfaces (IUI) ACM, New Orleans, LA USA, pp. 44-51, 2000.

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

6. Carr, L. A., DeRoure, D. C., Hall, W. and Hill, G. J. The Distributed Link Service: A Tool for Publishers, Authors and Readers, in Proc. of the fourth International World Wide Web Conference, Boston, Massachusetts, USA, pp. 647-656, 1995.

7. Davis, H. C., Hall, W., Heath, I., Hill, G. J. and Wilkins, R. J. Towards an Integrated Information Environment with Open Hypermedia Systems, in Proc. of the Fourth ACM Conference on Hypertext, Milan, Italy, pp. 181-190, 1992.

8. De Bra, P. AHAM: A Dexter-based Reference Model for Adaptive Hypermedia, in Proc. of Hypertext'99 Conference, ACM, Darmstadt, Germany, pp. 147-156, 1999.

9. DeRoure, D. C., Hall, W., Reich, S., Pikrakis, A., Hill, G. J. and Stairmand, M. MEMOIR -- An Open Framework for Enhanced Navigation of Distributed Information. Information Processing & Management. An International Journal, 2000.

10. El-Beltagy, S. Context, Queries, and the Web, In Technical Report , University of Southampton, Southampton, UK, ECSTR-IAM01-002, 2000.

11. El-Beltagy, S. On the Usability of Software Agents for Creating and Employing Links in Context, In Technical Report , University of Southampton, Southampton, UK, ECSTR-IAM00-6, 2000.

12. El-Beltagy, S., DeRoure, D. and Hall, W. A Multiagent system for Navigation Assistance and Information Finding, in Proc. of The Fourth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, pp. 281-295, 1999.

13. El-Beltagy, S., DeRoure, D. and Hall, W. The Evolution of a Practical Agent-based Recommender System, in Proc. of Workshop on Agent-based Recommender Systems, Autonomous Agents 2000 ACM, Barcelona, Spain, 2000.

14. Fountain, M. A., Hall, W., Heath, I. and Davis, H. C. MICROCOSM: An Open Model for Hypermedia with Dynamic Linking, in Proc. of ECHT'90,, pp. 298-311, 1990.

15. Golovchinsky, G. Queries? Links? Is there a difference?, in Proc. of CHI'97 ACM, Atlanta, GA USA, pp. 407-414, 1997.

16. Golovchinsky, G. What the query told the link: the integration of hypertext and information retrieval, in Proc. of the eighth ACM conference on Hypertext ACM, Southampton, United Kingdom, pp. 67-74, 1997.

17. Grønbæk, K., Sloth, L. and Orbeak, P. Webwise: browser and proxy support for open Hypermedia structuring mechanisms on the World Wide Web, in Proc. of 8th International World Wide Web Conference Elsevier Science, Toronto, Canada, pp. 253-267, 1999.

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

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

20. Maler, E. and DeRose, S. J. XML Linking Language (XLink), In Technical Report , World Wide Web Consortium, , http://www.w3.org/TR/1998/WD-xlink-19980303, 1999.

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

22. Rhodes, B. J. Margin Notes: Building a Contextually Aware Associative Memory, in Proc. of Intelligent User Interfaces (IUI '00), ACM, New Orleans, LA USA, pp. 219-224, 2000.

23. Robertson, J., Merkus, E. and Ginige, A. The Hypermedia Authoring Research Toolkit (HART), in Proc. of ECHT'94, ACM, Edinburgh, Scotland, UK, pp. 177-185, 1994.

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

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

26. Wiil, U. K. Open Hypermedia: Systems, Interoperability and Standards. Journal of Digital information, Special Issue on Open Hypermedia, **1**(2), 1997.

27. Wilkinson, R. and Smeaton, A. F. Automatic Link Generation. ACM Computing Surveys, **31**(4), 1999.