dSCAM: Finding Document Copies Across Multiple Databases *

Héctor García-Molina Luis Gravano Narayanan Shivakumar

Computer Science Department
Stanford University
Stanford, CA 94305-9040, USA

{hector,gravano,shiva}@cs.stanford.edu

Abstract

The advent of the Internet has made the illegal dissemination of copyrighted material easy. An important problem is how to automatically detect when a "new" digital document is "suspiciously close" to existing ones. The SCAM project at Stanford University has addressed this problem when there is a single registered-document database. However, in practice, text documents may appear in many autonomous databases, and one would like to discover copies without having to exhaustively search in all databases. Our approach, dSCAM, is a distributed version of SCAM that keeps succinct metainformation about the contents of the available document databases. Given a suspicious document S, dSCAM uses its information to prune all databases that cannot contain any document that is close enough to S, and hence the search can focus on the remaining sites. We also study how to query the remaining databases so as to minimize different querying costs. We empirically study the pruning and searching schemes, using a collection of 50 databases and two sets of test documents.

1 Introduction

In a renowned 1995 case [1], an author, who we will refer to as Mr. X for legal reasons, plagiarized several technical reports and conference papers, and resubmitted them under his own name to other conferences and journals. Unfortunately, most of these papers (nearly 18) passed undetected through the paper review process and were accepted to these conferences and journals. The topics of these papers ranged from Steiner routing in VLSI CAD, to massively parallel genetic algorithms, complexity theory, and network protocols. Mr. X also plagiarized papers from the database field, notably a paper in DAPD by Tal and Alonso [2] on three-phase locking, a paper in VLDB '92 by Ioannidis et al. [3] on parametric query optimization, and a paper in ICDE '90 by Leung and Muntz [4] on temporal query processing. The Stanford Copy Analysis Mechanism (SCAM) [5, 6] played an important role in identifying the papers that Mr. X had plagiarized. (See [7, 1] for further details.)

SCAM is a registration server mechanism that helps flag document-copyright violations in Digital Libraries. The target is not simply academic plagiarism, but any type of copying that can financially hurt authors and commercial publishers. SCAM is also useful for removing duplicates and near-duplicates in information retrieval systems [8]. Essentially, SCAM keeps a large database of documents along with indices to support efficient retrieval of stored documents that are "potential copies." SCAM attempts to find not just identical copies, but also cases of "substantial" overlap. For example, if a document contains several paragraphs or sections that were copied from a registered document, it should be flagged as a potential copy even if there are also significant portions where the documents differ. Documents flagged by SCAM have to be checked manually for actual violations since the copying may have been legal and since SCAM may produce some false positives.

The basic SCAM system requires a database of registered documents. In the future, publishers may indeed establish such "copyright registration servers" [9], and these servers can then automatically check public sources such as netnews articles and WWW/FTP sites for copies of the registered documents. However, if there are multiple registration servers, and one has a suspicious document to check, one has to decide what servers to check, since it may be impractical to go to all of them. Furthermore, we may also want to include in our search databases that may not be running a SCAM system. In this case, not only do we have to identify these databases, but we also need to pull out candidate documents so that SCAM can analyze them.

This is precisely what we had to do in Mr. X's case. Initially, we only had the abstracts of the papers that Mr. X had "written," i.e., we had the suspicious documents. Then we proceeded as follows:

*This material is based upon work supported by the National Science Foundation under Cooperative Agreement IRI-9411306. Funding for this cooperative agreement is also provided by DARPA, NASA, and the industrial partners of the Stanford Digital Libraries Project. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the other sponsors.

1. First we selected existing databases that we thought were likely to contain the matching registered documents. Based on the contents of the suspicious documents we decided that the Inspec and CS-TR databases were the most appropriate. (Inspec is a commercial database of electrical engineering and computer science abstracts; CS-TR is an emerging digital library of computer science technical reports, see `http://elib.stanford.edu`.)
2. We manually chose some keywords from Mr. X's abstracts, and issued queries such as "VLSI or Steiner or Routing" against the above databases.
3. We retrieved the abstracts (about 35,000 overall) that matched the above queries, registered them in SCAM, and then tested the suspicious documents against them. In this way we found a total of 14 cases of plagiarism, most of them previously unknown.

In this paper we develop *dSCAM*, a system that automates the entire copy search process. Automating this process is crucial as the number of document databases grows and as publishers rely more on digital publishing. As a matter of fact, appropriate safeguards for intellectual property rights are essential in a large scale public digital library, and an automated *dSCAM* can be one of the tools. In particular,

- *Discovery:* We present the *dSCAM* mechanism that, given many databases and a suspicious document, efficiently identifies the databases that may contain documents that SCAM would consider copies. This problem is fundamentally different from a conventional search problem: *dSCAM* has to flag a database even if it contains a single document that overlaps significantly with the suspicious document.
- *Extraction:* We present the *dSCAM* strategies for automatically generating a query that retrieves potential copies for subsequent analysis by SCAM.

For the discovery phase, we build on previous work in the resource-discovery area, more specifically, on the *GlOSS* approach [10, 11]. The idea is to collect in advance "metainformation" about the candidate databases. This can include, for example, information on how frequently terms appear in documents at a particular database. This metainformation is much smaller than a full index of the database or than what SCAM would actually need to detect copies. Then, based on this information, *dSCAM* can rule out databases that do not contain documents that SCAM would consider copies.

Notice that while *dSCAM* is structurally similar to *GlOSS*, there is a fundamental difference. *GlOSS* attempts to discover databases that satisfy a given query (Boolean or vector space). The *GlOSS* problem is a simpler one since all we need to know is that the candidate database contains the necessary terms. However, for *dSCAM*, finding documents that have similar terms to those of the suspicious document is not enough. For example, if the suspicious document only contains a subset that is a copy, then there are terms in the non-copy portion that are not relevant. Thus, simply treating the suspicious document as a *GlOSS* query will not lead us to the right databases.

Instead, *dSCAM* will need to keep more sophisticated statistics than *GlOSS* does, enough to let it identify sites that may have even a single copy, not just documents that are "similar" to the suspicious one in an information retrieval sense. The key challenge is to collect as little information as possible in *dSCAM* to be able to perform this difficult discovery task. Section 7 reports experiments that indeed show that our techniques are successful at isolating the databases with potential copies, with relatively few false positives.

In this paper we consider two types of discovery techniques: *conservative* and *liberal* ones. Conservative techniques only rule out a database if it is certain that SCAM would not consider *any* document there a potential copy. The clear advantage of conservative techniques is that they do not miss potential copies. In contrast, liberal techniques might in principle miss databases with potential copies. However, this is rarely the case, as we will see, and the liberal techniques search fewer unnecessary databases. In practice, the choice between conservative and liberal depends on how exhaustive the search must be and what resources are available.

For the extraction problem (i.e., the problem of generating a query to retrieve the potential copies from a database), a naive solution is to simply query each database (identified in the discovery phase) for all documents containing any of the terms in the suspicious document. That is, if the suspicious document contains words $w_1, \ldots w_N$, we could submit the query $w_1 \vee \ldots \vee w_N$ (or alternatively, request all the documents with w_1, then all the ones with w_2, and so on). Clearly, any potential copy would be extracted in this way. However, our goal here is to extract all the potential copies without having to perform such a massive query. Thus, to solve this problem we show how to find the minimal query, under two different cost metrics, that can extract all the desired documents. For example, one of our cost measures is the number of words in the submitted query. We will see that we can reduce such a number drastically by bounding the maximum "contribution" of every word to a potential copy.

We start in Section 2 by giving an overview of SCAM. In Section 3 we describe the data that *dSCAM* keeps about the databases. We use this data in Section 4 to define the conservative copy discovery schemes for *dSCAM*, while in Section 5 we relax these schemes to make them liberal. In Section 6 we present the extraction mechanisms. Finally, in Section 7 we discuss an experimental evaluation of our techniques, using a collection of 50 databases and two sets of suspicious documents.

2 Using SCAM for copy detection

Given a suspicious document S and a registered document D, SCAM detects whether D is a potential copy of S by deciding whether they overlap significantly. We have explored [5, 6] a variety of overlap measures. For example, we can say that S and D overlap if they contain at least some fraction of common sentences. A problem with this scheme is that it is often hard to detect sentence boundaries (e.g., periods in abbreviations get confused with the end of sentences). Also, it cannot detect partial-sentence overlaps.

A different measure we have studied uses *similarity*, in the information retrieval (IR) sense, as a starting point. Traditionally, two documents are said to be similar if the frequency with which words occur is correlated. If the distribution of word frequencies between S and D is identical, we say that the similarity is maximal at 1. As the distributions differ, the similarity decreases. This measure does not work for copy detection because the matching documents can have portions that are very different causing the word frequency distributions to differ significantly. However, this measure can be modified for copy detection as we will explain in this section. In this paper we will use this modified IR measure as the basis for $dSCAM$ because our experimental results show it works best, at least for the relatively small documents found on the Internet.

To evaluate S and D using this modified IR measure, SCAM first focuses on the words that appear a similar number of times in S and D, and ignores the rest of the words. More precisely, given a fixed $\epsilon > 2$, the *closeness set* for S and D, $c(S, D)$, contains the words w_i with a similar number of occurrences in the two documents [5]:

$$w_i \in c(S, D) \Leftrightarrow \frac{F_i(S)}{F_i(D)} + \frac{F_i(D)}{F_i(S)} < \epsilon$$

where $F_i(d)$ is the frequency of word w_i in document d. If either $F_i(S)$ or $F_i(D)$ are zero, then w_i is not in the closeness set. Given ϵ, S determines a range of frequencies $Accept(w_i, F_i(S))$ such that w_i is in the closeness set for S and D if and only if $F_i(D) \in Accept(w_i, F_i(S))$.

The intuition behind this is as follows. If S and D share a substantial portion of identical text, then there ought to be a set of words unique to that text that will occur with similar frequencies. Focusing on words in the closeness set diminishes the effects of unrelated portions of text.[1]

Example 1 *Consider a suspicious document S and a database db with two documents, D_1 and D_2. There are four words in these documents, w_1, w_2, w_3, and w_4. The following table shows the frequency of the words in the documents.*

Document	F_1	F_2	F_3	F_4
S	*1*	*3*	*3*	*9*
D_1	*1*	*3*	*0*	*0*
D_2	*0*	*8*	*5*	*0*

[1]It also helps to ignore altogether words that occur frequently across documents [6]. Our experiments of Section 7 use the stop words in [6].

For example, w_3 appears three times in S ($F_3(S) = 3$), five times in D_2 ($F_3(D_2) = 5$), and it does not appear in D_1 ($F_3(D_1) = 0$). Assuming $\epsilon = 2.5$ (a value that worked well in the experiments in [5]), $Accept(w_3, F_3(S)) = Accept(w_3, 3) = [2, 5]$. Thus, w_3 is in $c(S, D_2)$, the closeness set for S and D_2, because $F_3(D_2) = 5$ is in $Accept(w_3, F_3(S))$. Although $F_3(D_2)$ is higher than $F_3(S)$, these two values are sufficiently close for $\epsilon = 2.5$. In effect, $\frac{F_3(S)}{F_3(D_2)} + \frac{F_3(D_2)}{F_3(S)} = \frac{3}{5} + \frac{5}{3} = 2.27 < \epsilon = 2.5$. For the remaining cases, $Accept(w_1, F_1(S)) = [1, 1]$, $Accept(w_2, F_2(S)) = [2, 5]$, and $Accept(w_4, F_4(S)) = [5, 17]$. Then, $c(S, D_1) = \{w_1, w_2\}$, and $c(S, D_2) = \{w_3\}$.

After finding the closeness set for S and D, SCAM computes the similarity $sim(S, D)$ between the two documents. We would like to use traditional IR similarity measures (using only words in the closeness set), but this does not work because those measures give low values when S is a subset of D or vice versa. Instead we compute two measures, one for the case where S might be a subset of D and one for the reverse case, and take the maximum. In the former we ignore the norm (see below) of D since it could be a much larger document; in the latter we ignore the norm of S. That is, $sim(S, D) = \max\{subset(S, D), subset(D, S)\}$ where:

$$subset(D_1, D_2) = \sum_{w_i \in c(D_1, D_2)} \frac{F_i(D_1)}{|D_1|} \cdot F_i(D_2)$$

($|D| = \sum_{i=1}^{N} F_i^2(D)$ is the norm of document D and N is the number of terms.) If $sim(S, D) > T$, for some user-specified threshold T, then SCAM flags document D as a potential copy of the suspicious document S.

Example 1 (cont.) *Continuing with our example above, $|S| = F_1^2(S) + F_2^2(S) + F_3^2(S) + F_4^2(S) = 1^2 + 3^2 + 3^2 + 9^2 = 100$. Similarly, $|D_1| = 10$ and $|D_2| = 89$. To compute the similarity $sim(S, D_2)$ we just consider w_3, the only word in the closeness set for S and D_2. Then,*

$$\begin{aligned} sim(S, D_2) &= \max\{F_3(D_2) \cdot \frac{F_3(S)}{|S|}, \frac{F_3(D_2)}{|D_2|} \cdot F_3(S)\} \\ &= \max\{5 \cdot \frac{3}{100}, \frac{5}{89} \cdot 3\} = 0.17 \end{aligned}$$

Similarly, $sim(S, D_1) = 1$, because SCAM regards D_1 as a strict "subdocument" of S. So, for $T = 0.80$, SCAM would not consider D_2 to be a potential copy of S. However, SCAM would find D_1 suspiciously close to S.

Even though the SCAM similarity does not take into account word sequencing, the experiments in [5, 6] show that it detects potential copies relatively well. In these experiments, conducted with 50,000 netnews articles, false positives were very rare: the similarity measure flagged unrelated documents as copies (because they shared common vocabulary) in only 0.01% of the cases. False negatives were more common but still only 5% of the cases tested: in these cases, documents with relatively small overlap were not detected. Overall, the similarity measure performed better than the sentence overlap measure described earlier.

3 The *dSCAM* information about the databases

dSCAM needs information to decide whether a database *db* has potential copies of a suspicious document S. This information should be concise, but also sufficient to identify any such database. *dSCAM* keeps the following statistics (or a subset of them) for each database *db* and word w_i, where db_i is the set of documents in *db* that contain w_i:

- $f_i(db) = \min_{D \in db_i} F_i(D)$: $f_i(db)$ is the minimum frequency of word w_i in any document in *db* that contains w_i
- $F_i(db) = \max_{D \in db_i} F_i(D)$: $F_i(db)$ is the maximum frequency of word w_i in any document in *db* that contains w_i
- $n_i(db) = \min_{D \in db_i} |D|$: $n_i(db)$ is the minimum norm of any document in *db* that contains w_i
- $R_i(db) = \max_{D \in db_i} \frac{F_i(D)}{|D|}$: $R_i(db)$ is the maximum value of the ratio $\frac{F_i(D)}{|D|}$ for any document $D \in db$ that contains w_i
- $d_i(db)$ is the number of documents in *db* that contain word w_i

Example 1 (cont.) *The following table shows the dSCAM metadata for our sample database db. Note that there are no entries for w_4, since it does not appear in any document in db.*

Statistics	w_1	w_2	w_3
f_i	*1*	*3*	*5*
F_i	*1*	*8*	*5*
n_i	*10*	*10*	*89*
R_i	$\frac{1}{10}$	$\frac{3}{10}$	$\frac{5}{89}$
d_i	*1*	*2*	*1*

As mentioned earlier, db has two documents, D_1 and D_2. Document D_1 contains w_2 three times, and document D_2, eight times. Therefore, $f_2(db) = \min\{3, 8\} = 3$ and $F_2(db) = \max\{3, 8\} = 8$. Also, $|D_1| = 10$ and $|D_2| = 89$, so $n_2(db) = \min\{10, 89\} = 10$. Finally, $R_2(db) = \max\{\frac{3}{10}, \frac{8}{89}\} = \frac{3}{10}$, and $d_2(db) = 2$, since w_2 appears in both D_1 and D_2.

Notice that the table above is actually larger than our earlier table that gave the complete word frequencies. This is just because our sample database contains only two documents. In general, the information kept by *dSCAM* is proportional to the number of words or terms appearing in the database, while the information needed by SCAM is proportional to the number of words times the number of times the words appear in different documents. In a real database, many words appear in hundreds or thousands of documents, and hence the SCAM information can be much larger than the *dSCAM* information. We will return to this issue in Section 7. To obtain the necessary statistics, *dSCAM* periodically polls each potential source database, which then extracts the data from its index structures.

4 The conservative approach

Given a set of databases, a suspicious document S, and a threshold T, *dSCAM* selects all databases with potential copies of S, i.e., all the databases with at least one document D with $sim(S, D) > T$. To identify these databases, *dSCAM* uses the metadata of Section 3. In this section we focus on conservative techniques that never miss any database with potential copies. In other words, *dSCAM* cannot produce any *false negatives* with the techniques of this section. However, *dSCAM* might produce *false positives*, and consider that a database has potential copies when it actually does not. In Section 7 we report experimental results that study how often the latter takes place.

The information described in Section 3 can be used by *dSCAM* in a variety of ways. We present two alternatives, starting with the simplest. The more sophisticated technique will be less conservative: it will always identify the databases with potential copies of a document, but it will have fewer false positives than the simpler technique.

Given a database *db*, a suspicious document S, and a technique A, *dSCAM* computes an upper bound $Upper_A(db, S)$ on the similarity of any document in *db* and S. In other words, $Upper_A(db, S) \geq sim(S, D)$ for every document $D \in db$. Thus, if $Upper_A(db, S) \leq T$, then there are no documents in *db* close enough to S as determined by the threshold T, and we can safely conclude that database *db* has no potential copies of S. The two strategies below differ in how they compute this upper bound.

The *Range* strategy

Consider a word w_i in S. Suppose that w_i appears in some document D in *db*. We know that D contains w_i between $f_i(db)$ and $F_i(db)$ times. Also, w_i is in the closeness set for S and D if and only if $F_i(D) \in Accept(w_i, F_i(S))$. So, w_i is in the closeness set for S and D if and only if $F_i(D) \in [m_i, M_i] = [f_i(db), F_i(db)] \cap Accept(w_i, F_i(S))$. If this range is empty, then w_i is not in the closeness set for S and D, for any document $D \in db$, and therefore w_i does not contribute to $sim(S, D)$ for any D. If the range $[m_i, M_i]$ is not empty, then w_i can be in the closeness set for S and D, for some document D. For any such document D, $F_i(D) \leq M_i$. We then define the *maximum frequency* of word $w_i \in S$ in any document of *db*, $M_i(db, S)$, as:

$$M_i(db, S) = \begin{cases} M_i & \text{if } [m_i, M_i] \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Putting everything together, we define the upper bound on the similarity of any document D in *db* and S for technique *Range* as:

$Upper_{Range}(db, S) =$

$$\max\{Upper1_{Range}(db,S),\ Upper2_{Range}(db,S)\}$$

where:

$$Upper1_{Range}(db,S) = \sum_{i=1}^{N} M_i(db,S)\cdot\frac{F_i(S)}{|S|} \quad (1)$$

$$Upper2_{Range}(db,S) = \sum_{i=1}^{N} \frac{M_i(db,S)}{n_i(db)}\cdot F_i(S) \quad (2)$$

Note that since $n_i(db) \leq |D|$ for every $D \in db$ that contains w_i, then $Upper_{Range}(db,S) \geq sim(S,D)$ for every $D \in db$. Also note that the *Range* technique does not use the R_i statistics.

Example 1 (cont.) *Consider the db statistics and the suspicious document S. We have already computed $Accept(w_1, F_1(S)) = [1,1]$, $Accept(w_2, F_2(S)) = [2,5]$, $Accept(w_3,\ F_3(S)) = [2,5]$, and $Accept(w_4, F_4(S)) = [5,17]$. Also, dSCAM knows, for example, that word w_2 appears in db with in-document frequencies between $[f_2(db), F_2(db)] = [3,8]$. Then, the interesting range of frequencies of w_2 in db is $[m_2, M_2] = [3,8] \cap [2,5] = [3,5]$. The maximum such frequency is $M_2(db,S) = 5$. (Notice that there is no document D in db with $F_2(D) = 5$. $M_2(db,S)$ is in this case a strict upper bound for the frequencies of w_2 in db that are in $Accept(w_2, F_2(S))$.) Similarly, $M_1(db,S) = 1$, $M_3(db,S) = 5$, and $M_4(db,S) = 0$. Therefore,*

$$\begin{aligned} Upper1_{Range}(db,S) &= 1\cdot\frac{1}{100} + 5\cdot\frac{3}{100} + 5\cdot\frac{3}{100} \\ &= 0.31 \\ Upper2_{Range}(db,S) &= \frac{1}{10}\cdot 1 + \frac{5}{10}\cdot 3 + \frac{5}{89}\cdot 3 \\ &= 1.77 \\ Upper_{Range}(db,S) &= 1.77 \end{aligned}$$

Therefore, if our threshold T is, say, 0.80, we would search db. This is of course the right decision since D_1 in db is indeed a potential copy.

The *Ratio* strategy

This technique is similar to the previous one, but uses the R_i statistics. Thus,

$$Upper_{Ratio}(db,S) = \max\{Upper1_{Range}(db,S),\ Upper2_{Ratio}(db,S)\}$$

where:

$$Upper2_{Ratio}(db,S) = \sum_{i \mid M_i(db,S)\neq 0} \min\{\frac{M_i(db,S)}{n_i(db)}, R_i(db)\}\cdot F_i(S) \quad (3)$$

It is immediate from the definition above that $Upper_{Ratio}(db,S) \leq Upper_{Range}(db,S)$ for every database db and query document S. Therefore, *Ratio* is a less conservative technique than *Range*, and will tend to have fewer false positives than *Range*. Nevertheless, *Ratio* will always detect databases with potential copies of S, because $sim(S,D) \leq Upper_{Ratio}(db,S)$ for every $D \in db$.

Example 1 (cont.) *We have already computed $Upper1_{Range}(db,S) = 0.31$. Now,*

$$\begin{aligned} Upper2_{Ratio}(db,S) &= \frac{1}{10}\cdot 1 + \frac{3}{10}\cdot 3 + \frac{5}{89}\cdot 3 \\ &= 1.17 \end{aligned}$$

which is lower than $Upper2_{Range}(db,S)$.

5 The liberal approach

The techniques of Section 4 are conservative: they never fail to identify a database with potential copies of a suspicious document (i.e., these techniques have no false negatives). A problem with these techniques is that they usually produce too many false positives. (See Section 7.) Consequently, we now introduce *liberal* versions of the *Range* and *Ratio* techniques. In principle, the new techniques might have false negatives. As we will see, false negatives occur rarely, while the number of false positives is much lower than that for the conservative techniques.

We modify the techniques of Section 4 in two different ways. First, we allow these techniques to focus only on the "rarest" words that occur in a suspicious document, instead of on all its words (or on all the words that SCAM uses). (See Section 5.1.) This way *dSCAM* can prune away databases where these rare words do not appear, thus reducing the search space. Second, we allow these techniques to use probabilities to estimate (under some assumptions) how many potential copies of a suspicious document each database is expected to have. (See Section 5.2.) Thus, the probabilistic techniques no longer compute upper bounds, again reducing the search space.

5.1 Counting only rare words

The techniques of Section 4 considered every word in a suspicious document S (i.e., every word that SCAM uses) to decide which databases to search for potential copies of S. Alternatively, *dSCAM* can just focus on the *rarest* words in S, i.e., on the words in S that appear in the fewest number of databases. *dSCAM* then decides to search a database only if at least a few of these rare words appear in it. If *dSCAM* uses enough of the rare words in S, any potential copy of S will tend to contain a few of these words. Furthermore, since these words appear in only a few databases, they will help *dSCAM* dismiss a significant fraction of the databases, thus reducing the number of false positives.

One specific way to implement these ideas is as follows. Given a suspicious document S, *dSCAM* just considers k percent of its words. These are the $k\%$ words in S that appear in the fewest available databases. *dSCAM* can tell which words these are from the metadata about the databases (Section 3). The remaining words in S are simply ignored.

Example 1 (cont.) *Consider suspicious document S, with words w_1, w_2, w_3, and w_4. Suppose that w_1 appears in 1 database, w_2 in 2, w_3 in 70, and w_4 in 20 databases. If dSCAM uses only 50% of the words in S ($k = 50$), it chooses w_1 and w_2, and ignores w_3 and w_4.*

As we mentioned before, *dSCAM* now ignores words in S that SCAM uses for copy detection. Therefore, *dSCAM* might in principle miss a database with potential copies of S. However, as we will see in Section 7, we can find values for k for which *dSCAM* has very few false negatives, while producing much fewer false positives than with the conservative techniques of Section 4.

Given k, we adapt the $Upper_{Range}$ and $Upper_{Ratio}$ bounds of Section 4 (Equations 1, 2, and 3) to sum only over the k% rarest words in S. We refer to the new values as Sum_{Range} and Sum_{Ratio}, because they are no longer upper bounds on the similarities of the documents in the databases and S.

As we use fewer words in S (i.e., only k% of them), we need to adjust the threshold T (Section 2) for *dSCAM* accordingly. We refer to the adjusted threshold as T^k. For example, if we are just considering 10% of the words in S, we could compensate by reducing the threshold $T^{10} = 0.10 * T$. We explore different values for T^k in Section 7. If $Sum_{Range}(db, S)$ (respectively, $Sum_{Ratio}(db, S)$) is higher than T^k, *dSCAM* will search db for potential copies of S.

Example 1 (cont.) *In Section 4 we computed $Upper_{Range}(db, S) = 1.77$. Now, if dSCAM only considers the 50% rarest words in S (i.e., w_1 and w_2), only those words are counted, and we have:*

$$\begin{aligned} Sum1_{Range}(db, S) &= 1 \cdot \frac{1}{100} + 5 \cdot \frac{3}{100} = 0.16 \\ Sum2_{Range}(db, S) &= \frac{1}{10} \cdot 1 + \frac{5}{10} \cdot 3 = 1.6 \\ Sum_{Range}(db, S) &= 1.6 \end{aligned}$$

The original SCAM threshold was $T = 0.80$. Since we are now considering only half of the words, we could scale down T to, say, $T^{50} = 0.5 \cdot T = 0.40$. At any rate, we would still search db, because $1.6 > 0.40$. This is the right decision, since D_1 in db is indeed a potential copy.

5.2 Using probabilities

So far, the techniques for *dSCAM* compute the maximum possible contribution of each word considered, and add these contributions. However, it is unlikely that any document in a database will contain all of these words with this maximum contribution. In this section, we depart from this "deterministic" model, and, given a database db, try to bound the probability that db has potential copies of a suspicious document. If this probability is high enough, *dSCAM* will search db.

Our goal is to bound the probability that a document in db has a similarity with S that exceeds the adjusted threshold T^k. For this, we define two random variables $XRange1$ and $XRange2$ (corresponding to $Sum1_{Range}$ and $Sum2_{Range}$, respectively). These variables model the similarity of the documents in db and S. Then,

$$Prob_{Range} = \max\{P(XRange1 > T^k), P(XRange2 > T^k)\}$$

If $Prob_{Range} \geq \frac{1}{|db|}$, *dSCAM* will search db for potential copies of S, since there is at least one expected document that exceeds the adjusted threshold T^k.

Actually, instead of computing $P(XRange1 > T^k)$ and $P(XRange2 > T^k)$, we use an upper bound for these values as given by Chebyshev's inequality. This bound is based on the expected value and the variance of $XRange1$ and $XRange2$.

We now define random variable $XRange1$, following the definition of $Sum1_{Range}$. (Random variable $XRange2$ is analogous, using the definition of $Sum2_{Range}$.) The $XRange1$ is actually a sum of random variables: $XRange1 = XRange1_{i_1} + \ldots + XRange1_{i_s}$ where $w_{i_1}, \ldots, w_{i_s}$ are the k% rarest words in S. Random variable $XRange1_i$ corresponds to word w_i:

$$XRange1_i = \begin{cases} M_i(db, S) \cdot \frac{F_i(S)}{|S|} & \text{with prob. } \frac{d_i(db)}{|db|} \\ 0 & \text{with prob. } 1 - \frac{d_i(db)}{|db|} \end{cases}$$

This variable models the occurrence of word w_i in the documents of database db. Word w_i occurs in $d_i(db)$ documents in db, so the probability that it appears in a randomly chosen document from db is $\frac{d_i(db)}{|db|}$. To use Chebyshev's inequality and compute the variance of $XRange1$ and $XRange2$, we assume that words appear in documents following independent probability distributions. We define $Prob_{Ratio}$ in a completely analogous way.

6 Searching the databases with potential copies

Once *dSCAM* has decided that a database db might have potential copies of a suspicious document S, it has to extract these potential copies from db. If database db happens to run a local SCAM server, *dSCAM* can simply submit S to this server and get back exactly those documents that SCAM considers potential copies. However, if db does not run a SCAM server, we need an alternative mechanism to extract the potential copies automatically. For this, we will assume that db can answer boolean "or" queries, which most commercial search engines support. For example, we can retrieve from db all documents containing the words "copyright" or the word "SCAM" by issuing the query "copyright $\vee$ SCAM." (Alternatively, if some search engine does not support "or" queries, we could issue a sequence of queries, and then merge the sequence of results.)

Let $w_1, \ldots, w_N$ be the words in S. In principle, we could issue the query $w_1 \vee \ldots \vee w_N$ to db and obtain all documents that contain at least one of these words. However, such a query is bound to return too many documents that are not potential copies of S. In this section, we study how to choose a smaller set of words $\{w_{i_1}, \ldots, w_{i_n}\}$ that will not miss any potential copy from db. Furthermore, the resulting queries will tend not to extract documents that are not potential copies of S.

To choose a set of words to query, we define the *maximum contribution* $C_i(db, S)$ of word w_i in db as an upper bound on the amount that w_i can add to $sim(S, D)$, for any $D \in db$. We give two definitions of this maximum contribution, each corresponding to a technique of Section 4. The first of these is more conservative but uses less information. The other is less conservative but uses more information.

$$C_i(db, S) = \begin{cases} \max\{M_i(db,S) \cdot \frac{F_i(S)}{|S|},\ \frac{M_i(db,S)}{n_i(db)} \cdot F_i(S)\} & \text{for } Range \\ \max\{M_i(db,S) \cdot \frac{F_i(S)}{|S|},\ \min\{\frac{M_i(db,S)}{n_i(db)}, R_i(db)\} \cdot F_i(S)\} & \text{for } Ratio \end{cases}$$

Now, let $C(db, S) = \sum_{i=1}^{N} C_i(db, S)$, and let T be the SCAM similarity threshold that the users specified. Then, any set of words $\{w_{i_1}, \ldots, w_{i_n}\}$ with the following property is sufficient to extract all the potential copies of S from db:

$$\sum_{j=1}^{n} C_{i_j}(db, S) \geq C(db, S) - T \qquad (4)$$

To see why it is enough to use the query $w_{i_1} \vee \ldots \vee w_{i_n}$, consider a document $D \in db$ that does not contain any of these n words. Then, $sim(S, D) \leq C(db, S) - \sum_{j=1}^{n} C_{i_j}(db, S) \leq T$. Therefore, the similarity of D and S can never exceed the required threshold T. This approach is conservative: we cannot miss any potential copy of a document by choosing the query words as above. Alternatively, we explored a liberal approach that would retrieve all potential copies most of the time, and has much fewer "false positives." For space limitations, we do not describe this liberal technique further, but we report some experimental results in Section 7.

To choose among all sets of words that satisfy Condition 4, we associate a cost p_i with each word w_i. We then choose a set of words $\{w_{i_1}, \ldots, w_{i_n}\}$ that satisfies Condition 4 and minimizes $\sum_{j=1}^{n} p_{i_j}$. We consider two different cost models for a query:

The *WordMin* cost model

In this case we minimize the number of words that will appear in the query. Thus, $p_i = 1$ for all i. Then, our problem reduces to finding the smallest set of words that satisfies Condition 4, which we can do optimally with a simple greedy algorithm.

The *SelMin* cost model

In this case we consider the selectivity of each word w_i that will appear in the query, i.e., the fraction of the documents in the database that contain word w_i. Thus, $p_i = Sel(w_i, db)$. By minimizing the added selectivity we will tend to minimize the number of documents that we retrieve from db.

We will find an optimal solution for this problem by reducing it to the *0-1 knapsack problem* [12]. The new formulation of the problem is as follows. A thief robbing a store finds N items (the words). The ith item is worth p_i dollars (the selectivity of word w_i) and weighs $C_i(db, S)$ pounds (the maximum contribution of w_i). The thief wants to maximize the value of the load, but can only carry up to T pounds. The problem is to find the right items (words) to steal. This formulation of the problem actually finds the words that will not appear in the final query, and maximizes the added selectivity of these words. The weight of the words is at most T. Therefore, the words that are not chosen weigh at least $C(db, S) - T$, satisfy Condition 4, and have the lowest added selectivity among the sets satisfying Condition 4. Assuming that T, the C_i's, and the p_i's have a fixed number of significant decimals, we can use dynamic programming to solve the problem in $O(T \cdot N)$ time, where N is the number of words in the suspicious document [12].

7 Experiments

This section presents experimental results for *dSCAM*. We focus on three sets of issues: How many false positives do the *dSCAM* techniques report, how many false negatives do the liberal *dSCAM* techniques produce, and how effective is the document extraction step?

For the registered-document databases, our experiments used a total of 63,350 ClariNet news articles. We split these articles evenly in 50 databases so that each database consists of 1,267 documents.

For the suspicious documents, our experiments used two different document sets. The first set, which we refer to as *Registered*, contains 100 documents from the 50 databases. Therefore, each suspicious document has at least one perfect copy in some database. (There could be more copies due to crosspostings of articles.) The second set, which we refer to as *Disjoint*, contains 100 later articles that do not appear in any of the 50 databases. This set models the common case when the suspicious documents are actually new documents that do not appear anywhere else.

Our first experiments are for the Sum_{Ratio} technique, which proved to work the best among the *dSCAM* techniques, as we will see later. Figures 1 through 4 show different interesting metrics as a function of the adjusted threshold T^k, and for different values of k. In all of these plots, the SCAM threshold T is set to 1. For example, the curves for $k = 10$ correspond to considering only 10% of the words (the rarest ones) in the suspicious documents. Note that for $k = 100$ all of the words in the suspicious documents are used. In this case, Sum_{Ratio} coincides with the conservative technique $Upper_{Ratio}$.

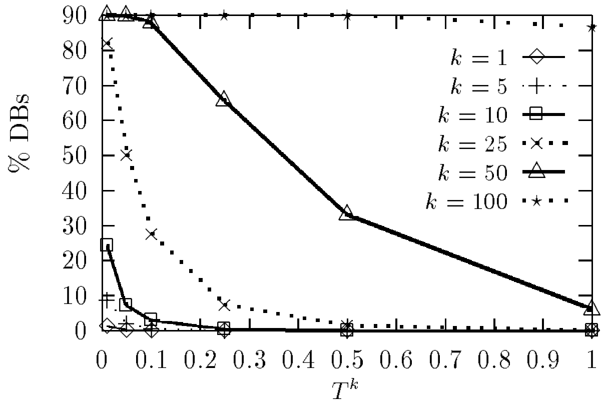

Figure 1: The percentage of the 50 databases that are searched as a function of the adjusted similarity threshold T^k (*Registered* suspicious documents; Sum_{Ratio} strategy; $T = 1$).

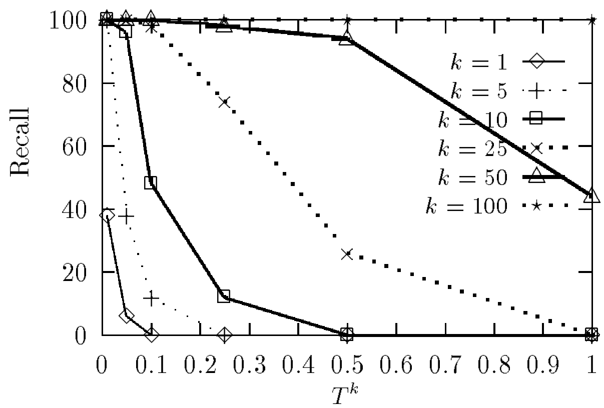

Figure 2: The average recall as a function of the adjusted similarity threshold T^k (*Registered* suspicious documents; Sum_{Ratio} strategy; $T = 1$).

One way to evaluate the $dSCAM$ strategies is to look at $d(S)$, the percentage of databases returned by $dSCAM$ for a suspicious document S. Figure 1 shows the average $d(S)$ (averaged over all S in the *Registered* set), as a function of T^k. The more words $dSCAM$ considers from the suspicious documents (i.e., the higher k), the more databases are searched: $dSCAM$ considers the words as ordered by how rare they are. Therefore, when $dSCAM$ starts considering "popular" words, more databases will tend to exceed the similarity threshold T^k. Also, for a fixed k, the higher T^k, the fewer databases that $dSCAM$ searches, since only databases that exceed T^k are searched. For low values of k, $dSCAM$ searches very few databases. For example, for $k = 10$ and $T^k = 0.05$, less than 10% of the databases are searched.

As we know, Sum_{Ratio} may produce false negatives for $k < 100$, i.e., it may tell us not to search databases where SCAM would find potential copies. It is interesting to study what percentage of the databases with potential copies $dSCAM$ actually searches (or equivalently, what

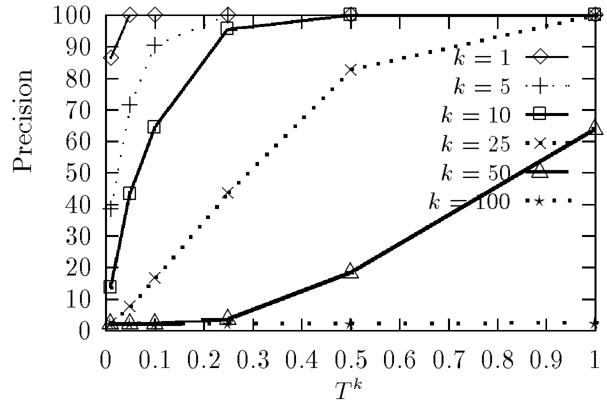

Figure 3: The average precision as a function of the adjusted similarity threshold T^k (*Registered* suspicious documents; Sum_{Ratio} strategy; $T = 1$).

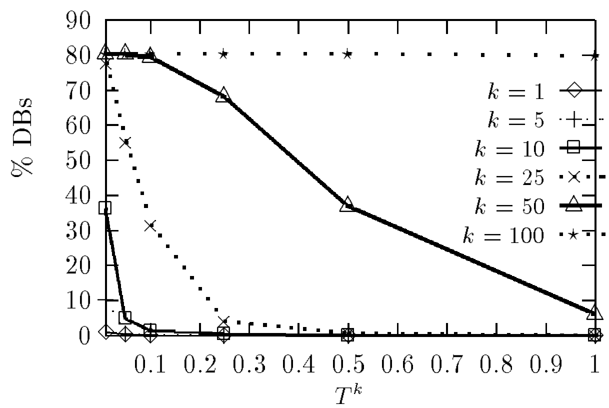

Figure 4: The percentage of the 50 databases that are searched as a function of the adjusted similarity threshold T^k (*Disjoint* suspicious documents; Sum_{Ratio} strategy; $T = 1$).

percentage of these databases are not false negatives). Let $s(S)$ be the number of databases with potential copies of S according to SCAM, and let $s'(S)$ be the number of databases with potential copies of S that $dSCAM$ searches. Then, the *recall* of the technique used by $dSCAM$ is the average value of $\frac{100 \cdot s'(S)}{s(S)}$ over our suspicious documents S.

Figure 2 shows the recall values for Sum_{Ratio} as a function of the adjusted threshold T^k. This figure is very similar to Figure 1: the more databases a technique searches, the higher its recall tends to be. Note, however, that some techniques have very few false negatives, while they search a low percentage of the databases. For example, for $k = 10$ and $T^k = 0.05$, recall is above 90%, meaning that for the average suspicious document, 90% of the databases with potential copies are chosen by $dSCAM$. As we have seen, just under 10% of the databases are searched for this value of k and T^k.

As we mentioned above, $dSCAM$ produces false positives. We want to measure what percentage of the databases selected by $dSCAM$ actually contains potential copies. The *precision* of the technique used by $dSCAM$ is the average value of $\frac{100 \cdot s'(S)}{d(S)}$ over our suspicious documents S. Figure 3 shows the precision values for Sum_{Ratio} as a function of the adjusted threshold T^k. As expected, the more databases a technique searches, the lower its precision tends to be. For $k = 10$ and $T^k = 0.05$, precision is over 40%, meaning that for the average suspicious document, over 40% of the databases that $dSCAM$ searches have potential copies of the document. Actually, this choice of values for k and T^k is a good one: $dSCAM$ searches very few databases while achieving high precision and recall values.

We are evaluating $dSCAM$ in terms of how well it predicts the behavior of SCAM at each database. However, SCAM can sometimes be wrong. For example, SCAM can wrongly flag a document D in db as a copy of a suspicious document S. $dSCAM$ might then also flag db as having potential copies of S. However, we do not "penalize" $dSCAM$ for this "wrong" choice: the best $dSCAM$ can do is to predict the behavior of SCAM, and that is why we define precision and recall as above. It would be unreasonable to ask a system like $dSCAM$, with very limited information about the databases, to detect copies more accurately than a system like SCAM, which has complete information about the database contents.

To illustrate the storage space differences between $dSCAM$ and SCAM, let us consider the data that we used in our experiments. In this case, there are around 4 million word-document pairs, which is the level of information that a SCAM server needs, whereas there are only around 791,000 word-database pairs, which is the level of information that a $dSCAM$ server needs. As the databases grow in size, we expect this difference to widen too, since the $dSCAM$ savings in storage come from words appearing in multiple documents. For example, if we consider our 50 databases as a single, big database, $dSCAM$ needs only 138,086 word-database pairs, whereas the SCAM data remains the same. Therefore, $dSCAM$ has just around 3.36% as many entries as SCAM. We are considering alternatives to reduce the size of the $dSCAM$ data even further. As an interesting direction for future work, $dSCAM$ can store information on, say, only the 10% rarest words. Most of the time, the 10% rarest words that appear in a suspicious document will be among these 10% overall rarest words, so $dSCAM$ can proceed as usual. With this scheme, the $dSCAM$ space requirements would be cut further by an order of magnitude.

Figure 4 shows results for the *Disjoint* set of suspicious documents, again for the Sum_{Ratio} technique and $T = 1$. There are no potential copies of these documents in any of the 50 databases. Therefore, recall is always 100%, and precision is 0% if some database is selected. It then suffices to report the percentage of databases chosen for these documents (Figure 4). These values tend to be lower in general than those for the *Registered* suspicious documents of Figure 1, which is the right trend, since no database contains

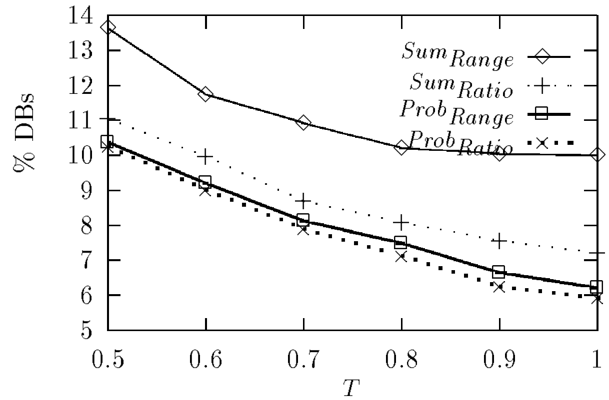

Figure 5: The percentage of the 50 databases that are searched as a function of the SCAM threshold T (*Registered* suspicious documents; $k = 10$; $T^k = 0.05 \cdot T$).

potential copies of the suspicious documents. For example, for $k = 10$ and $T^k = 0.05$, less than 5% of the databases are searched.

So far we have presented results just for the Sum_{Ratio} technique. Figures 5 through 7 show results also for Sum_{Range}, $Prob_{Range}$, and $Prob_{Ratio}$, as a function of the SCAM threshold T. In all of these plots, we have fixed $k = 10$ and $T^k = 0.05 \cdot T$, which worked well for both the *Registered* and the *Disjoint* suspicious documents when $T = 1$. In Figure 5, $Prob_{Range}$ and $Prob_{Ratio}$ search fewer databases than Sum_{Range} and Sum_{Ratio}, at the expense of significantly lower recall values (Figure 6). Sum_{Range} and Sum_{Ratio} have very high recall values (above 95% for all values of T). Precision is also relatively high, especially for the Sum_{Ratio} strategy (Figure 7). From all these plots, Sum_{Ratio} appears as the best choice for $dSCAM$, because of its high recall and precision, and low percentage of databases that it searches. Also, note that Sum_{Ratio} does not need the d_i statistics, resulting in lower storage requirements than those of $Prob_{Ratio}$, for example. However, if we want to be conservative, and be sure that we do not miss any potential copy of a document, then the best choice is also Sum_{Ratio}, but with $k = 100$ and $T^k = T$. (This technique coincides with the conservative $Upper_{Ratio}$ technique of Section 4.)

To determine whether the results above will still hold for larger databases, we performed the following experiment. Initially we have a single database with 1,267 documents (one of the databases that we used in this Section). $dSCAM$ decides whether this database should be searched or not for each of the *Disjoint* suspicious documents, with $T = 1$, the Sum_{Ratio} strategy, $k = 10$, and $T^k = 0.05$. The answer should be "no" for each of these documents, of course. Figure 8 shows that $dSCAM$ decides to search this database for less than 10% of the tested documents. This corresponds to a 0.10 probability of false positives. Then, we keep enlarging our only database by progres-

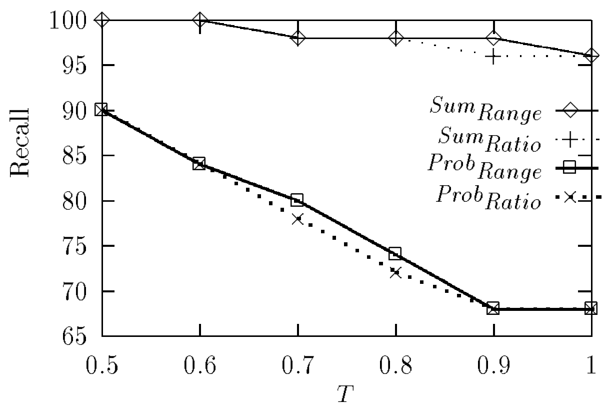

Figure 6: The average recall as a function of the SCAM threshold T (*Registered* suspicious documents; $k = 10$; $T^k = 0.05 \cdot T$).

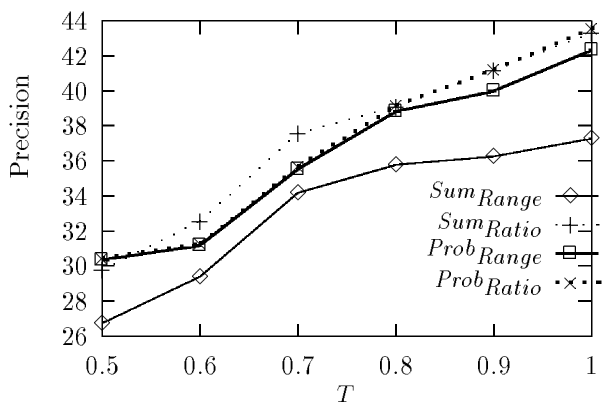

Figure 7: The average precision as a function of the SCAM threshold T (*Registered* suspicious documents; $k = 10$; $T^k = 0.05 \cdot T$).

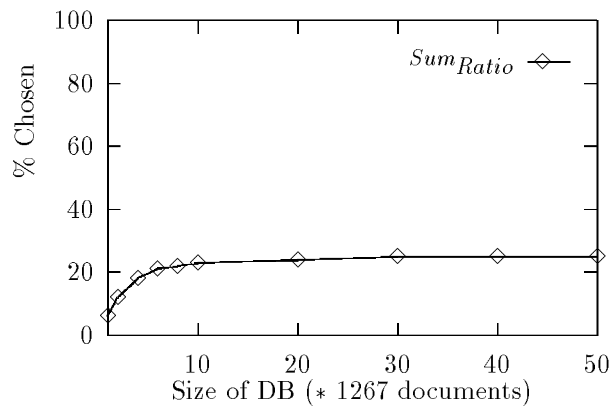

Figure 8: The average number of times that *dSCAM* (incorrectly) chooses to search the (growing) database, as a function of the size of the database (*Disjoint* suspicious documents; Sum_{Ratio} strategy).

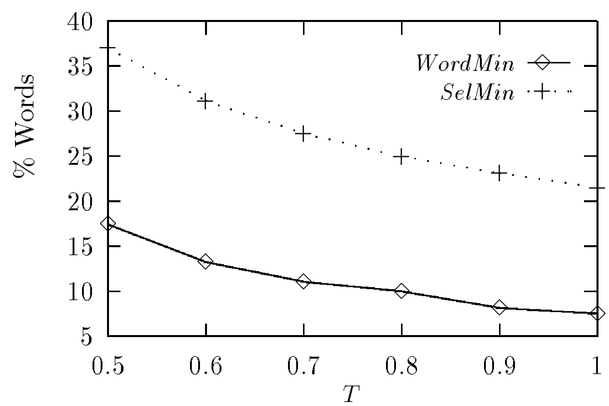

Figure 9: The percentage of words of the suspicious documents that are included in the query to extract the potential copies from the databases (*Registered* suspicious documents; *Ratio* strategy).

sively adding the documents from our original databases, until the database consists of all 63,350 documents. As we see from Figure 8, after an initial deterioration, *dSCAM* stabilizes and chooses to search the database around 25% of the time. These important results show that *dSCAM* scales relatively well to larger databases. That is, the probability of false positives is relatively insensitive (after an initial rise) to database size. Notice, incidentally, that the 25% false-positive probability can be made smaller by changing the T^k and k values (at a cost in false negatives). So the key observation from this figure is simply that the value is flat as the database size grows.

Our final set of experiments is for the results of Section 6. In that section we studied how to choose the query for each database that *dSCAM* selects. These queries retrieve all potential copies of the suspicious documents. There are many such queries, though. We presented two cost models, and showed algorithms to pick the cheapest query for each model.

Under our first cost model, *WordMin*, we minimize the number of words in the queries that we construct. Thus, we choose a minimum set of words for our query from the given suspicious document. Figure 9 shows the percentage of words in the suspicious document that are chosen to query the databases, for the *Registered* documents and for different values of T. The number of words in the queries decreases as T increases. In effect, Condition 4 in Section 6 becomes easier to satisfy for larger values of T. For example, for $T = 0.80$ and *Ratio*, we need on average 9.99% of the suspicious-document words for our queries. If a particular database cannot handle so many words in a query, we should partition the query into smaller subqueries, and take the union of its results. As expected, the number of words chosen using the *SelMin* cost model is higher, because this cost model focuses on the selectivity of the words, and not on the number of words chosen.

While our second cost model, *SelMin*, uses the word selectivities, the *WordMin* cost model ignores these selectivities. Therefore, we analyze the selectivity for the queries to know what fraction of each database we will retrieve

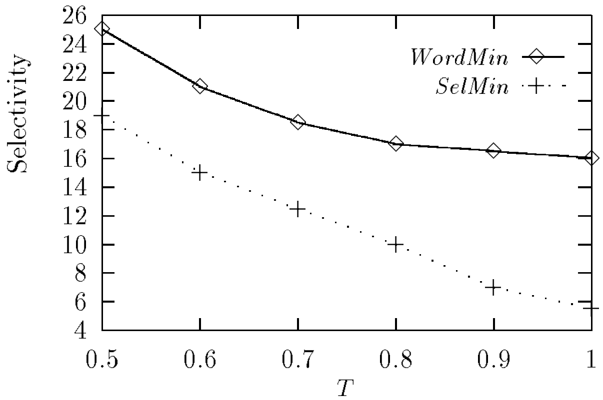

Figure 10: Average selectivity of the queries used to extract the potential copies from the databases, as a function of the SCAM threshold T (*Registered* suspicious documents; *Ratio* strategy).

with such queries. Figure 10 shows the average value of this selectivity for the *Registered* suspicious documents.

The number of query words and the added selectivity of the query words are relatively high. However, if all a database has is a Boolean-query interface, we have no choice but to ask the right queries to the database to extract all the potential copies of a suspicious document. (How to deal with a vector-space query interface [13] is part of our future work.) The results above show that we can do substantially better than the brute-force approach (i.e., when we use all the words in the suspicious document to build a big "or" query) by writing the queries as in Section 6.

We have also explored liberal techniques to extract the potential copies from a database. These liberal techniques might have false negatives (i.e., they might miss some potential copies) and they have much fewer false positives (i.e., they retrieve fewer documents that are not potential copies). Due to space limitations, we cannot describe these techniques here. However, we report some numbers for $T = 1$ to give an idea of the promising results that we obtained. For example, we queried the databases using only the 10% rarest words in the suspicious documents. These queries had an average selectivity of 0.49% (i.e., these "or" queries retrieved on average less than 1% of the database documents), and an average recall of 94% (i.e., these queries retrieved on average 94% of the potential copies). In contrast, the *WordMin* queries for $T = 1$ have fewer words on average (around 8% of the words), but their selectivity is much higher (around 16%). The *SelMin* queries for $T = 1$ have over 20% of the document words in them, and their average selectivity is still higher than that of the liberal technique (over 5%). Of course, recall is perfect for *WordMin* and *SelMin*, while this is not the case for the liberal techniques.

8 Related work

Protecting digital documents from illegal copying has received a lot of attention recently. Some systems favor the copy *prevention* approach, for example, by physically isolating information (e.g., by placing information on stand-alone CD-ROM systems), by using special-purpose hardware for authorization [14], or by using *active* documents (e.g., documents encapsulated by programs [15]). We believe such prevention schemes are cumbersome, and may make it difficult for honest users to share information. Furthermore, such prevention schemes can be broken by using software emulators [16] and recording documents. Instead of placing restrictions on the distribution of documents, another approach to protecting digital documents (one we subscribe to) is to *detect* illegal copies using registration server mechanisms such as SCAM [5, 6] or COPS [16]. Once we know a document to be an illegal copy, it is sometimes useful to know the originator of the illegal copy. There have been several proposals [17, 18] to add unique "watermarks" to documents (encoded in word spacing or in images) so that one can trace back to the original buyer of that illegal document.

A variety of mechanisms have been suggested for registration servers. In [19], a few words in a document are chosen as *anchors* and checksums of a following window of characters are computed. "Similar" files can then be found by comparing these checksums that are registered into a database. This tool is mainly intended for file management applications, and detection of files that are very similar, but not for detecting small text overlaps. The COPS [16] and SCAM registration servers however were developed to detect even small overlaps in text.

dSCAM builds on work in the resource-discovery area. (See [20, 21] for surveys.) This work usually focuses on finding the "best" sources for a query, where the best sources are usually those with the largest number of "relevant" documents for the query. (See for example [22], [10, 11], and [23].) These schemes are not tuned to choose databases with a potential copy of a suspicious document, in the sense of Section 2. Our problem requires that we identify databases even if they contain a single document that overlaps a suspicious document significantly.

9 Conclusion

Discovering a potential copy that might exist in one of many databases is a fundamentally difficult problem. One might say that it is harder than finding a "needle in a haystack:" the haystack is distributed across the Internet, we do not want *similar* items (e.g., a nail), and we also want to find any *piece* of the needle if it exists. It is a harder problem than simply finding similar items, as in traditional information retrieval. Given this difficulty it is somewhat surprising that *dSCAM* performs as well as we have found, especially when one considers the relatively small amount of index information it maintains. It is true that *dSCAM* can miss some potential copies or can lead us to sites without copies, but with the right algorithm and parameter settings, these errors can be made tolerable. For example, we found that *dSCAM* can miss fewer than 5% of the sites with potential copies, and for the sites it does lead us to, they actually have a potential copy roughly half the time.

dSCAM performs best when it only considers about 10% of the words in the suspicious document, those that are the "rarest." Intuitively, these rare words act as a "tell-tale signature" that makes it easier to pick out the target databases. We believe that this is the main reason that *dSCAM* performs better than one would expect, given the difficulty of the problem at hand. Some pirates may make it harder for *dSCAM* to detect these signatures by changing these rare words, but this is not a significant problem since our goal is to prevent widespread and direct copying of documents.

We believe that copy discovery will be an important service in distributed information systems. It will not prevent people from making illegal copies, but having effective discovery mechanisms (together with copy tracing schemes) may dissuade people from large scale duplication.

References

[1] P. J. Denning. Editorial: Plagiarism in the web. *Communications of the ACM*, 38(12), December 1995.

[2] A. Tal and R. Alonso. Commit protocols for externalized-commit heterogeneous database systems. *Distributed and Parallel Databases*, 2(2):209–34, April 1994.

[3] Y. E. Ioannidis, R. T. Ng, K. Shim, and T. K. Sellis. Parametric query optimization. In *Proceedings of the 18th International Conference on Very Large Data Bases*, pages 103–14, Vancouver, August 1992.

[4] T. Y. C. Leung and R. Muntz. Query processing for temporal databases. In *Proceedings of the 6th International Conference on Data Engineering*, pages 200–8, February 1990.

[5] Narayanan Shivakumar and Héctor García-Molina. SCAM: A copy detection mechanism for digital documents. In *Proceedings of the 2^{nd} International Conference in Theory and Practice of Digital Libraries (DL'95)*, Austin, Texas, June 1995.

[6] Narayanan Shivakumar and Héctor García-Molina. Building a scalable and accurate copy detection mechanism. In *Proceedings of the 1^{st} ACM Conference on Digital Libraries (DL'96)*, Bethesda, Maryland, March 1996.

[7] Narayanan Shivakumar and Héctor García-Molina. Information on SCAM. Available as `http://www-db.stanford.edu/~shiva/SCAM/scamInfo.html`.

[8] Tak W. Yan and Héctor García-Molina. Duplicate detection in information dissemination. In *Proceedings of the 1995 Very Large Databases Conference (VLDB'95)*, Zurich, Switzerland, September 1995.

[9] R. E. Kahn. Deposit, registration and recordation in an electronic copyright management system. Technical report, Corporation for National Research Initiatives, Reston, Virginia, August 1992.

[10] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. The effectiveness of *GlOSS* for the text-database discovery problem. In *Proceedings of the 1994 ACM SIGMOD Conference*, May 1994.

[11] Luis Gravano and Héctor García-Molina. Generalizing *GlOSS* for vector-space databases and broker hierarchies. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 78–89, September 1995.

[12] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms*. The MIT Press, 1991.

[13] Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison Wesley, 1989.

[14] G. J. Popek and C. S. Kline. Encryption and secure computer networks. *ACM Computing Surveys*, 11(4):331–356, December 1979.

[15] G. N. Griswold. A method for protecting copyright on networks. In *Joint Harvard MIT Workshop on Technology Strategies for Protecting Intellectual Property in the Networked Multimedia Environment*, April 1993.

[16] Sergey Brin, James Davis, and Héctor García-Molina. Copy detection mechanisms for digital documents. In *Proceedings of the ACM SIGMOD Annual Conference*, San José, May 1995.

[17] J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman. Document marking and identification using both line and word shifting. Technical report, AT&T Bell Laboratories, 1994.

[18] A. Choudhury, N. Maxemchuk, S. Paul, and H. Schulzrinne. Copyright protection for electronic publishing over computer networks. Technical report, AT&T Bell Laboratories, 1994.

[19] U. Manber and S. Wu. Glimpse: A tool to search through entire file systems. In *Proceedings of the Winter USENIX Conference*, January 1994.

[20] Michael F. Schwartz, Alan Emtage, Brewster Kahle, and B. Clifford Neuman. A comparison of Internet resource discovery approaches. *Computer Systems*, 5(4), 1992.

[21] Katia Obraczka, Peter B. Danzig, and Shih-Hao Li. Internet resource discovery services. *IEEE Computer*, September 1993.

[22] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In *Proceedings of the 18^{th} Annual SIGIR Conference*, 1995.

[23] Mark A. Sheldon, Andrzej Duda, Ron Weiss, James W. O'Toole, and David K. Gifford. A content routing system for distributed information servers. In *Proceedings of the 4^{th} International Conference on Extending Database Technology*, 1994.