Analysis of Lexical Signatures for Improving Information Persistence on the World Wide Web

SEUNG-TAEK PARK
Yahoo! Research Labs
DAVID M. PENNOCK
Yahoo! Research Labs
C. LEE GILES
The Pennsylvania State University
and
ROBERT KROVETZ
Ask Jeeves

A *lexical signature* (LS) consisting of several key words from a Web document is often sufficient information for finding the document later, even if its URL has changed. We conduct a large-scale empirical study of nine methods for generating lexical signatures, including Phelps and Wilensky's original proposal (PW), seven of our own static variations, and one new dynamic method. We examine their performance on the Web over a 10-month period, and on a TREC data set, evaluating their ability to both (1) uniquely identify the original (possibly modified) document, and (2) locate other relevant documents if the original is lost. Lexical signatures chosen to minimize document frequency (DF) are good at unique identification but poor at finding relevant documents. PW works well on the relatively small TREC data set, but acts almost identically to DF on the Web, which contains billions of documents. Term-frequency-based lexical signatures (TF) are very easy to compute and often perform well, but are highly dependent on the ranking system of the search engine used. The term-frequency inverse-document-frequency- (TFIDF-) based method and hybrid methods (which combine DF with TF or TFIDF) seem to be the most promising candidates among static methods for generating effective lexical signatures. We propose a dynamic LS generator

This article is an extended version of the authors' previous article [Park et al. 2002].
S.-T. Park was partially supported by the School of Information Science and Technology, The Pennsylvania State University, University Park, PA and by NEC Laboratories America, Princeton, NJ.
This work was conducted while S.-T. Park was at The Pennsylvania State University, University Park, PA and D. M. Pennock and R. Krovetz were at NEC Laboratories America, Princeton, NJ.
Authors' addresses: S.-T. Park, Yahoo! Research Labs, 74 N. Pasadena Ave., 3rd floor, Pasadena, CA, 91103; email: parkst@overture.com; D. M. Pennock, Yahoo! Research Labs, 74 N. Pasadena Ave., 3rd floor, Pasadena, CA, 91103; email: david.pennock@overture.com; C. L. Giles, School of Information Sciences and Technology, Pennsylvania State University, University Park, PA 16802; email: giles@ist.psu.edu; R. Krovetz, Teoma Technologies, 1551 S. Washington Ave., Suite 400 Piscataway, NJ 08554; email: bkrovetz@askjeeves.com.

called *Test & Select* (TS) to mitigate LS conflict. TS outperforms all eight static methods in terms of both extracting the desired document and finding relevant information, over three different search engines. All LS methods show significant performance degradation as documents in the corpus are edited.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Algorithms, Experimentation, Measurement, Performance, Reliability, Verification

Additional Key Words and Phrases: Broken URLs, dead links, digital libraries, indexing, inverse document frequency, information retrieval, lexical signatures, robust hyperlinks, search engines, term frequency, TREC, World Wide Web

1. INTRODUCTION

The World Wide Web is a dynamic information resource: Web documents and hyperlinks are constantly being added, modified, moved, and deleted by independent entities around the world. Because of the Web's scale, dynamics, distributed control, and lack of facilities for maintaining persistence of information, finding desired information remains a challenging problem. General-purpose search engines only cover a limited portion of the Web, and most take several months to update new information [Bharat and Broder 1998; Lawrence and Giles 1998b, 1999]. One solution is to build a greater variety of special-purpose search engines that can react more quickly to changes within their particular domain. For example, Citeseer [Giles et al. 1998; Lawrence et al. 1999] was developed to collect and maintain a searchable index of computer science research articles. However, these efforts require considerable startup and maintenance costs, and so may not be feasible for every domain.

Several initiatives have addressed the problem of broken links by proposing mechanisms for assigning location-independent names to documents in addition to URLs [Arms et al. 1997; Ingham et al. 1995, 1996; Shafer et al. 1996; Sollins and Masinter 1994]. None of these approaches have been widely adopted because they require users either to acquire new software or to explicitly maintain the validity of name dereferencing. We review these other related works [Aimar et al. 1995; Andrews et al. 1995; Fielding 1994; Goldberg and Yianilos 1998; Oberholzer and Wilde 2002] in the next section.

Phelps and Wilensky [2000a, 2000b] proposed a less burdensome solution: compute a *lexical signature* (LS) for each document. This is a string of about five key identifying words in the document. If the document cannot be found by URL, then it can often be located by feeding its signature words into a search engine. Phelps and Wilensky proposed that LS words be chosen by maximizing a modified term-frequency inverse-document-frequency (TFIDF) measure, capping term frequency (TF) at five, among other modifications.

We find that, because Phelps and Wilensky's method (PW) caps TF at five, it places too much emphasis on document rarity and, in regard to huge document collections like the Web, acts almost identically to the method that chooses lexical signatures by minimizing document frequency (DF). Both DF and PW are good at uniquely identifying a document when it exists and is indexed by the search engine, but neither is good at finding relevant documents when the

target document is not in the search engine's database. Moreover, we believe that unique identification is often unnecessary: as long as the search engine returns the desired document as the first-ranked document (even among many documents), then the LS is effective.

In this article, we study the relative efficacy of nine different methods for generating LSs, including PW. We conducted experiments on actual Web documents in November 2001 and again in September 2002, where search engine coverage and ranking algorithms were limited, and on a TREC data set, where search coverage was complete. PW performs well on the latter data set, but more like DF on the web. LSs based on maximizing term frequency (TF) are easy to compute and maintain, since they do not depend on measuring statistics across the database. TF often performs well, but depends to a large extent on how the search engine ranks documents. TFIDF-based methods and hybrid methods that use one or two minimum-DF words along with maximum-TF or maximum-TFIDF words perform well both on real Web data and on the idealized TREC data, in terms of both finding the desired document and finding alternative relevant documents. We also propose a method called *Test and Select* (TS) that chooses a LS dynamically. TS attempts to avoid LS conflict at the time a new LS is generated. We find that TS dominates all eight static approaches both in terms of extracting the desired document and finding alternative relevant information, over three different search engines. All LS methods experience significant performance degradation when documents are modified over time. To ensure reasonable performance, periodic updating of LSs may be required.

2. RELATED WORK

The Web's dynamic nature is a key factor making information retrieval in general, and the maintenance of persistent references in particular, difficult. Douglis et al. [1997] collected traces (including the full contents of request and response messages) from two large corporate networks over a 17-day and 2-day period, respectively.[1] The authors studied the relationship between the rate of change of Web resources and characteristics like access rate, age, content type, resource size, and their top level domains. The authors reported that the rate of change of Web resources depends primarily on two factors: the type of content and the domain. For example, images change rarely, while application data changes nearly every time it is accessed. About half of text/HTML documents remain unchanged. Also, more heavily accessed resources change more frequently and 18% of the full-body responses are duplicated resources. Note that their study considered dynamics over relatively short time periods: several days rather than months or years. Cho and Garcia-Molina [2000] provided observations on changes over a 4-month period. The authors reported that more than 20% of 720,000 documents change daily and 70% of documents change to some degree within 4 months. More than 70% of documents remained visible (regardless of changes of content) for more than 1 month. The authors

[1]One trace set was obtained at the gateway between AT&T Labs and the external Internet over 17 days; the other set was obtained at the primary Internet proxy of Digital Equipment Corporation over 2 days.

reported that documents in the commercial (.com) domain are updated more frequently than average: 40% of commercial documents change everyday, and half of commercial documents change within 11 days, compared to 50 days for half of all documents to change. The average lifespan of commercial documents is shorter than for documents in the educational (.edu) or government (.gov) domains. Brewington and Cybenko [2000] reported that the lifetime and the expected lifetime of a Web document can be modeled by an exponential and a Weibull distribution, respectively.

Because the Web is dynamic and decentralized, documents can disappear or move without notice, breaking any hyperlinks pointing to them. Broken links can persist nearly indefinitely, frustrating users. Pitkow [1998a, 1998b] reported that around 5% to 8% of requested hyperlinks on the Web are broken (i.e., return an error); Lawrence et al. [2001] found that many URL citations in research articles become invalid as early as a year or two after publication.

Several solutions have been proposed to address the *broken link* problem. One type of solution aims at making the authors' task of maintaining hyperlinks easier. The standard HTTP protocol [Berners-Lee et al. 1996; Fielding et al. 1999] itself provides a way of relocating moved documents by storing a forwarding pointer to the new location. The method requires manual updating of a sever configuration table and it does not help the case when a document is deleted or moved to a different server. Creech [1996] proposed a link management technique to help authors of Web documents ensure the consistency of their content by using a change log table (CLT) and a Web-walk (WW). A CLT is a table containing a log of operations (i.e., editing, moving, or removing) performed. A WW checks each document which is pointed to by the documents in the CLT and sends change reports to appropriate authors. This approach lightens, but does not eliminate, the burden on authors. Netscape LiveWire [Gulesian 1996] and MOMspider [Fielding 1994] aid hyperlink management for cross-site (external) links. LinkGuard[2] tries to build a complete map of all links on the Web. If broken links are found, it notifies document owners who subscribe to its service, providing some advice on how to fix them. Most of above approaches offer a degree of automation, for example, notification services, though are not fully automated, requiring some manual intervention on the part of the authors.

Uniform Resource Names (URNs) [Sollins and Masinter 1994] offer another approach to the broken link problem. A URN is a invariable, location-independent name that does not change when a document is moved. A few systems have been developed for the implementation of URNs: Persistent Uniform Resource Locators (PURL) [Shafer et al. 1996] and the Handle [Arms et al. 1997] system are examples. These systems use resolve servers to translate URNs to URLs, which are updated whenever a document is moved. Handle requires additional software to support name resolution, and PURL requires prefixing the address of the PURL resolver into hyperlinks (e.g., http://purl.org/something). WebLinker [Aimar et al. 1995] uses a related concept called a *Local Resource Name* (LRN) that can be updated once to reflect a

[2]LinkGuard, http://www.ap.dell.com/ap/ap/en/gen/casestudies/casestudy_linkguard.htm.

new physical location of a document; all pages referring to the LRN need not be changed. In part because URN-related approaches require additional hardware or software, either for the client or the server, or both, they have not been widely adopted.

Ingham et al. [1995, 1996] proposed an object-oriented network existing in parallel with the Web, which maintains referential integrity and performs garbage collection. The Hyper-G network information system [Andrews et al. 1995] and Xanadu[3] have been proposed as an alternatives to the Web. Both approaches provide a bidirectional link structure along with mechanisms to explicitly enforce link consistency. Neither system is used in practice to any significant extent. More advanced authoring languages (e.g., DHTML or XML) have been widely discussed as replacements for current HTML perhaps with better persistence properties [Mace et al. 1998; Oberholzer and Wilde 2002]. In our opinion, new bidirectional hypertext systems are unlikely to supplant the existing (directional) Web—in fact, the absence of maintenance and garbage-collection requirements arguably have encouraged the explosive growth of the Web as we know it as compared to more structured hypertext mechanisms like Xanadu.

Archiving offers an alternative solution to the broken link problem. The mission of the Internet Archive[4] is to continually store snapshots of the Web taken at various intervals, and make available a search infrastructure for finding old versions of documents. Archival intermemory uses distribution and redundancy to ensure data survivability [Goldberg and Yianilos 1998]. Lots Of Copies Keep Stuff Safe (LOCKSS) provides a tool where multiple libraries each collect, store, preserve, and archive important document independently [Reich and Rosenthal 2001]. Consistency and completeness of LOCKSS caches are achieved through a robust and secure peer-to-peer polling and reputation system. Participating libraries cooperate to detect and repair preservation failures. Because the Web is distributed and dynamic, no archive can guarantee a complete and accurate snapshot of the Web's state at any given moment in time, let alone fully track its state over time. For some perspective, today's most advanced search engines required on the order of a month to update their Web-wide indexes and databases.

The most common way to find documents on the Web (including documents missing due to broken links) is to use a search engine. Google,[5] AltaVista,[6] AlltheWeb,[7] Northern Light,[8] Ask Jeeves,[9] MSN,[10] and Yahoo![11] are among the popular choices. Since the indexes of individual search engines vary in their coverage of the Web [Lawrence and Giles 1998b, 1999], meta-search engines

[3]Xanadu PERMAPUB and PERMASTORE, http://www.xanadu.net/.
[4]Internet Archive, http://www.archive.org/.
[5]Google, http://www.google.com.
[6]AltaVista, http://www.altavista.com/.
[7]AlltheWeb, http://www.alltheweb.com/.
[8]Northern Light, http://www.northernlight.com/.
[9]Ask Jeeves, http://www.ask.com/.
[10]MSN, http://www.msn.com/.
[11]Yahoo!, http://www.yahoo.com/.

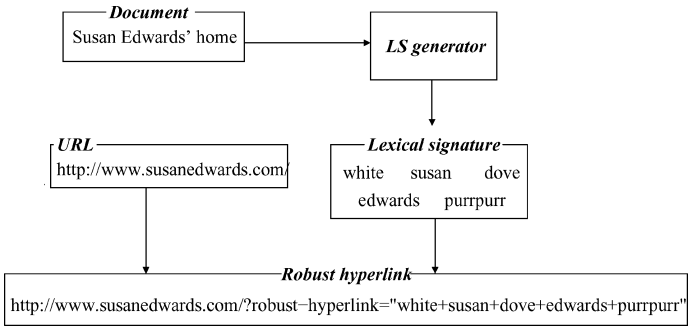


Fig. 1. Generating a robust hyperlink.

like Metacrawler[12] and Inquirus [Lawrence and Giles 1998a] may improve results, though sometimes the individual search engines block queries from meta-search portals. VeriSign, the company charged with maintaining .com and .net domain names, recently began a program where HTTP requests to invalid domains would return algorithmic and paid search listings, rather than 404 errors. Amid controversy over the program, VeriSign has since suspended the practice.[13] Relying on search engines to locate lost documents requires the searcher to know enough about the target document and the search engine's retrieval algorithm to pose an appropriate query that returns the correct document. Lexical signatures are in a sense a method of tagging documents with appropriate search queries useful for finding the document later.

Phelps and Wilensky [2000a, 2000b] first proposed the use of lexical signatures to enable robust hyperlinks. A *lexical signature* (LS), is a string of about five key identifying words in the document. If the document cannot be found by URL, then it can often be located by feeding its signature words into a search engine. Phelps and Wilensky proposed that the LS words be chosen by maximizing a modified term-frequency inverse-document-frequency (TFIDF) measure, capping term frequency (TF) at five, among other modifications. They also proposed methods for appending lexical signatures onto a document's URL (using an HTTP GET variable) and instrumenting browsers to automatically perform content-based dereferencing (i.e., query a search engine and process the results) when standard URL dereferencing fails. Even without robust hyperlink compatible browsers, users can easily submit the LS to a search engine by copying the LS from the document's URL. The authors have also studied the compatibility of various types of robust hyperlinks with current browsers. Phelps and Wilensky reported that, in most cases, a search engine returns the desired document and only that document. If the search engine returns no documents (because the desired document no longer exists or is not indexed), then the authors suggested removing one or more words from the LS and using this reduced signature to search for substitute documents. Thus, a secondary purpose for LSs is to discover relevant or similar documents when the desired document is truly lost. Figure 1 shows how a robust hyperlink is generated;

[12]MetaCrawler, http://www.metacrawler.com.
[13]VeriSign to Shut Down Site Finder, http://www.wired.com/news/business/0,1367,60682,00.html.

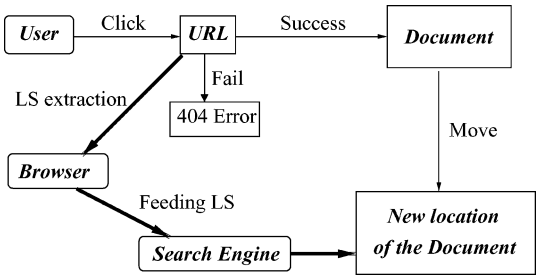


Fig. 2. Mechanism of robust hyperlinks. When first address-based dereferencing fails, the browser automatically performs content-based dereferencing. The thicker arrows show how content-based dereferencing works.

Figure 2 shows how robust hyperlinks work when first address-based dereferencing fails. A fully automated version of lexical signatures would involve additional software either in the user's browser or on the server side, in order to reroute HTTP requests to search engines appropriately when URL dereferencing fails. However, the level of required software support is fairly minimal, and even without it users can simply copy the LS from the URL and feed it manually into a search engine.

3. TERMINOLOGY

3.1 Lexical Signatures

Phelps and Wilensky's [2000a, 2000b] main motivation was to associate LSs with documents, so that when the LS is fed to a search engine, the desired document—and only that document—is returned. Then, when URLs change and links to documents become invalid, new locations for documents can be easily found via search engines. To achieve this goal, Phelps and Wilensky argued that LSs should have following characteristics:

(1) LSs should extract the desired document and only that document.
(2) LSs should be robust enough to find documents that have been slightly modified.
(3) New LSs should have minimal overlap with existing LSs.
(4) LSs should have minimal search engine dependency.

We prefer a slightly weaker notion of unique identification: as long as the desired document is the top-ranked document returned by the search engine, we are satisfied. We also pay closer attention to the other potential benefit of LSs: to help the user find relevant documents when the desired document is truly lost. We therefore modify the first desired characteristic of lexical signatures as follows:

(1a) LSs should easily extract the desired document. When a search engine returns more than one document, the desired document should be the top-ranked document.

(1b) LSs should be useful enough to find relevant information when the precise documents being searched for are lost.

3.2 What Is a Term?

LSs are composed of a small number of *terms*. Phelps and Wilensky [2000a, 2000b] used individual words as terms, where words are case-insensitive, are contained in the context of the document (not in meta-tags), contain at least four letters, and do not include any numbers. In our experiments, these rules of thumb for defining terms proved to be fairly effective. Number queries caused many problems with document retrieval: for example, if the query "2,000" is given to search engines, some return documents that contain only "2000," "2;000," "2:000," or "2,000," but not "2000." Also, many words that have less than four letters are stop words (e.g., "the," "of," or "in"). In our experiments, filtering out these short words for the most part slightly improved the efficacy of the LSs.

3.3 Basic and Hybrid LSs

In our experiments, we explored LSs containing five terms.[14] We generated eight kinds of LSs for each document: four *basic* LSs and four *hybrid* LSs. Basic LSs are generated using a single metric. For example, TF-based signatures are generated based on the term frequency values of words in the given document. Hybrid signatures combine terms generated from two different basic methods. For example, TF3DF2 uses three TF-based words and two DF-based words. A detailed explanation of the basic LS methods is as follows:

(1) *TF*: Select terms in decreasing term frequency (TF) order. If there is a tie, then pick words based on increasing document frequency (DF). If there is another tie, randomly select the words.
(2) *DF*: Select words in increasing DF order. If there is a tie, then pick words based on decreasing TF order. If there is another tie, randomly select the words.
(3) *TFIDF*: Select words in decreasing term-frequency inverse-document-frequency (TFIDF) order. If there is a tie, then pick words based on increasing DF order. If there is another tie, randomly select the words.
(4) *PW*: Select words based on Phelps and Wilensky's [2000a, 2000b] method (i.e., decreasing TFIDF order where the TF term is capped at five). If there is a tie, then pick words based on increasing DF order. If there is another tie, randomly select the words.

[14]We decided on five terms, since that was the number initially proposed by Phelps and Wilensky [2000a, 2000b]. There is no obvious optimal number of terms. The more terms, generally the better the LS will perform, although more terms will make extended URLs very long, and may reduce the utility of the scheme for users. Also, search engines often restrict the length of queries: for example *Google* restricts queries to 10 words. Phelps and Wilensky's idea was to choose the smallest number of terms that are able to uniquely identify any document. Exploring other numbers of terms is left for future work.

Since we are also interested in the ability of LSs to extract relevant documents when they fail to find missing documents, we find it useful to divide an LS into two parts. The first part is useful for finding relevant documents and the second for uniquely identifying the desired document. If the desired document is not extracted because a search engine has not indexed it or the document is lost or gone, the second part of the LS is removed and we attempt to find relevant documents using only the first part. Since we want to use the first part of LSs to find relevant documents, we filter out in that part all words that have document frequency 1. This leads us to propose the following hybrid LS methods:

(1) *TF3DF2* : Select two words in increasing DF order. If there is a tie, then pick words based on decreasing TF order. If there is another tie, randomly select the words. Then filter out all words which have DF value 1. Select three words maximizing TF. If there is a tie, it is resolved the same way as with TF method. For TFIDF3DF2, and TFIDF4DF1, ties are resolved the same way as with TFIDF.
(2) *TF4DF1* : Select one word based on increasing DF order first. Then filter out all words which have DF value 1. Select four words maximizing TF.
(3) *TFIDF3DF2* : Select two words based on increasing DF order first. Then filter out all words which have DF value 1. Select three words maximizing TFIDF.
(4) *TFIDF4DF1* : Select one word based on increasing DF order first. Then filter out all words which have DF value 1. Select four words maximizing TFIDF.

3.4 Similarity Metric: Cosine Measure

To measure the similarity between returned documents and the desired document, we use the cosine measure within the vector-space model [Witten et al. 1999]. For n unique words in our corpus, each document can be represented as an n-dimensional vector. For example document A can be represented as $A = (a_1, .., a_n)$, where if the ith word in the corpus appears in document A, then a_i is either the word's TF value, or the word's TFIDF value. Otherwise, $a_i = 0$. There is no easy way to extract the exact DF value of each word for all documents on the World Wide Web. Also, even if we could extract the DF value of each word for documents indexed by a search engine, the heavy search query burden for a search engine would not be tolerated.[15] Thus, we use TF values with the cosine measure on actual Web documents (which gives a slight bias to TF-based lexical signatures in our experiments), and use TFIDF values on TREC data (which gives a slight bias to TFIDF-based methods). Using the vector space model, the cosine similarity measure between documents A and B is

$$\cos\theta = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2}}.$$

[15]We discuss some technical difficulties for it in Section 7.2, "Limitations."

In practice, documents on the Web are regularly updated or modified. We would like to consider extremely similar documents that result from minor modifications to be the same document. Therefore, if the cosine value of two documents is greater than 0.9, we consider them to be the same document.

4. EMPIRICAL RESULTS FROM WEB DATA

Phelps and Wilensky [2000a, 2000b] reported qualitative results that their LSs extracted in most cases one or two documents, one of which was the desired document. We conducted extensive quantitative experiments comparing the ability of all eight LS methods to retrieve the desired document or, failing that, to identify an appropriate alternative document.

4.1 Web Data Set

For this experiment, we extracted the first 1500 URLs from precrawled data[16] (containing 1.5 million URLs) and downloaded each corresponding document. Several Web documents do not have any word or have only a few words in their content, for example, some documents only contain Java scripts or flash links. We excluded all documents and corresponding URLs that contained fewer than 50 words. The URLs that could not be downloaded because of server failures or corresponding document removals were also excluded. After removing stop words from word tokens, we again removed any URLs and corresponding documents that contained fewer than five unique words. The final test data set consisted of 980 Web documents.

4.2 Experimental Method

Since there is no obvious way to get the document frequency (DF) of each word for the entire Web, we used a search engine to generate a DF list for all words in our document set. In this experiment, we assumed that the DF value of each word from the search engine Google[17] is proportional to the actual DF value for the entire Web. Based on document frequency list and term frequency list of each document, we examine eight different LSs per document.

After generating LSs for all methods, we used them as queries for three search engines: YahooGoogle,[18] which uses *Google's* searching algorithm, AltaVista,[19] and MSN.[20] Unlike MSN and YahooGoogle, AltaVista returns all documents that contain any words in the LS. To solve this problem, we used the advanced search option for AltaVista (using this option, AltaVista only returns

[16]The data used in this experiment was created by crawling all of dmoz.org (the Open Directory) and all documents one or two forward links from DMOZ, then retrieving all links external to the dmoz.org domain. From these URLS any href tag was extracted eliminating any files with a file name extension that indicated that it was not HTML (e.g., pdf, ps, doc, etc). Thus the data set consists of all sites listed in DMOZ plus all sites up to two forward links away from DMOZ. We thank Gary Flake and NEC Laboratories America for access to this data.
[17]http://www.google.com/.
[18]http://google.yahoo.com/.
[19]http://www.altavista.com/.
[20]http://www.msn.com/.

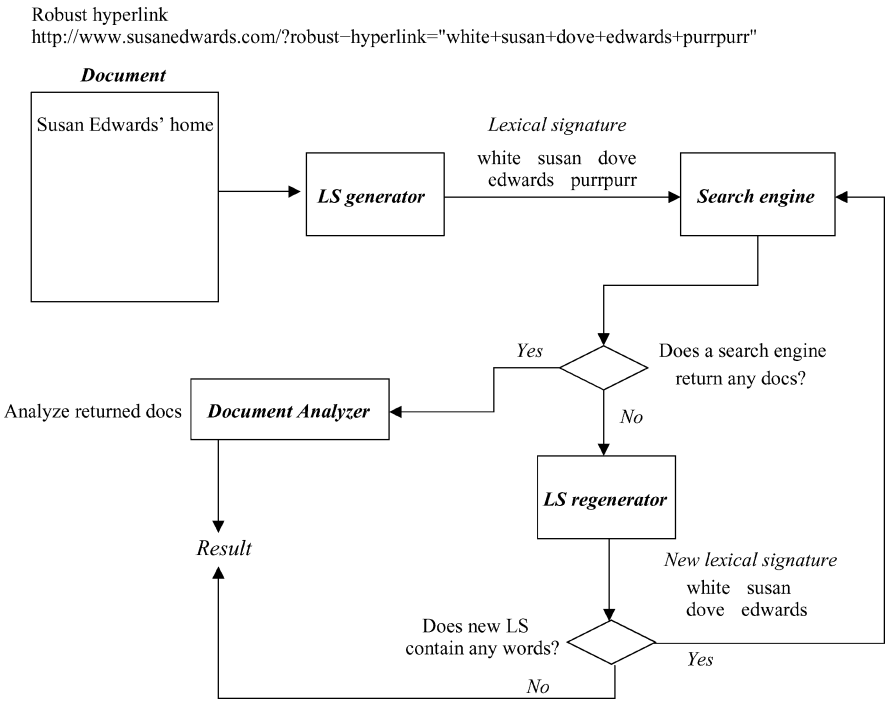


Fig. 3. Layout of experimental procedure.

documents that contain all words in query). If a search engine did not return any documents, we removed the word with the lowest DF and requeried the search engine. This procedure was continued until the search engine returned documents or all of words in the given lexical signature were removed. After the search engine returned the list of documents, those documents were downloaded and the similarity between returned documents and the target document was calculated using the cosine measure. If the search engine returned more than ten documents, we analyzed only the top 10 ranked documents and ignored the rest. The experimental procedure is shown in Figure 3. For all queries to YahooGoogle, all but two queries to MSN, and all but four queries to Altavista, at least some documents were returned. This experiment was conducted twice: first between October 18 and November 8, 2001, and second between September 3 and September 9, 2002, to investigate performance variations over time. In the second experiment, LSs generated in the first (October 2001) experiment were used.

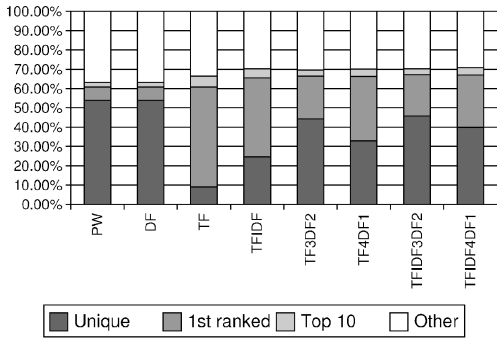
4.3 Retrieval Performance

Our concern is not only with whether or not the desired document is returned but with its location in a possible list of returned documents. We define retrieval performance of LSs as the percentage of times the desired document is returned based on how and when the document is returned. Since we already have a signature for the desired document, our performance measure is similar to recall.

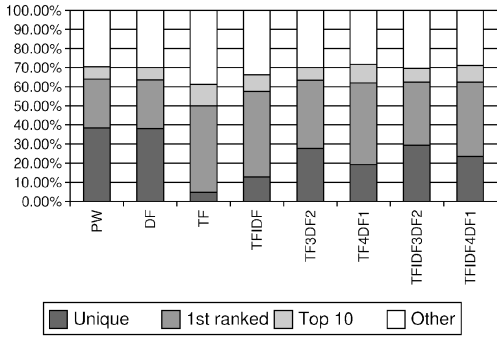
We define the following disjoint exhaustive classes. *Unique* represents the percentage of lexical signatures that successfully extract and return the single desired document. (This is the class discussed by Phelps and Wilensky [2000a, 2000b].) *1st ranked* represents the percentage of LSs that extract a list of documents with the desired document first ranked. *Top 10* is the percentage of LSs that successfully return a list with the desired document in the top 10, though not first ranked. *Other* represents the percentage of LSs that fail to return the desired document in the top 10 documents displayed. Because these classes are disjoint and exhaustive, the above classes represent 100% of all cases. Figure 4 shows the retrieval performance of each lexical signatures in extracting the desired documents for three different search engines averaged over the 980 unique LSs.

Figure 4 shows that considering only the *unique* extraction property of LSs is not the only important factor in extracting the desired document. Note that DF and PW are most effective for the *Unique* property. However, if we focus on just retrieving the desired document, that is, the case where *Unique*, *1st ranked*, and *Top 10* are combined, then hybrid methods are the most consistent and effective methods across all three search engines. Using this definition of retrieval performance, PW and DF methods performed worse than others for YahooGoogle. Phelps and Wilensky [2000a, 2000b] argued that the original TFIDF did not sufficiently emphasize the contribution of rarity; that is, if LSs are chosen to minimize DF, it would be more helpful to filter out other documents and extract only the single desired document. However, PW limits the words in the LS to a TF of 5. We argue that this overemphasizes rarity. As the number of documents on the Web increases, the PW and DF methods become similar, as is evident in Figure 4.

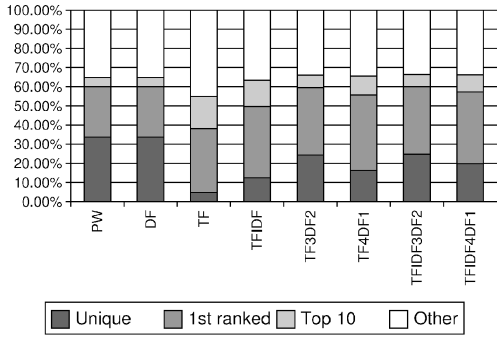
We calculated the statistical significance of the differences between PW and the other methods, using the randomization test [Fisher 1966; Noreen 1989]. We describe the test for comparing PW with TFIDF; the other tests are analogous. Let N_{pw} and N_{tfidf} be the number of PWs and TFIDFs, respectively, that extracted the desired documents successfully (*Unique* + *1st ranked* + *Top 10*). Recall that the number of documents tested was 980, so the total number of results for both PW and TFIDF was $980 + 980 = 1960$. We collected all 1960 results, shuffled them at random, then randomly partitioned the results into two groups of size 980 each. Let N_{1st} and N_{2nd} be the number of results in the two subgroups, respectively, that extracted the desired documents successfully. Now we defined $\delta_{ls} = |N_{pw} - N_{tfidf}|$ and $\delta_{random} = |N_{1st} - N_{2nd}|$. We repeated this randomization procedure 10,000 times, computing the percentage of cases where $\delta_{ls} > \delta_{random}$. Let q be the frequency with which $\delta_{ls} > \delta_{random}$; that is, $q = n_{greater}/10,000$ where $n_{greater}$ is the number of times that δ_{ls} is greater than δ_{random} out of the 10,000 trials. The null hypothesis was that PW results and TFIDF results arise from the same distribution, in which case δ_{ls} would be on average equal to δ_{random}. To the extent that δ_{ls} was consistently greater than δ_{random} across the 10,000 trials, we have evidence to reject the null hypothesis and conclude that the results of PW and TFIDF were statistically distinguishable. If q was 0.95 or higher, we rejected the null hypothesis with a p-value less than 0.05 and conferred statistical significance on the results. Note that the



(a) YahooGoogle, 2001



(b) MSN, 2001



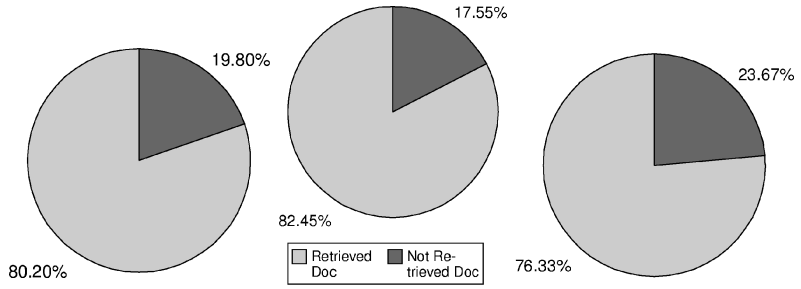
(c) AltaVista, 2001

Fig. 4. Retrieval performance of LS methods in 2001.

p-value here was defined as $1-q$. We consider a p-value < 0.05 to be significant. In other words, if the p-value < 0.05, then the difference between the results for PW and TFIDF is not likely attributable to random fluctuations. Table I shows the number of documents extracted by each LS and the statistical significance (p-value) of the observed difference between PW and the other LSs. Since differences among hybrid methods were relatively small (the p-value between the best and worst hybrid method was generally higher than 0.4), we show only the

Table I. Statistical Significance Between PW and Other Methods for Retrieving the Desired Document in 2001 (Each entry records the raw number of successful results and, in parentheses, the p-value denoting the degree statistical significance. Bold numbers indicate an improvement over PW. Bold statistical significance values indicate that the p-value $< .05$. SE = search engine.)

SE	PW	TF	TFIDF	TFIDF4DF1
YG	618	**651** (.1086)	**688** (**.0007**)	**694** (**.0002**)
MSN	690	599 **(0)**	649 **(.0493)**	**696** (.7256)
AV	635	538 **(0)**	621 (.4786)	**648** (.5181)



(a) YahooGoogle (b) MSN (c) AltaVista

Fig. 5. Coverage of each search engine in 2001.

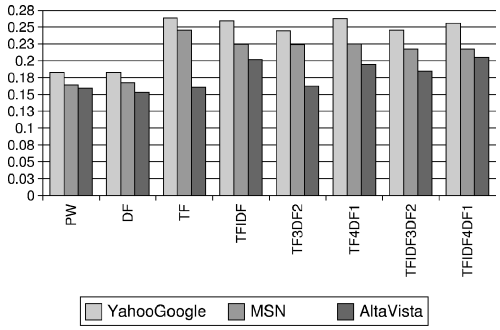
results of TFIDF4DF1 among hybrid methods. (Note: in all tables, YahooGoogle is abbreviated as YG, MSN as MSN, and AltaVista as AV.)

Hybrid methods performed comparably to PW for retrieving the desired documents. TF showed better retrieval performance than PW with YahooGoogle, but worse with other search engines. In general, the performance of TF depended strongly on the ranking algorithm of the search engine. TFIDF showed some performance variations over search engines, but the differences were relatively small compared to TF.

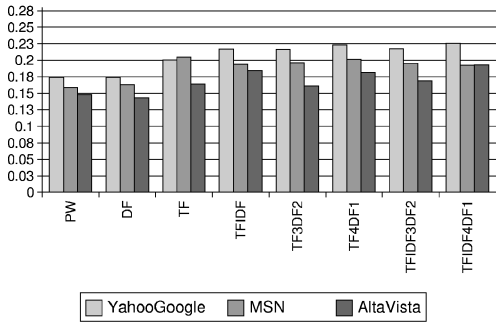
4.4 Finding Relevant Documents

Suppose the desired document cannot be extracted; can the LS find a related one? If so, which method works best? Figure 5 shows the percentage of all 980 documents not found in the top 10 results returned by each search engine (defined as the *Not Retrieved Documents* class), using all LS methods; specifically 194, 178, and 236 documents could not be retrieved from YahooGoogle, MSN, and Altavista, respectively. There could be many reasons for this. These documents might not yet be indexed by the search engines; they might have been moved; they might have been deleted; they might have been modified or updated; or they might consist of very few unique words. When a document cannot be found, we'd like the LS to extract relevant documents.

To assess the ability of LSs to retrieve related documents, we analyzed the cosine similarity between the documents retrieved in the *Not Retrieved Document* class to the original document. Figure 6(a) gives the average cosine values



(a) Average cosine value of the first ranked document, for *Not Retrieved Documents*



(b) Average cosine value of top 10 documents, for *Not Retrieved Documents*

Fig. 6. Ability of LS methods to extract similar documents in 2001.

of the first ranked documents and Figure 6(b) shows average cosine values of the top 10 documents.

We computed the statistical significance of these results using the same randomization test procedure described in Section 4.3. The results are shown in Table II. The differences among hybrid methods were small, except for AltaVista. For AltaVista, the cosine of documents returned using TF was low, affecting TF-based hybrid methods like TF3DF2 and TF4DF1.

TF showed the best average cosine for the first ranked documents with YahooGoogle and MSN and top 10 documents with MSN. Hybrid methods showed the best average cosine for top 1 and top 10 documents with AltaVista, and for top 10 documents with YahooGoogle. TFIDF and hybrid methods performed much better than PW in finding relevant documents, with high statistical confidence.

4.5 Robustness of LSs Under Document Modification; 980 URLs Revisited

To investigate the robustness of LSs over time, we repeated the same experiments 10 months later, in September 2002. We used the LSs generated in 2001. Figure 7 shows retrieval performance degradation after 10 months. The retrieval performance of all LS methods significantly decreased. Still, the hybrid

Table II. Statistical Significance Between PW and Other Methods for Finding Relevant Documents in 2001 (Each entry records the raw cosine value and, in parentheses, the p-value denoting the degree statistical significance. Bold numbers indicate an improvement over PW. Bold statistical significance values indicate that the p-value $< .05$. SE = search engine.)

Type	SE	PW	TF	TFIDF	TFIDF4DF1
1st	YG	.183	**.264 (.0004)**	**.260 (.0008)**	**.256 (.0012)**
	MSN	.164	**.245 (.0008)**	**.224 (.0106)**	**.217 (.0205)**
	AV	.159	**.161** (.7993)	**.202 (.0103)**	**.205 (.0076)**
Avg	YG	.174	**.201** (.1274)	**.217 (.0194)**	**.225 (.0097)**
	MSN	.158	**.205 (.0123)**	**.194** (.0531)	**.192** (.0666)
	AV	.148	**.164** (.1628)	**.184 (.0067)**	**.193 (.0017)**

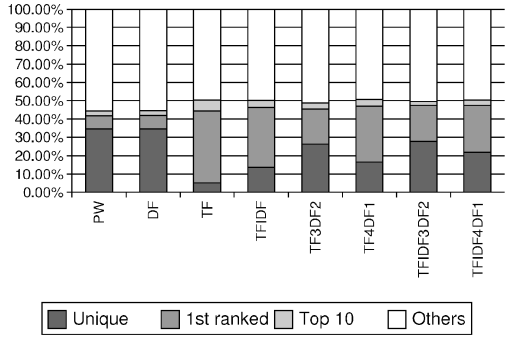
methods performed at least as good as TF and TFIDF and better than PW and DF. Also, TF and TFIDF performed better than PW over all search engines in 2002. Tests of statistical significance are shown in Table III.

As in 2001, we remeasured the average cosine of documents in the *Not Retrieved Documents* class. In 2002, 375, 414, and 434 documents were not extracted by YahooGoogle, MSN, and AltaVista, respectively. As before, the hybrid methods, TF, and TFIDF outperformed PW and DF for retrieving relevant documents. One interesting change from 2001 to 2002 was the marked performance increase for AltaVista in finding relevant documents. We saw relatively small performance variations over the three search engines in 2002. We observe that the cosine relevance over all three search engines improved in 2002 compared to 2001. Table IV gives statistical significance figures for the relevance tests in 2002, comparing PW against the other methods.

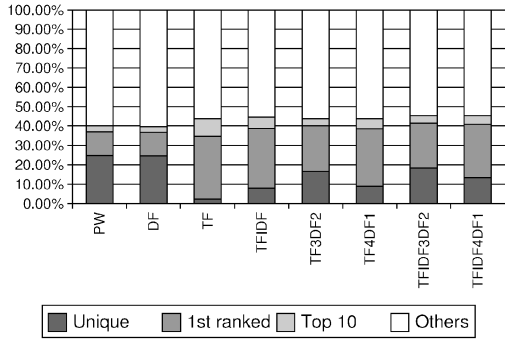
We examined in more detail the factors affecting performance degradation. We redownloaded all 980 URLs, analyzed them using the cosine measure, and classified them into five classes: *Gone or Moved*, *Same*, *Slightly Modified*, *Modified*, and *Heavily Modified*. The classes were defined to get a better grasp of how many URLs were "broken" and, of those that remained, how significantly documents were modified, after 10 months on the Web.

If in 2002 a document could not be retrieved using its original URL (e.g., 404 error), we classified the document as *Gone or Moved*. If a particular URL was broken, it was very hard to tell whether it was actually gone, or had moved, or had changed URLs. For example, if search engines did not index a moved document, then we could not find the new location. Even if the search engine returned a similar document at another location, it was not obvious how to determine whether it was the same document edited and moved, or a new but similar document. Because of these reasons, *Gone* and *Moved* were combined in a single class.

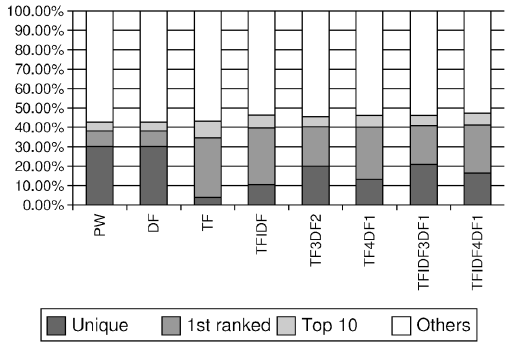
For the valid URLs remaining, we further subdivided them according to how drastically they had been modified. If the cosine angle between the old document (2001) and the new document (2002) was 1, then the document was not changed (*Same* class). Otherwise, the document was considered to be modified. We found that among 980 URLs, 124 documents (12.65%) were gone or moved, and 660 documents (67.35%) were modified. Only 196 documents (20%)



(a) YahooGoogle, 2002



(b) MSN, 2002



(c) AltaVista, 2002

Fig. 7. Retrieval performance of LS methods in 2002.

remained the same. Note that our findings are similar those of earlier studies [Douglis et al. 1997; Brewington and Cybenko 2000; Cho and Garcia-Molina 2000]. Since Web documents are frequently modified, robustness against modification is one of the key characteristics to consider when choosing an LS. To study this issue more closely, we divided modified documents into three classes: *Slightly Modified*, when the cosine was greater than or equal to 0.8, *Modified*, when the cosine was between 0.5 and 0.8, and *Heavily Modified*, when the

Table III. Statistical Significance Between PW and Other Methods for Retrieving the Desired Document in 2002 (Each entry records the raw number of successful results and, in parentheses, the p-value denoting the degree statistical significance. Bold numbers indicate an improvement over PW. Bold statistical significance values indicate that the p-value $< .05$. SE = search engine.)

SE	PW	TF	TFIDF	TFIDF4DF1
YG	435	**494 (.0073)**	**491 (.0107)**	**495 (.0067)**
MSN	393	**429** (.0928)	**437 (.0396)**	**455 (.0161)**
AV	419	**424** (.7861)	**453** (.1092)	**463 (.0415)**

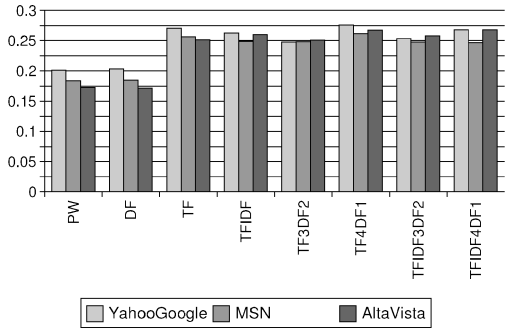
Table IV. Statistical Significance Between PW and Other Methods for Finding Relevant Document in 2002 (Each entry records the raw cosine value and, in parentheses, the p-value denoting the degree statistical significance. Bold numbers indicate an improvement over PW. Bold statistical significance indicate that the p-value $< .05$. SE = search engine.)

Type	SE	PW	TF	TFIDF	TFIDF4DF1
1st	YG	.201	**.271 (0)**	**.262 (0)**	**.268 (0)**
	MSN	.183	**.257 (0)**	**.249 (0)**	**.246 (0)**
	AV	.172	**.251 (0)**	**.260 (0)**	**.268 (0)**
Avg	YG	.183	**.214 (.01)**	**.219 (.0053)**	**.228 (.0009)**
	MSN	.170	**.224 (0)**	**.211 (.0004)**	**.220 (.0001)**
	AV	.167	**.231 (0)**	**.228 (0)**	**.241 (0)**

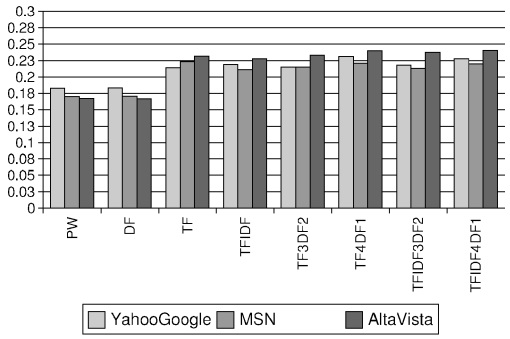
cosine was less than 0.5. Among modified documents, around 57% of modified documents (377 documents) were slightly modified and 26% (172 documents) were modified heavily. The analysis of all 980 documents is shown in Figure 9.

We measured the robustness of the LSs for each level of modification. To evaluate the returned documents, we used two methods: URL comparison and cosine measure. We considered that the first-ranked document was the target document if and only if the first URL was the same as target URL *or* the cosine between first-ranked document and target document was greater than 0.9. For the *Gone or Moved* class, all target URLs were by definition broken, so we used the cosine measure only. In Figures 10 and 11, we show the results for YahooGoogle; the results for the other search engines were similar. When documents were unchanged (*Same* class), retrieval performance of the LSs did not change much. This suggests that LSs are robust over time when documents are rarely modified or moved, as is generally the case with scientific publications, for example. The hybrid LS methods still showed generally better retrieval performance than the basic LS methods after 10 months. It is possible that a document with a high rank in 2001 could have a low rank in 2002, as the search engines' databases and ranking algorithms changed. However, this seemed to be rare over the test period: the *Same* class showed similar performances in both 2001 and 2002.

Figure 10 compares the performance of all LS methods for the *Same* class and the *Gone or Moved* class between 2001 and 2002. When documents were moved or lost, the retrieval performance of all LSs decreased dramatically. This result may be due to either of two reasons: (1) the document was moved but the search engines had not yet indexed the new location of the document, or



(a) Average cosine value of the first ranked document, for *Not Retrieved Documents*



(b) Average cosine value of top 10 documents, for *Not Retrieved Documents*

Fig. 8. Ability of LS methods to extract similar documents in 2002.

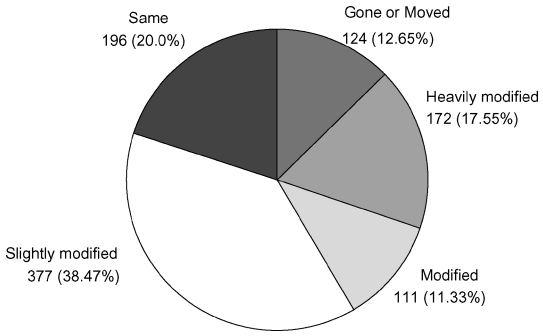
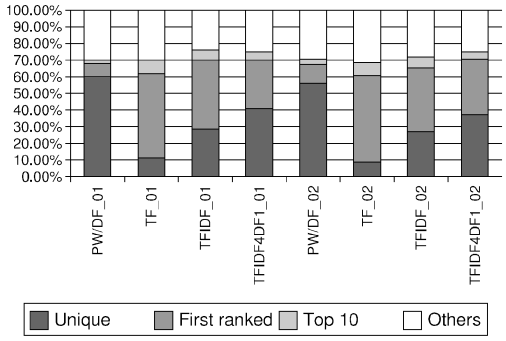


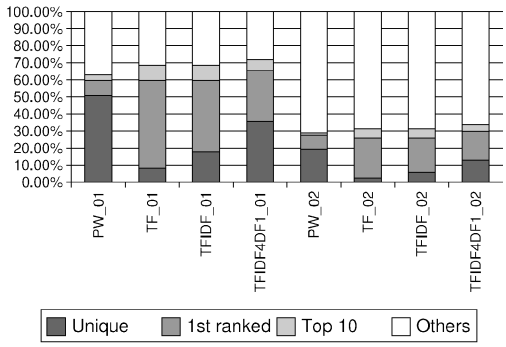
Fig. 9. Categorization of 980 revisited documents.

(2) the document was lost and duplicates of the document did not exist. In this case, the ability to find relevant documents becomes a more important factor in selecting an LS method. Hybrid LS methods still performed slightly better than basic LS methods in 2002.

How did LS methods behave when documents were modified? When documents were slightly modified (when the cosine angle between the old and new



(a) YahooGoogle, *Same*



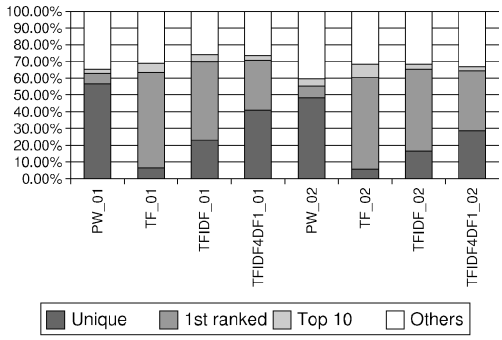
(b) YahooGoogle, *Gone or Moved*

Fig. 10. Retrieval performance of LSs in the *Same* and *Gone or Moved* classes.

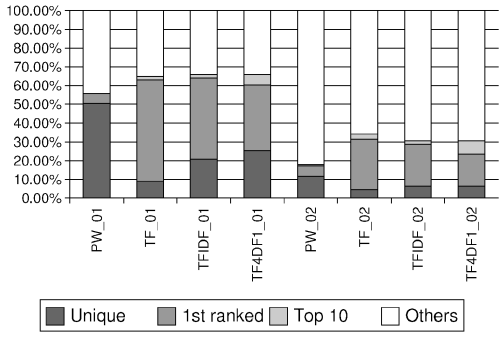
Table V. Statistical Significance Between PW and Other Methods for Retrieving the Desired Document, Separated into Classes According to Their Degree of Modification (Each entry records the raw number of successful results and, in parentheses, the p-value denoting the degree of statistical significance. Bold numbers indicate an improvement over PW. Bold statistical significance values indicate that the p-value $< .05$.)

Year	Class	Total	PW	TF	TFIDF	TFIDF4DF1
2002	SA	196	138	134 (.5903)	**141** (.346)	**147** (.2627)
	GOM	124	36	**39** (.5834)	**39** (.5834)	**42** (.3472)
	SM	377	224	**257 (.0085)**	**257 (.0085)**	**252 (.029)**
	M	111	20	**38 (.003)**	**34 (.0173)**	**34 (.0173)**
	HM	172	17	**26** (.1004)	**20** (.4876)	**23** (.2375)

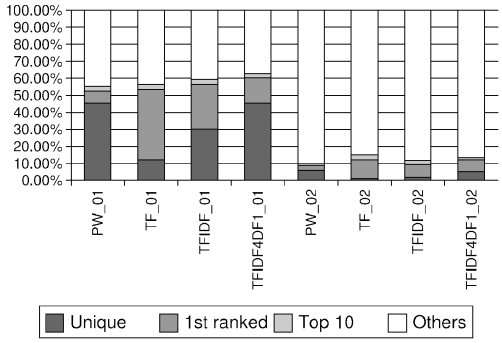
documents was greater than or equal to 0.8), retrieval performances of all LS methods were slightly decreased. In general, retrieval performances of DF/PW decreased more than for other LSs, and TF showed the best robustness under slight document modification. TFIDF was the second-most robust. The hybrid LS methods were less robust than TF and TFIDF, but still showed reasonable retrieval performance, and were better than DF/PW. This trend became more clear as documents were modified more. In the *Modified* class (where the cosine angle was between 0.5 and 0.8), TF was best, both in terms of retrieval



(a) YahooGoogle, *Slightly Modified*



(b) YahooGoogle, *Modified*



(c) YahooGoogle, *Heavily Modified*

Fig. 11. Retrieval performance of LSs in the *Slightly Modified*, *Modified*, and *Heavily Modified* classes.

performance and robustness, and DF and PW were worst. In the *Heavily Modified* class (where the cosine was less than 0.5), retrieval performances of all LS methods were reduced dramatically, almost to the level seen in the *Gone or Moved* class. DF/PW showed the worst performance, but the other LS methods were hardly better. Even in this case, TF was marginally better than the other LS methods. Figure 11 compares the extraction performances of LSs for the target documents in 2001 and 2002 when the target documents were modified after

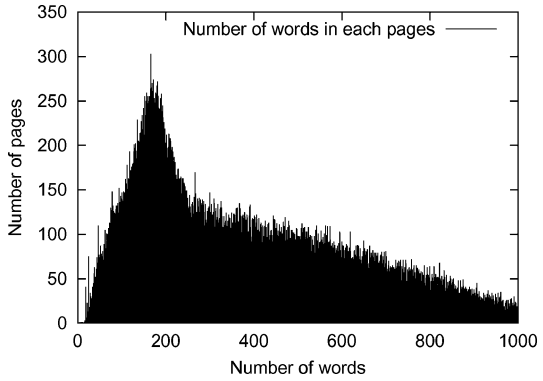


Fig. 12. Distribution of number of words in each document for TREC data.

a 10-month period and Table V summarizes the statistical significance between PW and other methods of each class for the desired document extraction in 2002.

In summary, TF shows the best robustness, while DF and PW are the most vulnerable under document modification. Even though a document is modified significantly, the most frequently appearing (highest TF) words are most likely to remain in any new versions of the document. However, the rarest (least DF) words might be misspellings or mistakes and may be likely to be removed over time. Hybrid methods are more robust than DF/PW. Hybrid methods are slightly more vulnerable than TF/TFIDF, though not significantly. All LS methods show significant performance degradation when documents are frequently modified. Since a large fraction of Web documents are frequently modified, LSs may need to be updated periodically to ensure reasonable performance. Note, however, that requiring LS updates too frequently reduces the practical benefit of using LSs in the first place, since LSs themselves are no longer persistent.

5. EMPIRICAL RESULTS FROM TREC DATA

We conducted similar experiments on 100,000 documents from TREC 3, 4, and 5: 20,000 documents from Ziff-Davis, 40,000 from AP Newswire, 20,000 from the *Wall Street Journal*, and 20,000 from the *San Jose Mercury News*. This data has the advantage of being a controlled source: unlike on the real Web, we could compute DF exactly, we can build a complete index, and we can test LSs for all documents in the corpus.

5.1 Data Set and Experimental Environment

From the 100,000 documents, we removed all tags, serial numbers that could identify the documents, dates, and names of newspapers and magazines such as *AP Newswire* in all documents. After removing stop words, we generated a term frequency list for each document and a document frequency list for all documents. Our corpus contained 404,657 unique words and 219,930 words had document frequency 1. Most documents were around 200 words in length; Figure 12 shows the distribution of length of all documents. We built a simple

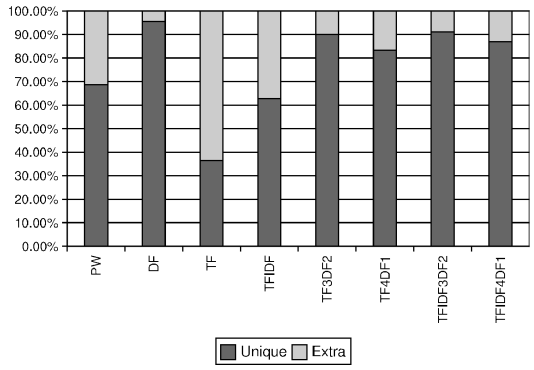


Fig. 13. Retrieval performance for unique identification on TREC data.

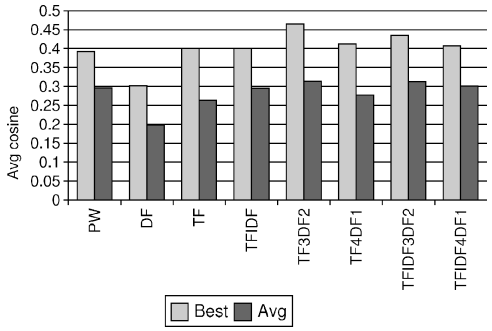


Fig. 14. Ability of LSs to find related documents. Average cosine value of best cases and average cases when the desired document was not extracted.

search engine for the experiment which had its own index file for all unique words, except stop words. The engine had two inputs, document-ID (name of the document) and its LS. When we fed a document-ID and its LS to the engine, it returned all documents that contained all words of the lexical signature except the target document of the given document-ID. If our routine did not return any documents, then the LS was unique. Unlike real search engines, the coverage of our search engine was complete. Since it did not have any ranking algorithm, we analyzed all returned documents.

5.2 Unique Identification

Figure 13 demonstrates the *unique* retrieval performance for different LS methods. As we expect, DF showed the best performance for this property and all hybrid methods had better performances than any basic method, except DF. Since the number of documents in TREC data is much smaller than on the Web, PW is more similar to TFIDF. In this case, hybrid methods performed better than PW even in uniquely identifying documents.

5.3 Finding Relevant Documents

Figure 14 shows the cosine value of the most relevant returned document, and the average cosine values of all returned documents, when the target document

Table VI. Number of Conflict Document Pairs and Conflict Ratio of Each LS

Basic Methods	PW	DF	TF	TFIDF
Π	29552	22116	42156	33864
CR	2.96e-06	2.21e-06	4.22e-06	3.39e-06
Hybrid Methods	TF3 DF2	TF4 DF1	TFIDF3 DF2	TFIDF4 DF1
Π	23640	23328	23630	23810
CR	2.36e-06	2.33e-06	2.36e-06	2.38e-06

was removed from the database. In general, hybrid methods extracted more relevant documents than basic methods, and their average cosine values were higher than those of basic methods.[21]

The poor performance of DF can be explained by the *uniqueness-robustness tradeoff*, defined by Phelps and Wilensky [2000a, 2000b]. LSs chosen to minimize DF were good at extracting the single desired document, but were not robust to minor modifications. Similarly, DF was not robust for finding relevant documents when the original document could no longer be found.

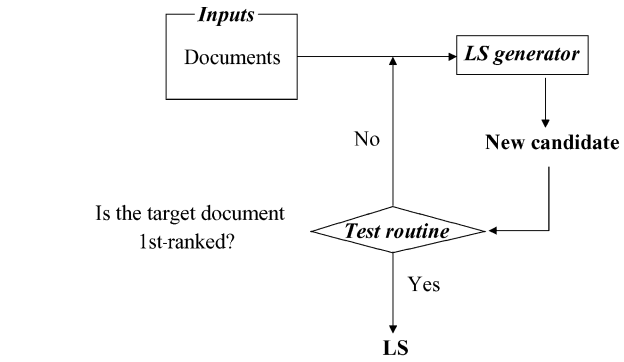
5.4 LS Conflict

LS generating methods can be considered as a hash function: each LS and its corresponding document can be considered as the output (value) and input (key), respectively, of the hash function. LSs in turn can be considered as inputs to a second (inverse) hash function—the search engine—which extracts the desired documents as outputs. Our aim is to find a good hash function that generates LSs that work well with search engines to find the target document or a replacement document. The LS designer generally does not have control over (or even access to) the search engine's internal "hash function," but does have control over the LS hash function.

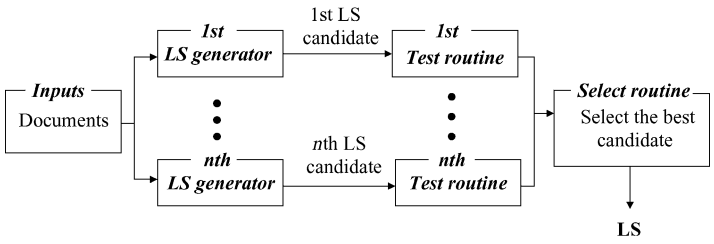
In the previous tests with both real Web data and TREC data, we have investigated four essential properties of LSs: retrieval performance, ability to find relevant information, search engine independence, and robustness under document modification. The last remaining LS property is minimal conflict: new LSs should have minimal conflict with existing LSs.

This property can be thought of in the following way: how often do two outputs of the LS hash function cause a collision? We examined the level of conflict for each of the LS methods by measuring how many document pairs generated the same LS. For this purpose, we measured the collision rate of each LS method for all 100,000 TREC documents. The collision rate *CR* is defined as follows: let Θ be the number of all possible document pairs and Π be the number of documents pairs whose LSs are identical. Then, $CR = \Pi/\Theta$. Table VI shows the collision rates for each LS method. As we expected, DF showed the smallest collision rate and TF the largest. All hybrid methods performed slightly worse than DF, but much better than other basic methods.

[21]The differences between TF4DF1 and other hybrid methods were not statistically significant.



(a) Recursive LS generation method



(b) Test & Select method

Fig. 15. Layout of two dynamic LS generation methods.

6. DYNAMIC LS GENERATION: THE *TEST & SELECT* METHOD

All methods tested so far generate LSs in a static manner without explicitly considering possible conflicts; in this section we consider methods for generating LSs dynamically in an attempt to actively avoid conflicts. The basic idea is to test the LSs in a search engine before adopting them. If an LS does not perform well, then the LS is regenerated. This procedure continues until a good LS is selected. There are many possible ways for generating LSs dynamically. An iterative method might operate as follows: after an LS is generated, it is tested; if the LS does not perform well, one word from the LS is deleted and a new word is added; then the process repeats. Which word should be removed among the five words? If the old LS extracted too many documents, the highest DF words (most common word) should be deleted. However, if it did not extract any documents, the rarest word should be deleted to find more relevant information. What if a few documents are returned by the search engine, including the desired document, but not first-ranked? To select the right word or words for replacement is not obvious in this and other situations.

A second method is somewhat easier to implement. Several LSs based on different methods are generated, tested in parallel, and then the best-performing LS among them is selected. More specifically, the "best" LS is chosen the following way: first, if any LS extracts the document uniquely, choose that LS. Otherwise, if any LS extracts the document as first-ranked, choose that LS. If that fails, choose any LS that extracts the document in the top 10. Otherwise, choose

Table VII. Statistical Significance Between TFIDF4DF1 and TS Methods for Retrieving the Desired Documents (Each entry records the raw number of successful results and, in parentheses, the p-value denoting the degree statistical significance. Bold numbers indicate an improvement over PW. Bold statistical significance values indicate that the p-value $< .05$. SE = search engine.)

Year	SE	TFIDF4DF1	TS_B	TS_H	TS_A
2001	YG	694	**772 (0)**	**747 (.0053)**	**786 (0)**
	MSN	696	**784 (0)**	**748 (.0468)**	**802 (0)**
	AV	648	**730 (.00001)**	**690 (.0409)**	**744 (0)**
2002	YG	495	**520** (.2427)	**513** (.3906)	**535** (.0632)
	MSN	445	**471** (.2233)	**458** (.5326)	**472** (.2006)
	AV	463	**478** (.4695)	**470** (.7207)	**482** (.1364)

the LS that yields the most similar first-ranked document. In case of a tie, we choose among the LS methods in following order: TFIDF4DF1, TFIDF3DF2, TF4DF1, TF3DF2, TFIDF, TF, PW, DF. We call this the *Test & Select* (TS) method. Two dynamic LS generative methods are illustrated in Figure 15. We experimented with the following TS methods:

(1) *TS_B*: Select the best-performing LS among basic methods.
(2) *TS_H*: Select the best-performing LS among hybrid methods.
(3) *TS_A*: Select the best-performing LS among all methods.

Dynamically generated LSs are better able to avoid collisions when a new LS is generated. Also, by testing LS candidates before choosing LSs of documents, the retrieval performance and ability to find relevant information can be improved. Figure 16 and Table VII show the retrieval performance and statistical significance figures for TS methods in 2001 and 2002. In 2001, retrieval performance increased significantly compared to any single static method. Perhaps surprisingly, TS_B showed better retrieval performance than TS_H. The retrieval performance TS_A was best.

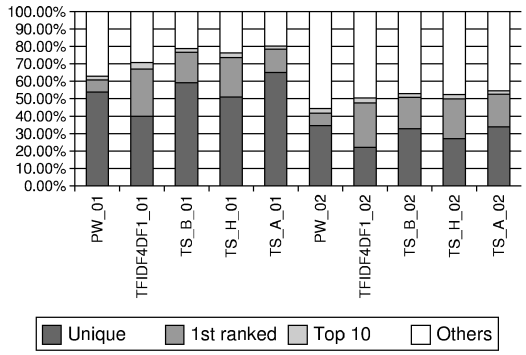
Even though TS improved retrieval ability over any single static LS method, the advantage of TS lessened over time. After 10 months, the TS methods were still better than the static methods, although their performance advantage became smaller.

We also observed significant performance improvement for finding relevant information. In 2001, TS_A performed best, TS_B second best, and TS_H third. Ten months later, however, the advantage of the TS methods disappeared: none of the TS methods had any significant advantage over the static hybrid methods, and some performed worse. Performance differences among the TS methods were negligible. Results are summarized in Figure 17. Results and statistical significance figures are reported in Tables VIII and IX.

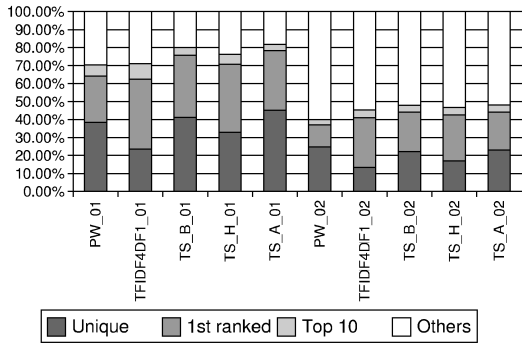
7. CONCLUSIONS

7.1 Summary

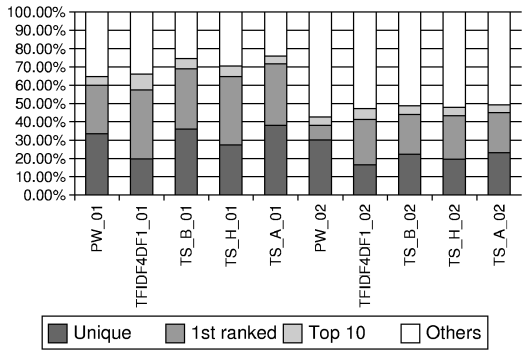
Because the Web does not have a well-adopted standard for maintaining persistence of information [Lawrence et al. 2001], the act of moving and deleting



(a) YahooGoogle, 2001 & 2002



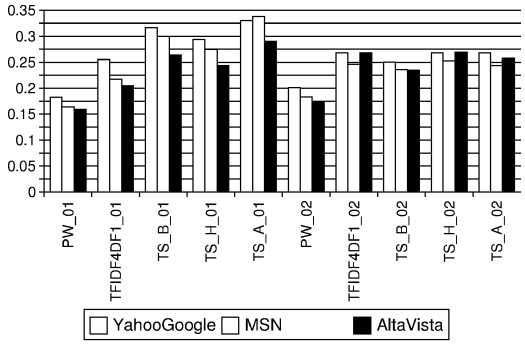
(b) MSN, 2001 & 2002



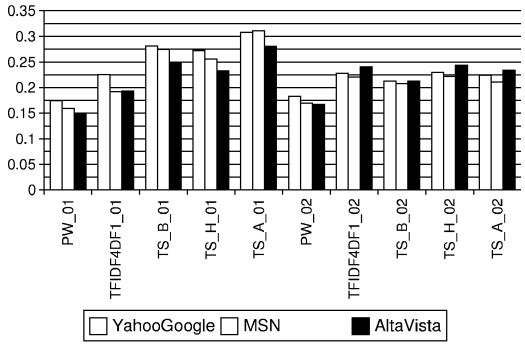
(c) AltaVista, 2001 & 2002

Fig. 16. Retrieval performance of TS methods.

documents creates a large number of broken links throughout the Web. Such broken links pose a significant problem for the growing number of people that rely on the Web as a universal database. Phelps and Wilensky [2000a, 2000b] showed that even a small number of words can often uniquely identify each document on the Web. In this article, we studied nine methods for generating such



(a) 1st ranked, 2001 and 2002



(b) Top 10, 2001 and 2002

Fig. 17. Average cosine of returned documents for *Not Retrieved Docs*.

Table VIII. Statistical Significance Between TFIDF4DF1 and TS Methods for Finding Relevant Documents in 2001 (Each entry records the raw cosine values and, in parentheses, the p-value denoting the degree statistical significance. Bold numbers indicate an improvement over PW. Bold statistical significance values indicate that the p-value $< .05$. SE = search engine.)

Type	SE	TFIDF4DF1	TS_B	TS_H	TS_A
1st	YG	.256	**.317 (.0121)**	**.294** (.0979)	**.330 (.001)**
	MSN	.217	**.300 (.0013)**	**.275 (.0216)**	**.338 (0)**
	AV	.205	**.264 (.0029)**	**.244 (.0466)**	**.290 (0)**
avg	YG	.225	**.281 (.0032)**	**.272 (.0164)**	**.308 (0)**
	MSN	.192	**.275 (0)**	**.256 (.0012)**	**.311 (0)**
	AV	.193	**.249 (.0002)**	**.232 (.0122)**	**.281 (0)**

lexical signatures (LSs), including Phelps and Wilensky's original proposal, seven of our own static methods, and one dynamic method. We argue that the *unique* extraction property is not the only important property for LSs. As long as the desired document appears first in a returned document list, the LS is effective. Also, since the coverage of search engines is limited, and documents are added, moved, modified, and deleted frequently, the ability to retrieve highly relevant documents when the desired document cannot be extracted is another important property for LSs. Moreover, since different search engines

Table IX. Statistical Significance Between TFIDF4DF1 and TS Methods for Finding Relevant Documents in 2002 (Each entry records the raw cosine values and, in parentheses, the p-value denoting the degree statistical significance. Bold numbers indicate an improvement over PW. Bold statistical significance values indicate that the p-value $< .05$. SE = search engine.)

Type	SE	TFIDF4DF1	TS_B	TS_H	TS_A
1st	YG	.267	.250 (.295)	**.268** (.9894)	**.268** (.9948)
	MSN	.246	.236 (.5044)	**.253** (.6922)	.244 (.8678)
	AV	.268	.235 **(.0341)**	**.270** (.9149)	.258 (.5107)
avg	YG	.228	.213 (.2575)	**.230** (.8904)	.224 (.7633)
	MSN	.220	.207 (.3134)	**.222** (.9064)	.211 (.4618)
	AV	.241	.212 **(.0215)**	**.244** (.8143)	.234 (.6117)

have different coverage and ranking systems, the consistency of LSs across search engines should be considered.

We found that DF-based LSs were best at uniquely identifying documents, on both Web data and TREC data. However, DF was worst at retrieving relevant documents when the desired document was missing. Moreover, DF was most vulnerable to document modification. PW acted almost identically to DF when the number of document was large, as is the case on the Web. However, when the number of documents was relatively small (e.g., around 100,000 documents), PW acted like TFIDF and its relevance performance improved. TF was worst at uniquely retrieving documents, but worked well for finding relevant documents. However, this result may be overstated, due to the bias of our measure for TF. TF also showed the best robustness under document modification, and was easy to compute and maintain. However, TF's performance variability across different search engines, and its relatively large LS conflict, may outweigh its benefits. TFIDF is the best candidate for LSs among the basic methods, due to its effectiveness at extracting both the desired document and relevant alternative documents. Also, it showed good robustness against document modification. Hybrid methods seemed even better candidates for generating LSs. They showed good retrieval of unique documents on both Web and TREC data—even better than PW on TREC data. Hybrid methods returned the desired document within the top few returned documents more often than even DF and PW. In addition, they showed excellent performance in retrieving relevant documents when the desired document was missing. Finally, their ability to extract both desired and relevant documents was relatively stable over different search engines, and reasonably robust to document changes. Hybrid methods were slightly less robust against document modification than TF and TFIDF, though not significantly. The hybrid methods' retrieval performance and ability to find relevant documents were as good as TF and TFIDF after a 10-month time period.

Performance can be improved by generating LSs dynamically. We propose a Test & Select (TS) method, which generates several LS candidates in parallel, tests them, and selects the best candidate among them. All TS methods showed significant improvement both in retrieving the desired document and finding relevant information when the document was missing. However, these advantages largely disappeared after 10 months.

Since a large fraction of Web documents are frequently modified, periodic updates of LSs may be required to ensure reasonable performance. However, when updates are required too frequently, the benefit of LSs as persistent identifiers is clearly reduced. The performance degradation of all LS methods over a fairly short time period—10 months, from November 2001 to September 2002—suggests that the use of lexical signatures as a cure-all for persistence of information on the Web may be viewed with a fair degree of skepticism. For more stable document collections, for example, scientific publications, white articles, digital books, archival records, etc., lexical signatures may provide a viable alternative to more burdensome solutions to the broken links problem.

7.2 Limitations

Judging that one document is a modified version of another, or is relevant to another, is ultimately subjective. It is widely considered that human judgment is the most accurate measure of relevance, but obtaining such relevance judgments for arbitrary pairs of documents is costly and time-consuming. Even for TREC data, only relevance judgments between topics and documents are available, and trying to infer document-document relevance from document-topic relevance can be dubious, since relevance is not in general a transitive property. We focused on using the cosine measure to estimate relevance, which is at least consistent if only a coarse approximation.

A related issue is our definition of when two documents are the same. In many cases, the cosine value of two documents is less than 1, even though two documents have nearly identical content, for example, two presentations of an AP news story, each bordered with different navigational HTML. Somewhat arbitrarily, we considered two documents to be the same if the cosine value of two documents was larger than 0.9. Clearly, it is possible that two documents are the same even though their cosine is less than 0.9, and it is possible that two documents are different even though their cosine is nearly 1, though (i.e., weather reports on two different days). We feel that these situations would be rare, and would not significantly effect the nature of our conclusions. Note also that, since we use a TF-based cosine measure for Web documents, and a TFIDF-based cosine measure for the TREC corpus, our results may slightly favor the TF- and TFIDF-based lexical signatures, respectively.

A second limitation stems from our reliance on commercial search engines for the experiments. To measure DF, we had to query a search engine for every unique word in the corpus—more than a million for just the 980 documents in this study. If we had used another source for DF values, we would have run the risk of choosing lexical signature words that our target search engine did not even index. Although our testing procedure itself entailed many search engine queries, the main impediment to scaling up the size of the experiment was computing DF values. Since roughly half of all terms are unique in almost any natural language corpus, the number of queries required to obtain DF values grows very quickly with the size of the corpus. Many search engines actively

block what look like large sets of automated queries, reducing our ability to conduct a larger-scale experiment.[22]

The particular URLs we chose were the first 1500 within a set of URLs crawled for another purpose. A better methodology might have been to sample documents randomly. However, defining a random Web document is nontrivial, and is the subject of ongoing research among several groups [Henzinger et al. 1999, 2000; Rusmevichientong et al. 2001]. We believe that our methodology, though perhaps not ideal, still provided valuable and interpretable lessons that proved statistically significant.

7.3 Performance Evaluation Methods for Search Engines

It has been widely argued that search engines should be evaluated by their ability to retrieve highly relevant documents rather than all possible documents [Jarvelin and Kekalainen 2000; Voorhees 2001]. LSs are good query terms that can extract relevant documents when the desired document cannot be retrieved. One limitation of LSs mentioned by Phelps and Wilensky [2000a, 2000b] is that their performance can depend on particular search engines. However, this limitation can be exploited to evaluate search engine performance. Because a document's TF values are independent of other documents in the database, and because TF-based LSs usually extract more than 10 documents, the ability of TF-based signatures to extract relevant documents is highly dependent on the search engine's ranking system. By measuring similarities of returned documents with the target document, using a similarity measure or human judgments, we can evaluate the search engine's ability to retrieve and rank relevant documents. In our experiments, YahooGoogle showed the best performance (among the engines tested) for retrieving both desired documents and relevant documents, in almost all cases in 2001. However, as we can see in Figures 7 and 8, by 2002 AtaVista had become competitive with YahooGoogle, especially in its ability to extract similar replacement documents. In 2002, AltaVista showed the best average cosine of the top 10 documents, for all LS methods except DF/PW, even though our DF values were obtained from Google.

ACKNOWLEDGMENTS

We would like to thank Gary Flake and NEC Laboratories America for providing the Web data; Steve Lawrence, Justin Sobel, and the anonymous referees for useful insights and comments; and Douglas Oard for suggesting the use of a dynamic LS generating mechanism.

REFERENCES

AIMAR, A., HANNELL, I., KHODABANDEH, A., PALAZZI, P., ROUSSEAU, B., RUGGIER, M., CASEY, J., AND DRAKOS, N. 1995. Weblinker: A tool for managing WWW cross-references. *Comput. Netw. ISDN Syst. 28*, 1&2 (Dec.), 99–107.

[22] In fact, after completing the DF computations, Google temporarily blocked the first author's entire department and half of his engineering school, making us wary of scaling up the experiment, and prompting our shift to YahooGoogle.

ANDREWS, K., KAPPE, F., AND MAURER, H. 1995. The Hyper-G network information system. *J. Univers. Comput. Sci. 1*, 4 (April), 206–220.

ARMS, W., BLANCHI, C., AND OVERLY, E. 1997. An architecture for information in digital libraries. *D-Lib Mag*. Available at http://www.dlib.org/dlib/february97/cnri/02arms1.html.

BERNERS-LEE, T., FIELDING, R., AND FRYSTYK, H. 1996. Hypertext transfer protocol—http/1.0. Available at http://www.w3.org/Protocols/HTTP/1.0/draft-ietf-http-spec.html.

BHARAT, K. AND BRODER, A. Z. 1998. A technique for measuring the relative size and overlap of public Web search engines. In *Proceedings of the 7th International World Wide Web Conference* [WWW7]. 379–388.

BREWINGTON, B. E. AND CYBENKO, G. 2000. How dynamic is the Web? *Comput. Netw. (Amsterdam, The Netherlands: 1999) 33*, 1–6, 257–276.

CHO, J. AND GARCIA-MOLINA, H. 2000. The evolution of the Web and implications for an incremental crawler. In *Proceedings of the 26th International Conference on Very Large Databases*. 117–128.

CREECH, M. L. 1996. Author-oriented link management. In *Proceedings of the 5th International World Wide Web Conference*.

DOUGLIS, F., FELDMANN, A., KRISHNAMURTHY, B., AND MOGUL, J. C. 1997. Rate of change and other metrics: A live study of the world wide web. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*. 147–158.

FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. 1999. Rfc 2616—hypertext transfer protocol—http/1.1. Available at http://www.w3.org/Protocols/rfc2616/rfc2616.html.

FIELDING, R. T. 1994. Maintaining distributed hypertext infostructures: Welcome to MOMspider's Web. *Comput. Netw. ISDN Syst. 27*, 2, 193–204.

FISHER, R. A. 1966. *Design of Experiments*, 8th ed. Hafner (Macmillan), New York, NY.

GILES, C. L., BOLLACKER, K., AND LAWRENCE, S. 1998. CiteSeer: An automatic citation indexing system. In *Digital Libraries 98—The Third ACM Conference on Digital Libraries* (Pittsburgh, PA), I. Witten, R. Akscyn, and F. M. Shipman III, Eds. ACM Press, New York, NY, 89–98.

GOLDBERG, A. AND YIANILOS, P. N. 1998. Towards an archival intermemory. In *IEEE Advances in Digital Libraries*. (ADL '98, Santa Barbara, (CA). IEEE Computer Society Press, Los Alamitos, CA, 147–156.

GULESIAN, M. 1996. Netscape livewire pro 1.0. Available at http://www.dbmsmag.com/9612d08.html.

HENZINGER, M. R., HEYDON, A., MITZENMACHER, M., AND NAJORK, M. 1999. Measuring index quality using random walks on the Web. In *Proceedings of the 8th International World Wide Web Conference*. 213–225.

HENZINGER, M. R., HEYDON, A., MITZENMACHER, M., AND NAJORK, M. 2000. On near-uniform URL sampling. In *Proceedings of the 9th International World Wide Web Conference*. 295–308.

INGHAM, D., CAUGHEY, S., AND LITTLE, M. 1996. Fixing the 'broken-link' problem: The W3Objects approach. In *Proceedings of the 5th International World Wide Web Conference*.

INGHAM, D., LITTLE, M., CAUGHEY, S., AND SHRIVASTAVA, S. 1995. W3Objects: Bringing object-oriented technology to the Web. In *Proceedings of the 4th International World Wide Web Conference*. 89–105.

JARVELIN, K. AND KEKALAINEN, J. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of SIGIR 2000*. 41–48.

LAWRENCE, S. AND GILES, C. L. 1998a. Context and page analysis for improved web search. *IEEE Internet Comput. 2*, 4, 38–46.

LAWRENCE, S. AND GILES, C. L. 1998b. Searching the World Wide Web. *Science 280*, 5360 (April), 98–100.

LAWRENCE, S. AND GILES, C. L. 1999. Accessibility of information on the Web. *Nature 400*, 107–109.

LAWRENCE, S., GILES, C. L., AND BOLLACKER, K. 1999. Digital libraries and autonomous citation indexing. *IEEE Comput. 32*, 6, 67–71.

LAWRENCE, S., PENNOCK, D. M., FLAKE, G., KROVETZ, R., COETZEE, F. M., GLOVER, E., NIELSEN, F. A., KRUGER, A., AND GILES, C. L. 2001. Persistence of Web references in scientific research. *IEEE Comput. 34*, 2 (Feb.), 26–31.

MACE, S., FLOHR, U., DOBSON, R., AND GRAHAM, T. 1998. Weaving a better Web. *Byte 23*, 3, 55–68.
NOREEN, E. W. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley, New York, NY.
OBERHOLZER, G. AND WILDE, E. 2002. Extended link visualization with DHTML: The Web as an open hypermedia system. Tech. rep. TIK-125. Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland.
PARK, S. T., PENNOCK, D. M., GILES, C. L., AND KROVETZ, R. 2002. Analysis of lexical signatures for finding lost or related documents. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 11–18.
PHELPS, T. A. AND WILENSKY, R. 2000a. Robust hyperlinks: Cheap, everywhere, now. In *Proceedings of Digital Documents and Electronic Publishing 2000* (DDEP00).
PHELPS, T. A. AND WILENSKY, R. 2000b. Robust hyperlinks cost just five words each. Tech. rep. UCB//CSD-00-1091. University of California, Berkeley, Berkely, CA.
PITKOW, J. 1998a. Web characterization activity: Answers to the W3C HTTP-NGs protocol design group's questions. World Wide Web Consortium (WWW.W3.org).
PITKOW, J. E. 1998b. Summary of WWW characterizations. *Comput. Netw. ISDN Syst. 30*, 1–7, 551–558.
REICH, V. AND ROSENTHAL, D. S. H. 2001. Lockss: A permanent Web publishing and access system. *D-Lib Mag. 7*, 6 (June), 55–68.
RUSMEVICHIENTONG, P., PENNOCK, D. M., LAWRENCE, S., AND GILES, C. L. 2001. Methods for sampling pages uniformly from the World Wide Web. In *Proceedings of the AAAI Fall Symposium on Using Uncertainty Within Computation*. 121–128.
SHAFER, K., WEIBEL, S., JUL, E., AND FAUSEY, J. 1996. Introduction to persistent uniform resource locators. In *INET 96*. Internet Society, Reston, Va.
SOLLINS, K. AND MASINTER, L. 1994. *Functional Requirements for Uniform Resource Names*, Network Working Group, RFC 1737. Available at http://www.faqs.org/rfcs/rfc1737.html.
VOORHEES, E. M. 2001. Evaluation by highly relevant documents. In *Proceedings of the SIGIR 2001*. 74–80.
WITTEN, I. H., MOFFAT, A., AND BELL, T. C. 1999. *Managing Gigabytes*, 2nd ed. Harcourt Science and Technology Company, San Diego, CA.

Received February 2003; revised December 2003; accepted February 2004