Automatic Retrieval of Similar Content Using Search Engine Query Interface

Ali Dasdan, Paolo D'Alberto, Santanu Kolay, and Chris Drome
Yahoo! Inc
Sunnyvale, CA, USA
{dasdan,pdalbert,santanuk,cdrome}@yahoo-inc.com

ABSTRACT

We consider the coverage testing problem where we are given a document and a corpus with a limited query interface and asked to find if the corpus contains a near-duplicate of the document. This problem has applications in search engines for competitive coverage testing. To solve this problem, we propose approaches that work in three main steps: generate a query signature from the document, query the corpus using the query signature and scrape the returned results, and validate the similarity between the input document and the returned results. We discuss techniques to control and bound the performance of these methods. We perform large-scale experimental validation and show that these methods perform well across different search engine corpora and documents in multiple languages. They also are robust against performance parameter variations.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search Process

General Terms

Algorithms, Experimentation, Measurement

Keywords

Coverage, keyword extraction, random sampling, strong query, query generation

1. INTRODUCTION

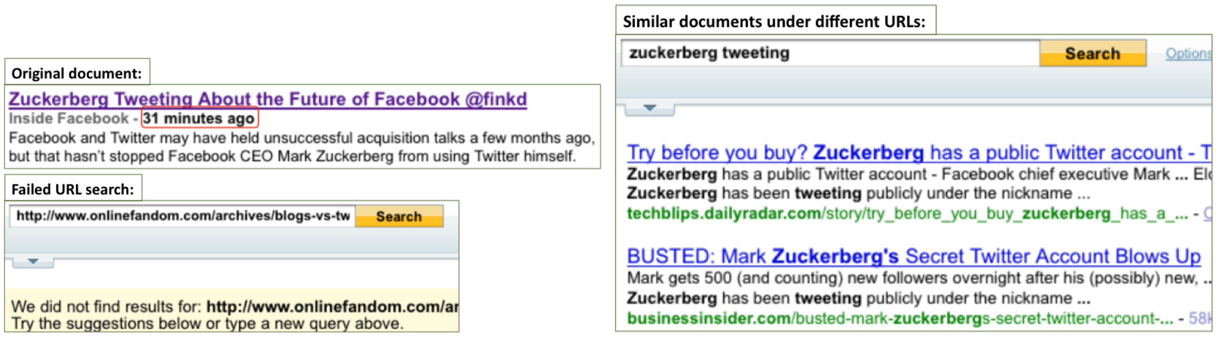
Consider the following problem that often occurs during competitive testing of search engines. A human judge is given a document and its URL and is asked to find out if a public search engine has the content. If she searches for the URL and the search engine fails to return the exact URL, then she is faced with two issues: (1) How can she surface the content through the standard query interface? and (2) Does any of the returned results contain similar content? For the first issue, she can query the search engine using queries generated from the content, e.g., using the title. For the second issue, she can validate the returned results by performing a side-by-side visual comparison. This is potentially an error-prone, tedious, and labor-intensive process. The situation gets more serious when the problem involves thousands of URLs. To rectify this situation, we propose an approach that automates query generation and content similarity validation, and removes subjectivity.

The problem above is an instance of the *coverage testing problem* [14] in which we are given a document like a web page and a corpus of documents with a standard public query interface like a search engine. The goal is to determine if the corpus contains a near-duplicate of the input document. This problem commonly occurs in practice when performing competitive coverage testing of search engines and validating the document indexing process. For example, when the coverage testing is performed over a large number of new URLs, the results provide a coverage metric on how much of the new content on the Web a target search engine acquires. By repeating this test often enough, the results can also provide a latency metric on how long it takes for the search engine to reach a certain threshold of coverage.

An illustration of the coverage testing problem is shown in Fig. 1. The original document is a fairly new blog entry about the CEO of Facebook using Twitter. A simple coverage test in this case is to see if a target search engine already has this content. As shown on the left-bottom part of this figure, this test fails. However, it is incorrect to conclude from this check alone that the search engine does not have the content at all. In fact, as shown on the right side, the search engine does have the content, actually under more than one URL, only two of which are shown. This example shows the importance of content check for coverage testing.

The complexity of the coverage testing problem is exacerbated by the size of the corpus, the number of input documents to test, time sensitivity, the heterogeneity of the corpus and documents in terms of language, encoding, structure, and variety of other aspects. As a result, the coverage testing problem is a difficult problem, and it subsumes interesting subproblems, as discussed next.

To solve the coverage testing problem for an input document (already retrieved) and a target search engine, the following template flow, which we refer to as the *strong query maker (SQM)*, can be used: (1) generating a query signa-


Original document:

Zuckerberg Tweeting About the Future of Facebook @finkd
Inside Facebook - 31 minutes ago
Facebook and Twitter may have held unsuccessful acquisition talks a few months ago, but that hasn't stopped Facebook CEO Mark Zuckerberg from using Twitter himself.

Failed URL search:

http://www.onlinefandom.com/archives/blogs-vs-tw Search

We did not find results for: http://www.onlinefandom.com/ar
Try the suggestions below or type a new query above.

Similar documents under different URLs:

zuckerberg tweeting Search Options

Try before you buy? Zuckerberg has a public Twitter account - T
Zuckerberg has a public Twitter account - Facebook chief executive Mark ... El
Zuckerberg has been tweeting publicly under the nickname ...
techblips.dailyradar.com/story/try_before_you_buy_zuckerberg_has_a_... -

BUSTED: Mark Zuckerberg's Secret Twitter Account Blows Up
Mark gets 500 (and counting) new followers overnight after his (possibly) new, ..
Zuckerberg has been tweeting publicly under the nickname ...
businessinsider.com/busted-mark-zuckerbergs-secret-twitter-account-... - 58k

Figure 1: The motivation for content check: The comparison of URL and content searchers for the "original document". For this document, the target search engine does not have its URL but does have its content under more than one URL. In other words, the URL search fails whereas the content search succeeds.

ture from the document, (2) querying the target search engine and scraping for the URLs in the search results page, (3) downloading the content of each scraped URL from the Web, and finally (4) verifying the similarity between the input document and each document downloaded. This template is similar to another one, independently proposed in [25] for the problem of catching plagiarism from web documents.

In this template flow, the last and first steps are interesting problems themselves and are the ones where various approaches have been proposed in the literature. The last step is equivalent to the document similarity detection problem; for its solution, multiple similarity measures or methods exist, e.g., see [6, 13] for the original proposals and [17, 21] for their large-scale experimental comparison. In this paper, this problem is out of our scope as we want to focus on the entire flow and its first step. For our similarity detection, we simply use the shingling approach of [6].

The first step of the template flow where a query from the input document needs to be generated is the most important as the chance of finding another document similar to the input document depends on the quality of the generated query. To emphasize this point just made, we refer to this step as the generation of a *query signature* from the input document, or the query generation problem.

Generating queries from an input document has been studied in many different contexts, which we review in detail in Section 2. For our work, we first adapted an approach proposed in [25] in which the least frequent k terms from the input document for some small k are selected as a query signature. Other approaches that involve term frequencies or functions derived from them are proposed in [3, 15, 20, 30] for different but related problems. We will refer to this approach of using the least frequent terms as *the least-frequent-terms (LFT) approach*.

The disadvantage of the LFT and similar frequency-based approaches is their dependence on a language-based lexicon. Without a privileged access to a search engine, creating and maintaining such a lexicon is almost impossible, e.g., see [4] for an example of the work involved. In our search for a simpler approach without this dependence, we were able to propose a new, randomized approach that can generate a query signature from an input document using the document content *alone*. In this approach, we simply select a random sequence of k terms from the input document as the query signature. In the actual implementation as will be discussed in Section 4.1, the selection is slightly more involved to guarantee repeatability, i.e., to generate the same query signature for the same input document, but the approach is very simple to implement. We will refer to this new approach as *the random-sequence-of-terms (RST) approach*.

Our large-scale experiments over two major search engine corpora and over 40+ markets with many different languages show that the RST approach outperforms the LFT approach. This conclusion was also validated by human judges. This is a really good result due to the practicality of the RST approach, i.e., being simpler and lexicon-independent. It seems that this difference in performance is due to two advantages of the RST approach: randomization and proximity-preserving. Since the value of randomization is well-known, e.g., see [23], we focus on the latter advantage. The RST approach preserves the proximity of the query terms by insisting on the selection of *consecutive* terms from the input document. In contrast, the LFT approach can select terms from all over the input document. To see the power of proximity-preserving, consider the following example: To get the CIKM2009 home page in the top ten results of each of the top three major search engines, it is enough to submit the query "The purpose of the conference is to identify" without the quotes even though the terms "purpose", "conference", and "identify" are fairly common, and the query does not contain terms strictly specific to CIKM2009 like "CIKM".

Overall, we make the following contributions in this paper:

- We propose a new approach to query signature generation from an input document. The new approach is simpler than the previous approaches in that it can work for documents in many languages and without the need for a lexicon or privileged access to search engine corpora. The new approach performs very well and is robust under different parameter settings. Our theoretical analysis of performance also validates our intuition about performance tradeoffs.

- We extend a well-known search quality metric, called the Discounted Cumulative Gain (DCG) [18], to measure the results of our comparison. The extension allows us to consider the effect of document ranking, resulting in a single number as the comparison result.

- We have performed experiments over thousands of documents from over forty markets with many languages. We have also created a lexicon of over 22 Million terms from a sample of 244 Million web pages, sifting through 140 Billion terms in the process. To the best of our knowledge, these experiments are the largest scale to date for the coverage testing and related problems. For example, a few thousand documents mostly in the Portuguese language were used in [25].

The rest of the paper is organized as follows. Section 2 introduces the prior work in the literature. Section 3 formalizes the coverage testing problem. Section 4 describes the template flow with all its steps discussed in detail. Section 5 proves bounds on the performance of the proposed flow. Section 6 discusses the implementation and experimental methodology. Section 7 details the experiments and results, showing evidence that the proposed flow and approaches work very well. Section 8, concludes this paper and suggests some future work.

2. RELATED WORK

We discuss the related work under the query generation and similarity detection groups.

On query generation. In the literature, according to Swanson [28], Luhn [19] is the first to propose a method of summarizing a document by creating an abstract for indexing and search [2, 20]. His abstracts contained full sentences extracted from the document, thereby taking advantage of term proximity. Furthermore, the sentences were selected based on their containing significant terms of the document. Here significant terms refer to those terms which had neither low nor high frequency in the term lexicon.

Thirty years later and independent of Lunh's paper, Bharat and Broder [3] proposed a similar frequency based method to determine the relative size and overlap of search engines using representative queries extracted from a lexicon (Yahoo! Directory). These queries were referred to as *strong queries* because they were assumed to be strong enough to uniquely represent the documents they were generated from. We also adopt this term for query signatures. For more recent work built upon this method, see [1, 7].

In both approaches in [3] and [19], a strong query from an input document is created by selecting terms whose frequency is neither too high nor too low. The high end is rejected to reduce the number of documents matching the strong query and the low end is rejected to eliminate spelling errors. Note that since we want to identify at least a single document similar to the input document, these approaches do not apply to the coverage problem directly as we can make the query stronger by moving towards the terms at the low end of the frequency spectrum.

Some other approaches for generating query signatures have been proposed in the literature. These approaches can be adapted to solve the coverage testing problem although they are originally proposed for other related problems.

Ghani et al. [15] propose approaches for the problem of creating a corpus for a minority language. Their approaches include selecting k tokens uniformly randomly, selecting the most frequent k tokens, and selecting the top k tokens using their tf-idf scores. Note that the tf-idf score of a term is a standard IR metric and corresponds to a combination of its frequency in a single document and over all documents in a corpus. For more information on the tf-idf scores, see [12] or other standard IR references.

Pereira and Ziviani [25] study the problem of finding documents from which a suspicious document may be plagiarized. Their approaches include selecting the most frequent k tokens and selecting equal number of tokens before and after an *anchor* token. Possible anchors are the most frequent tokens, the least frequent tokens, every ith token for some i, and the most unique tokens (using idf scores, over all the documents over the corpus).

Yang et al. [30] introduce the problem of cross-referencing blog content on the Web. Their main approach includes selecting the top k phrases where each phrase is scored by a linear combination of the tf-idf or mutual information based scores of its terms. They also discuss how to improve their results by using the hyperlinks in the Wikipedia corpus.

In many of these approaches, the ordering by token scores can be done deterministically or probabilistically [15]. In the latter case, the probability of selecting a token becomes proportional to its score. Moreover, ties can be broken by selecting longer tokens [25].

On document similarity detection. The previous approaches for similarity detection are studied under the problems that have been usually referred to as the near-duplicate detection or copy detection problems.

The first automatic detection mechanism for copy detection in large digital libraries is SCAM [5, 26, 27]. The idea is to represent a document as a probability distribution function (PDF); that is, a tuple of term and relative frequency. The similarity function is based on a comparison of the PDFs.

The PDF is a common representation for documents, however it has three drawbacks: it is not necessarily compact (i.e., it can be of the size of the original document), PDF comparisons have a complexity that is linear to the size of the PDF (and there are a quite few [24]), and finally, by construction the PDF does retain no information about the proximity of the sentences and terms.

Broder et al. [6, 8, 9, 10, 11] and Charikar [13] developed strong theoretical techniques for solving the near-duplicate detection problem. In effect, Broder et al. were able to reduce the duplicate detection problem to a set intersection problem, by using a sequence of tokens (shingles) to represent the content of a document. By contrast, Charikar used random projections to reduce the near-duplicate problem to one of determining the overlap of two high-dimensional vectors in the term space.

Recently, powerful estimates of the Jaccard Index have been proposed (e.g., [22, 29]) especially by searching a better sampling approach.

Henzinger [17] and Manku et al. [21] present experimental comparisons between Broder's and Charikar's approach, and concludes that Charikar's approach has usually better performance.

Our work in perspective. We build upon the developments laid down in the literature. For query generation, our randomized approach seems to be new. We also adapt one of the best previous approaches for comparison [25]. Note that although there have been many frequency-based approaches for query generation, the paper [25] seems to be the first for explicitly stating the use of the least frequent terms for query generation. For similarity detection, we use the shingling method of [6] simply because it performs well, we had a working implementation of it in our experimental environment, and we had to use the same method for a fair comparison of the query generation approaches.

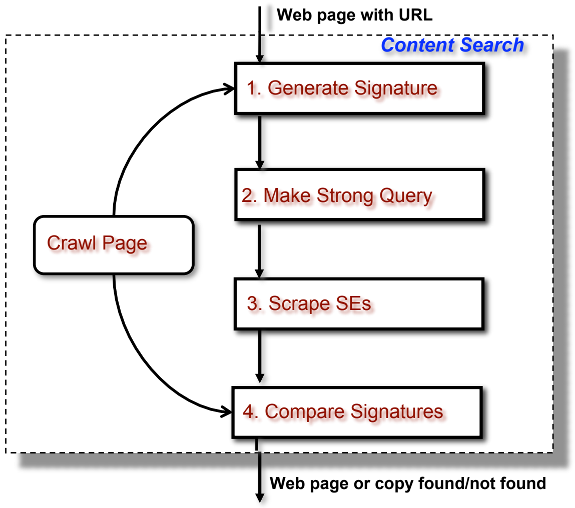


Figure 2: Strong Query Maker (SQM) flow. The output web page, if found, is a near-duplicate of the input page from one of the target SEs.

3. PROBLEM DEFINITION

We define the coverage testing problem as follows.

PROBLEM 1. *Consider a corpus D of documents with a public query interface. Given an input document d, the goal is to determine if there exists a document d' in D that is similar to d using a given similarity measure.*

Note that the corpora we target in this paper are search engine indexes. Their query interfaces are limited mainly in two ways: the number of terms in a query and the number of queries sent from the same IP address. As such, our secondary goal is to minimize both these numbers.

As for the similarity measures, the similarity $sim(d, d')$ between d and d' can be computed using any one of the similarity measures available in the literature. An example similarity measure is the Jaccard Index [12]; it measures similarity between two documents by the ratio of the set of terms common to both documents to the total set of terms in both documents. More formally, it is defined by

$$sim(d, d') = \frac{|d \bigcap d'|}{|d \bigcup d'|} \quad (1)$$

where the intersection and union operations are computed over the terms of the documents. The shingling method of [6] provides an unbiased estimator for this measure.

4. STRONG QUERY MAKER (SQM)

In this section, we shall describe our template flow in detail. We refer to the flow as well as its implementation in a tool as the *strong query maker (SQM)*, where the phrase "strong query" is adopted from [3].

The system and the context, where our strong query maker is used, is presented in Fig. 2. The system can be used to perform coverage testing of any search engine with a standard query interface and, basically, it can also be tailored to any proprietary platform. In this flow, the search engine that is queried and scraped is referred to as the target search engine.

The SQM tool we implemented extends the basic flow with capabilities to work in a production environment such as the ability to take in as input a large set of input web pages, the ability to query any search engine, and the ability to parallelize its work.

Summary. Consider D and d as in Section 3. Let R_q represent the set of documents that D returns upon receiving the query q. Also let Q represent the set of queries q generated from d. Then, SQM solves the coverage testing problem by searching for a document d' in $\bigcup_{q \in Q} R_q$ such that $sim(d, d') \geq \theta$, where θ is the similarity threshold.

Details. In terms of its interface, SQM takes in a document d, which is a web page in this paper, and produces either another document d' that is guaranteed to be a near-duplicate of d or a message saying that no near-duplicates of d can be found in the target search engine's index.

SQM performs two types of checks: a *URL search* and a *content search*. The URL search can happen only if the URL of d is available. Since the URL search is performed as a speed-up trick to reduce the number of cases to be handled by the content search, it can safely be ignored.

For the URL search, if the URL is found among the returned search results, then the operation is a success with the caveat that the target search engine may have a different version of d. As such, SQM validates if two versions of d are still near-duplicates and returns success or failure.

The content search is needed only if the URL of d is not found among the returned search results. Recall that this does not necessarily mean that the target search engine does not have the content of d because it may cover a near-duplicate of d under a different URL. As the number of duplicates on the Web is very large, this is a very likely scenario.

Now, SQM can perform a content search for d using its content. Of course, content search would be very easy if the query interface of search engines could take in the entire content of d; however, search engines can accept queries with the number of terms in the lower 10s and may even ignore most of the terms save a few during ranking. This is the reason for query generation from d.

Flow. The basic flow of SQM in Fig. 2 can be divided into five basic steps: retrieving the input document d from the Web, generating its query signatures, making strong queries, querying and scraping target search engines, and finally validating for similarity between d and each of the documents in the search results (after also retrieving these documents from the Web).

The first step involves fetching the input document from the Web using a crawler and processing it in a format that enables the following steps. To speed up this step, we may consult an existing production crawler, if this task is done inside a search engine, to see if it has already crawled the page. If the crawled version is too old, it may be a good idea to re-crawl the page using the dedicated crawler shown in Fig. 2.

We next analyze each step of the SQM flow in a separate subsection. In the sequel, for simplicity, we will assume that we are querying a single target search engine for d.

4.1 Generating Signatures

The goal of this step is to generate a textual signature of the input document. This step takes in d with its URL and produces a textual signature of d. Signatures, which can be numeric or textual, form the basis of strong query generation as well as near-duplicate checking.

Numeric signature. This step assumes that the document d has already been converted into a sequence of tokens where each token corresponds to a term in d. In the literature, there are two major approaches to produce a numeric signature out of this sequence, the shingling approach proposed by Broder et al. [11] and the random projection based approach proposed by Charikar [13]. In SQM, we used the shingling approach but we do not see any fundamental difficulty towards working with the other approach (or any other similarity detection measures).

If the token sequence of d contains $|d|$ tokens, the shingling approach produces a number, called a shingle, for every subsequence of k tokens using Rabin fingerprints [8], resulting in a total of $|d| - k + 1$ shingles. Then, the numeric signature with respect to this set of shingles can be the minimum m shingles. By repeating this process using multiple fingerprinting functions, the numeric signature can be made more robust. For simplicity, we will assume that a single fingerprinting function is used to produce a numeric signature with m shingles.

Textual signature. Query generation needs a textual signature of d because a search engine cannot be queried using shingles or any other numeric signatures unless we have privileged access to the index of the search engine.

As mentioned in Section 1, for query generation, we used the LFT approach from [25] and proposed a new approach called the RST approach. The reasons for selecting the LFT approach for comparison were threefold: (1) It is a recent proposal in the literature; (2) It increases query strength by focusing on the least common terms; and (3) Its implementation was relatively easier with our existing infrastructure.

The LFT approach. To be able to use this approach, we needed to have a representative lexicon of terms on the Web and a way to use it to generate terms from d for its textual signature. For that, we generated a dictionary of 22M terms together with their frequencies from a sample of 244M current documents, sifting through 140B terms in the process. We then sorted the terms of d in the increasing frequency order using the frequencies obtained from the dictionary. Finally, we generated the first k terms as the query signature of d from the least frequent side in the sorted order. For additional queries, we used the next sets of k terms.

The RST approach. For this proposed (RST) approach, a query signature can be generated by randomly selecting a set of k consecutive terms from d. Since repeatability is a requirement for ensuring debuggability in a production environment, we wanted to generate the same query from the same document over multiple runs. To manage this, we piggybacked on the repeatability of the numeric signatures. In other words, we generated a textual signature of d by mapping its numeric signature (shingles) back to the term subsequence it was generated from. Since shingling is a one-way hash, this is only possible by keeping track of a mapping between every subsequence of k terms and the shingle generated from it. Since the minimum operation is associative, the numeric signature can be updated incrementally so this mapping uses constant space. Even with multiple fingerprinting functions, this mapping can use space linear in the number of functions.

Comparison of two approaches. The LFT approach needs a representative dictionary whereas the RST approach does not; it works only with the terms in d. This requirement for the LFT approach is a disadvantage because representative dictionaries of the Web documents are not publicly available (except a version from Google for a set of documents from 2006 [4]), and even when one creates a version, then it is expensive to keep it up-to-date. Note that access to a dictionary is, however, an advantage as it can provide term and document frequencies (as raw frequencies or as tf-idf).

The RST approach takes advantage of the *proximity* of the terms in the textual signature whereas the LFT approach selects terms based on frequency order so ignores proximity completely. Since search engines are known to use the proximity feature in their ranking of documents, proximity-preservation is an advantage for the RST approach.

Using proximity may also backfire as the subsequences of terms from d may contain insignificant terms such as stopwords. However, search engines are also known to remove stopwords anyway so this is not a serious weakness. We also suggest as future work a hybrid approach that can actually combined the LFT and RST approaches.

Other approaches for query generation. Many other approaches for generating textual signatures have been proposed in the literature, e.g., see [15, 25, 30]. These approaches are mainly proposed to solve other problems but may be adapted to the coverage testing problem. It is an interesting future work to compare all these approaches together.

4.2 Making Strong Queries

In this step, we use the textual signatures to generate strong queries. Intuitively, a strong query q generated from d is a query that uniquely identifies d such that a search engine that has indexed d will return d when it is queried with q [3]. In general, the higher the rank of d in the returned results, the stronger the query q. In practice, the ideal is to have d in the top-10 returned results or in the first page of the search results.

In the previous section, we described a way to generate one textual signature from d. Because of the way they are generated, they are already strong. By repeating the process t times with different generated queries, we reduce the error of SQM.

In this step, we also need to make sure that the generated queries are in a form that can be submitted to search engines. For that, we need to tackle two important issues.

Tokenization. In English and some other languages, terms are separated by spaces and punctuation marks. In traditional Chinese, there is no use of separators and the organization in terms is highly contextual. Now imagine that the language of d is Chinese. The document processor that processes d uses one way of tokenizing the terms. The tokenization algorithms used by two search engines A and B may be different. This creates a problem when a query generated by A's algorithm is used to query B's index. Our solution to such problems is simple: keep the query with tokens as well as generate another query by concatenating the tokens without any separators. This doubles the number t of strong queries of d.

Stopwords. As mentioned before, the RST approach is susceptible to having stopwords such as *a, an, and, or* in the textual signatures. These terms are harmless as search engines will remove them from submitted queries anyway yet they consume precious space in the already short queries. We also wanted our results to be resilient against any errors in the stopword removal of our target search engines. In the end, we decided to simply remove the stopwords from the generated textual signatures for English.

4.3 Scraping Search Engines

The goal of this step is to query the public query interface of the target search engine with each query of d, scrape the r results out of the returned results (where $r = 10$ for the first page of results) and aggregate them into a results set.

The search engine may or may not return any results for one or more of the submitted queries. If the lack of results is due to an error, we repeated the query after a short delay. In the end, the process concludes with a final results set. If the set is empty, it indicates that the target search engine does not have a near-duplicate of d. This conclusion may be a false negative but if the target search engine is one of the major search engines, the rate of a false negative is small. Moreover, we can make it even smaller as discussed in Section 5.

4.4 Comparing Documents for Similarity

At this point, we have d and a results set of URLs returned from the target search engine in response to all the strong queries of d. The goal of this step is then to validate if d is covered in the results set.

We first download the web pages for each URL in the results set using our dedicated crawler. As a side effect of this downloading, we will also get numeric signatures for each page. Then, the problem reduces to the well-known near-duplicate detection problem in which we compare d against these pages and find out if any of the pages is a near-duplicate of d using the shingling method. If so, then SQM in Fig. 2 outputs "found" together with the URL and numeric signature of the found near-duplicate. If not, the output from SQM is "not found".

5. PERFORMANCE BOUNDS

We will show how the performance of SQM depends on various parameters, some of which we can control and the rest depend on the corpus. Define a document *relevant* if it is similar to the input document, and *non-relevant* otherwise.

False positive rate. The false positive rate FP measures the error of reporting "success" (or "found" for SQM) when none of the retrieved documents are relevant. Note that by construction, SQM computes the similarity between each retrieved document and the query document using the *sim* measure. Assume the retrieval of r non-relevant documents and assume that SQM reported that it had found a relevant document. This can happen only if *sim* has failed for at least one of the r retrieved documents, or the complement of the case where *sim* has performed correctly for every one of the r retrieved results. This argument results in the following result.

LEMMA 1. *The false positive rate of SQM for r retrieved results is*

$$FP(SQM) = 1 - (1 - FP(sim))^r \leq r \cdot FP(sim) \tag{2}$$

where $FP(sim)$ is the false positive rate of the sim measure.

In the above result, the upper bound is obtained via Bernoulli's inequality [16].

This result implies that the performance of SQM depends highly on the *sim* measure, which is desirable as SQM can benefit directly from better similarity measures. On the other hand, the false positive rate increases with the number of the retrieved results, hence, the need to minimize this number.

False negative rate. The false negative rate FN measures the error of reporting "no success" (or "not found" for SQM) when the corpus does contain relevant documents. Assume that the corpus has matched the query q with the set D_q of documents. Also assume that only c of these documents are relevant for the target document. Now, by construction, SQM will review only r documents out of D_q. When none of these r documents are relevant, we get the false negative rate of SQM for one iteration as

$$FN(SQM) = \prod_{j=0}^{r-1}(1 - \frac{c}{n-j}) \tag{3}$$

where $|D_q| = n$. By applying the exponential inequality [16] to each factor above leads to

$$FN(SQM) \leq exp(-c(H_n - H_{n-r})) \tag{4}$$

where H_n is the nth harmonic number and $exp(x) = e^x$ is the exponential function. Using the bounds on the harmonic numbers and assuming $n \gg r$, this inequality further reduces to

$$FN(SQM) \leq (1 - \frac{r}{n})^c \tag{5}$$

for one iteration. For multiple iterations, assume n_i gives the number of the documents matching the ith query. Then, we get

$$FN(SQM) \leq \prod_{i=1}^{i=t}(1 - \frac{k}{n_i})^c \tag{6}$$

for t iterations. Using the exponential inequality one more time, we get the following result.

LEMMA 2. *The false negative rate of SQM for r retrieved results and t iterations is bounded as*

$$FN(SQM) \leq exp(-kc\sum_{i=1}^{t}\frac{1}{n_i}), \tag{7}$$

which simplifies to

$$FN(SQM) \leq exp(-kct/n) \tag{8}$$

when $n_i = n$ for each i.

This result shows that the false negative rate of SQM can be decreased by acting on four parameters as: increase the number r of results scraped, increase the number t of iterations, decrease the number n of documents matching the query, and increase the number c of duplicates.

Among these parameters, changing c is out of our control; it is also the case that search engines try to reduce the number of duplicates in their corpora. Adjusting r and t are trivial. The main difficulty is with the parameter n, hence, the importance of and focus on generating strong queries for the target document.

Note that the goal of both the LFT and RST approaches is to generate as unique a query as possible so that n can be minimized. For multiple queries, the union of these n's needs to be minimized as well.

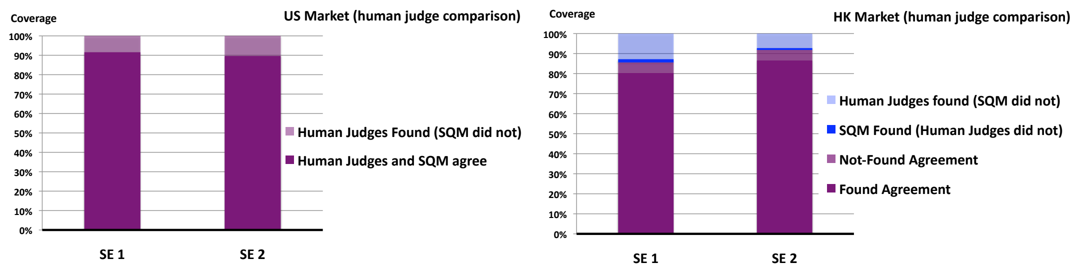


Figure 3: Comparison between SQM and human judges on two search engines using documents from each of the US market (10,000 documents on the left) and Hong Kong market (1,200 documents on the right).

6. EXPERIMENTAL METHODOLOGY

Next we explain how we set the parameters of the search space for our experiments to perform a thorough exploration of the search space and how we measured the quality of the scraped results using a new metric, adapted from the well-known Discounted Cumulative Gain (DCG) metric [18].

6.1 Parameters

For our experiments, we had to decide on the instantiation of many parameters. The parameters and our choices are listed below. For each of these parameters, we also performed sensitivity analyses to validate our value selection as well as conclusions. In the list below, we also report on some of these analyses.

- What are the target search engines? We used two major search engines through their standard public query interface. We refer to them as SE1 and SE2.
- How many strong queries to generate for d (the t parameter)? We suggest 10 strong queries for d after experiments with $t = 5, 10, 20$.
- How many top results to scrape from the results returned from the target search engines (the r parameter)? We suggest scraping of only the top-10 results after experiments with $r = 5, 10, 20, 50$. This is also convenient as the first page of search results from major search engines contains 10 results.
- How many terms to use per query (the k parameter)? We suggest 10 terms per query after experiments with $k = 5, 10, 15, 20$. Note that typical user queries contain 2-3 terms, so the use of 10 terms helps improve the uniqueness of the query.
- How many terms per shingle for similarity detection? We used 10 terms per shingle, more than suggested in [17]. The choice of 10 terms enables the one-to-one mapping of shingles to queries.
- How to factor in the rank of a found near-duplicate? We describe the answer in the next section.

6.2 Results Quality

Given a query for d, we can measure the quality of the scraped results by the combination of the document ranking and document matching. We adapt a measure commonly used for assessing the relevance of the search engine results pages to an input query: Discounted Cumulative Gain (**DCG**) [18], defined as

$$DCG = \sum_{i=1}^{k} \frac{2^{rel(URL_q, URL_i)}}{\log_2(1 + rank(URL_i))} \qquad (9)$$

where k is the number of URLs scraped, URL_q is the input URL from which the query q is generated, $rank(URL_i)$ is the rank of the URL_i in the returned search results, and the relevance measure rel is further defined numerically and qualitatively as follows.

$$rel(A, B) = \begin{cases} 1(bad) & \textbf{if } 0 \le sim(A,B) < 4 \\ 2(fair) & \textbf{if } 4 \le sim(A,B) < 6 \\ 3(good) & \textbf{if } 6 \le sim(A,B) < 8 \\ 4(excellent) & \textbf{if } 8 \le sim(A,B) < 10 \\ 5(perfect) & \textbf{if } sim(A,B) = 10 \end{cases} \qquad (10)$$

where $sim(A, B)$ is the Jaccard Index between the terms of two documents A and B, as defined in Section 3. Note that the rel measure gets larger proportional to the similarity between its arguments.

Over multiple queries for the input document, the DCG can be computed in two ways: by averaging over all queries or by selecting the best DCG over all queries. We give results for both in the sequel.

7. EXPERIMENTAL RESULTS

We divide this section in four parts: In Section 7.1, we compare our approach against the coverage testing performed by human judges; In Section 7.2, we compare our approach to find documents in SE1 and SE2 that are known to exist in both indexes; In Section 7.3, we compare both the LFT and RST approaches on the same set of documents; and finally, in Section 7.4, we report on the results of parameter sensitivity experiments.

7.1 Human Judges vs. SQM

In Fig. 3, we show a performance comparison between SQM with the RST approach and human judges for coverage testing. The tests were performed on both SE1 and SE2 using 10,000 documents from the US (United States) market and 1,200 documents from the HK (Hong Kong) markets.

In 90% of the time, SQM and human judges agree on the "found" and "not found" outcomes. For the US market, human judges claim to have found more documents than SQM did. For the HK market, the opposite occurred (SQM found documents that human judges did not). Regarding the discrepancy, note that human editors could check their judgment only visually whereas SQM can validate its judgments

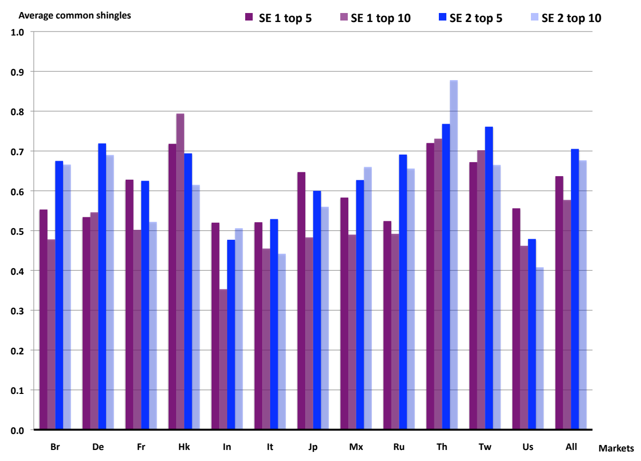


Figure 4: Coverage testing using SQM with the RST approach over a sample of 12 markets (the x-axis). The performance is measured in terms of 10 (the maximum value) minus the average number of common shingles. Both the top-5 and top-10 results were scraped. The lower the bar the better the performance.

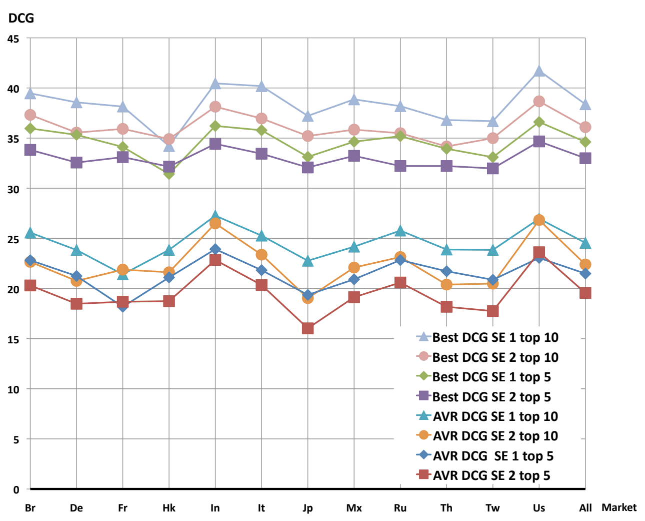


Figure 5: Best and average DCG measures of SQM with the RST approach over a sample of 12 markets (the x-axis). The top and bottom four graphs are for the best and average DCG, respectively.

provably and automatically using near-duplicate detection. Even if the judges were not convinced with this outcome, SQM could reduce their workload by at least 90%.

7.2 Coverage

In this section, we show that (1) SQM works consistently well across different markets, indicating its independence of the document language; (2) SQM works well with both SE1 and SE2; and (3) Its quality is not very sensitive to the number of results scraped per query. Here SQM refers to SQM with the RST approach.

In Fig. 4, we show the coverage testing results for a sample of 12 markets from different continents and with a mix of languages. The comparison metric is based on the average of the common shingles of the most relevant document per market. The maximum value is 10, and this figure shows the difference between 10 and the comparison metric. A lower bar means higher quality result.

The data set was obtained by using queries from the query logs to scrape SE1 and SE2 for a total of 2,000 URLs per market. Since the URLs were scraped from their returned results, they were *guaranteed* to exist in their indexes. We then downloaded the content for each of these URLs. After that, we ran SQM using the downloaded content as input against SE1 and SE2.

The results in Fig. 4 show that SQM performs very well independent of content language and the number of results scraped, which is either 5 or 10.

Effect of ranking. A query signature is stronger if it surfaces the target content in top ranks. In Fig. 5, we present a summary of the best DCG and average DCG over all queries for input document. Since we had many documents per market, each of these DCG values were further averaged over these documents to reduce them to a single number. The results in this figure shows that SQM is fairly robust across these different markets.

Effect of duplicates. As discussed in Section 5, the content duplicates in a corpus can affect the results. In Fig. 6, we show the effect of duplicates on the DCG measure by including or excluding them from the documents we use to compute the final DCG. We computed DCG over documents with grade excellent or perfect to focus on the most similar documents.

As this figure shows, duplicates affect SE2 and SE1 differently. For both search engines, the results improve with duplicates excluded but SE2 seems more robust against this effect.

7.3 The RST vs. LFT Approach

In this set of experiments, we compare the RST and LFT approaches on the same SQM implementation run over the same data set. We used the same parameters for both approaches: we used 10 queries of 10 terms each in the generated queries.

In Fig. 7, we show 10 minus the common shingles for the most relevant document, where 10 is the maximum number of common shingles possible. This figure indicates that the RST approach outperforms the LFT approach over all markets and search engines.

7.4 Parameter Sensitivity

In Fig. 8, we show the results of experiments performed to test the sensitivity of SQM with the RST approach against varying parameter combinations. The parameters were the number of terms per query, the number of queries per document, and the number of results scraped per document. The data comes from the HK market and the experiments were run on SE1.

For these experiments, both the URL and content searches were performed. The content search is what the RST approach does.

As one would expect, the URL search returns better results. The content search also performs well, coming within 12.5% of the URL search performance. Both searches seem to be almost insensitive to parameter variations, hence, the robustness of SQM and the RST approach. This figure also shows that the "10-10 rule" with 10 terms per query and 10

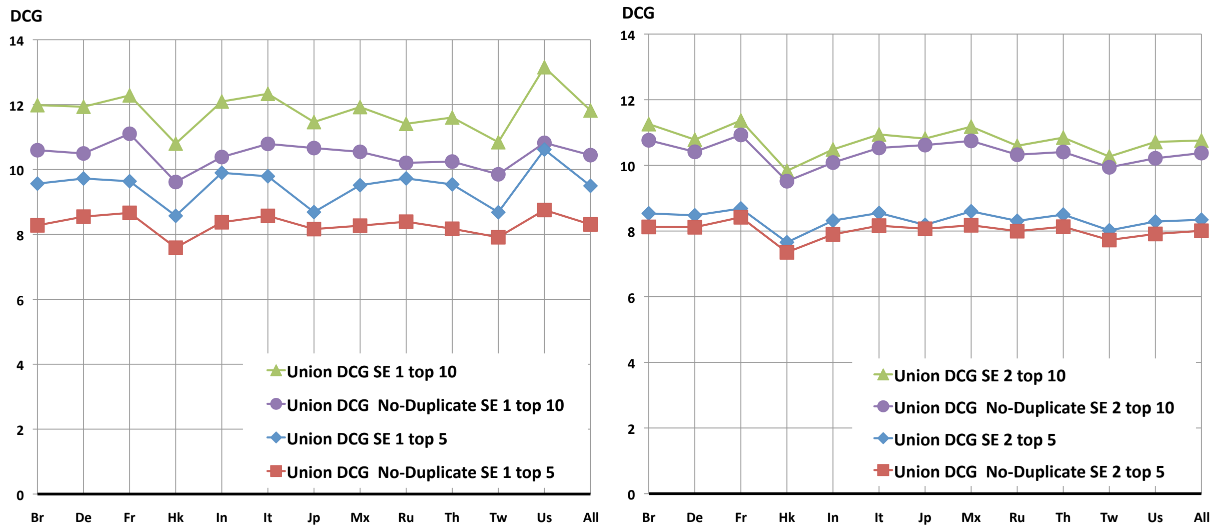


Figure 6: Effect of duplicates on DCG measure over a sample of 12 markets (the x-axes). Here SQM is with the RST approach. The left part is SE1 and the right part is SE2.

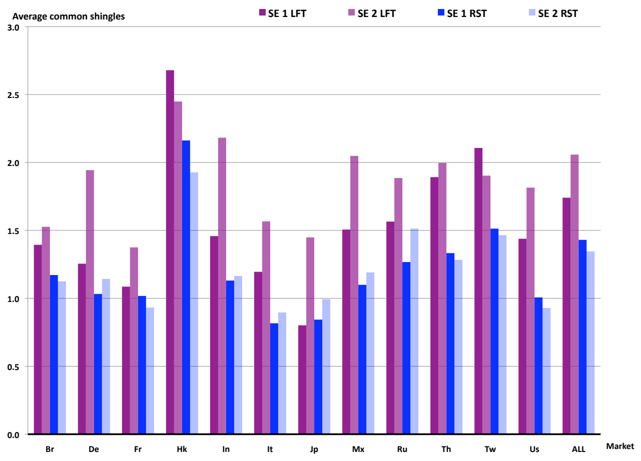


Figure 7: Comparison of the RST approach with the LFT approach on both search engines over a sample of 12 markets (the x-axis). The lower the bar the better the performance.

queries per document is a good setting as its performance is almost indistinguishable from the best performance.

8. CONCLUSIONS

In this paper, we addressed the coverage testing problem. For its solution, we adapted the LFT approach from the literature and proposed a new approach called the RST approach. We use each approach for the query generation step of a template flow and tool called the strong query maker (SQM). Our theoretical analysis provides guidance on the performance parameters. Our large-scale experimental validation over two major search engine corpora and documents in many different languages shows that the proposed RST approach outperformed human judges as well as the LFT approach. It also shows that the proposed approach is largely insensitive to the parameter variations.

As future work, it is possible to bring the strengths of these two approaches to a hybrid approach in which both

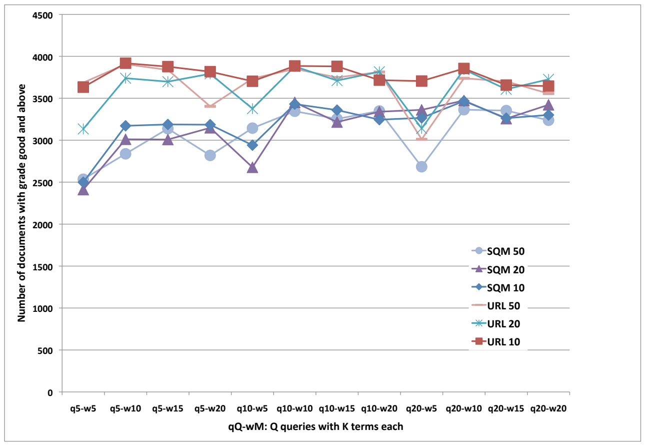


Figure 8: Experiments on parameter sensitivity of SQM with the RST approach. The x-axis varies the number of terms and queries. The y-axis shows the good or better documents found. The legend URL N and SQM N refer to the number N of results scraped with URL search and content search, respectively.

proximity and frequency are considered in query generation. Such hybrid approaches can also benefit from the other proposals in [25].

Acknowledgments

We greatly appreciate the help from Andrei Broder, Andrew Tomkins, and Ravi Kumar for the discussions about the Bharat–Broder approach; Keeyong Han and Ching-Fong Su for international support; Gene Meyers, Kexiang Hu, and Sirisha Machiraju for help with the scraping tools; Choongsoon Chang and Brian Harrington for help with lexicon creation; Emre Velipasaoglu for help with the DCG-based measures; and the anonymous reviewers (especially the first reviewer) for their comments that helped improve the presentation.

9. REFERENCES

[1] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. *J. ACM*, 55(5):1–74, 2008.

[2] P. Baxendale. Machine-made index for technical literature experiment. *IBM J. Research and Development*, 2:354–361, October 1958.

[3] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. *Comput. Netw. ISDN Syst.*, 30(1-7):379–388, 1998.

[4] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In *Proc. Joint Conf. Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 858–867. ACL, 2007.

[5] S. Brin, J. Davis, and H. García-Molina. Copy detection mechanisms for digital documents. In *Proc. Int. Conf. Management of Data (SIGMOD)*, pp. 398–409. ACM, 1995.

[6] A. Broder. On the resemblance and containment of documents. In *Proc. Compression and Complexity of Sequences (SEQUENCES)*, page 21. IEEE, 1997.

[7] A. Broder, M. Fontura, V. Josifovski, R. Kumar, R. Motwani, S. Nabar, R. Panigrahy, A. Tomkins, and Y. Xu. Estimating corpus size via queries. In *Proc. Int. Conf. Info. and Knowledge Management (CIKM)*, pp. 594–603. ACM, 2006.

[8] A. Z. Broder. Some applications of rabin's fingerprinting method. In *Sequences II: Methods in Communications, Security, and Computer Science*, pp. 143–152. Springer-Verlag, 1993.

[9] A. Z. Broder. Identifying and filtering near-duplicate documents. In *Proc. Symp. Combinatorial Pattern Matching (COM)*, pp. 1–10. Springer-Verlag, 2000.

[10] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations (extended abstract). In *Proc. Symp. Theory of Computing (STOC)*, pp. 327–336. ACM, 1998.

[11] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Comput. Netw. ISDN Syst.*, 29(8-13):1157–1166, 1997.

[12] S. Chakrabarti. *Mining the Web.* Morgan Kaufmann, 2003.

[13] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. Symp. Theory of Computing (STOC)*, pp. 380–388. ACM, 2002.

[14] A. Dasdan, K. Tsioutsiouliklis, and E. Velipasaoglu. Web search engine metrics. Tutorial in *Int. Conf. World Wide Web (WWW)*, ACM, 2009.

[15] R. Ghani, R. Jones, and D. Mladenic. Automatic web search query generation to create minority language corpora. In *Proc. of Conf. on Research and Dev. in Info. Retrieval (SIGIR)*, pp. 432–433. ACM, 2001.

[16] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Wesley, 1994.

[17] M. Henzinger. Finding near-duplicate web pp.: a large-scale evaluation of algorithms. In *Proc. of Conf. on Research and Dev. in Info. Retrieval (SIGIR)*, pp. 284–291. ACM, 2006.

[18] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[19] H. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Research and Development*, 1(4):309–317, 1957.

[20] H. Luhn. The automatic creation of literature abstracts. *IBM J. Research and Development*, 2(2), 1958.

[21] G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. In *Proc. Int. Conf. World Wide Web (WWW)*, pp. 141–150. ACM, 2007.

[22] F. D. McSherry, K. Talwar, and M. D. Manasse. Consistent weighted sampling of multisets and distributions. U.S. Patent Appl., Sep 2008.

[23] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[24] T. Noreault, M. McGill, and M. Koll. A performance evaluation of similarity measures, document term weighting schemes and representations in a boolean environment. In *Proc. of Conf. on Research and Dev. in Info. Retrieval (SIGIR)*, pp. 57–76. ACM, 1981.

[25] A. Pereira Jr. and N. Ziviani. Retrieving similar documents from the Web. *J. Web Engineering*, 2(4):247–261, 2004.

[26] N. Shivakumar and H. García-Molina. SCAM: A copy detection mechanism for digital documents. In *Proc. Int. Conf. Digital Libraries (DL)*. ACM, 1995.

[27] N. Shivakumar and H. García-Molina. Building a scalable and accurate copy detection mechanism. In *Proc. Int. Conf. Digital Libraries (DL)*, pp. 160–168. ACM, 1996.

[28] D. R. Swanson. Historical note: Information retrieval and the future of an illusion. *J. American Society for Information Science*, 39(2):92–98, 1988.

[29] Q. Tan, Z. Zhuang, P. Mitra, and C. L. Giles. Designing efficient sampling techniques to detect webpage updates. In *Proc. Int. Conf. World Wide Web (WWW)*, pp. 1147–1148. ACM, 2007.

[30] Y. Yang, N. Bansal, W. Dakka, P. Ipeirotis, N. Koudas, and D. Papadias. Query by document. In *Proc. Int. Conf. Web Search and Data Mining*, pp. 34–43. ACM, 2009.