Similarity Search on the Web: Evaluation and Scalability Considerations

Taher H. Haveliwala
Stanford University
taherh@db.stanford.edu

Aristides Gionis
Stanford University
gionis@cs.stanford.edu

Dan Klein
Stanford University
klein@db.stanford.edu

Piotr Indyk
M.I.T.
indyk@theory.lcs.mit.edu

ABSTRACT

Allowing users to find pages on the web similar to a particular query page is a crucial component of modern search engines. A variety of techniques and approaches exist to support "Related Pages" queries. In this paper we discuss shortcomings of previous approaches and present a unifying approach that puts special emphasis on the use of text, both within anchors and surrounding anchors. In the central contribution of our paper, we present a novel technique for automating the evaluation process, allowing us to tune our parameters to maximize the quality of the results. Finally, we show how to scale our approach to millions of web pages, using the established Locality-Sensitive-Hashing technique.

1. INTRODUCTION

The goal of supporting similarity search on the web is to allow users to find all web pages similar to a query page [10]. The first step to providing such "Related Pages" functionality is deciding on document representation. Broadly speaking, there are three approaches to selecting the terms to include in the vector (or equivalently, multiset) representing a web page u:

1. Include words appearing in u (content-based approach)
2. Include document identifiers (for instance urls) for each document v that links to u (link-based approach)
3. Include words appearing inside or near an anchor in v, when the anchor links to u (anchor-based approach)

The usual content-based approach ignores the available hyperlink data and is susceptible to spam. In particular, it relies solely on the information provided by the page's author, ignoring the opinions of the authors of other web pages [3]. The link-based approach, investigated in [10], suffers from the shortcoming that pages with few inlinks will not have sufficient citation data, either to be allowed in queries or to appear as results of queries. This problem is especially pronounced when attempting to discover similarity relations for new pages which have not yet been sufficiently cited. As we will see in Section 4, under a link-based approach, the vectors for *most* documents (even related ones) are in fact orthogonal to each other.

The third approach, which relies on text near anchors, referred to as the *anchor-window* [8], appears most useful for the web similarity-search task. Indeed, the use of anchor-windows has been previously considered for a variety of web IR tasks [2, 1, 8, 9]. The anchor-window often constitutes a hand-built summary of the target document [1], collecting both explicit hand-summarization and implicit hand-classification present in referring documents.

We expect that when aggregating over *all* inlinks, the frequency of relevant terms will dominate the frequency of irrelevant ones. Thus, the resulting distribution is expected to be a signature which is a reliable, concise representation of the document. Because each anchor-window contributes several terms, the anchor-based strategy requires fewer citations than the link-based strategy to prevent interdocument orthogonality. However, as a result of reducing orthogonality, the anchor based strategy is nontrivial to implement efficiently. We discuss later how a previously established high-dimensional similarity-search technique based on hashing can be used to efficiently implement the anchor-based strategy.

These three general strategies for document representation involve additional specific considerations. We must choose how to weight terms, whether or not to stem terms, and in the case of the anchor-based strategy, how to bound the anchor-window regions. Furthermore, the most effective document representation may consist of a combination of the various strategies.

Clearly, devising an approach to allow feature selection in this huge parameter space is necessary. User studies are costly, and feasible only for exploring a small fraction of the feature space. In the central contribution of our paper, we develop an automated evaluation methodology to determine the optimal document representation strategy.

In particular, we view manually constructed collections such as Yahoo! and Open Directory ([12, 18]) as a kind of generic precompiled "user study." Our evaluation methodology uses the notion of document similarity implicitly represented by these hierarchical directories to induce "correct",

This material is based upon work supported by the National Science Foundation under Grant No. 0085896.

ground truth orderings. Then, using a statistical measure, we compare orderings obtained from different parameter settings to the correct orderings. Our underlying assumption is that parameter settings which yield higher values of this measure correspond to parameters that will produce better results. To demonstrate our evaluation methodology, we applied it to a reasonably sized set of parameter settings and determined which of them is most effective for supporting similarity search on the web.

Scaling up our similarity measure from the evaluation test set to the web as a whole requires utilizing an efficient algorithm for similarity search, given our feature space. We rely on the previously established *Locality-Sensitive Hashing (LSH)* technique, introduced by Indyk and Motwani [15] and independently discovered by Broder [4]. The basic idea is to hash the web page vector such that the pages that are similar, according to our similarity measure, have a much higher probability of being mapped to the same bucket than pages that are very different. LSH requires the ability to create a small signature for each web page vector, such that similar documents have the same signatures. Creating such signatures for the cosine measure is to our knowledge an open problem.

On the other hand, such signatures can be constructed for the *Jaccard coefficient* similarity measure (see [5]). The Jaccard coefficient of two multisets[1] A and B is defined as

$$sim_J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

This efficiency consideration is why we focus exclusively on the Jaccard measure rather than the cosine measure in our current work [2].

To demonstrate the scalability of our approach, we constructed a similarity search index over 10 million documents using a single machine. We have built a query interface over this index with favorable results. Because each stage of our algorithm is trivially parallelizable, with 100 machines the approach can scale to build the index for 1 billion documents[3]. Although we do not discuss issues involving parallelization in this paper, in our ongoing work we are constructing a similarity-search index for close to 150 million pages.

2. DOCUMENT REPRESENTATION

A web document u is represented by a bag

$$B_u = \{(w_u^1, f_u^1), \ldots, (w_u^k, f_u^k)\}$$

where w_u^i are terms used in representing u (e.g., terms found in the content and anchor-windows of u, or links to u), and f_u^i are corresponding weights. As discussed in the introduction, there are several considerations as to *which* words should be placed in a document's bag, and with what weight, which we now discuss in detail.

2.1 Choosing Terms

For both the content and anchor-based approaches, we chose to remove all HTML comments, Javascript code, tags (except 'alt' text), and non-alphabetic characters. A custom stopword list containing roughly 800 terms was also applied.

For the anchor-based approach, we must also decide how many words to the left and right of an anchor A_{vu} should be included in B_u. We experimented with three strategies for this decision. In all cases, the anchor-text itself A_{vu}, is included, as well as the title of document u. The three strategies follow:

BASIC: We choose some fixed window size W, and always include W words to the left, and W words to the right, of A_{vu}[4]. Specifically, we use $W \in \{0, 4, 8, 16, 32\}$. Note that when W is 32 we are in fact using a large fraction of the entire referring document.

SYNTACTIC: We use sentence, paragraph, and HTML-region-detection techniques to dynamically bound the region around A_{vu} that gets included in B_u. The primary document features that are capable of triggering a window cut-off are paragraph boundaries, table cell boundaries, list item boundaries, and hard breaks which follow sentence boundaries. By inspection of sample documents, these heuristics were in fact reliable in isolating these descriptions when they existed. This technique resulted in very narrow windows that averaged only close to 3 words in either direction.

TOPICAL: We use a simple technique for guessing topic boundaries at which to bound the region which gets included. The primary features which trigger this bounding are heading beginnings, list ends, and table ends. The goal of these broader windows was to isolate topic regions. A particularly common case handled by these windows was that of documents composed of several regions, each beginning with a descriptive header and consisting of a list of urls on the topic of that header. Regions found by the TOPICAL heuristics averaged about 21 words in size to either side of the anchor.

2.2 Stemming Terms

We explored the effect of three different steming variations:

NOSTEM: The term is left as is. If it appears in the stoplist, it is dropped.

STEM: The term is stemmed using Porter's well known stemming algorithm [19] to remove word endings. If the stemmed version of the term appears in the stemmed version of our stoplist, it is dropped.

STOPSTEM: The term is stemmed as above, for the purposes of checking whether the term stem is in the stoplist. If it is, the term is dropped, otherwise the original *unstemmed* term is added to the bag.

The STOPSTEM variation provides valuable insight into the effects of stemming, which are twofold: *(i)* Terms which should be collapsed, for instance 'computer' and 'computers,' get mapped to the same stem, 'computer' and *(ii)* many terms useless for detecting similarity which do not themselves appear frequently enough to get flagged as stopwords,

[1] We extend Jaccard to multisets by applying bag union and bag intersection (max and min multiplicity, resp.)

[2] We did experimentally verify that the cosine measure leads to the same relative ordering of parameter settings as the Jaccard coefficient.

[3] Note that commercial search engines in general have several hundreds or thousands of machines.

[4] Stopwords do not get counted when determining the window cutoff.

e.g., 'informs,' will be flagged when using STOPSTEM, since the stem 'inform' is shared with the high frequency stopword 'information'. The first effect is potentially beneficial as a form of dimensionality reduction. The second is beneficial if it is the case that the usefulness of a term can be determined by the properties of its stem more accurately than by the properties of the term itself. As we will see in Section 4, both effects manifest themselves to differing degrees in our experiments.

2.3 Term Weighting

A further consideration in generating document bags is how a term's frequency should be scaled. A clear benefit of the TF.IDF family of weighting functions is that they attenuate the weight of terms with high document frequency. These monotonic term weighting schemes, however, amplify the weight of terms with very low document frequency. This is in fact a good approach for ad-hoc queries, where a rare term in the query should be given the most importance. In the case where we are judging document similarities, rare terms are much less useful as they are often typos, rare names, or other non-topical terms that adversely affect the similarity measure. Therefore, we also experimented with non-monotonic term-weighting schemes that attenuate both high and low document-frequency terms. The idea that mid-frequency terms have the greatest "resolving power" is not new [20, 17]. We call such schemes *non-monotonic document frequency (NMDF)* functions.

Another component of term weighting which we consider, and which has a substantial impact on our quality metric, is *distance weighting*. When using an anchor-based approach of a given window size, instead of treating all terms near an anchor A_{vu} equally, we can weight them based on their distance from the anchor (with anchor-words themselves given distance 0). As we will see in Section 4, the use of a distance-based attenuation function in conjunction with large anchor-windows significantly improves results under our evaluation measure.

3. EVALUATION METHODOLOGY

The quality of the rankings returned by our system is almost exclusively determined by the similarity metric and document features used. Previous work [10] has relied on user studies to assess query response quality. However, user studies are time-consuming, costly, and not well-suited to research that involves the comparison of many parameters. We propose an automated method of evaluation which uses the orderings implicit in human-built hierarchical directories to judge the quality of our system's rankings.

The overall outline of our evaluation method is as follows. We use a hierarchical directory to induce correct, ground truth orderings. Then, we compare our own orderings to these correct partial orderings using a measure outlined below. We claim that parameter settings which yield higher values of this measure correspond to parameters which will produce better results from the standpoint of a user of the system.

3.1 Finding a Ground Truth Ordering

Unfortunately, there is no available ground truth in the form of either exact document-document similarity values or correct similarity search results. However, there is a great deal of ordering information implicit in the hierarchical web

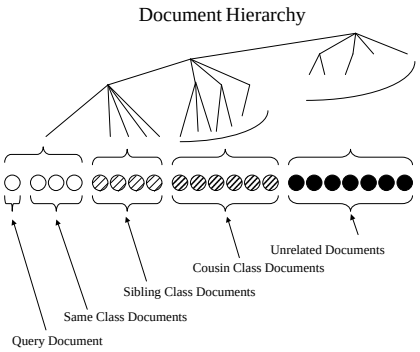

Figure 1: Mapping a hierarchy onto a partial ordering, given a source document.

directories mentioned above. For example, a document in the `recreation/aviation/un-powered` class is on average more similar to other documents in that same class than those outside of that class. Furthermore, that document is likely to be more similar to other documents in other `recreation/aviation` classes than those entirely outside of that region of the tree. Intuitively, the most similar documents to that source are the other documents in the source's class, followed by those in sibling classes, and so on.

There are certainly cases where location in the hierarchy does not accurately reflect document similarity. Consider documents in `recreation/autos`, which are almost certainly more similar to those in `shopping/autos` than to those in `recreation/smoking`. In our sample, these anomalies are rare and do not affect our evaluation criteria since we average over the statistics of many documents.

To formalize the notion of distance from a source document to another document in the hierarchy we define *familial distance*.

DEFINITION 1. *Let the* familial distance $d_f(s, d)$ *from a source document s to another document d in a class hierarchy be the distance from s's class to the most specific class dominating both s and d.*

In our system, however, we have collapsed the directory below a fixed depth of three and ignored the (relatively few) documents above that depth. Therefore, there are only four possible values for familial distance, as depicted in Figure 1:

0 *Same* – Documents in the same class as the source.

1 *Siblings* – Documents whose classes are siblings to the source's class.

2 *Cousins* – Documents whose classes are first cousins to the source's class.

3 *Unrelated* – Documents whose lowest common ancestor to the source in the hierarchy is the root.

Given a source document, we wish to use the familial distances to other documents to construct a partial similarity ordering over those documents. Our general principle is:

> The true similarity of documents to a source document decreases monotonically with the familial distance from that document.

Given this principle, and our familial distance, for any source document in a hierarchical directory we can derive a partial ordering of all other documents in the directory.

DEFINITION 2. *Let the* familial ordering $\prec_{d_f(s)}$ *of all documents with respect to a source document* s *be* $\{(a,b)|\ d_f(s,a) < d_f(s,b)\}$.

This ordering is very weak in that for a given source, most pairs of documents are not comparable. The majority of the distinctions that *are* made, however, are among documents that are very similar to the source and documents which are much less similar. The very notion of a correct total similarity ordering is somewhat suspect. Beyond a certain point, pages are simply unrelated. Our familial ordering reflects this by making no distinctions between the documents in the most distant category, which forms the bulk of the documents in the repository.

Of course our principle that true similarity decreases monotonically with familial distance does not always hold. However it is reasonable to expect that, on average, a ranking system[5] that accords better with familial ordering will be better than one that accords less closely.

3.2 Comparing Orderings

At this point, we have derived a partial ordering from a given hierarchical directory and query (source) document s. We wish to then use this partial ordering to evaluate the correctness of an (almost) total ordering produced by our system[6]. Perhaps the most common method of comparing two rankings is the Spearman rank correlation coefficient. This measure is best suited for comparing rankings with few or no ties and its value corresponds to a Pearson ρ coefficient [21]. There are two main problems with using the Spearman correlation coefficient for the present work. First, as mentioned, there are a tremendous number of ties in one of the rankings, and second, since we are more concerned with certain regions of the rankings than others (e.g., the top), we would like a natural way to directly measure how many of the "important" ranking choices are being made correctly. Given these goals, a more natural measure is the Kruskal-Goodman Γ [11].

DEFINITION 3. $\Gamma(\prec_a, \prec_b) = 2*Pr[\prec_a, \prec_b \textit{ agree on } (x,y)|\ \prec_a, \prec_b \textit{ order } (x,y)] - 1$

Intuitively, there are a certain number of document pairs, and a given ordering only makes judgments about some of those pairs. When comparing two orderings, we look at only the pairs of documents that both orderings make a judgment about. A value of 1 is perfect accord, 0 is the expected value of a random ordering, and -1 indicates perfect reversed accord. We claim that if two rankings $\prec_a$ and $\prec_b$ differ in their Γ values with respect to a ground truth $\prec_t$, then the ordering with the higher Γ will be the better ranking.

3.3 Regions of the Orderings

[5] Of course the ranking system cannot make use of the directory itself.

[6] Our ordering produces ties when two documents d_1 and d_2 have exactly the same similarity to the source document s. When this happens, it is nearly always because s is orthogonal to both d_1 and d_2 (similarity 0 to both).

```
Source document:  www.aabga.org
(American Association of Botanical Gardens and Arboreta)
Category of source document: /home/gardens/clubs_and_associations

window size = 32, stem, dist and freq term weighting
Sibling-Gamma: 0.53

1    0.16  /home/gardens/clubs_and_associations
2    0.15  /home/gardens/clubs_and_associations
5    0.13  /home/gardens/clubs_and_associations
10   0.11  /home/gardens/plants
20   0.10  /home/gardens/clubs_and_associations
50   0.07  /home/gardens/plants
100  0.06  /home/apartment_living/gardening

window size = 0, nostem, no term weighting
Sibling-Gamma: 0.30

1    0.17  /reference/libraries/independent_libraries
2    0.15  /home/gardens/clubs_and_associations
5    0.14  /industries/construction_and_maintenance
10   0.14  /business/industries/agriculture_and_forestry
20   0.13  /recreation/travel/reservations
50   0.13  /recreation/travel/reservations
100  0.13  /business/industries/construction_and_maintenance
```

Figure 2: Orderings obtained from two different parameter settings with respect to the same source document. For contrast we give the best and the worst settings. For each document shown, we give the rank, the similarity to the source document, and the category (we omit the url of the document).

Thus, given a directory, a query document s, and a similarity measure sim, we can construct two orderings: the ground truth familial ordering $\prec_{d_f(s)}$, and the ordering induced by our similarity measure $\prec_{sim(s)}$. We can then calculate the corresponding Γ value. This gives us a measure of the quality of the ranking for that query document with respect to that similarity measure and directory. However, we need to give a sense of how good our rankings are across all query documents. In principle, we can directly extend the Γ statistic by iterating through all documents, aggregating all the concordant and discordant pairs, and dividing by the total number of pairs.

In order to more precisely evaluate our results, however, we calculated three partial-Γ values that emphasized different regions of the ordering. Each partial-Γ is based on the fraction of correct comparable pairs *of a certain type*. Our types are:

> *Siblings*-Γ: Calculated from only pairs of documents (d_1, d_2) where d_1 was from the same class as the source document and d_2 was from a sibling class.
>
> *Cousins*-Γ: Calculated from only pairs of documents (d_1, d_2) where d_1 was from the same class as the source document and d_2 was from a cousin class.
>
> *Unrelated*-Γ: Calculated from only pairs of documents (d_1, d_2) where d_1 was from the same class as the source document and d_2 was from an unrelated class.

These partial-Γ values allowed us to inspect how various similarity measures performed on various regions of the rankings. For example, sibling-Γ performance indicates how well fine distinctions are being made near the top of the familial ranking, while unrelated-Γ performance measures how well coarser distinctions are being made. Unrelated-Γ is also a good indicator of situations when the top of the list is high-quality from a precision standpoint but many similar documents have been ranked very low and therefore omitted from the top of the list (almost always because the features

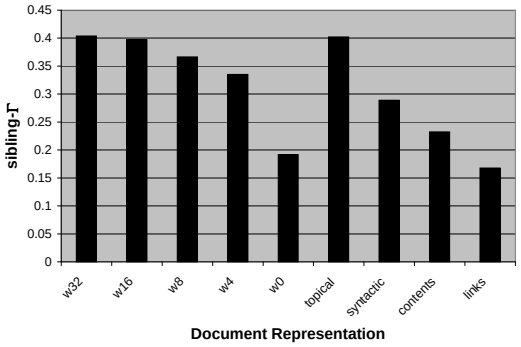

Figure 3: Pure Bags

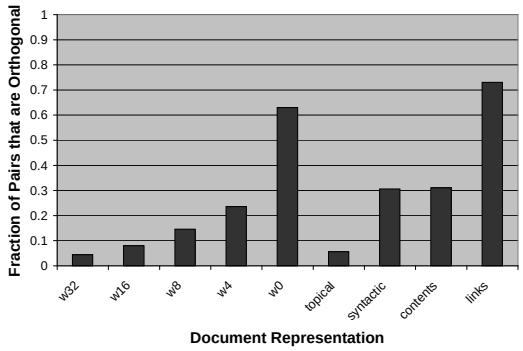

Figure 4: Intracluster Orthogonality

were too sparse, and documents that were actually similar appeared to be orthogonal).

One important thing to note about our various result measures is that, with few exceptions, better parameter settings were better for all measures. In Figure 2, we show an example that justifies our assumption that larger values of Γ statistic correspond to parameter settings that yield better results.

4. EXPERIMENTAL RESULTS OF PARAMETER EVALUATION

For evaluating the various strategies as discussed in Section 2, we employ the methodology described in the previous section. We sampled Open Directory [18] to get 300 pairs of clusters from the third level in the hierarchy, as depicted previously in Figure 1[7]. As our source of data, we used a web crawl containing 42 million pages [14]. Of the urls in the sample clusters, 51,469 of them were linked to by some document in our crawl, and could thus be used by our anchor-based approaches. These test-set urls were linked to by close to 1 million pages in our repository, all of which were used to support the anchor based strategy we studied[8]. This section describes the evaluation of the strategies suggested in Section 2. For most results, we will concentrate only on how the strategies performed under the sibling-Γ statistic, as discussed in Section 3. Note that the results of our large scale experiment, demonstrating the scalability of our similarity measure, is discussed in Section 6.

4.1 Results: Choosing Terms

The sibling-Γ values for bags generated using various anchor-window sizes, using TOPICAL and SYNTACTIC window-bounding, using purely links, and using purely page contents, are given in Figure 3.

The results for an anchor based approach using large windows provides the best results according to our evaluation criteria. This may seem counterintuitive; by taking small windows around the anchor, we would expect fewer spurious words to be present in a document's bag, providing a more concise representation. Further experiments revealed why, in fact, larger windows provide benefit.

[7]Any urls present below the third level were collapsed into their third level ancestor category.

[8]Open Directory pages themselves, however, were excluded from the data set to avoid bias

Figure 4 shows the fraction of document pairs within the same Open Directory cluster that are *orthogonal* (i.e., no common words) under a given representation. We see that with smaller window sizes, many documents that should be considered similar are in fact orthogonal. In this case, no amount of re-weighting or scaling can improve results; the representations simply do not provide enough accessible similarity information about these orthogonal pairs. We also see that under the content and link approaches, documents in the same cluster are largely orthogonal. Under the link based approach, *most* of the documents within a cluster are pairwise orthogonal, revealing a serious limitation of a purely link-based approach. Incoming links can be thought of as being opaque descriptors. If two pages have many inlinks, but the intersection of their inlinks is empty, we can say very little about these two pages[9]. It may be that they discuss the same topic, but because they are new, they are never cocited. In the case of the anchor-window-based approach, the chance that the bags for the two pages are orthogonal is much lower. Each inlink, instead of being represented by a single opaque url, is represented by the descriptive terms that are the constituents of the inlink.

We also experimented with dynamically sized SYNTACTIC and TOPICAL windows, as described in Section 2. These window types behave roughly according to their average window size, both in Γ values and orthogonality. Surprisingly, although the dynamic-window heuristics appeared to be effective in isolating the desired regions, any increase in region quality was overwhelmed by the trend of larger windows providing better results[10].

In addition to varying window size, we can also choose to include terms of multiple types (anchor, content, or links, as described in Section 2) in our document representation. Figure 5 shows that by combining content and anchor-based bags, we can improve the sibling-Γ score[11]. The intuition for this variation is that if a particular document has very few incoming links then the document's contents will dominate the bags, otherwise, if the document has many incoming

[9]Using the SVD we could potentially glean some information in a pure-link approach despite orthogonality, assuming enough linkage [10].

[10]However, the gap was substantially closed for high inlink pages.

[11]All values in Figure 5 were generated with the distance-based term weighting scheme to be described in the next section.

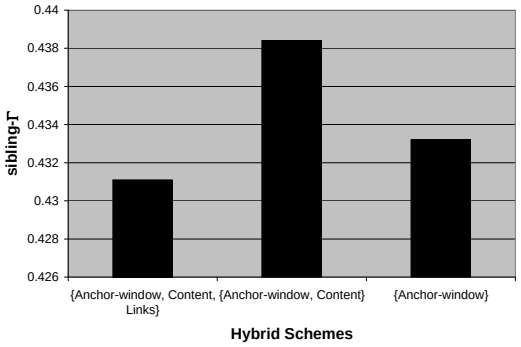

Figure 5: Hybrid Bags

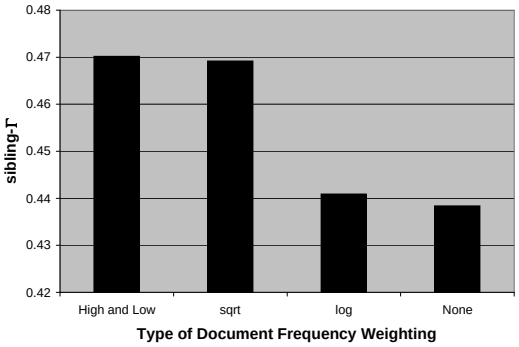

Figure 7: Frequency Based Term Weighting

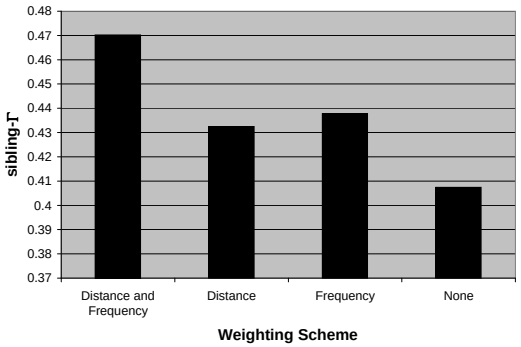

Figure 6: Distance and Frequency Weighting

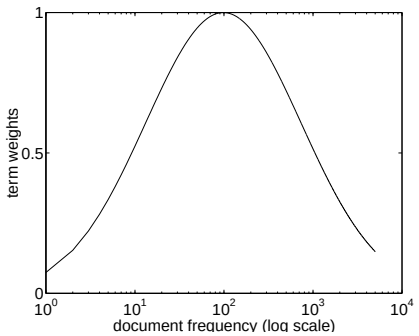

Figure 8: NMDF Term Weighting

links the anchor-window-based terms will dominate. In this way, the document's bag will automatically depend on as much information as is available.

4.2 Results: Term Weighting

In the previous section, we saw that the anchor-based approach with large windows performs the best. Our initial intuition, however, that smaller windows would provide a more concise representation is not completely without merit. In fact, we can improve performance substantially under our evaluation criteria by weighting terms based on their distance from the anchor. We prevent ourselves from falling into the trap of making similar documents appear orthogonal (small windows), while at the same time, not giving spurious terms too much weight (large windows). Figure 6 shows the results when term weights are scaled by $log_2(\frac{32}{1+distance(t,A_{vu})})$.

The results for frequency based weighting, shown in Figure 7, suggest that attenuating terms with low document frequency, in addition to attenuating terms with high document frequency (as is usually done), can increase performance. Let tf be a term's frequency in the bag, and df be the term's overall document frequency. Then in Figure 7, *log* refers to weighting with $\frac{tf}{1+log_2(df)}$. *sqrt* refers to weighting with $\frac{tf}{\sqrt{df}}$. *High and Low* refers to weighting with the log-scale gaussian $tf \times e^{-.5(\frac{log(df)-\mu}{\sigma})^2}$ (see Figure 8).

4.3 Results: Stemming

We now investigate the effects of our three stemming approaches. Figure 9 shows the sibling-Γ values for the NOSTEM, STOPSTEM, and STEM parameter settings. We see that STOPSTEM improves the Γ value, and that STEM provides an additional (although much less statistically significant[12]) improvement. As mentioned in Section 2.2, the effect of STOPSTEM over NOSTEM is to increase the effective reach of the stopword list. Words that are not themselves detected as stopwords, yet share a stem with another word that was detected as a stopword, will be removed. Although this technique can lead to false drops, our results show it is beneficial overall. The small additional impact of STEM over STOPSTEM is due to collapsing word variants into a single term.

5. SCALING TO LARGE REPOSITORIES

We assume that we have selected the parameters that maximize the quality of our similarity measure as explained in Section 3. Moreover, we call two documents α-*similar* if the Jaccard coefficient of their bags is greater than α.

PROBLEM 1. (SIMILARDOCUMENT **(efficiency considerations)**)*: Preprocess a repository of the web $\mathcal{W}$ so that for each query web document q in $\mathcal{W}$ all web documents in $\mathcal{W}$ that are α-similar to q can be found efficiently.*

In this section, we develop a scalable algorithm, called FINDALLSIMILAR to solve the above problem for a realistic web

[12]The NOSTEM-STOPSTEM and STEM-STOPSTEM average differences are of the same approximate magnitude, however the pairwise variance of the STEM-STOPTEM is extremely high in comparison to the other pairwise variances.

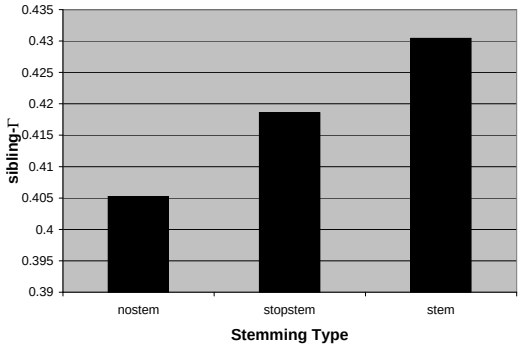

Figure 9: Stemming Variants

repository size.

One observation about Problem 1 is that all queries are known in advance; there are as many queries as web documents in the web repository $\mathcal{W}$. Furthermore, let k be the average size of a query result. It is quite reasonable to assume that k is small for the desired ranges of α; for most documents there is no reason to return more than a few hundred similar documents. Therefore, an efficient way to answer queries for Problem 1 is to *precompute* the answer for all possible queries. The space required to store all possible answers would be $O(kn)$, and since we only need to store document IDs, the space would be comparable to (or even smaller than) the size of the repository.

To avoid the quadratic blow-up in computing all pairwise document similarities, which is prohibitive even for an off-line process, we use scalable algorithms based on *Locality-Sensitive Hashing (LSH)* [15]. The basic idea is to hash the web documents in a way such that documents that are similar have a much higher probability of collision than documents which are very different.

The input to the FindAllSimilar algorithm is a repository of the web. The output is a sorted representation of what we call a *Web Document Similarity Graph (WDSG)*. A WDSG is simply a graph whose nodes correspond to web documents, and whose edges correspond to pairs of web documents that are α-similar. Furthermore, each edge is labeled by the similarity of its two end-point documents. Note that there is no edge between two documents whenever their similarity is less than α.

Now consider all edges of the WDSG where each edge (u, v, s_{uv}) is replaced with both of its *directed* counterparts (u, v, s_{uv}) and (v, u, s_{uv}). Sorting of the WDSG is defined by sorting the directed edges of the graph on their first node, and then by descending order on their similarity.

Notice that the sorted order of a WDSG indeed provides the answer to all queries of Problem 1. To get all documents that are α-similar to a query document q, we simply find the first time that q appears as the first node in the sorted WDSG and then we report all documents d in the run (q, d, s_{qd}).

A schematic view of the FindAllSimilar algorithm is shown in Figure 10. In the next two sections, we explain FindAllSimilar as a two stage algorithm. In the first stage we generate bags for each web document in the repository. In the second stage, the heart of the algorithm, we apply the LSH technique on the generated bags in order to produce the

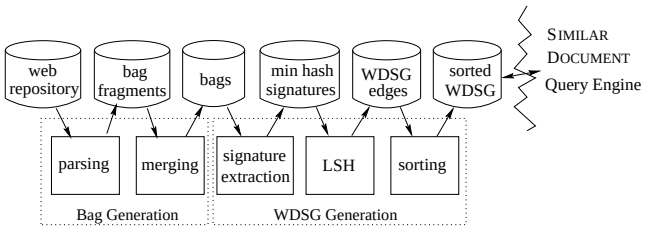

Figure 10: Schematic view of our approach.

WDSG. Due to space limitations we only give an overview of our algorithm: details can be found in [13].

5.1 Bag Generation

As we explained in the previous sections, the bag of each document contains words (i) from the content text of the document and (ii) from anchor windows of other documents that point to it. Our bag generation algorithm scans through the web repository and produces *bag fragments* for each document. For each document there is at most one *content bag fragment* and possibly many *anchor bag fragments*. After all bag fragments are generated, we sort and collapse them to form bags for the urls, apply our NMDF scaling as discussed in Section 2.3, and finally normalize the frequencies to sum to constant.

5.2 Generation of the Web Document Similarity Graph

For the description of the WDSG generation algorithm we assume that each document is represented by a bag of words $B = \{(w_1, f_1), \ldots, (w_k, f_k)\}$, where w are the words found in the content and anchor text of the document, and f are the corresponding normalized frequencies (after scaling with the NMDF function).

For the Jaccard coefficient similarity measure sim_J there exists a family $\mathcal{H}$ of hash functions (see [6]) such that for each pair of documents u, v we have $Pr[h(u) = h(v)] = sim_J(u, v)$, where the hash function h is chosen at random from the family $\mathcal{H}$. Notice that we need all frequencies in a bag to sum to a constant, since we would like two bags with exactly the same terms and corresponding frequencies that differ by a scalar factor to have similarity 1.

Based on the above property, we can compute for each bag a *Min Hash signature (MH-signature)* such that same value of the MH-signature of two documents indicates similar documents. However, the method is probabilistic and therefore both false positives and false negatives are likely to occur. According to the LSH scheme, we generate m MH-signatures for each document, and compute an *LSH-signature* by concatenating k of these MH-signatures. Then we generate candidate WDSG edges such that the number of false positives and false negatives is very small. Again, the complete details can be found in [13].

To reduce the number of false positives we filter the candidate pairs produced by LSH using the MH-signatures. Notice that if m is large enough, the fraction of the MH-signatures that two documents agree is an unbias estimate for their similarity. So, during filtering a pair (u, v) is kept only if the documents u and v agree on at least an α fraction of their MH-signatures. Finally, the filtered WDSG edges are sorted (on disk) yielding an ordering of the WDSG. The

Algorithm step	Time
Generation of bag fragments	17 hours
Merging of anchor-bag fragments	8 hours
Sorting of content bags	4 hours
MH-signature generation	29 hours
Generation of cand. WDSG edges	2 hours
Filtering WDSG edges	24 hours
Sorting of filtered WDSG edges	28 hours

Figure 11: Timing results

Type of data	Space
Anchor-bag fragments	14 GB
Anchor-bag fragments after merging	5.5 GB
Content bags	8 GB
MH-signatures	1.5 GB
Sorted WDSG edges	5.2 GB

Figure 12: Space usage

web document IDs can then be indexed for use by the SIMILARDOCUMENT query engine.

6. EXPERIMENTAL RESULTS

We employed the strategies that produced the best Γ values (see Section 4) and the scalable algorithm we described above (see Section 5) to run an experiment on a sizable web repository. In particular we used size-32 anchor windows with distance and frequency term weighting, stemming, and with content terms included. We provide a description of our dataset and the behavior of our algorithms, as well as a few examples from the results we obtained.

6.1 Efficiency Results

For our experiments, we used a 10 million page subset of our web repository. These were the first 10 million pages in the crawl for which we had both anchor-bag fragments and content bags. After merging all anchor-bag fragments and content bags, we generated 80 MH-signatures ($m = 80$) each 3 bytes long for each of the 10 million document bags. We set the threshold parameter α to be 20%, because we observed that for the chosen bag parameters many documents in the same class of Open Directory have pairwise similarities of roughly 20%.

All experiments took place on a Pentium III 700 MHz with 1 GB of memory. The timing results and space requirements of the various stages are given in Figures 11 and 12, resp.

6.2 Quality of Retrieved Documents

Evaluating the results of retrieved web documents is a nontrivial task. We have demonstrated that human-constructed directories are very useful for tuning the parameters of our algorithm. Such directories index only a very small subset of the existing web pages, however, and cannot be used to judge *final* quality. Comparisons with existing search engines are difficult, since one needs to make sure both systems use the same web page collection. Thus far, the evaluation of our final results has been subjective. By inspecting a large number of queries we found that most results are of high quality. Some anecdotal evidence of this claim is shown in

```
Query: www.cnnfn.com            Query: www.cnet.com        Query: www.mp3.com

0.562 www.pathfinder.com/money 0.312 www.pcworld.com      0.325 www.audiofind.com
0.525 www.barrons.com          0.312 www.zdnet.com/pcmag  0.325 www.rioport.com
0.500 quote.yahoo.com          0.263 www.pcmag.com        0.325 www.audiogalaxy.com
0.487 bis.dowjones.com         0.225 www.news.com         0.325 www.real.com
0.475 www.quote.com            0.200 www.hotwired.com     0.312 mp3.com
```

Figure 13: For each query we report the 5 most similar documents and the estimated similarities.

Figure 13.

7. RELATED WORK

Most relevant to our work are algorithms for the "Related Pages" functionality provided by several major search engines. Unfortunately, the details of these algorithms are not publicly available. Dean et al [10] propose algorithms, which we discussed in Sections 1 and 4.1, for finding related pages based on the connectivity of the web only and *not* on the text of pages. The idea of using hyperlink text for document representation has been exploited in the past to attack a variety of IR problems [1, 3, 7, 8, 9, 16]. The novelty of our paper, however, consists in the fact that we do not make any *a priori* assumption about what are the best features for document representation. Rather, we develop an evaluation methodology that allows us to select the best features from among a set of different candidates. Approaches algorithmically related to the ones presented in Section 5 have been used in [6, 4], although for the different problem of identifying mirror pages.

8. ACKNOWLEDGEMENTS

We would like to thank Professors Chris Manning and Jeff Ullman for their insights and invaluable feedback.

9. REFERENCES

[1] E. Amitay. Using Common Hypertext Links to Identify the Best Phrasal Description of Target Web Documents. *Proceedings of SIGIR'98 Post-Conference Workshop on Hypertext Information Retrieval for the Web*, 1998.

[2] G. Attardi, A. Gull, and F. Sebastiani. Theseus: Categorization by context. *Proceedings of WWW8*, 1999.

[3] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search engine. *Proceedings of WWW7*, 1998.

[4] A. Broder. Filtering Near-duplicate Documents. *Proceedings of FUN*, 1998.

[5] A. Broder. On the Resemblance and Containment of Documents. *In Compression and Complexity of Sequences*, 1998.

[6] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic Clustering of the Web. *Proceedings of WWW6*, 1997.

[7] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced Hypertext Categorization Using Hyperlinks. *Proceedings of SIGMOD*, 1998.

[8] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. *Proceedings of WWW7*, 1998.

[9] B. Davison. Topical Locality in the Web. *Proceedings of SIGIR*, 2000.

[10] J. Dean and M. Henzinger. Finding Related Pages in the World Wide Web. *Proceedings of WWW8*, 1999.

[11] L. A. Goodman and W. H. Kruskal. Measures of association for cross classifications. *J. of Amer. Stat. Assoc.*, 49:732–764, 1954.

[12] Google. `http://www.google.com/`.

[13] T.H. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Similarity Search on the Web: Evaluation and Scalability Considerations. *Extended Technical Report*, 2000.

[14] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. WebBase: A Repository of Web Pages. *Proceedings of International World Wide Web Conference*, 2000.

[15] P. Indyk and R. Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. *STOC*, 1998.

[16] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Proceedings of SODA*, 1998.

[17] H. P. Luhn. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, 2:159–165, 1958.

[18] Open Directory Project. `http://www.dmoz.com/`.

[19] M. Porter. An Algorithm for Suffix Stripping. *Program: Automated Library and Information Systems*, 14(3):130–137, 1980.

[20] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[21] S. Siegel and N. J. Castellan. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 1988.