Finding Related Pages in the World Wide Web

Jeffrey Dean   Monika R. Henzinger

Compaq Computer Corporation Systems Research Center
130 Lytton Ave.
Palo Alto, California 94301
E-mail: {`monika,jdean`}`@pa.dec.com`

Abstract

When using traditional search engines, users have to formulate queries to describe their information need. This paper discusses a different approach to web searching where the input to the search process is not a set of query terms, but instead is the URL of a page, and the output is a set of related web pages. A related web page is one that addresses the same topic as the original page. For example, `www.washingtonpost.com` is a page related to `www.nytimes.com`, since both are online newspapers.

We describe two algorithms to identify related web pages. These algorithms use only the connectivity information in the web (*i.e.*, the links between pages) and *not* the content of pages or usage information. We have implemented both algorithms and measured their runtime performance. To evaluate the effectiveness of our algorithms, we performed a user study comparing our algorithms with Netscape's "What's Related" service [12]. Our study showed that the precision at 10 for our two algorithms are 73% better and 51% better than that of Netscape, despite the fact that Netscape uses both content and usage pattern information in addition to connectivity information.

Keywords: search engines, related pages, searching paradigms.

1 Introduction

Traditional web search engines take a query as input and produce a set of (hopefully) relevant pages that match the query terms. While useful in many circumstances, search engines have the disadvantage that users have to formulate queries that specify their information need, which is prone to errors. This paper discusses how to find related web pages, a different approach to web searching. In our approach the input to the search process is not a set of query terms, but the URL of a page, and the output is a set of related web pages. A related web page is one that addresses the same topic as the original page, but is not necessarily semantically identical. For example, given `www.nytimes.com`, the tool should find other newspapers and news organizations on the web. Of course, in contrast to search engines, our approach requires that the user has already found a page of interest.

Recent work in information retrieval on the web has recognized that the hyperlink structure can be very valuable for locating information [18, 3, 7, 23, 19, 25, 24, 6, 17, 5]. This assumes that if there is a link from page v and w, then the author of v recommends page w, and links often connect related pages. In this paper, we describe the Companion and Cocitation algorithms, two algorithms which use only the hyperlink structure of the web to identify related web pages. For example, Table 1 shows the output of the Companion algorithm when given `www.nytimes.com` as input (in this case, the results for the Cocitation algorithm are identical and the results for Netscape are very similar, although this is not always true).

Input: `www.nytimes.com`	
`www.usatoday.com`	USA Today newspaper
`www.washingtonpost.com`	Washington Post newspaper
`www.cnn.com`	Cable News Network
`www.latimes.com`	Los Angeles Times newspaper
`www.wsj.com`	Wall Street Journal newspaper
`www.msnbc.com`	MSNBC cable news station
`www.sjmercury.com`	San Jose Mercury News newspaper
`www.chicago.tribune.com`	Chicago Tribune newspaper
`www.nando.net`	Nando Times on-line news service
`www.the-times.co.uk`	The Times newspaper

Table 1: Example results produced by the Companion algorithm

Input: `linas.org/linux/corba.html`			
Companion		*Netscape*	
1	www.cs.wustl.edu/~schmidt/TAO.html	0	labserver.kuntrynet.com/~linux
1	dsg.harvard.edu/public/arachne	0	web.singnet.com.sg/~siuyin
1	jorba.castle.net.au/	–	www.clark.net/pub/srokicki/linux
1	www-b0.fnal.gov:8000/ROBIN	–	www.earth.demon.co.uk/linux/uk...
1	www.paragon-software.com/products/oak	0	www.emry.net/webwatcher
1	www.tatanka.com/orb1.htm	0	www.interl.net/~jasoneng/NLL/lwr
1	www.oot.co.uk/dome-index.html	0	www.jnpcs.com/mkb/linux
0	www.intellisoft.com/~mark	1	www.linuxresources.com/
1	www.dstc.edu.au/AU/staff/mart...	0	www.liszt.com/
1	www.inf.fu-berlin.de/~brose/jacorb	0	www.local.net/~jgo/linuxhelp.html

Table 2: Comparison of results for the Companion and Netscape algorithms. A "1" means that the page was valuable, a "0" means that the page was not valuable, a "–" means that the page could not be accessed.

One of our goals was to design algorithms with high precision that are very fast and that do not require a large number of different kinds of input data. Since we have a tool that gives us access to the hyperlink structure of the web (the *Connectivity Server* [4]), we focused on algorithms that only use connectivity information to identify related pages. Our algorithms use only the information about the links that appear on each page and the order in which the links appear. They neither examine the content of pages, nor do they examine patterns of how users tend to navigate among pages.

Our Companion algorithm is derived from the HITS algorithm proposed by Kleinberg for ranking search engine queries [17]. Kleinberg suggested that the HITS algorithm could be used for finding related pages as well, and provided anecdotal evidence that it might work well. In this paper, we extend the algorithm to exploit not only links but also their order on a page (see Section 2.1.1) and present the results of a user-study showing that the resulting algorithm works very well.

The Cocitation algorithm finds pages that are frequently cocited with the input URL u (that is, it finds other pages that are pointed to by many other pages that all also point to u).

Netscape Communicator Version 4.06 introduced a related pages service that is built into the browser [12] (see Section 2.3 for a more detailed discussion). On the browser screen, there is a "What's Related" button, which presents a menu of related web pages in some cases. The "What's Related" algorithm in Netscape is based on technology developed by Alexa, Inc., and computes its answers based on connectivity information, content information, and usage information [11].

To compare the performance of our two algorithms and Netscape's algorithm, we performed a user study on 59 URLs chosen by 18 volunteers. Our study results show that the precision at 10 computed over all 59 URLs of our two algorithms are 73% better and 51% better than Netscape's. Not all algorithms gave answers for all URLs in our study. If we restrict the comparison to only the 37 URLs for which all three algorithms returned answers, then the precision at 10 of our two algorithms are 40% and 22% better than Netscape's algorithm. This is surprising since our algorithms are based only on connectivity information.

Netscape's algorithm gives answers for about 17 million URLs [11], while our algorithms can give answers for a much larger set of URLs (we have connectivity information on 180 million URLs). This is important because it means that we can give related URL information for more URLs. Our algorithms are also fast: in our environment, both average less than 200 msec of computation per input URL.

The example shown in Table 1 is for a URL with a very high level of connectivity (`www.nytimes.com` contains 47,751 inlinks in our Connectivity Server), and all three algorithms generally perform quite well for well-connected URLs. Our algorithms can also work well when there is much less connectivity, as shown by the example in Table 2. This table shows the answers for the Companion and Netscape algorithms for `linas.org/linux/corba.html`, one of the input URLs chosen by a user as part of our user study. Alongside each answer is the user's rating for each answer, with a '1' meaning that the user considered the page related, '0' meaning that the user considered the page unrelated, and '–' meaning that the user was unable to access the page at all. The original page was about CORBA implementations for Linux, and there were 123 pages pointing to this page in our Connectivity Server. Nine of the ten answers given by the Companion algorithm were deemed related by our user, while only one page from Netscape's set of answers was deemed related. Most of Netscape's answers were about the much broader topic of Linux, rather than specifically about CORBA implementations on Linux.

Section 2 presents our algorithms in detail and describes Netscape's service, while Section 3 discusses the implementation of our algorithms. Section 4 describes the user study we performed and presents its results, and also provides a brief performance evaluation of our algorithms. Finally, Section 6 presents our conclusions.

2 Related Page Algorithms

In this section we describe our two algorithms (the *Companion* algorithm and the *Cocitation* algorithm), as well as Netscape's algorithm. Unlike Netscape's algorithm, both of our algorithms exploit only the hyperlink-structure (i.e. graph connectivity) of the web and do not examine the information about the content or usage of pages. Netscape's algorithm uses all three kinds of information to arrive at its results.

In the sections below, we use the terms *parent* and *child*. If there is a hyperlink from page w to page v, we say that w is a *parent* of v and that v is a *child* of w.

2.1 Companion Algorithm

The Companion algorithm takes as input a starting URL u and consists of four steps:

1. Build a *vicinity graph* for u.

2. Contract duplicates and near-duplicates in this graph.

3. Compute edge weights based on host to host connections.

4. Compute a *hub* score and an *authority* score for each node in the graph and return the top ranked authority nodes (our implementation returns the top 10). This phase of the algorithm uses a modified version of the HITS algorithm originally proposed by Kleinberg [17].

These steps are described in more detail in the subsections below. Only step 1 exploits the order of links on a page.

2.1.1 Step 1: Building the Vicinity Graph

Given a query URL u we build a directed graph of nodes that are nearby to u in the web graph. Graph nodes correspond to web pages and graph edges correspond to hyperlinks. The graph consists of the following nodes (and the edges between these nodes):

1. u,
2. up to B parents of u, and for each parent up to BF of its children different from u, and
3. up to F children of u, and for each child up to FB of its parents different from u.

Here is how we choose these nodes in detail: There is a stoplist $STOP$ of URLs that are unrelated to most queries and that have very high indegree. Our stoplist was developed by experimentation, and currently contains 21 URLs. Examples of nodes that appear on our stoplist are `www.yahoo.com` and `www.microsoft.com/ie/download.html`. If the starting URL u is not one of the nodes on our stoplist, then we ignore all the URLs on the stoplist when forming the vicinity graph. If u does appear on the stoplist, however, then we disable the stoplist (i.e. set $STOP$ to the empty list) and freely include any nodes in the vicinity graph. We disable the stoplist when the input URL appears on the stoplist because many nodes on the stoplist are popular search engines and portals, and we want to permit these nodes to be considered when the input URL is another such popular site.

Go back (B), and back-forward (BF): If u has more than B parents, add B random parents not on $STOP$ to the graph; otherwise add all of u's parents. If a parent x of u has more than $BF+1$ children, add up to $BF/2$ children pointed to by the $BF/2$ links on x immediately preceding the link to u and up to $BF/2$ children pointed to by the $BF/2$ links on x immediately succeeding the link to u (ignoring duplicate links). If page x has fewer than BF children, we add all of its children to the graph. Note that this exploits the order of links on page x.

Go forward (F), and forward-back (FB): If u has more than F children, add the children pointed to by the first F links of u; otherwise, add all of u's children. If a child of u has more than BF parents, add the BF parents not on $STOP$ with highest indegree; otherwise, add all of the child's parents.

If there is a hyperlink from a page represented by node v in the graph to a page represented by node w and v and w do not belong to the same host, then there is a directed edge from v to w in the graph (we omit edges between nodes on the same host).

In our experience, we have found that using a large value for B (2000) and a small value for BF (8) works better in practice than using moderate values for each (say, 50 and 50). We have observed that links to pages on a similar topic tend to be clustered together, while links that are farther apart on a page are less likely to be on the same topic (for example, most hotlists are grouped into categories). This has also been observed by other researchers [9]. Using a larger value for B also means that the likelihood of the computation being dominated by a single parent page is reduced.

2.1.2 Step 2: Duplicate Elimination

After building this graph we combine *near-duplicates*. We say two nodes are near-duplicates if (a) they each have more than 10 links and (b) they have at least 95% of their links in common. To combine two near-duplicates we replace their two nodes by a node whose links are the union of the links of the two near-duplicates. This duplicate elimination phase is important because many pages are duplicated across hosts (e.g. mirror sites, different aliases for the same page), and we have observed that allowing them to remain separate can greatly distort the results.

2.1.3 Step 3: Assign Edge Weights

In this stage, we assign a weight to each edge, using the edge weighting scheme of Bharat and Henzinger [5] which we repeat here for completeness. An edge between two nodes on the same host[1] has weight 0. If there are k edges from documents on a first host to a single document on a second host we give each edge an *authority weight* of $1/k$. This weight is used when computing the authority score of the document on the second host. If there are l edges from a single document on a first host to a set of documents on a second host, we give each edge a *hub weight* of $1/l$. We perform this scaling to prevent a single host from having too much influence on the computation.

We call the resulting weighted graph the *vicinity graph* of u.

2.1.4 Step 4: Compute Hub and Authority Scores

In this step, we run the *imp* algorithm [5] on the graph to compute *hub* and *authority* scores. The *imp* algorithm is a straightforward extension of the HITS algorithm with edge weights.

The intuition behind the HITS algorithm is that a document that points to many others is a good hub, and a document that many documents point to is a good authority. Transitively, a document that points to many good authorities is an even better hub, and similarly a document pointed to by many good hubs is an even better authority. The HITS computation repeatedly updates hub and authority scores so that documents with high authority scores are expected to have relevant *content*, whereas documents with high hub scores are expected to contain *links* to relevant content. The computation of hub and authority scores is done as follows:

> Initialize all elements of the hub vector H to 1.0.
> Initialize all elements of the authority vector A to 1.0.
> While the vectors H and A have not converged:
>   For all nodes n in the vicinity graph N,
> $A[n] := \sum_{(n',n)\in edges(N)} H[n'] \times authority\_weight(n', n)$
>   For all n in N,
> $H[n] := \sum_{(n,n')\in edges(N)} A[n'] \times hub\_weight(n, n')$
>   Normalize the H and A vectors.

Note that the algorithm does not claim to find *all* relevant pages, since there may be some that have good content but have not been linked to by many authors.

The Companion algorithm then returns the nodes with the ten highest authority scores (excluding u itself) as the pages that are most related to the start page u.

[1] We assume throughout the paper that the *host* can be determined from the URL-string.

2.2 Cocitation Algorithm

An alternative approach for finding related pages is to examine the siblings of a starting node u in the web graph. Two nodes are *co-cited* if they have a common parent. The number of common parents of two nodes is their *degree of co-citation*. As an alternative to the Companion algorithm, we have developed a very simple algorithm that determines related pages by looking for sibling nodes with the highest degree of co-citation. The Cocitation algorithm first chooses up to B arbitrary parents of u. For each of these parents p, it adds to a set S up to BF children of p that surround the link from p to u. The elements of S are siblings of u. For each node s in S, we determine the degree of cocitation of s with u. Finally, the algorithm returns the 10 most frequently cocited nodes in S as the related pages.

In some cases there is an insufficient level of cocitation with u to provide meaningful results. In our implementation, if there are not at least 15 nodes in S that are cocited with u at least twice, then we restart the algorithm using the node corresponding to u's URL with one path element removed. For example, if u's URL is `a.com/X/Y/Z` and an insufficient number of cocited nodes exist for this URL, then we restart the algorithm with the URL `a.com/X/Y` (if the resulting URL is invalid, we continue to chop elements until we are left with just a host name, or we find a valid URL).

In our implementation, we chose B to be 2000 and BF to be 8 (the same parameter values we used for our implementation of the Companion algorithm).

One way of looking at the Cocitation algorithm is that it finds "maximal" n X 2 bipartite subgraphs in the vicinity graph.

2.3 Netscape's Approach

Netscape introduced a new "What's Related?" feature in version 4.06 of the Netscape Communicator browser. Details about the approach used to identify related pages in their algorithm are sketchy. However, the What's Related FAQ page indicates that the algorithm uses connectivity information, usage information, and content analysis of the pages to determine relationships. We quote from the "What's Related" FAQ page:

> *The What's Related data is created by Alexa Internet. Alexa uses crawling, archiving, categorizing and data mining techniques to build the related sites lists for millions of web URLs. For example, Alexa uses links on the crawled pages to find related sites. The day-to-day use of What's Related also helps build and refine the data. As the service is used, the requested URLs are logged. By looking at high-level trends, Netscape and Alexa can deduce relationships between web sites. For example, if thousands of users go directly from site A to site B, the two sites are likely to be related.*
>
> *Next, Alexa checks all the URLs to make sure they are live links. This process removes links that would try to return pages that don't exist (404 errors), as well as any links to servers that aren't available to the general Internet population, such as servers that are no longer active or are behind firewalls. Finally, once all of the relationships are established and the links are checked, the top ten related sites for each URL are chosen by looking at the strength of the relationship between the sites.*
>
> *Each month, Alexa recrawls the web and rebuilds the data to pull in new sites and to refine the relationships between the existing sites. New sites with strong relationships to a site will automatically appear in the What's Related list for that site by displacing any sites with weaker relationships.*
>
> *Note that since the relationships between sites are based on strength, What's Related lists are not necessarily balanced. Site A may appear in the list for Site B, but Site B may not be in the list*

for Site A. Generally, this happens when the number of sites with strong relationships is greater than ten, or when sites do not have similar enough content.

3 Implementation

In experimenting with these algorithms, we were fortunate to have access to Compaq's Connectivity Server [4]. The Connectivity Server provides high speed access to the graph structure of 180 million URLs (nodes) and the links (edges) that connect them. The entire graph structure is kept in memory on a Compaq AlphaServer 4100 with 8 GB of main memory and dual 466 MHz Alpha processors. The random access patterns engendered by the connectivity-based algorithms described in this paper mean that it is important for most or all of the graph to fit in main memory to prevent high levels of paging activity.

We implemented a multi-threaded server that accepts a URL uses either the Cocitation algorithm or the Companion algorithm to find pages related to the given URL. Our server implementation consists of approximately 5500 lines of commented C code, of which approximately 1000 lines implement the Companion algorithm, 400 lines implement the Cocitation algorithm, and the remainder are shared code to perform tasks such as parsing HTTP query requests, printing results, and logging status messages. We link our server code directly with the Connectivity Server library, and access the connectivity information by `mmap`ping the graph information into the address space of our server.

Our implementation of the Companion algorithm has been subjected to a moderate amount of performance tuning, mostly in designing the neighborhood graph data structures to improve data-cache performance. The implementation of the Cocitation algorithm has not been tuned extensively, although it does share a fair amount of code with the Companion algorithm, and this shared code has been tuned somewhat.

4 Evaluation

In this section we describe the evaluation we performed for the algorithms. Section 4.1 describes our user study, while Section 4.2 discusses the results of the study. Section 4.3 evaluates the run time performance of our algorithms.

4.1 Experimental Setup

To compare the different approaches, we performed a user study. We asked 18 volunteers to supply us with at least two URLs for which they wanted to find related pages. Our volunteers included 14 computer science professionals (mostly our colleagues at Compaq's research laboratories), as well as 4 people with other professional careers. We received 69 URLs and used each of the algorithms to determine the top 10 answers for each URL. We put the answers in random order and returned them to the volunteers for rating. The volunteers were instructed as follows:

We want to measure how well each algorithm performs. To measure performance we want to know the percentage of valuable URLs returned by each algorithm. To be valuable the URL must be both relevant to the topic you are interested in and a high quality page. For example, if your URL was www.audi.com and you get back a newsgroup posting where somebody talks about his new Audi car, then the page was on topic, but probably not high quality. On the other hand, if you get www.jaguar.com as an answer, then it is up to you to decide whether this answer is on topic or not.

Scoring:

Algorithm	*# of URLs with Answers*	*# of Answers*	*# of Dead Links*
Companion	50	498	42
Cocitation	58	580	62
Netscape	40	364	29

Table 3: Summary of all answers for the algorithms

0: *Page was not valuable/useful*
1: *Page was valuable/useful*
–: *Page could not be accessed (i.e. did not exist, or server was down)*

Please ignore the order in which the pages are returned. So if a later page contains similar content to an earlier page please rate the latter page as if you had not seen the earlier page. This will imply that we do not measure how "happy" you are with a set of answers returned by an algorithm. Instead we measure how many valuable answers each algorithm gives.

4.2 User Study Results

We received responses rating the answer URLs for 59 of the input URLs. These 59 input URLs form the basis of our study. Table 3 shows how many of these queries the algorithms answered and how many answer URLs they returned. In many cases, the algorithms returned links that our users rated as inaccessible. The column labeled *# of Dead Links* shows the number of inaccessible pages among all the answers for each algorithm. For the purposes of our evaluation, we treat an inaccessible link as a score of '0', since inaccessible pages are not valuable/useful.

The Cocitation algorithm returned results for all but one of the URLs. The reason why it returned results for almost all input URLs is that when insufficient connectivity was found surrounding an input URL (e.g. `a.com/X/Y`), the Cocitation algorithm used a chopped URL as input (*e.g.* `a.com/X`). Although we did not include this chopping feature in our implementation of the Companion algorithm, it is directly applicable and would enable the Companion algorithm to return answers for more URLs. We have empirically observed that Netscape's algorithm also applies a similar chopping heuristic in some cases.

Table 4 contains a listing of the 59 URLs in our study. For each URL, the three columns labeled *Cp*, *Ct*, and *N* show the URLs for which the Companion, Cocitation, and Netscape algorithms returned results, respectively. The table also shows the number of hyperlinks pointing to the URL in the Connectivity Server (*Inlinks*). For the Companion algorithm, it shows the number of nodes and edges in the vicinity graph, as well as the wall clock time in milliseconds taken to compute the set of related pages (computed by surrounding the computation with `gettimeofday` system calls). For the cocitation algorithm, it shows the number of siblings found, the number of siblings cocited at least twice (*Cocited*), and the wall clock time taken to compute the answers.

The three algorithms return answers for different subsets of our 59 URLs. To compare these algorithms, we can subdivide the URLs into several groups. The *intersection* group consists of those URLs where all algorithms returned at least one answer. There were 37 URLs in this group. The *non-Netscape* group consists of the URLs where Netscape's approach did not return any answers. It consists of 19 URLs.

To quantify the performance of the three algorithms, we now defined two metrics. The *precision at r* for a given algorithm is the total number of answers receiving a '1' score within the first r answers, divided by r times the number of URLs. Notice that when an algorithm does not give any answers for a URL, this is as if it gave all non-relevant answers for that URL.

					Companion algorithm			Cocitation algorithm		
URL	Cp	Ct	N	In links	Nodes	Edges	Time (msec)	Siblings	Cocited	Time (msec)
1. babelfish.altavista.digital.com/cgi...	√	√	√	29284	6382	10305	269	6601	1079	573
2. developer.intel.com/design/strong/t...	√	√	√	19	333	479	12	85	39	17
3. english-server.hss.cmu.edu/	√	√	√	3774	4197	8263	215	6630	2063	584
4. hack.box.sk/	√	√	√	1666	7963	14090	488	6099	1493	515
5. ieor.berkeley.edu/~hochbaum/html/bo...	√	√		5	64	73	4	49	15	17
6. linas.org/linux/corba.html	√	√	√	123	2028	4222	77	444	138	48
7. members.tripod.com/~src-fall-regatt...		√		0	0	0	0	6800	1523	736
8. metroactive.com/music/	√	√	√	13	201	302	67	168	58	31
9. travelwithkids.miningco.com/	√	√		99	744	1051	27	459	128	471
10. www-db.stanford.edu/~wiener	√	√		2	17	18	3	816	301	100
11. www.acf.dhhs.gov/	√	√	√	1822	5522	14967	394	3659	1401	424
12. www.adventurewest.com/pub/NASTC/	√	√	√	13	65	92	4	52	16	14
13. www.amazon.com/exec/obidos/cache/br...		√	√	0	0	0	0	1149	411	262
14. www.anglia.ac.uk/~systimk/Humour/Hi...		√		0	0	0	0	103	27	9
15. www.ayso26.org/	√			5	142	169	8	0	0	0
16. www.babynames.com/	√	√	√	528	2695	4781	110	1826	496	167
17. www.braintumor.org/	√	√	√	201	779	1763	35	551	212	68
18. www.bris.ac.uk/Depts/Union/BTS/Scri...	√	√	√	131	505	1084	162	379	152	72
19. www.carrier.com/	√	√	√	964	2755	5672	123	1837	580	165
20. www.chesschampions.com/kasparov.htm...	√	√	√	11	57	90	3	535	220	55
21. www.cl.cam.ac.uk/users/rja14/tamper...	√	√	√	119	741	1256	27	452	155	36
22. www.davecentral.com/	√	√	√	6909	4703	11391	263	4349	1148	477
23. www.duofold.com/stepout/ski-wsa.htm		√		0	0	0	0	185	66	25
24. www.ebay.com/	√	√	√	4658	4389	8660	199	5951	1314	493
25. www.etoys.com/	√	√	√	2579	2294	6153	111	3351	1005	427
26. www.fifa.com/	√	√	√	12815	6105	12360	349	6452	1452	522
27. www.focus.de/	√	√	√	7662	4881	14039	361	4208	1301	525
28. www.geocities.com/Paris/Metro/1324/	√	√		11	92	138	7	39	15	30
29. www.geocities.com/TheTropics/2442/d...	√	√		64	450	685	19	184	65	25
30. www.harappa.com/har/har0.html	√	√	√	38	235	318	86	202	37	23
31. www.harmony-central.com/MIDI/Doc/tu...	√	√	√	31	184	249	7	153	27	21
32. www.hotelres.com/	√	√	√	708	3072	6036	148	2380	889	196
33. www.innovation.ch/java/HTTPClient/	√	√	√	73	341	639	13	215	68	21
34. www.inquizit.com/	√	√		12	62	95	4	54	18	15
35. www.israelidance.com/	√	√	√	40	210	323	9	173	40	16
36. www.jewishmusic.com/	√	√	√	399	1617	3111	66	1232	397	134
37. www.joh.cam.ac.uk/	√	√		162	557	1027	23	409	114	37
38. www.levenger.com/	√	√	√	259	1367	2083	51	1116	274	87
39. www.mdl.sandia.gov/micromachine/gal...		√		0	0	0	0	143	61	15
40. www.midiweb.com/	√	√	√	1967	5304	15549	340	3370	1141	408
41. www.minimed.com/	√	√	√	217	778	1663	33	573	223	69
42. www.mit.edu/people/mkgray/net/	√	√	√	258	1138	2124	44	896	287	72
43. www.mot-sps.com/	√	√	√	391	883	1896	38	499	206	54
44. www.movielink.com/	√	√	√	12274	6205	11589	337	5622	1207	514
45. www.netusa1.net/~spost/bench.html	√	√		1	9	9	3	944	229	86
46. www.nsc.com/catalog/sg708.html		√	√	0	0	0	0	2774	1149	333
47. www.odci.gov/cia/publications/factb...	√	√	√	3765	4352	8322	221	6708	1962	613
48. www.paccc.com/	√	√	√	18	409	625	15	64	21	7
49. www.perl.com/perl/index.html	√	√	√	14	53	92	4	39	18	5
50. www.pianospot.com/1700305.htm		√		0	0	0	0	39	23	85
51. www.rei-outlet.com /	√	√		20	105	163	6	81	16	16
52. www.sddt.com/files/library/94headli...	√	√	√	9	31	66	3	3647	1260	386
53. www.sultry.arts.su.edu.au/links/lin...	√	√		54	258	461	11	210	74	21
54. www.telemarque.com/articles/andrnch...		√	√	0	0	0	0	607	241	61
55. www.trane.com/	√	√	√	900	2965	6201	140	2096	684	181
56. www.traxxx.de/	√	√		1219	4607	10560	304	3355	1184	478
57. www.us-soccer.com/	√	√	√	1175	3458	8515	201	2344	865	230
58. www.wisdom.weizmann.ac.il/~naor/puz...	√	√		8	49	65	3	96	16	10
59. www.wwa.com/~android7/pilot/index.h...		√		0	0	0	0	1846	709	214

Table 4: User-provided URLs for evaluation (and statistics)

Algorithm	*All* *Avg. Precision*	*All* *Precision at 10*	*Intersection* *Avg. Precision*	*Intersection* *Precision at 10*	*Non-Netscape* *Avg. Precision*	*Non-Netscape* *Precision at 10*
Companion	0.541	0.417	0.666	0.501	0.540	0.401
Cocitation	0.518	0.363	0.605	0.435	0.434	0.325
Netscape	0.343	0.241	0.502	0.357	n/a	

Table 5: Precision metrics for each algorithm for three groups of URLs

Algorithms	*All* *Sign*	*All* *Rank sum*	*Intersection* *Sign*	*Intersection* *Rank sum*	*Non-Netscape* *Sign*	*Non-Netscape* *Rank sum*
Companion better than Netscape	<0.0001	0.0026	0.0041	0.0340	n/a	
Cocitation better than Netscape	0.0136	0.0164	0.1685	0.2340	n/a	
Companion better than Cocitation	0.1922	0.3898	0.0793	0.2628	0.2643	0.4180

Table 6: Sign Test and Wilcoxon Sum of Ranks Test for algorithm pairs

For a given URL u, the *average precision for* u of an algorithm is the sum of the precision at each rank where the answer of the algorithm for u received a '1' score divided by the total number of the answers of the algorithm for u receiving a '1' score. If the algorithm does not return any answers for u, its average precision for u is 0. The overall *average precision* for an algorithm is the sum of all the average precisions for all the URLs divided by the total number of URLs.

For each of the three groups of URLs (*all*, *intersection*, and *non-Netscape*), Table 5 shows the average precision and the precision at 10 for each algorithm. Figure 1 shows the precision at r for each of these groups of URLs in graphs (a), (b), and (c). Figures 1 (a) and 1 (b) illustrate that the Companion and Cocitation algorithms substantially outperform Netscape's algorithm at all ranks, and the Companion algorithm almost always outperforms the Cocitation algorithm.

The *intersection* group is the most interesting comparison, since it avoids penalizing an algorithm for not returning at least one answer. For the *intersection* group, Netscape's algorithm achieves a precision at 10 of 0.357, while the Companion algorithm achieves a precision at 10 of 0.501 (40% better), and the Cocitation algorithm achieves a precision at 10 of 0.435 (22% better). The average precision in the *intersection* group does not penalize an algorithm for returning fewer than 10 answers. Under this metric, the Companion algorithm is 32% better than Netscape's algorithm, while the Cocitation algorithm is 20% better than Netscape's algorithm.

In the group that includes all URLs, all three algorithms had drops in their precision at 10 values. There are two reasons for this. The first is that algorithms were given a precision of 0 for a

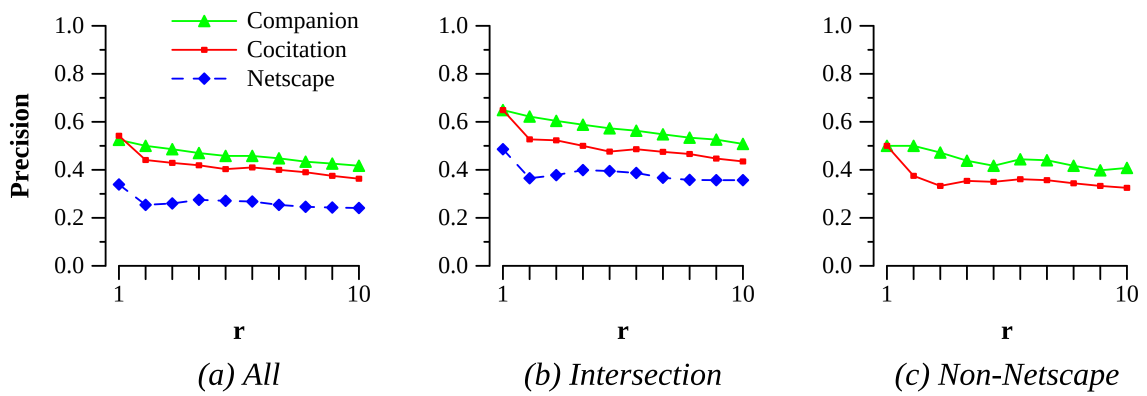

Figure 1: Precision at r for the three groups of URLs

Algorithm	Companion	Cocitation	Netscape
Companion		253 (51%)	55 (11%)
Cocitation	253 (44%)		56 (10%)
Netscape	55 (15%)	56 (15%)	

Table 7: Overlap between answers returned by algorithms

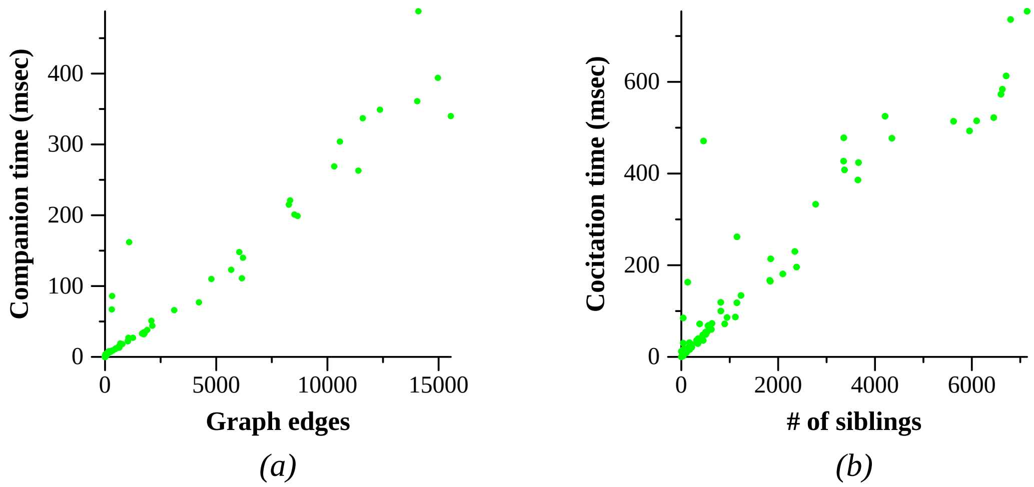

Figure 2: Graph size vs. running time of Companion and Cocitation algs.

given URL if they did not return any answers. This mostly affected the Netscape and Companion algorithms. The second reason is that for the URLs in the non-Netscape set, both the Companion and Cocitation algorithms did not perform as well as they did for URLs in the Intersection set. Despite these drops in absolute average precision, the average precision of the Companion algorithm is 57% better than that of Netscape, and the average precision of the Cocitation algorithm is 51% better than that of Netscape. Similar results hold when examining average precision rather than precision at 10.

To evaluate the statistical significance of our results, we computed the Sign Test and the Wilcoxson Sums of Ranks Test for each pair of algorithms [20]. These results are shown in Table 6 and show that the difference between the Companion and Netscape algorithms and between the Cocitation and Netscape algorithms are statistically significant.

We also wanted to evaluate whether or not the algorithms were generally returning the same results for a given URL or whether they were returning largely disjoint sets of URLs. Table 7 shows the amount of overlap in the answers returned by each pair of algorithms. The percentage in parentheses is the overlap divided by the total number of answers returned by the algorithm in that row. As the table shows, there is a large overlap between the answers returned by the Companion and Cocitation algorithms. This is not surprising, since the two algorithms are both based on connectivity information surrounding the input URL and since both use similar parameters to choose the surrounding nodes. There is relatively little overlap between the answers returned by Netscape and the other two algorithms.

4.3 Run-time Performance

In this section, we present data about the run-time performance of the Companion and Cocitation algorithms. Since we do not have direct access to Netscape's algorithm and only access it through the public web interface, we are unable to present performance information for Netscape's algorithm.

All measurements were performed on a Compaq AlphaServer 4100 with 8 GB of main memory and dual 466 MHz Alpha processors. The measured running times are wall clock times from the time the input URL is given to the server until the time the answers are returned. These times do not include the time taken to format the results as an HTML page, since that was done by a server process running on another machine (and the time to do this was negligible).

The average running time for the Companion algorithm on the 50 URLs for which it returned answers was 109 msec, while the average running time for the Cocitation algorithm on the 58 URLs for which it provided answers was 195 msec. The performance of both these algorithms is sufficiently fast that either one could handle a large amount of traffic (close to 800,000 requests per day for the Companion algorithm). Furthermore, the average performance could probably be improved by caching answers for frequently requested URLs.

Although we did not explicitly include this factor in our user study, we have informally observed that the subjective quality of answers returned for both the Companion and the Cocitation algorithms does not decrease when we decrease the parameter B (the number of inlinks considered) during the building of the vicinity graph. This is important for on-line services because it means that the graph size could be reduced during times of high load, thereby reducing the amount of time taken to service each request. Under conditions of low load, the graph size could be increased.

The Companion algorithm generally converges on its answers within a few iterations (typically less than 10 iterations), but the number of iterations increases with the graph size. Each iteration takes time that is linear in the number of edges in the vicinity graph. We plot the running time vs. the number of graph edges in Figure 2 (a).

The running time of the Cocitation algorithm is $O(n \log n)$, where n is the number of siblings examined for cocitation, since it sorts the siblings by the degree of cocitation. This effect is illustrated in Figure 2 (b). In our experience, the running times for the the cocitation and companion algorithms are generally correlated, since URLs which have a large number of siblings to consider in the cocitation algorithm also generally produce a large neighborhood graph for processing in the companion algorithm.

5 Related Work

Many researchers have proposed schemes for using the hyperlink structure of the web [18, 3, 7, 23, 19, 25, 24, 6, 17, 5]. For the most part, this work does not discuss the finding of related pages, with four exceptions discussed below.

We know of only one previous work that expoits the order of links: Chakrabarti et al. [9] use the links and their order to categorize web pages and they show that the links that are near a given link in page order frequently point to pages on the same topic.

Previous authors have suggested using cocitation and other forms of connectivity to identify related web pages. Spertus observed that cocitation could indicate that two pages are related [23]. That is, if page A points to both pages B and C, then B and C might be related. Various researchers in the field of bibliometrics have also observed this [15, 13, 14, 22], and this observation forms the basis of our Cocitation algorithm. The notion of collaborative filtering, although it is based on user's recommendations rather than hyperlinks, also relies on this observation [21]. Pitkow and Pirolli [19] cluster web pages based on co-citation analysis. Terveen and Hill [25] use the connectivity structure of the web to find groups of related web sites.

Our companion algorithm descended from the HITS algorithm developed by Kleinberg [17]. The HITS algorithm was originally proposed by Kleinberg as a way of using connectivity structure to identify the most authoritative sources of information on a particular topic, where the topic was defined by the combined link structure of a large number of starting nodes on the topic. Kleinberg also proposed that the HITS algorithm could be used to find related pages when the topic was defined by just a single node. The Companion algorithm used HITS algorithm as a starting point and extended and modified it in four main ways:

1. Kleinberg suggested using the following graph to find related pages: Take a fixed number (say 200) of parents of the given URL and call the set consisting of the URL and these parents the *start set*. Now build the graph consisting of all nodes pointing to a node in the start set or pointed to by a node in the start set. This means that "grandparents" of u are included in the graph, while nodes that share a child with u are not included in the graph. We believe that the latter nodes are more likely to be related to u than are the "grandparent" nodes. Therefore our vicinity graph is structured to exclude grandparent nodes but to include nodes that share a child with u.
2. We exploit the order of the links on a page to determine which "siblings" of u to include. When we added this feature, the precision of our algorithm improved noticably.
3. The original HITS algorithm did not have edge weights. We use edge weights to reduce the influence of pages that all reside on one host, since Bharat and Henzinger have shown that edge weights improve the precision [5].
4. We also merge nodes in our vicinity graph that have a large number of duplicate links. Duplicate nodes are not such a serious problem when using the HITS or *imp* algorithms to rank query results, since the start set consists of a large number of URLs. However, when forming the vicinity graph starting with just a single URL, the influence of duplicate nodes is increased because duplicate nodes with a large number of out links will quickly dominate the hub and authority computation.

Kleinberg also showed that HITS computes the principal eigenvector of the matrix AA^T, where A is the adjacency matrix of the above graph, and suggested using non-principal eigenvectors for finding related pages. Finally, he gave anecdotal evidence that HITS might work well.

Consecutively, a sequence of papers [10, 8] presented improvements on HITS and used it to populate a given hierarchy of categories. These improvements are not directly relevant to the task of finding related pages.

6 Conclusion

We have presented two different algorithms for finding related pages in the WWW. They significantly outperform Netscape's algorithm for finding related pages. The algorithms can be implemented efficiently and are suitable for use within a large scale web service providing a related pages feature.

Our two algorithms can be extended to handle more than one input URL. In this case, the algorithms would compute pages that are related to all input URLs. We are currently exploring these extensions.

Acknowledgements

This work has benefited greatly from discussions we have had with Krishna Bharat, Andrei Broder, Puneet Kumar, and Hannes Marais. We are also indebted to Puneet for his work on the Connectivity Server. As some of the earliest users of the server, Puneet answered our many questions and implemented many improvements to the server in response to our suggestions. We are also grateful

to Hannes Marais for developing WebL, a web scripting language [16]. Using WebL, we were able to quickly develop a prototype user interface for our related pages server. Krishna Bharat, Allan Heydon, and Hannes Marais provided useful feedback on earlier drafts of this paper. Finally, we would like to thank all the participants in our user study.

References

[1] *Proceedings of the Sixth International World Wide Web Conference*, Santa Clara, California, April 1997.

[2] *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, April 1998.

[3] G. O. Arocena, A. O. Mendelzon, and G. A. Mihaila. Applications of a web query language. In *Proceedings of the Sixth International World Wide Web Conference* [1], pages 587–595.

[4] K. Bharat, A. Z. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The connectivity server: Fast access to linkage information on the web. In *Proceedings of the Seventh International World Wide Web Conference* [2], pages 469–477.

[5] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in hyperlinked environments. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 104–111, 1998.

[6] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference* [2], pages 107–117.

[7] J. Carriere and R. Kazman. Webquery: Searching and visualizing the web through connectivity. In *Proceedings of the Sixth International World Wide Web Conference* [1], pages 701–711.

[8] S. Chakrabarti, B. Dom, D. Gibson, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in topic distillation. In *ACM–SIGIR'98 Post-Conference Workshop on Hypertext Information Retrieval for the Web*, 1998.

[9] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

[10] S. Chakrabarti, B. Dom, Raghavan P., S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the Seventh International World Wide Web Conference* [2], pages 65–74.

[11] Netscape Communications Corporation. 'What's Related FAQ' web page. `http://home.netscape.com/escapes/related/faq.html`.

[12] Netscape Communications Corporation. 'What's Related' web page. `http://home.netscape.com/escapes/related/`.

[13] E. Garfield. Citation analysis as a tool in journal evaluation. *Science*, 178, 1972.

[14] E. Garfield. *Citation Indexing*. ISI Press, Philadelphia, PA, 1979.

[15] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14, 1963.

[16] T. Kistler and H. Marais. Webl - a programming language for the web. In *Proceedings of the Seventh International World Wide Web Conference* [2], pages 259–270.

[17] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, January 1998.

[18] P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the web. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 96)*, pages 118–125, April 1996.

[19] J. Pitkow and P. Pirolli. Life, death, and lawfulness on the electronic frontier. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 97)*, pages 383–390, March 1997.

[20] Sheldon M. Ross. *Introductory Statistics*. McGraw Hill, 1996.

[21] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the 1995 Conference on Human Factors in Computing Systems (CHI'95)*, 1995.

[22] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *J. Amer. Soc. Info. Sci.*, 24, 1973.

[23] Ellen Spertus. Parasite: Mining structural information on the web. In *Proceedings of the Sixth International World Wide Web Conference* [1], pages 587–595.

[24] Loren Terveen and Will Hill. Evaluating emergent collaboration on the web. In *Proceedings of ACM CSCW'98 Conference on Computer-Supported Cooperative Work*, Social Filtering, Social Influences, pages 355–362, 1998.

[25] Loren Terveen and Will Hill. Finding and visualizing inter-site clan graphs. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI-98) : Making the Impossible Possible*, pages 448–455, New York, April18–23 1998. ACM Press.

Vitae

Monika R. Henzinger received her Ph.D. from Princeton University in 1993 under the supervision of Robert E. Tarjan. Afterwards, she was an assistant professor in Computer Science at Cornell University. She joined the Digital Systems Research Center (now Compaq Computer Corporation's Systems Research Center) in 1996. Her current research interests are information retrieval on the World Wide Web and algorithmic problems arising in this context.

Jeffrey Dean received his Ph.D. from the University of Washington in 1996, working on efficient implementation techniques for object-oriented languages under Professor Craig Chambers. He joined Compaq's Western Research Laboratory in 1996, where he worked on profiling techniques, performance monitoring hardware, compiler algorithms, and information retrieval. In February, 1999, he joined mySimon, Inc., where he is currently working on scalable comparison shopping systems for the World Wide Web. His current research interests include information retrieval and the development of scalable systems for the World Wide Web. He is two continents shy of his goal of playing basketball on every continent.