Robust Hyperlinks Cost Just Five Words Each

Thomas A. Phelps and Robert Wilensky
Division of Computer Science
University of California, Berkeley
Berkeley, CA 94720-1776
phelps@cs.berkeley.edu, wilensky@cs.berkeley.edu

Keywords: signature, robust, hyperlink, reference, location

Lexical signature: thinkinginpostscript cityquilt fernec planetext peroperties

Abstract

We propose *robust hyperlinks* as a solution to the problem of broken hyperlinks. A robust hyperlink is a URL augmented with a small "signature", computed from the referenced document. The signature can be submitted as a query to web search engines to locate the document. It turns out that very small signatures are sufficient to readily locate individual documents out of the many millions on the web.

Robust hyperlinks exhibit a number of desirable qualities: They can be computed and exploited automatically, are small and cheap to compute (so that it is practical to make all hyperlinks robust), do not require new server or infrastructure support, can be rolled out reasonably well in the existing URL syntax, can be used to automatically retrofit existing links to make them robust, and are easy to understand. In particular, one can start using robust hyperlinks now, as servers and web pages are mostly compatible as is, while clients can increase their support in the future.

Robust hyperlinks are one example of using the web to bootstrap new features onto itself.

Introduction

Hypertext research has long been concerned with the problem of the persistence of hyperlinks, that is, of dealing with problems that arise when one endpoint of a link, especially the destination, is unresolvable, either because it was deleted, renamed, moved, or otherwise changed. Dangling pointers on the Web are considered by some to be a significant problem, and a number of solutions have been proposed to deal with them. Some of these suggest reliance on some additional naming scheme, such as Uniform Resource Names (URNs) **[Sollins and Masinter, 1994]**, handles **[Kahn and Wilensky, 1995]**, Persistent Uniform Resource Locator (PURLs) **[OCLC]** or Common Names **[CNRP]**. Other approaches involve monitoring and notification to insure referential integrity (e.g., **[Ingham et al., 1996]**), **[Mind-it]**, **[Macskassy and Shklar, 1997]**, **[Francis et al., 1995]**).

In this paper, we demonstrate a different approach to this problem. This is to augment URLs so that they themselves become *robust hyperlinks*. A robust hyperlink is one which offers a reasonable chance of being successfully dereferenced in the presence of uncoordinated change. That is, suppose that, when creating a hyperlink to a networked resource, one could design it so that it was still possible, with high probability, to resolve the reference of the hyperlink, even if the resource referred to by the hyperlink had been moved and edited, with the probability of successful dereferencing declining with the degree of substantive change to the document content. Subsequence users of such a hyperlink would find it robust, in that it would function reasonably well after the state of the network it reflected had changed.

Note that robust hyperlinks puts the burden of additional effort on the party creating the hyperlink, rather than the party administrating the resource. One implication of this fact is that no buy-in is required by an administrative unit, as for example a web server or site adminstrator. Similiarly, hyperlinks could be made robust on a piecemeal basis, a link at a time, rather than require use on a more systematic basis.

We believe these practical advantages of robust hyperlinks should facilitate their adoption, should the underlying technology be available. That technology has the following requirements:

- Robust hyperlinks should provide a very high likelihood of successful dereferencing in those cases in which an item is moved, but has otherwise been left largely unchanged. Moreover, performance should degrade gracefully as document content changes from its state at the time the hyperlink was created.
- When the robust character of the hyperlink is not needed, robustness should not impose a significant performance penalty.
- The additional storage required for a robust hyperlink must be relatively small, so that it is practical to make all URLs robust.
- Robust hyperlinks will require support, however minimal, from client or from proxy with which the user can interact. Thus, implementation in clients or via proxies should be straightforward so as to encourage widespread adoption.
- To encourage immediate adoption, robust hyperlinks should be largely non-interfering with clients and services that do not support them.
- The additional work required to make a hyperlink robust should not be computationally large, and it must be possible to automate it completely. That is, an author should be able to point to a hyperlink, and have it automatically become a robust hyperlink.

Providing Robust Hyperlinks

As it turns out, robust hyperlinks can be readily created in a manner that fulfills all of the above characteristics. The basic idea is extremely simple. It is to include some part of the document content along with the URL. Then, if the URL is no longer valid, one can feed the content to a web search engine, and peruse the results.

Of course, for many cases, it is not challenging to carry out this process by hand. For example, if the page is known to be a particular individual’s home page, then a user can manually call up a search engine, enter the person’s name and perhaps affiliation, and have a good chance of finding it. However, in the general case, how to determine good search terms may not be obvious: Users may have never encountered a link before, and hence may not know much about its content. Even if users are familiar with a resource, hypothesizing search terms from memory generally involves much trial and error.

The scheme described in this paper determines a small number of good words to search for, that is, words that will find the desired page, but as few of the other millions of pages in the web as possible. This "lexical signature" is then attached to the URL. A robust-hyperlink-aware agent will generally perform an initial attempt at "traditional (i.e., address-based) dereferencing", that is, looking up a URL, ignoring the signature. However, if traditional dereferencing fails, the client enters into a second phase of "signature-based (i..e, content-based) dereferencing", in which it uses the signature to search for documents whose signature most closely matches that in the robust hyperlink. The user is then presented with the matching documents from which to complete the reference.

Computing Lexical Signatures

One way to create lexical signatures that meets the desired criteria is to select the first few terms of the document that have the highest "term frequency-inverse document frequency" (TF-IDF) values. Certainly, such lexical signatures are easy to compute: The frequency of a term in a document is of course easy to determine, and document frequency of terms can be estimated by the values given for these terms by search engines. Intuitively, TF-IDF seems like a reasonable characterization of a document’s contents. The question is whether a relatively small set of such terms can effectively discriminate a given document from all the others in a large collection.

Empirical Results

Perhaps surprisingly, for a distributed hypertext system the size of the Web, a very small number of terms is sufficient. Specifically, a signature of five terms is sufficient to determine a web resource virtually uniquely. Indeed, fewer terms will probably suffice; we advocate at least these many terms because the redundancy that is provided is useful with respect to document change and because the additional terms may needed to distinguish documents as the web continues to grow.

Let us examine this claim a bit more closely. Our criteria state that we need searching by signature to return a reasonably small result set, meaning one that can be readily perused by a user so as to select a document. Empirically, it seems that using five word signatures actually overshoots this goal: In most cases, a query to a search engine requesting documents which contain all of the terms in the signature will cause a unique document to be returned, namely, the desired document. In those few cases in which more than one document is returned, the desired document is among the highest ranked. In those case in which a particular search engine returns no matching documents, this is generally because the document has not yet been indexed, or has been substantially edited since it was last indexed.

As an example, we computed signatures for a varied, unpremeditated sample of different sorts of web pages. (For a different set of examples, see the hyperlinks in this paper.) Most of these are for papers referenced by a bibliography maintained by one of the authors, as we feel this is a realistic application of the technology. To these we added a personal home page, a research web site, and a commercial home page. For each case, we computed signatures, and then perform straightforward queries using several search engines. (That is, we just supply the engines with the signature terms, without using any advanced features of the engines.) In Table 1, we report the rank of the document in the result set (or "?" if it is absent in the first page of results, which is usually the first ten results).

Table 1:Sample Signatures and Query Results.

URLs are presented along with their signatures and the rank of that exact URL in the result set of each search engine to which the signature is submitted. In addition, the server software of the URL’s host is listed, along with whether that server accepts the "robust URL" syntax described below. (In the on-line version of this paper, the contents of the "Accepts Robust URLs?" cells are hyperlinks that use robust URLs; the search engine rank cells are hyperlinks that query the respective search engine with the signature.)

URL	signature	Server	Accepts Robust URLs?	Google	Alta Vista	Yahoo	Hotbot	Infoseek
http://www.cs.berkeley.edu/~daf/								
	bregler interreflections zisserman cvpr iccv	Apache 1.3.4	**yes**	**1**	**6**	**1**	**1**	**1**
http://www-diglib.stanford.edu/diglib/pub/								
	sdlip interbib sdlt infobus testbed	Apache 1.3.4	**yes**	**1**	**1**	**1**	**1**	**4**
http://www.hotofftheweb.com/								
	servicemarks moskowitz mustache scrapbook surfers	ApacheSSL 2.4.1/1.3.3	**yes**	**1**	**2**	**1**	**1**	**1**
http://developer.apple.com/techpubs/macos8/Legacy/OpenDoc/opendoc.html								
	opendoc webobjects software constants reference	Netscape-Enterprise 3.5.1G	**yes**	**1**	**?**	**1**	**?**	**?**
http://www.rightbrain.com/pages/book-download.shtml								
	thinkinginpostscript ematter click infringe	BESTWWWD 2.4	**yes**	**?**	**2**	**?**	**?**	**?**
http://msdn.microsoft.com/workshop/author/css/css.asp								
	mystyles dblspaced selamoglu intdev italicizes	Microsoft-IIS 5.0	**yes**	**1**	**1**	**1**	**1**	**1**
http://www.adobe.com/products/acrobat/main.html								
	accessiblity hewson epaper gillmor workflow	Netscape-Enterprise 3.6 SP2	**yes**	**?**	**1**	**1**	**1**	**?**
http://www.isg.sfu.ca/~duchier/misc/hypertext_review/								
	balasubramanian bala hypermedia pegasus interface	Apache 1.3.6 (Unix) PHP/3.0.7 mod_ssl/2.3.11 OpenSSL/0.9.3a	**yes**	**1**	**10**	**?** (1-4 are dups.)	**?** (1-4 are dups.)	**?**
http://www.ai.univie.ac.at/~paolo/lva/vu-htmm1998/html/conklin87/Conklin87.html								
	planetext fernec synview peroperties textnet	Apache 1.3.3 (Unix) Debian/GNU	**yes**	**?**	**?**	**?**	**1**	**?**
http://www.lcc.gatech.edu/gallery/hypercafe/HT96_HTML/HyperCafe_HT96.html								
	cityquilt hypervideo cyberbelt infrawhere videotexts	Apache 1.3.6 (Unix)	**yes**	**1**	**1**	**1**	**1**	**1**

These results suggest that a number of signature-based dereferencing strategies are feasible. For example, an agent could query a set of engines, and return the top few results from each one. In our sample set, each hyperlink is successfully dereferenced by this strategy.

Alternatively, an agent could make "stringent" queries to one or more engines (i.e., queries insisting that all the terms be present), and, if this fails to return a result, make progressively less stringent queries. (While it is not obvious in the table, in most cases, only one or two documents are returned by the more stringent searches. In most other cases, the stringent searches returns no items, probably because the document was substantially modified since the last time the crawler reached it.) For example, a Google query succeeds only when all the terms are provided; the Alta Vista query will find the most relevant pages, which do not necessary include all of the query terms. Performing the Google query, and then performing the Alta Vista query if Google fails, locates the desired reference in all but one case.

Of course, most of these search engines (but not Google) offer "advanced options" that afford the user more control, leaving open the possibility of addition strategies.

Note that the results actually be even better than the table would suggest. For example, in several cases an identical paper with a different URL occurs earlier in the result set. Similarly, some of the other pages may produce access to the document a link or two away, so even though we do not count this as successful, it might be helpful to the user.

To understand why such a small number of terms can uniquely identify a web page, we suggest the following line of reasoning. There are probably a very large number of distinct terms on the web; 500,000 is probably a conservative estimate. Then the number of distinct combinations of 5 terms is greater than 3x10*28. Assuming the web is populated by documents whose most characteristic terms are uniformly drawn at random, the probability that more than one document matches a set of 5 characteristic terms is very small indeed.

Of course, the assumption of uniform distribution is highly questionable. Perhaps the empirical results indicate that it is not that far off in practice. Interesting, even among intuitively similar documents (e.g., separate chapters of the same book), signatures seem not to overlap much. In addition, TF-IDF-based signatures are by definition skewed toward infrequent terms--most of the signatures we have seen contain domain-specific abbreviations, proper names, jargon, et cetera, which may each occur only in a few dozen documents, narrowing down the set of matching documents very rapidly.

Indeed, in our examples above, cutting the signature length down to three terms changes the query results only slightly. (One signature would fail to readily locate its target in this case.) Determining the optimal length will presumably require some empirical experimentation and study. However, the method does not depend on a standard length signature, so different implementations are free to use different lengths as well as different methods of computing them.

We find these results are encouraging, if only impressionistic. We have automated the process of signature creation and subsequent searching, and have found these result to be consistent with our (still somewhat limited) experience. For example, we created signatures for all the references at the end of paper. The results are virtually identical to those in the table. However, we have resisted the temptation to report more thorough empirical testing for several reasons. First, we are depending on web search engines, whose performance changes from moment to moment. Second, these results, and any other empirical testing we might do, compute signatures and search for pages that have not actually been moved, whereas in real use, old signatures would be used to search for moved, possibly changed, pages. Therefore, we do not think that generating a large quantity of artificial data will be definitive proof that robust hyperlinks can be used effectively, which will of necessity require empirical testing by real users. Finally, as we discuss below, there are many ways in which one might vary and possibly improve the details of this scheme (all of which are mutually compatible). Thus, we present sample empirical results to demonstrate feasibility, and to encourage experimentation and use.

Integrating Robust Hyperlinks into the Web

Encoding Signatures in URLs

Given that lexical signatures are a good way to augment URLs, we are left with the issue of how to include these in hyperlinks. Here we discuss several alternatives. For the purposes of this discussion, suppose that the hyperlink has the URL *http://www.something.dom/a/b/c*, and that the designated resource has the signature *w1,...,w5*.

Robust URLs (Incompatible with existing web syntax): One can introduce new syntax for URLs to identify the signature, as for example XPointer **[XPointer]** does for sub-resource references. Doing so is incompatible with existing URLs, and makes for an awkward transition in adopting the scheme. (However, should robust URLs come into widespread use, such a proposal might merit further consideration.)

Robust URLs (Mostly Compatible): Another approach is to append the signature to the URL as if it were a query term, that is:

```
http://www.something.dom/a/b/c?lexical-signature="w1+w2+w3+w4+w5"
```

(If the URL already includes a query, the proposal is to append the signature expression with a "&".)

In this approach, if one's client is "robust-hyperlink-aware", it strips the signature expression before attempting traditional dereferencing. An advantage of this approach is that, if one's client is not robust-hyperlink-aware, one can use a local proxy to intercept all hyperlinks, and perform any robust hyperlink processing there, including stripping out the lexical-signature expression for traditional dereferencing. In addition, the robust hyperlink is just a URL, and hence can be passed around easily, as in an email message (although we suspect that this will not be an important use).

The primary disadvantage of this approach is that it may have some adverse interactions with non-aware clients and servers. However, most widely used HTTP servers ignore query terms for a non-script URL, and most services seem to ignore what appear to be gratuitous search terms. For example, the table above reports, in the "Accepts Robust URLs?" column, the results of submitting a robust URL in place of each original URL. These are uniformly successful. Moreover, the sample covers the major web servers in use today: Apache (with over 50% of the market), Microsoft Internet Information Server (24%), and Netscape Enterprise (7%) **[Netcraft 1999]**.

Of course, the term "lexical-signature" might in fact be a valid search term for some service. While one can arbitrarily decrease the possibility of a collision by using an even more obscure name, the term is probably of sufficient rarity already. Once the term is established in the future, well-informed web designers will know not to use it.

So, for the most part, "robust URLs" will be harmless to non-aware clients and servers, if not universally so. If the scheme becomes popular, the minority of web sites hostile to unknown parameters may become less so, or perhaps explicitly recogize one more.

Robust Link Elements. Another possibility is to include the signature in the markup rather than as part of the URL. For example, suppose the URL was part of the following HTML anchor element:

```
<a href="http://www.something.dom/a/b/c">click here</a>
```

We could robustify this to

```
<a href="http://www.something.dom/a/b/c"
lexical-signature="w1+w2+w3+w4+w5">click here</a>
```

The advantage of this proposal is that it should be ignored completely by non-aware clients. A disadvantage is that, if one's client is not robust-hyperlink-aware, then one can't set up a local proxy to intercept a dereferencing attempt, as one can for robust URLs. Furthermore, this embedding works only for document formats that can harmlessly accept new attributes, limiting its use to HTML, XML, SGML (and even there, documents will fail validation against un-updated DTDs), and excluding other text and multimedia types.

System Support for Robust Hyperlinks

To take advantage of robust hyperlinks, some piece of software needs to exploit them. Ideally, browsers will support them directly. This would require no changes in servers or other infrastructure, and users would benefit as soon as they update their browser. Until this happens, support can be provided by a proxy server that retains the major benefits, though it requires the assent of the web site administrator.

Robust Proxy Module. A robust proxy module transparently makes robust all URLs that pass through it, from any client connecting to any server. When a client requests a page with a non-robust URL, the module signs the returned incoming page and sends a redirect HTTP signal to the client to enable it to accept the robust hyperlink. This hyperlink can then be saved as a bookmark, used in an HTML document, or emailed. Moreover, when clients send robust hyperlinks to the proxy module, the signature is stripped off so as to minimize the chance of adverse interactions with servers. If the server returns an HTTP error code 404, the proxy engages in signature-based dereferencing, i.e., sends the signature to one or more search engines.

The advantage of this approach is that the proxy server can always correctly interpret the robust URL, and handle interactions for all clients, regardless of whether they are robust-hyperlink-aware. Sites can buy in one at a time. One disadvantage is that someone has to set up and manage the proxy service. Another disadvantage is that one always suffers the (small) overhead of going through the proxy, even if conventional dereferencing succeeds. (It might be possible to save some overhead if one's client supports some sort of "fail-over" capabilities, so that the proxy or software agent would only be contacted once traditional dereferencing fails.)

Robust Proxy Service. Another possibility is to include the signature in the URL, as before, but preface the URL with an aware proxy-service URL. For example, the URL might look like this:

```
http://www.myproxyserver.dom/cgi-bin?url="http://www.something.dom/a/b/c"
&lex-signature=w1+w2+w3+w4+w5
```

The proxy presumably performs traditional dereferencing on the URL, and uses the signature only if that fails. The advantage of this approach is that the proxy server can always correctly interpret the robust URL description, and handle interactions for all clients, regardless of whether they are robust-hyperlink-aware. This scheme has a number of significant disadvantages. Someone has to set up and manage the proxy service (although perhaps some enterpreneur will find it valuable to provide such a service). It roughly doubles web traffic, as each URL request suffers an additional round trip through the service, even if conventional dereferencing succeeds. Many people would not care to expose a complete record of their web browsing. (Even if the content is encrypted, the sites visited cannot be). Finally, once clients support robust hyperlinks, all hyperlinks encoded this way would need to be translated.

Limitations

The proposed scheme is not without its limitations:

Non-indexed documents. The scheme will only work for documents indexed by web search engines. Many important classes of documents, for example, PostScript, DVI, compressed documents, and images are generally not indexed. As web search engines improve, however, robust hyperlink coverage will improve with it. Furthermore, it has been claimed that search engines are beginning to lag considerably behind the state of the web, and the performance of signature dereferencing can only be as successful their coverage permits. (Of course, that signatures are computed independently of indexing, so that once a signed document becomes indexed, the existing signature immediately gains currency.)

Moreover, large numbers of documents live behind a firewall or are accessible only through a script. In the case of those behind a firewall, they are presumably accessible only to users within the local administrative domain. Moreover, if the documents are indexed behind the firewall, it is possible for a local robust hyperlink module to handle them. I.e., one's robust hyperlink agent could maintain a list of search engines to use, and could try the local search engine before using global ones.

In the case of documents behind a script, there may also be a helpful search engine at the site. Finding the location of such a service and automatically exploiting it remains an interesting research challenge.

Duplicates. There are many duplicates of documents on the web. In this case, signature-based dereferencing may return a substantial result. Probably this fact does not represent a substantial problem, as presumably any of the duplicates will be reasonable replacements for the moved document.

Variation in search engine performance. Signature dereferencing performance relies on the performance of search engines, which is highly variable. Moreover, signatures should be computed to work with the minimal common functionality of search engines, namely that only page content (not comments, not scripts or style sheets or other special features), with HTML tags stripped out, and with words defined as continuous strings of alphabetic characters (no numbers).

Resources with highly variable content. Some resources, e.g., a newspaper home page, vary in content very quickly over time. In these cases, a straightforward signature will not be of much use. However, if the page is changing frequently, it is likely that it is a live page so that the original URL remains valid.

Non-textual resources. Non-textual resources (e.g., images and video) are not generally indexed based on their content. Two possible extensions are as follows: (i) If an image, et cetera, is embedded within an obvious textual context, the signature of that context could be used instead. (ii) Various attempts to analyze images by content are under way (e.g., see **[Carson et al., 1999]**) in ways that provide a set of terms that are rankable and indexable; should one of these become available as the basis for a global image search engine, then analogous image signatures could conceivably be created. At this stage, it is premature to determine whether image signatures will be viable.

Extensions

We mean for the results presented above to suggest that a useful level of performance is readily obtainable, and hence, that the approach is viable. There are any number of ways in which one might improve these results, both in the practice of computing signatures and by the action of robust-hyperlink-aware agents in dereferencing dangling links.

Signature Creation. Upon creation of the initial candidate signature, the signature creation agent can immediately perform a search to see how well it works in locating the reference. If, as in some of our examples above, the stringent search produces no results, one may try computing an alternate signature. (One possibility here is to choose among the top 2n significant terms to produce the best performing n term signature.)

One interesting case is that of resources with highly variable contents. As mentioned above, a signature is likely to be less helpful in such cases (and less needed, as well). One possibility for such cases is to incrementally compute *adaptive signatures*, that is, signatures that are computed over time, so that they contain terms that persist over time.

Signature Variations. One may want to modify a strict TF-IDF ranking to include other criteria. For example, one might want to preclude all signature terms from occurring in the same sentence, or from containing words that appear to be misspellings, or words that occur only once in a document, as these terms might be subject to a greater probability of modification that will render the signature ineffective. (In our own implementation, we have experimented with some of these variants, but have not found significant differences in the results.) Indeed, one can compute longer or shorter signatures as desired, or even hand-engineer them, and still have a signature that will operate with robust-hyperlink-aware agents.

In addition, the particular lexical signatures we suggest are just one form of lexical signature, which are in turn just one form of signature. TF-IDF-based lexical signatures have the advantage of graceful degradation, and of a search infrastructure supporting them already in place. However, it might be possible to devise other strategies for lexical signature computation that will be superior. In addition, other, non-lexical forms of signatures might be devised that could have other attractive properties. If some of these are important enough, it might be worth the effort of search engines to compute these as they crawl the web.

Signature Dereferencing Strategies. If the document has been edited to remove one or more lexical signature terms, searches requiring all terms to be present in the document will of course fail. In this case, any number of "back off" strategies can be employed to widen the search. For example, above we suggested using more liberal search engine semantics, which still tends to return the desired document as one of the top few choices. However, signature dereferencing agents are free to create their own back-off strategies, for example, incrementally eliminating terms, or combining multiple search engine results in various ways.

Note that our basic signature dereferencing strategy uses web searching as a proxy for signature matching. However, once a result set is obtained, one's agent might compute signatures of the result set members, and use the actual signature matches rather than the search results. As an example, the URL **http://developer.apple.com/techpubs/mac/Cyberdog/Cyberdog-6.html** has a signature, which, when given to Google, returns 6 documents, with the document generating the signature coming in second. This result is perfectly acceptable, and isn't surprising, as the document is just a book chapter, and the first place document, the book's glossary. However, none of the other 5 items found by Google have a signature that is even close to that of the original. Thus, should one want to spend the effort computing the signatures of the result set and matching them to the original, the result can be improved (moving up from second to first place in this example).

Robust Hyperlink Agents Above we mentioned that a robust-hyperlink-aware agent, either a client or proxy, generally attempts traditional dereferencing, followed by signature-based dereferencing should that fail. However, other actions might be carried out by such an agent. It might provide a means to persistently change a reference for the given user. For example, the client may maintain a list of hyperlink remappings for the user, so that subsequent uses of a URL will automatically be remapped to the one the user selected. If one uses a robust proxy service, the service might keep track of previous users' replacement suggestions, and offer these to future users. If the resources belongs to the user, the agent might offer to edit the hyperlink for the user. (Similarly, a robust proxy service might keep track of users' suggested hyperlink replacements, and, at some point, offer this history to the author as suggestions for link replacements.)

Aware agents might use signatures even when traditional referencing succeeds. For example, since computing a signature is relatively cheap, one's client might always compute the signature of a document it locates, and compare it to that in the hyperlink. If the signatures depart significantly, the user might be advised, and perhaps given the choice of performing signature-based dereferencing.

A more conservative strategy might be to perform signature checking only if there is some other indication that the document located may not be the same as that originally referenced. For example, suppose a document has several hyperlinks to sub-resources of a given document. Sub-resources might be named anchors in HTML, or XPointers **[XPointer]**, or robust location references, such as those used in Multivalent Documents **[MVD 1998a]**, **[Phelps and Wilensky, 1998]**. In each case, it is possible for the resource reference to be successfully dereferenced, but for the sub-resource not to be found. For example, most web clients, when given a URL of the form http://...#name, will simply ignore the absence of an anchor named "name". Failure to resolve sub-resource references might instead be interpreted as an indication to perform signature checking.

Other Applications of Lexical Signatures

Simple signatures along the lines of the ones we suggest for robust hyperlinks may have other applications as well. In particular, we speculate that they may have some utility for detecting duplicate or plagiarized documents. Investigating such applications is a topic for future research.

Relation to Other Work

The primary difference between robust hyperlinks and other approaches is essentially a practical one. Namely, one does not require administrative buy-in, the creation of infrastructure, or agreement on conventions for robust hyperlinks to work. In addition, the storage, computational and communication requirements are modest.

An alternative approach to robust hyperlinks might be feasible if systems like the Alexa's archive of the web **[Alexa]** can be made relatively complete. That is, if one can always find an old reference in an archive, one can use it as a query for the current version of the resource. The feasibility of this approach is a function of the completeness of a web archive (which, in effect, allows one to compute signatures retroactively).

Implementation

We have provided support for robust URLs in the Multivalent Document System **[Phelps, 1998]**, **[Phelps and Wilensky, 1998]**. In particular, the system will compute the signature of a document on request, and if performance measurements are desired, submit signatures to multiple search engines. URLs saved as bookmarks are currently automatically made robust.

Under development are two tools to bring most of the advantages of robust hyperlinks to users of standard browers. The first is a module for the Apache web server that makes robust all web traffic that pass through it, as described above. The second piece of software transparently updates the URLs in a web site to make the robust, by crawling the site and rewriting HREFs. Robust hyperlinks are encoded by appending the signature to the URL as in the form of CGI arguments, as described above.

Conclusion

We believe our initial findings indicate that robust hyperlinks are a viable solution to a large part of the problem of dangling pointers. Namely, we can, immediately, at small cost, and fully automatically, make links that will enable us to find textual documents with highly probability when the resource has been both moved and modified.

The approach embodied in robust hyperlinks is an example of the web being able to bootstrap new features upon those previously developed. Perhaps many other such additional capabilities will be possible.

Acknowledgements

This research was supported by Digital Libraries Initiative, under grant NSF CA98-17353.

References

(In the on-line version of this paper, the hyperlinks in the references below are robust URLs.)

[Macskassy and Shklar, 1997] Sofus Macskassy and Leon Shklar. **Maintaining information resources**. *Proceedings of the Third International Workshop on Next Generation Information Technologies* (NGITS'97), June 30-July 3, 1997, Neve Ilan, Israel.

[Alexa] **Alexa**. (http://www.alexa.com/)

[Carson et al., 1999] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. **Blobworld: Image segmentation using Expectation-Maximization and its application to image querying**, February 4, 1999. (http://elib.cs.berkeley.edu/~carson/papers/pami.html)

[CNRP] **Common Name Resolution Protocol**, 18-Oct-99. (http://www.ietf.org/html.charters/cnrp-charter.html)

[Francis et al., 1995] Paul Francis, Takashi Kambayashi, Shin-ya Sato and Susumu Shimizu. **Ingrid: A Self-Configuring Information Navigation Infrastructure** December 11-14, 1995. (http://www.ingrid.org/francis/www4/Overview.html)

[Kahn and Wilensky, 1995] Robert Kahn and Robert Wilensky. **A Framework for Distributed Digital Object Services**. cnri.dlib/tn95-01, May 13, 1995.

[Mind-it] **Mind-it**. (http://www.netmind.com/html/individual.html)

[Phelps, 1998] Thomas A. Phelps. **Multivalent Documents: Anytime, Anywhere, Any Type, Every Way User-Improvable Digital Documents and Systems.** Ph.D. Dissertation, University of California, Berkeley. UC Berkeley Division of Computer Science Technical Report No. UCB/CSD-98-1026, December 1998. Also see the **general** and **technical** home pages.

[Wilensky and Phelps , 1998] Robert Wilensky and Thomas A. Phelps . **Multivalent Documents: A New Model for Digital Documents**. UC Berkeley Division of Computer Science Technical Report, CSD-98-999, March 13, 1998.

[OCLC] **OCLC PURL Service**. (http://www.purl.org)

[Netcraft 1999] **Netcraft Web Server Survey**. December 1999 (http://www.netcraft.com/survey/).

[Sollins and Masinter, 1994] K. Sollins and L. Masinter. **Functional Requirements for Uniform Resource Names**, Network Working Group Request for Comments 1737, December 1994.

[Ingham et al., 1996] David Ingham, Steve Caughey, Mark Little. **Fixing the "Broken-Link" Problem: The W3Objects Approach**. Computing Networks & ISDN Systems, Vol. 28, No. 7-11, pp. 1255-1268: Proceedings of the Fifth International World Wide Web Conference, Paris, France, 6-10 May 1996.

[XPointer] **XML Pointer Language (XPointer)**. W3C Working Draft 9 July 1999. (http://www.w3.org/1999/07/WD-xptr-19990709)