Evaluating Strategies for Similarity Search on the Web*

Taher H. Haveliwala†
Stanford University
Computer Science Department
Stanford, CA 94305-9040
taherh@cs.stanford.edu
(650) 723-9273

Aristides Gionis‡
Stanford University
Computer Science Department
Stanford, CA 94305-9045
gionis@cs.stanford.edu
(650) 723-4532

Dan Klein§
Stanford University
Computer Science Department
Stanford, CA 94305-9040
klein@cs.stanford.edu
(650) 725-6965

Piotr Indyk
MIT
Laboratory of Computer Science
Cambridge, MA 02139-3594
indyk@theory.lcs.mit.edu
(617) 452-3402

ABSTRACT

Finding pages on the Web that are similar to a query page (Related Pages) is an important component of modern search engines. A variety of strategies have been proposed for answering Related Pages queries, but comparative evaluation by user studies is expensive, especially when large strategy spaces must be searched (e.g., when tuning parameters). We present a technique for automatically evaluating strategies using Web hierarchies, such as Open Directory, in place of user feedback. We apply this evaluation methodology to a mix of document representation strategies, including the use of text, anchor-text, and links. We discuss the relative advantages and disadvantages of the various approaches examined. Finally, we describe how to efficiently construct a similarity index out of our chosen strategies, and provide sample results from our index.

Keywords

related pages, similarity search, search, evaluation, Open Directory Project

1. INTRODUCTION

The goal of Web-page similarity search is to allow users to find Web pages similar to a query page [12]. In particular, given a query document, a similarity-search algorithm should provide a ranked listing of documents similar to that document.

Given a small number of similarity-search strategies, one might imagine comparing their relative quality with user feedback. However, user studies can have significant cost in both time and resources. Moreover, if, instead of comparing a small number of options, we are interested in comparing parametrized methods with large parameter spaces, the number of strategies can quickly exceed what can be evaluated using user studies. In this situation, it is extremely desirable to automate strategy comparisons and parameter selection.

The "best" parameters are those that result in the most accurate ranked similarity listings for arbitrary query documents. In this paper, we develop an automated evaluation methodology to determine the optimal document representation strategy. In particular, we view manually constructed directories such as Yahoo! [26] and the Open Directory Project (ODP) [21] as a kind of precompiled user study. Our evaluation methodology uses the notion of document similarity that is implicitly encoded in these hierarchical directories to induce "correct", ground truth orderings of documents by similarity, given some query document. Then, using a statistical measure ([13]), we compare similarity rankings obtained from different parameter settings of our algorithm to the correct rankings. Our underlying assumption is that parameter settings that 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 (including choices for document representation and term weighting schemes) and determined which of them is most effective for similarity search on the Web.

There are many possible ways to represent a document for the purpose of supporting effective similarity search. The following briefly describes the representation axes we considered for use with the evaluation methodology just described.

Three approaches to selecting the terms to include in the vector (or equivalently, multiset) representing a Web page u follow:

1. Words appearing in u (a content-based approach)
2. Document identifiers (e.g. urls) for each document v that links to u (a link-based approach)
3. Words appearing inside or near an anchor in v, when the anchor links to u (an 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 [12], 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 that have not yet been cited suffi-

*This work was supported by the National Science Foundation under Grant IIS-0085896
†Supported by an NSF Graduate Research Fellowship
‡Supported by NSF Grant IIS-0118173 and a Microsoft Research Graduate Fellowship
§Supported by an NSF Graduate Research Fellowship

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

ciently. As we will see in Section 5, 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* [9], appears most useful for the Web similarity-search task. Indeed, the use of anchor-windows has been previously considered for a variety of other Web IR tasks [2, 1, 9, 11]. 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 that 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 [14]. 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, such as term weighting and width of anchor-windows, which we discuss further in Section 3.

Note that there are many additional parameters that could be considered, such as weighting schemes for font sizes, font types, titles, etc. Our goal was not to search the parameter space exhaustively. Rather, we chose a reasonable set of parameters to present our evaluation methodology and to obtain insight into the qualitative effects of these basic parameters.

Once the best parameters, including choice of document representation and term weighting schemes, have been determined using the evaluation methodology, we must scale the similarity measure to build a similarity index for the Web as a whole. We develop an indexing approach relying on the Min-hashing technique [10, 5] and construct a similarity-search index for roughly 75 million urls to demonstrate the scalability of our approach. Because each stage of our algorithm is trivially parallelizable, our indexing approach can scale to the few billion accessible documents currently on the Web.[1]

2. EVALUATION METHODOLOGY

The quality of the rankings returned by our system is determined by the similarity metric and document features used. Previous work [12] 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 instead use an automated method of evaluation that uses the orderings implicit in human-built hierarchical directories to improve the quality of our system's rankings.

In the clustering literature, numerous methods of automatic evaluation have been proposed [17]. Steinback et al. [25] divide these methods into two broad classes. *Internal quality measures*, such as average pairwise document similarity, indicate the quality of a proposed cluster set based purely on the internal cluster geometry and statistics, without reference to any ground truth. *External quality measures*, such as entropy measures, test the accordance of a cluster set with a ground truth. As we are primarily investigating various feature selection methods and similarity metrics themselves in our work, we restrict our attention to external measures.

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

2.1 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.

PROBLEM 1. SIMILARDOCUMENT **(notion of similarity)**: *Formalize the notion of similarity between Web documents using an external quality measure.*

There is a great deal of ordering information implicit in the hierarchical Web 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 cases 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.[2]

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. We name these distances as follows:

[1]Commercial search engines generally have several hundreds or even thousands of machines at their disposal.

[2]We treated the hierarchy as a tree, ignoring the "soft-links" denoted with an "@" suffix

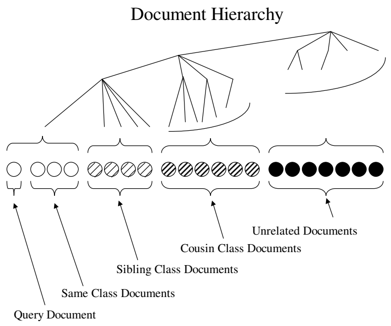

Figure 1: Mapping a hierarchy onto a partial ordering, given a source document.

Distance 0: *Same* – Documents are in the same class.

Distance 1: *Siblings* – Documents are in sibling classes.

Distance 2: *Cousins* – Documents are in classes which are first cousins.

Distance 3: *Unrelated* – The lowest common ancestor of the documents classes is the root.

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

> *On average, the true similarity of documents to a source document decreases monotonically with the familial distance from that document.*

Given this principle, and our definition of familial distance, for any source document in a hierarchical directory we can derive a partial ordering of all other documents in the directory. Note that we do not give any numerical interpretation to these familial distance values. We only depend on the above stated monotonicity principle: a source document is on average more similar to a same-class document than to a sibling-class document, and is on average more similar to a sibling-class document than a cousin-class document, and so on.

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

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 that are much less similar. The very notion of a correct total similarity ordering is somewhat suspect, as beyond a certain point, pages are simply unrelated. Our familial ordering makes 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[3] that accords better with familial ordering will be better than one that accords less closely.

[3] Of course the ranking system cannot make use of the directory itself for this statement to hold.

2.2 Comparing Orderings

At this point, we have derived a partial ordering from a given hierarchical directory and query (source) document s, that belongs in the hierarchy. We then wish to use this partial ordering to evaluate the correctness of an (almost) total ordering produced by our system.[4] Perhaps the most common method of comparing two rankings is the Spearman rank correlation coefficient. This measure is best suited to comparing rankings with few or no ties, and its value corresponds to a Pearson ρ coefficient [24]. 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 (namely the ground truth ranking), 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 measure directly how many of the "important" ranking choices are being made correctly. Given these goals, a more natural measure is the Kruskal-Goodman Γ [13].

DEFINITION 3. *For orderings* $\prec_a$ *and* $\prec_b$, $\Gamma(\prec_a,\prec_b)$ *is* $2\times Pr[\prec_a,\prec_b$ *agree on* $(x,y)\ |\ \prec_a,\prec_b$ *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 only at 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.

2.3 Regions of the Orderings

Thus, given a directory, a query document s, and a similarity measure *sim*, we can construct two orderings (over documents in the directory): 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 value 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 as follows. We iterate s over 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.

[4] 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	`http://www.aabga.org`
Source title	`American Assoc. of Botanical Gardens and Arboreta`
Source category	`/home/gardens/clubs_and_associations`

Settings: window size = 32, stem, dist and term weighting $\Gamma = \mathbf{0.53}$		
Rank	Sim	Category
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`

Settings: window size = 0, no stem, no term weighting $\Gamma = \mathbf{0.30}$		
Rank	Sim	Category
1	0.17	`/reference/libraries/independent_libraries`
2	0.15	`/home/gardens/clubs_and_associations`
5	0.14	`business/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).

***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 being unusually low in relation to sibling-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 were too sparse, and documents that were actually similar appeared to be orthogonal).

In Figure 2, we show an example that reflects our assumption that larger values of the Γ statistic correspond to parameter settings that yield better results.

3. DOCUMENT REPRESENTATION

In this section we will discuss the specific document representation and term weighting options we chose to evaluate using the technique outlined above. Let the Web document u be 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. It now remains to discuss *which* words should be placed in a document's bag, and with what weight.

3.1 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 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} (the anchor linking from page v to page u) should be included in B_u. We experimented with three strategies for this decision. In all cases, the anchor-text itself of 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}[5]. Specifically, we use $W \in \{0, 4, 8, 16, 32\}$.

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. This technique resulted in very narrow windows that averaged close to only 3 words in either direction.

TOPICAL: We use a simple technique for guessing topic boundaries at which to bound the region that gets included. The primary features that trigger this bounding are heading beginnings, list ends, and table ends. 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.

3.2 Stemming Terms

We explored the effect of three different stemming 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 [22] 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 variant 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.

3.3 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 amplification is in fact good for ad-hoc queries, where a rare

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

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 nontopical terms that adversely affect the similarity measure. Therefore, we also experimented with nonmonotonic 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 [23, 20]. We call such schemes *nonmonotonic document frequency (NMDF)* functions.

Another component of term weighting that 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 5, the use of a distance-based attenuation function in conjunction with large anchor-windows significantly improves results under our evaluation measure.

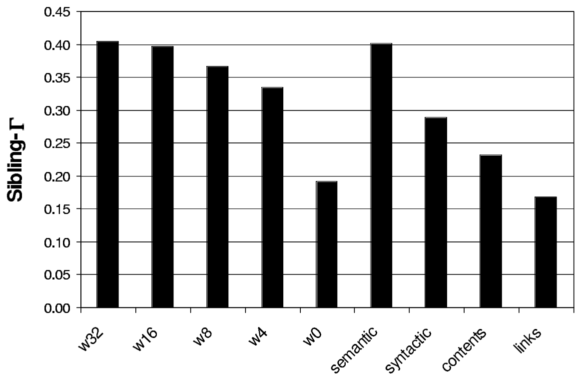

Figure 3: Document representations. Larger fixed anchor windows always gave better results, but topical dynamic windows acheived similar results with shorter average window size.

4. DOCUMENT SIMILARITY METRIC

The metric we use for measuring the similarity of document bags is the *Jaccard coefficient.* The Jaccard coefficient of two sets A and B is defined as

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

In the previous section we explained how we represent Web documents using bags (i.e. multisets). For the purposes of this paper we extend Jaccard from sets to bags by applying bag union and bag intersection. This is done by taking the max and min multiplicity of terms, for the union and intersection operations, respectively.

The reasons that we focus on the Jaccard measure rather than the classical *cosine* measure are mainly scalability considerations. For scaling our similarity-search technique to massive document datasets we rely on the *Min-Hashing* technique. The main idea here is to hash the Web documents such that the documents that are similar, according to our similarity measure, are mapped to the same bucket with a probability equal to the similarity between them. Creating such a hash function for the cosine measure is to our knowledge an open problem. On the other hand, creating such hashes is possible for the Jaccard measure (see [5]).

We used our evaluation methodology to verify that the Jaccard coefficient and the cosine measure yield comparable results.[6] Further evidence for the intuitive appeal of our measure is provided in [19], where the Jaccard coefficient outperforms all competitor measures for the task of defining similarities between words. Note that the bulk of the work presented here does not depend on whether Jaccard or cosine is used; only in Section 7 do we require the use of the Jaccard coefficient.

5. EXPERIMENTAL RESULTS OF PARAMETER EVALUATION

For evaluating the various strategies discussed in Section 3, we employ the methodology described in Section 2. We sampled Open Directory [21] 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 from the Stanford WebBase containing 42 million pages [15]. 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 3.

We verified that all three of our Γ measures yield, with very few exceptions, the same relative order of parameter settings. In a sense, this agreement is an indication of the robustness of our Γ measures. Here we report the results only for the sibling-Γ statistic. The graphs for the cousins-Γ and unrelated-Γ measures behave similarly.

For some of the graphs shown in this section the difference of Γ scores between different parameter settings might seem quite small, i.e. second decimal digit. Notice, however, that in each graph we explore the effect of a single "parameter dimension" independently, so when we add up the effect on all "parameter dimensions" the difference becomes substantial.

5.1 Results: Choosing Terms

Sibling-Γ values when bags are 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. Figure 4 shows the fraction of document pairs within the same Open Directory cluster that are *orthogonal* (i.e., no common words) un-

[6] We omit the description of these experiments as it is not the focus of our work.

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

[8] ODP pages themselves were of course excluded from the data set to avoid bias. Furthermore, the high orthogonality figures for the link-based approach, shown in Figure 4, show that partial ODP mirrors could not have had a significant impact on our results.

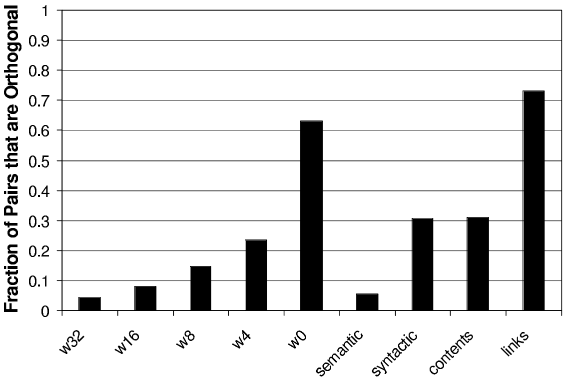

Figure 4: Intracluster Orthogonality for various anchor window types. Small windows and pure links resulted in document bags which were largely orthogonal, making similarity hard to determine.

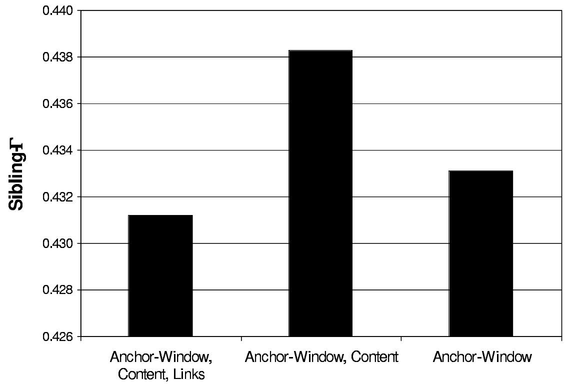

Figure 5: Hybrid bag types. Adding documents' own contents gave better results than anchor-windows alone, though adding link IDs lowered gamma.

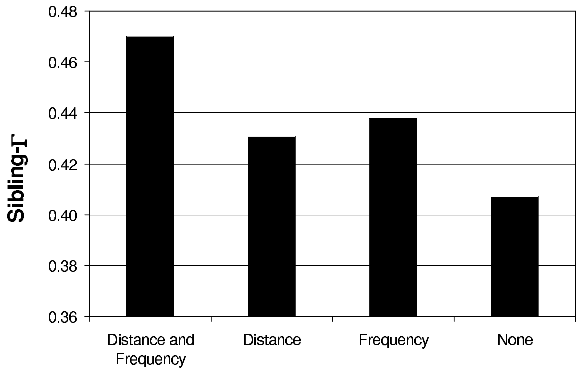

Figure 6: Term Weighting: Frequency and distance weighting improved results, and further improved results when combined.

der 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 reweighting 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. Note that the pure link based approach shown is very similar to the *Cocitation Algorithm* of [12].[10]

We also experimented with dynamically sized SYNTACTIC and TOPICAL windows, as described in Section 3. 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.[11]

In addition to varying window size, we can also choose to include terms of multiple types (anchor, content, or links, as described in Section 3) in our document representation. Figure 5 shows that by combining content and anchor-based bags, we can improve the sibling-Γ score[12]. 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 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.

5.2 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 $\log_2(\frac{32}{1+\text{distance}(t,A_{vu})})$.

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 doc-

[9] Using the SVD we could potentially glean some information in a pure link approach despite orthogonality, assuming enough linkage [12].
[10] Furthermore we verified that the *Cocitation Algorithm* as described in [12] yields similar Γ scores to the scores for the 'links' strategy shown above
[11] However, the gap was substantially closed for high inlink pages.
[12] All values in Figure 5 were generated with the distance-based term weighting scheme to be described.

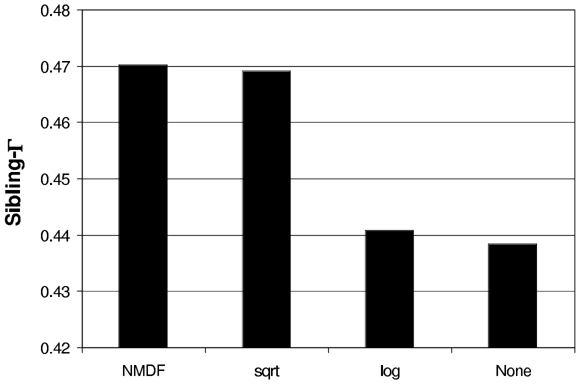

Figure 7: Types of Frequency weighting: sqrt gave the best results of the monotonic frequency weighting schemes; NMDF gave slightly better results.

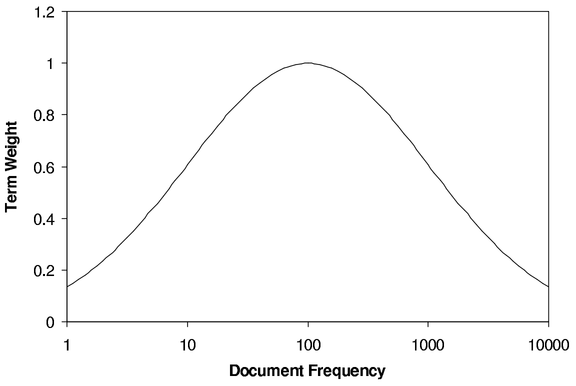

Figure 8: Non-monotonic document frequency (NMDF) weighting.

ument 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 $\frac{tf}{\sqrt{df}}$. *NMDF* refers to weighting with the log-scale gaussian $tf \times e^{-\frac{1}{2}(\frac{\log(df)-\mu}{\sigma})^2}$ (see Figure 8).

5.3 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[13]) improvement. As mentioned in Section 3.2, 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. The small additional impact of STEM over STOPSTEM is due to collapsing word variants into a single term.

[13]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.

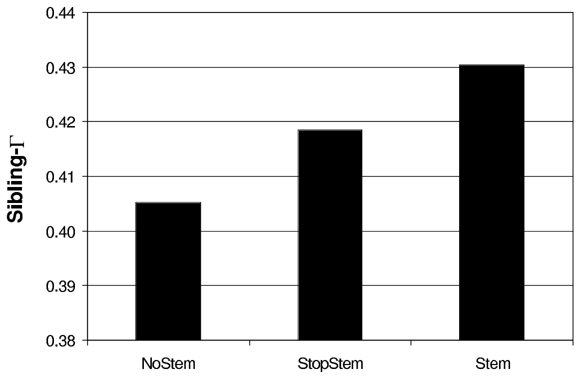

Figure 9: Stemming Variants: stemming gave the best results.

6. SCALING TO LARGE REPOSITORIES

We assume that we have selected the parameters that maximize the quality of our similarity measure as explained in Section 2. We now discuss how to efficiently find similar documents from the Web as a whole.

DEFINITION 4. *Two documents are* α-similar *if the Jaccard coefficient of their bags is greater than* α.

PROBLEM 2. 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 INDEXALLSIMILAR to solve the above problem for a realistic Web repository size.

In tackling Problem 2, there is a tradeoff between the work required during the preprocessing stage and the work required at query time to find the documents α-similar to q. We have explored two approaches. Note that since q is chosen from $\mathcal{W}$, all queries are known in advance. Using this property, we showed in previous work ([14]) how to efficiently precompute and store the answers for all possible queries. In this case, the preprocessing stage is compute-intensive, while the query processing is a trivial disk lookup. An alternative strategy, which we discuss in detail in this section, builds a specialized index during preprocessing, but delays the similarity computation until query time. As we will describe, the index is compact, and can be generated very efficiently, allowing us to scale to large repositories with modest hardware resources. Furthermore, the computation required at query time is reasonable.

A schematic view of the INDEXALLSIMILAR algorithm is shown in Figure 10. In the next two sections, we explain INDEXALLSIMILAR as a two stage algorithm. In the first stage we generate bags for each Web document in the repository. In the second stage, we generate a vector of signatures, known as Min-hash signatures, for each bag, and index them to allow efficient retrieval both of document ids given signatures, and the signatures given document ids.

6.1 Bag Generation

As we explained in the previous sections, the bag of each document contains words (i) from the content text of the

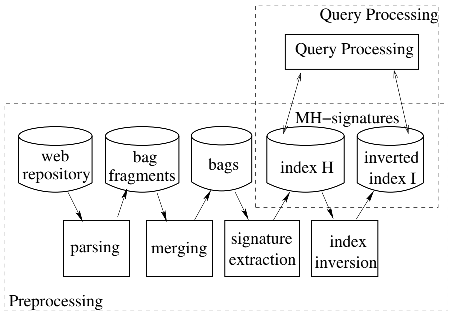

Figure 10: Schematic view of our approach.

Algorithm: ProcessQuery
Input: Query document q
Output: Similar documents
Let $mh_q = H[q]$ /* Fetch the MH-vector for q */
For each j from 1 to m /* Iterate over mh_q */
  /* For documents with the same j'th MH-signature as q */
  For each $doc_u \in I[j][mh_q[j]]$
    $sim[doc_u] ++$
Sort the set of docids $\{doc_i\}$ by their sim scores $sim[doc_i]$
Output $\{[doc_i, sim[doc_i]] | \frac{sim[doc_i]}{m} > \alpha\}$

Figure 11: Query Processing

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 3.3, and finally normalize the frequencies to sum to constant.

6.2 Generation of the Document Similarity Index

For the description of the Document Similarity Index (DSI) 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).

There exists a family $\mathcal{H}$ of hash functions (see [7]) 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 $\mathcal{H}$ and $sim_J(u, v)$ is the Jaccard similarity between the two documents' bags. The family $\mathcal{H}$ is defined by imposing a random order on the set of all words and then representing each url u by the lowest rank (according to that random order) element from B_u. In practice, it is quite inefficient to generate fully random permutation of all words. Therefore, Broder et al. [7] use a family of random linear functions of the form $h(x) = ax + b \mod p$. We use the same approach (see Broder et al. [6] and Indyk [16] for the theoretical background of this technique).

Based on the above property, we can compute for each bag a vector of *Min-hash signatures (MH-signatures)* such that the same value of the i-th MH-signature of two documents indicates similar documents. In particular, if we generate a vector mh_u of m MH-signatures for each document u, the expected fraction of the positions in which the two documents share the same MH-signatures is equal to the Jaccard similarity of the document bags.

We generate two data structures on disk. The first, H, consecutively stores mh_u for each document u (i.e., the m 4-byte MH-signatures for each document). Since our document ids are consecutively assigned, fetching these signatures for any document, given the document id, requires exactly 1 disk seek to the appropriate offset in H, followed by a sequential read of m 4-byte signatures. The second structure, I, is generated by inverting the first. For each position j in an MH-vector, and each MH-signature h that appears in position j in some MH-vector, $I[j][h]$ is a list containing id's for every document u such that the $mh_u[j] = h$. The algorithm for retrieving the ranked list of documents α-similar to the query document q, using the indexes H and I, is given in Figure 11.

When constructing the indexes H and I, the choice of m needed to ensure w.h.p. that documents that are α-similar to the query document are retrieved by ProcessQuery depends solely on α; in particular, it is shown in [7] that the choice of m is independent of the number of documents, as well as the size of the lexicon. Since we found in previous experiments that documents within an Open Directory category have similarity of at least 0.15, we chose $\alpha = 0.15$. We can safely choose $m = 80$ for this value of α [10].[14]

7. EXPERIMENTAL RESULTS

We employed the strategies that produced the best Γ values (see Section 5) in conjunction with the scalable algorithm we described above (see Section 6) 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.

7.1 Efficiency Results

The latest Stanford WebBase repository contains roughly 120 million pages, from a crawl performed in January 2001. For our large scale experiment, we used a 45 million page subset, which generated bags for 75 million urls. After merging all bag fragments, we generated 80 MH-signatures ($m = 80$) each 4 bytes long for each of the 75 million document bags.

Three machines, each AMD-K6 550MHz, were used to process the web repository in parallel to produce the bag fragments. The subsequent steps (merging of fragments, MH-signature generation, and query processing) took place on a dual Pentium-III 933 MHz with 2 GB of main memory. The timing results of the various stages and index sizes are given in figure 12. The query processing step is dominated by the cost of accessing I, the smaller of the on-disk indexes. To improve performance, we filtered I to remove urls of low indegree (3 or fewer inlinks). Note that these urls remain in H, so that all urls can appear as queries; some simply

[14] We chose α and m heuristically; the properties of the Web as a whole differ from those of Open Directory. Given additional resources, decreasing α and increasing m would be appropriate.

Algorithm step	Time
Generation of bag fragments	24 hours
Merging of anchor-bag fragments	8 hours
MH-signature generation	22 hours
Query Processing	< 3 seconds

Type of data	Space
Web repository (45M pages,compressed)	100 GB
Merged bags	42 GB
MH-signatures (H)	24 GB
Inverted MH-signatures (filtered) (I)	5 GB

Figure 12: Timing results and space usage

will not appear in *results*. Of course at a slight increase in query time (or given more resources), I need not be filtered in this way. Also note that if I is maintained wholly in main-memory (by partitioning it across several machines, for instance), the query processing time drops to a fraction of a second.

7.2 Quality of Retrieved Documents

Accurate comparisons with existing search engines are difficult, since one needs to make sure both systems use the same web document collection. We have found however, that the "Related Pages" functionality of commercial search engines often return *navigationally*, as opposed to *topically*, similar results. For instance, `www.msn.com` is by some criteria similar to `moneycentral.msn.com`. They are both part of Microsoft MSN; however the former would not be a very useful result for someone looking for other financial sites. We claim that the use of our evaluation methodology has led us to the use of strategies that reflect the notion of "similarity" embodied in the popular ODP directory. For illustration, we have provided some sample queries in figure 13. In figure 14 we have given the top 10 words (by weight) in the bags for these query urls.[15]

8. 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. [12] propose algorithms, which we discussed in Sections 1 and 5.1, 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, 8, 9, 11, 18]. 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 6 have been used in [7, 4], although for the different problem of identifying mirror pages.

9. ACKNOWLEDGMENTS

We would like to thank Professor Chris Manning, Professor Jeff Ullman, and Mayur Datar for their insights and invaluable feedback.

[15] For display, the terms were unstemmed with the most commonly occurring variant.

10. 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, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise Independent Permutations. *Proceedings of STOC*, 1998.

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

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

[9] 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.

[10] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang. Finding Interesting Associations without Support Pruning. *Proceedings of ICDE*, 2000.

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

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

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

[14] T.H. Haveliwala, A. Gionis, and P. Indyk. Scalable Techniques for Clustering the Web. *Informal Proceedings of International Workshop on the Web and Databases, WebDB*, 2000.

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

[16] P. Indyk. A Small Minwise Independent Family of Hash Functions. *Proceedings of SODA*, 1999.

[17] A.K. Jain, M. Narasimha Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3), 1999.

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

[19] L. Lee. Measures of Distributional Similarity. *Proceedings of ACL*, 1999.

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

[21] Open Directory Project (ODP). `http://www.dmoz.com/`.

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

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

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

[25] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *TextMining Workshop, KDD*, 2000.

[26] Yahoo! `http://www.yahoo.com/`.

MSN Money moneycentral.msn.com	Weather.com www.weather.com
MSN Money www.moneycentral.com	CNN.com - Weather www.cnn.com/WEATHER
Money Magazine www.pathfinder.com/money	Welcome to the Weather Underground www.princeton.edu/Webweather/ww.html
Welcome to Moneyextra www.moneyworld.co.uk	Rain or Shine www.rainorshine.com
Money www.money.com	UM Weather cirrus.sprl.umich.edu/wxnet
ETrade www.etrade.com	Weather for Active Lives www.intellicast.com/weather/
Money Club www.moneyclub.com	WeatherPost www.weatherpost.com
Morningstar - ... successful investing www.morningstar.net	Full-service weather company www.wni.com
The Money Page – ... Guide to Investment www.moneypage.com	Welcome to The Weather Underground www.wunderground.com
Reuters MoneyNet www.moneynet.com	Yahoo! Weather - Indianapolis Forecast weather.yahoo.com/forecast/...
MutualFunds www.mfmag.com	USAToday.com www.usatoday.com/weather

CNN Money www.cnnfn.com	MP3.com: free mp3 downloads... www.mp3.com
Financial markets, commodities, news www.bloomberg.com	International Music Network - About Us imnworld.com/about.html
Investors Business Daily www.investors.com	EMusic — World's Most Popular MP3 Service! www.emusic.com
Welcome to the new Barron's online www.barrons.com	CMJ: New Music First www.mp3now.com
Financial Times www.usa.ft.com	EMusic — World's Most Popular MP3 Service! www.goodnoise.com
CNN Money cnnfn.cnn.com	Lycos Music — Downloads mp3.lycos.com
CNBC on MSN Money Wizard www.cnbc.com	Audiogalaxy www.audiogalaxy.com
Financial Information Link Library www.mbnet.mb.ca/~russell	Listen www.listen.com
Wallstreet Journal Home Page update.wsj.com	LAUNCH.com - Discover New Music... www.launch.com
Money Magazine www.pathfinder.com/money	Nullsoft Winamp www.winamp.com
Quote.com www.quote.com	Welcome to Gracenote www.cddb.com

The Source for Java(TM) Technology java.sun.com	CD Now www.cdnow.com
The Source for Java(TM) Technology www.javasoft.com	CD Universe - Your Online Music Store www.cduniverse.com
developerWorks: Java technology www.ibm.com/java	The Orchard - ... music, artists, bands www.theorchard.com
The IT Industry Portal www.gamelan.com	Columbia House — Home Page www.columbiahouse.com
DevEdge Online - JavaScript Developer Central developer.netscape.com/tech/javascript/	Every CD www.everycd.com
Microsoft Visual J++ Home Page www.microsoft.com/visualj	CDconnection.com www.cdconnection.com
JavaScript World – Welcome! www.jsworld.com	Music Boulevard www.musicblvd.com
Java Boutique www.j-g.com/java	Music: CDs, records and tapes, oh my! www.gemm.com
JavaWorld.com www.javaworld.com	CD World cdworld.com
Sun Microsystems www.sun.ru	Broadcast Music, Inc. www.bmi.com
JavaLobby Homepage www.javalobby.org	MP3.com: free mp3 downloads... www.mp3.com

Figure 13: Sample queries and results

URL	Top Terms in Bag (Decreasing Order by Weight)
moneycentral.msn.com	money, finance, msn, website, moneycentral, stock, employment, microsoft, business, investor
www.weather.com	weather, channel, forecasts, fbc, enter, travel, seek, best, national, usa
www.cnnfn.com	finance, business, cnn, cnnfn, stock, market, street, money, wall, journal
www.mp3.com	music, audio, player, artist, napster, radio, band, million, century, song
java.sun.com	java, jdk, technology, microsystems, api, applet, spacer, platform, language, website
www.cdnow.com	music, cdnow, amazon, records, books, sports, best, entertainment, favorite, audio

Figure 14: Top 10 words from sample bags