Scalable Document Fingerprinting
(Extended Abstract)

Nevin Heintze
Bell Laboratories
Murray Hill, NJ 07974
nch@research.bell-labs.com

Abstract

As more information becomes available electronically, document search based on textual similarity is becoming increasingly important, not only for locating documents online, but also for addressing internet variants of old problems such as plagiarism and copyright violation.

This paper presents an online system that provides reliable search results using modest resources and scales up to data sets of the order of a million documents. Our system provides a practical compromise between storage requirements, immunity to noise introduced by document conversion and security needs for plagiarism applications. We present both quantitative analysis and empirical results to argue that our design is feasible and effective. A web-based prototype system is accessible via the URL http://www.cs.cmu.edu/afs/cs/user/nch/www/koala.html.

1 Introduction

As more information becomes available on the internet, searching for documents based on textual similarity is becoming increasingly useful. For example, suppose that I have an early version of a research article, and that I want to determine where it was eventually published (so that I can cite it appropriately), or find if there is a more up-to-date version that fixes previous errors, or perhaps locate an old technical report version that contains more complete proofs or implementation details. Given that the various version of an article are likely to have a significant amount of the text in common, a textually related document search is very likely to locate the earlier/later versions of the article (assuming they are accessible to the search engine).

Another application of this kind of search is detection of copyright violations and plagiarism on the internet. It is now all too easy to obtain a paper over the internet, modify the cover page to insert your name in place of the original authors, perhaps change the title and abstract, and submit the paper as if it were your own. Given the large number of conferences and journals, and the imperfections of the refereeing process, the chances of such a paper slipping through undetected are significant. This kind of plagiarism has been successfully carried out in the past and was the subject of a recent editorial in the Communications of the ACM [4]. Arguably it was inevitable that the individual involved would be discovered. What is surprising is the scope of his activities, the time it took before he was discovered, and that he was able to continue with some success even after his case was well known.

More generally, the internet raises major plagiarism and copyright problems because of the ease by which documents may be copied and modified. Resources such as newsfeed wire services, newspaper articles, netnews articles, online books and so forth are increasingly at risk. As a result, many important sources of information are not made available online because the individuals and organizations that own the information find these risks unacceptable. On another front, there is increasing concern about plagiarism of coursework papers at the college undergraduate and graduate level.

There are two approaches to this plagiarism/copyright problem. In the first, digital sig-

The design and implementation of this system was carried out in the Summer of 1995 while the author was at Carnegie Mellon University, supported by the US Postal Service. This work is the opinion of the author and does not necessarily represent the view of his employer or the US Postal Service.

natures or watermarks are included in a document [3, 10]. These signatures may involve the use of particular word spacings or checksums of components of a document. Unfortunately, these signatures can often be deleted (particularly if the document is translated from one format to another). Moreover, these approaches are not well suited for detecting partial matches involving modified documents. The second approach involves the registration and storage of documents and subsequent textual matching with new documents to track copies and modified versions [1, 6, 11]. This approach has been used in a number of implemented systems [2, 7, 8, 9].

In this paper we consider the general problem of textual matching for document search and plagiarism/copyright applications. We focus on the question of how to build a practical online system that provides reliable search results for a data set of the order of a million documents, while using modest resources (0.5G of disk, i.e. about 500 bytes per document). Three central issues arise: first, how can we meet the space constraints; second, how can we provide reliable search in the presence of noise (to perform textual matching, we must convert documents into text, and this is typically a very noisy and unreliable process), and third, how can we meet the security needs of plagiarism applications (if it is easy to circumvent the matching of related documents by making a handful of selective text changes, then the system will be of little value for plagiarism detection). We present both quantitative and empirical analysis of our system. A web-based prototype system called Koala is accessible via the URL `http://www.cs.cmu.edu/afs/cs/user/nch/www/koala.html`.

Related Work

Most closely related to our work is the Stanford Digital Library work on SCAM [7, 8, 9]. Their approach uses relative word frequencies. Similarity between documents is based on a modified cosine similarity measure. The storage requirements for this approach are approximately 30%-65% of the size of the original documents ([9] reports data storage requirements of between 37MB and 79MB for a document set of 120MB, depending on the chunking approach used). The use of words as the basis for document analysis requires textual representations of documents that faithfully preserve word boundaries.

Our approach is based on selecting a set of subsequences of characters from a document and generating a fingerprint based on the hash values of these subsequences. Similarity between two documents is measured by counting the number of common subsequences in fingerprints (the reliability of this measure is critically dependent on how the subsequences are selected from a document). One major difference from SCAM is that we store less than 500 bytes per document (400 bytes for the actual fingerprint, about 20 bytes for document identification such as URL and email information, and some overhead for indexing), which typically means that our storage requirements are about 0.5-1% of the size of the original documents, almost two orders of magnitude less than the requirements for SCAM. Another significant difference is that we accept documents in a variety of different formats (including Postscipt generated from TeX, PageMaker, Microsoft Word and FrameMaker). To perform textual comparison on such documents, we must first convert them to text. The problem is that such conversions introduce many errors. In particular, punctuation and word spacing are very unreliable. An early version of our work was word based, but did not give satisfactory results. Instead, we have adopted techniques that are largely insensitive to word boundaries and other common errors introduced by document conversion.

The idea of selecting a set of pieces of a document and then hashing these to obtain a document fingerprint is used by Manber in *sif*, a tool for finding similar files in a large file system. However, the motivations for *sif* and our work are very different. First, *sif* focuses on similarities of 25% and higher, whereas we strive to provide reliable information for matches of 3-5% and lower (the key is to reduce the number of false positives while retaining important matches). Second, our work addresses the problem of tolerance to noise. Third, our work addresses special issues related to plagiarism applications. As a result, the *sif* and Koala system designs differ quite substantially.

2 Textual Relationships

We first address the question of what kinds of textual relationships should be considered significant. Checking for exact matches is easy, but not satisfactory. For example it would miss matches where one document is the result of minor edits of the other. It would also misses identical documents that differ because of noise introduced by document translation processes (Postscript to text conversion, OCR, etc.). Moreover, it would ignore most of the interesting textual relationships between documents. In this paper, we consider the following general kinds of relationships to be significant:

1. Identical documents.
2. Documents that are the result of small edits/corrections to other documents.
3. Documents that are reorganizations of other documents.
4. Documents that are revisions of other documents.
5. Documents that are condensed/expanded versions of other documents (e.g. journal versus conference versions of papers).
6. Documents that include portions (say several hundred words) of other documents.

We require that the first five classes of relationships be identified with very high probability; for the remaining class we will tolerate a small number of false positives and false negatives.

3 Fingerprinting

3.1 Full Fingerprinting

Consider the following simple fingerprinting scheme: given a document, let the fingerprint of the document consist of the set of all possible document substrings of length α. There are $l - \alpha + 1$ such substrings, where l is the length of the document. Comparing two documents under this scheme is simply a matter of counting the number of substrings common to both fingerprints: if we compare a document A of size $|A|$ against a document B, and if n is the number of substrings common to both documents then $n/|A|$ is the measure of how much of A is contained in B. If α is chosen appropriately, this simple fingerprint gives reliable document matching results. We refer to this scheme as *full fingerprinting.* Although it is not practical (for space reasons), it is a very useful measure of document similarity, and we shall use it for the evaluation of our system in Section 7. (We remark that it would not be a good idea to construct fingerprints by chopping up the document into *floor*(l/α) substrings by making a cut at every α^{th} character, because insertion of a character at the start of the document would shift the substrings by 1 and the resulting fingerprint would be a poor match to the original, even though the two documents are almost identical).

The choice of α in this fingerprinting scheme is particularly important, and is subject to two conflicting constraints. If α is too small, then there will be many false matches (e.g. if α is the size of a word, then the scheme reduces to little more than comparing lists of words in documents, a poor similarity metric). If α is too large, then there will be many false negatives because one character change can affect α substrings in the fingerprint (e.g. if α is the size of a paragraph, then a single character change in a paragraph would effectively prevent matching for the entire paragraph). We remark that there is no "right" value for α: we cannot quantitatively or empirically calculate a definitive value for α. In essence, the choice of α defines the notion of document similarity for the system. Different values of α will be useful for different kinds of searches. The value of α used in this paper is effectively around 30–45 (more precisely, our strings consist of 20 character sequences of consonants; we discuss this in detail in Section 5). We investigate the effect of different values of α in Section 7.

3.2 Selective Fingerprinting

As mentioned earlier, full fingerprinting is conceptually useful but it is not practical because of the sizes of the fingerprints generated. To reduce the size of a fingerprint, we select a subset of the substrings from the full fingerprint. Since the goal of our work is to treat documents that vary in size from several thousand words to several hundred thousand words while meeting very tight space constraints, we have chosen to select a fixed number of substrings, independent of the size of the document. We call this *fixed size selective fingerprinting.* (An alternative is to select a fixed proportion of the substrings, so that the size of the selective fingerprint is proportional to the size of the document. The main drawback of this alternative is space consumption: to provide accurate fingerprinting of documents with several thousand words we would need a fingerprint containing 50-100 substrings, and this means fingerprints of size 5000-10000 for documents containing several hundred thousand words.)

The design a fixed size selective fingerprinting system revolves around two choices: fingerprint size and selection strategy (that is, which substrings do we select from the full fingerprint). We discuss these in the next two subsections.

3.3 Fingerprint Size and Security

We employ different size fingerprints for storage and search: the fingerprints we store in the database have size 100, but the fingerprints used for searching have size 1000. Importantly, the search fingerprint for a document is a strict superset of the fingerprint used for storage. There are two reasons for this choice. The first is reliability, and is intimately connected with design decisions discussed in Subsection 6.1. The second motivation is security: we want our system to be resilient under attack by would-be plagiarists. To illustrate the issue, first suppose that we use fixed size fingerprints of 100 for both storage and search and that the selection strategy is publicly known. In this case, it would be easy for a plagiarist to determine which 100 substrings are part of the fingerprint, and make 100 changes at the appropriate places in the plagiarized version so that it no longer matches the original. If, instead of making the selection strategy public, we keep it secret (for example, we could use some secret seed value to guide the selecting strategy), then by a trial and error process, it is still possible to find an appropriate set of 100 changes (for example, one could chop the original document into pieces and search separately on these pieces to identify the selected substrings).

We provide better security by periodically changing the stored fingerprint of a document. The use of two fingerprints provides a particularly convenient way to achieve this: we obtain a new stored fingerprint by simply choosing a different subset of the search fingerprint (since the ratio of sizes involved is 100:1000, this still gives considerable scope for change). The advantage of this approach is that we do not need to change the search engine (i.e. we still generate the same search fingerprint) to search against the modified stored fingerprint. This is important, because it allows us to change the database incrementally: we can update the stored fingerprints of a few documents at a time in a transparent manner. To support this process, we maintain a list of URLs for each fingerprinted document so that we can retrieve the document and recompute its fingerprint as desired. We also maintain a contact email address for each document to help resolve stale URLs. We envision updating fingerprints on a regular basis (perhaps once every six months or year), with irregular updates if there is suspicious activity relating to a document (such as an unusually large number of searches for it).

3.4 Selection Strategy

One simple strategy is random selection. However this gives poor results. For example suppose that we have a document of length 50,000 (which gives rise to about 50,000 possible substrings of length α) and we use 100 substring fingerprints for storage and 1000 substring fingerprints for search. Now consider matching the document against itself. The probability that any particular substring appears in the storage fingerprint is $100/50000 = 1/500$. Hence, the expected number of substrings from the search fingerprint that match the storage fingerprint is $1/500 \times 1000 = 2$ (i.e. a match ratio of about 2%). The results are of course much worse for documents that are related but not identical.

To provide more reliable matches, the selection strategy must select similar substrings from similar documents. One approach is employ a string hash function, and then a fingerprint of size n can be obtained by picking the n substrings with the lowest hash values. The approach we use is related to the hash function approach and gives similar results, but reduces false positives. We defer the details to Section 6.

3.5 Limitations of Fixed Length Fingerprints

An important measure of the reliability of a fingerprinting scheme is how closely its results correlate with those from full fingerprinting. For the fixed length selective fingerprinting approach we have chosen, the correlation is good for documents of similar size, but can become problematic for documents of significantly different size. Specifically, we can show that for documents of identical size, the expected match ratios for fixed length selective fingerprinting are identical to those for full fingerprinting. However for documents of different size, the results can vary by a ratio as high as the ratio of sizes of the two documents. To illustrate the problem, consider the extreme case of matching a document of 1000 words against a document of 100,000 words, and suppose that the smaller document appears once in the larger document. Now if the stored fingerprint of the larger document has size 100, then on average there will be one substring for every 1000 word piece of the larger document. In other words, the stored fingerprint of the larger document will have about one substring in common with the smaller document (i.e. about a 1% match ratio), compared with the 100% match given by full fingerprinting. We are currently investigating ways to address this issue, including using variable sized fingerprints and flagging low match ratios as significant if document sizes vary significantly.

4 Fingerprint Storage

Fingerprints are hashed and stored using a very simple indexed file. Specifically, each substring selected for inclusion in a fingerprint is hashed to a 28-bit unsigned integer. The top 16 bits of this 28-bit hash values are used as an index into a table at the start of the file. This table consists of pointers into 256 word blocks. The first word of each block contains a chain pointer to an overflow block (if there is one), and the second word contains a count of the number of words used in the block. The remaining 254 words are used to store fingerprint entries: each entry consists of the lower 12-bits of the 28-bit substring hash value and a 20-bit document identifier (for a maximum of 1M documents), which gives a total of 32 bits, or one word per fingerprint substring. Since we have 100 substrings per (stored) fingerprint, each fingerprint occupies 400 bytes.

We also store a log of each document in the database. This log includes the document identifier, date, URL and a contact email address. Currently this information is stored as raw ascii and consists of about 50-80 bytes per document. This can be compressed substantially. Early experiments indicate that a factor of 4 should be possible.

The simple indexing scheme we have used has a number of drawbacks. First, if the file only contains a small number of fingerprints then there will be a lot of wasted space because most of the blocks will be nearly empty. Second, as more documents are added, the overhead of following overflow blocks can become significant. For example, if there are 1M documents, there will be 100M words of fingerprints, which will occupy about 400–500K blocks, giving rise to an average chain length of 6–8, or about 6000–8000 disk probes per document search (search fingerprints have size 1000). These issues can be addressed by more sophisticated indexing/disk-management schemes.

5 Document Noise

To generate fingerprints and perform document matching, we must first obtain text versions of documents. Unfortunately, this is an unreliable process that introduces many errors. One of the main formats we wish to support is Postscript. Postscript interpreters can be adapted to produce text, but they are typically slow and often produce poor results. Alternatively, for Postscript output by specific tools, we can exploit the format of Postscript generated by the tool to recover the text quickly and fairly accurately (this is the case for example with TeX/dvips generated Postscript). The problem is that the formats change as the tools evolve, and we need different programs to deal with different Postscript tools.

For Postscript conversion, the main errors introduced involve punctuation, non-alphabetic characters and spacing. In particular, word boundaries are often distorted. There are some secondary problems with vowels and uppercase/lowercase distinctions. We factor out these problems by ignoring all but non-vowel characters and converting everything to lower case. This allows us to use fast Postscript to text converters based on string extraction (the translator we use is a modified version of Jason Black's ps2txt program, which in turn is based on a program by Iqbal Qazi). By focusing on non-vowel characters and converting to lower case, we have obtained very reliable results for Postscipt generated from TeX, PageMaker, Microsoft Word and FrameMaker. Note that by considering only consonants, our approach is not actually based on document substrings, but rather on character subsequences of the original document. We use subsequences of length 20, and given the typical distribution of consonants, this corresponds to spans of about 30–45 characters in the original document.

6 Reducing False Positives

One of the goals of our system is to provide reliable low level match information, and in particular, to reduce the number of false positives. This is important for a number of reasons. First, the identification of low level matches appears to be an interesting search paradigm for locating related documents. Second, it helps offset some of the limitations of fixed length fingerprints. Third, it has important performance implications: in the context of a database of millions of documents, false positives can significantly increase the cost of searching.

6.1 Fingerprint Generation

The selection strategy used to choose the substrings to include in a fingerprint can have a significant impact on the number of false positives. The issue is that some substrings occur significantly more frequently than others – sometimes by many orders of magnitude. Moreover, these frequent substrings are more likely to be selected in a fingerprint if fingerprint construction does not consider substring frequency. To illustrate the potential impact of this, first suppose that each substring is equally likely to appear in a document, and consider comparing two unrelated documents. If the hash function used is well behaved, then the search fingerprint contains 1000 randomly selected elements from a space of 2^{28} elements and the storage fingerprint contain 100 randomly selected elements from the same space. Now, the probability that at least one element from the 2^{28} space will appear in both fingerprints is given by the probability that a particular element appears multiplied by the number of possible elements. This can be approximated by:

$$\left(\frac{100}{2^{28}} \times \frac{1000}{2^{28}}\right) \times 2^{28}$$

which is about 0.00037. Hence, for a database of $1M$ documents, we can expect about 370 random noise hits for each search.

Now suppose that a fingerprint consists of substrings that are 10 times more frequent than the average. Then the probability of a one substring match between two documents increases by a factor of 10×10 to 0.037, or near $40K$ random noise hits for a search against $1M$ documents.

We can significantly reduce false positives by using a substring selection strategy based on frequency measures. One way to do this is to compute the set of all substrings for the document and then pick the least frequently occurring substrings. However this is computationally expensive and does not yield good results because the space of substrings in one document is not a useful indication of the overall frequency of substrings.

Instead, we use a frequency measure based on the first five letters of a substring. This is not only cheaper to compute, but gives useful results. The intuition is that the distribution of five letter sequences in a specific document is a useful approximation to the general distribution of five letter sequences. Hence if we pick substrings whose first five letters occur infrequently in a document, it is likely that the first five letters of these substrings will occur infrequently in general. Substrings whose first five letters occur infrequently are likely to be such that the entire substring occurs relatively infrequently. In Section 7 we give experimental evidence that indicates this technique can reduce false positives by more than a factor of two.

Underlying this approach is the assumption that the substring frequency distribution of (significant) overlapping text segments does not vary substantially from the frequency distribution of other text segments (which would imply that match ratios for related documents are not affected by focusing on infrequent substrings). This assumption appears to be valid in practice.

One problem with the use of five letter frequency distributions for substring selection is that different documents may have slightly different five letter frequency distributions. The use of different size fingerprints for storage and search provides an effective way to address this. To illustrate why, consider a substring s that is common to two documents A and B. If s is selected in B's stored fingerprint (that is, s is "very infrequent" according to B's frequency measure), then although it may not appear in A's stored fingerprint (because the frequency measures for A and B differ), it most probably will appear in A's (much larger) search fingerprint because it is likely to be "moderately infrequently" according to A's frequency measure.

We remark that another way to identify infrequent substrings is to use the fingerprint database to provide frequency measures. One disadvantage of this approach is that fingerprint generation can no longer be performed as a stand-alone operation. However it has the potential to very reliably identify infrequent substrings and deserves further investigation.

6.2 Fingerprint Matching

Some substrings are very common in certain collections of documents. For example, in a technical report series, substrings generated from addresses, funding agencies acknowledgements and strings such as "This work is the opinion of the author and does not necessarily represent the view of his employer or ..." appear in many documents. Such substrings are difficult to recognize within the context of a single document (and so the technique described in the previous section does not detect them), but require the context of a collection of documents.

We use a frequency check during search to isolate these strings. When a search is performed and a particular (hashed) substring is looked up in the database, we check to see if this particular string appears in 4 or more documents. If it does, then we ignore it during the search. However, this raises a security issue: if one could repeatedly add copies of the one document to the database, then eventually all of the substrings of the document would be ignored, and the document would not generate a match. To address this situation, we cap the number of ignored substrings at 10. In Section 7, we show that this technique can reduce false negatives by between 15% and 85%, depending on which of the other checks in this section are deployed. We remark that the values of 4 and 10 deserve further evaluation using different sizes and classes of document sets. We also remark that this technique is an application of a very standard information processing idea: discount the significance of common features.

6.3 Document Prologs

While the use of frequency checks provides one way to ignore common substrings, other techniques can also be useful. In particular, most of the problematic strings such as addresses, funding agencies acknowledgements etc., appear at the start of a document. Also, when a Postscript file is converted to text, the first words of text are often from the preamble of the Postscript file and indicate the tools used to generate the file; they have nothing to do with the actual text of the document.

One simple approach is to ignore the first part of a document. In Section 7 we show that ignoring the first 1000 characters of a document gives useful reductions of false negatives without significantly affecting other matches. Moreover, it is useful in tandem with the technique described in the previous subsection.

7 Results

We now present some experimental results from an implementation of the system. The main data set we use is a collection of 366 technical reports from the Carnegie Mellon University School of Computer Science (our set essentially consists of all reports available online as of August 1995). This set consists of just over 30MB of text. Table 1 gives the distribution of matches when each document is searched against all others. The number in parenthesis in the right hand column is the number of non-identical document matches (i.e. 372 - 366). Note that there were 763 matches at the 1% level, which is about two 1% matches for each document. This is higher than expected. It reflects the fact that the data set has a high degree of low-level correlation: it is generated by a relatively small group of people with shared experiences and background (for example, people in this group tend to cite each other's work). Some of these low-level commonalities would be removed by the technique described in Section 6.2, however the data set is too small for this technique to remove all of them. The web-based implementation of the system has a larger database (about 3000 documents) that includes the technical report data set. When a technical report is matched against this larger database, we typically find twice as many 1% matches as we obtain when matching against just the technical report database. This means that the 2600 other documents are generating about as many 1% matches as the technical reports (i.e. a 1% match rate that is 7 times lower). This is partly because this data set is larger, but also because it does not have the same degree of low-level correlation (the 2600 "other" documents are technical reports and papers from a wide variety of different institutions).

Table 2 compares fixed size selective fingerprinting with full fingerprinting for a small collection of documents, and provides evidence of the reliability of selective fingerprinting. The left hand column gives the match ratios reported by our system, and the right hand column gives the results for full fingerprinting as both a match ratio and as raw data (common-substrings/total-substrings). For this small collection of documents, selective fingerprinting gives match ratios that are within a factor of two of full fingerprinting, and usually much closer.

To establish the utility of the techniques for reducing false positives, we present two collections of data. Table 3 considers both (a) dropping frequent substrings during searching and (b) cutting the first 1000 characters of a document. The first line of the table gives the match ratio distribution for matching all documents against all documents with both techniques (a) and (b) enabled. The second line disables just (a), and the third line disables just (b). The final line disables (a) and (b). Both techniques are very useful in isolation, but it is clear that they address overlapping issues. They are, however, sufficiently different to be useful in tandem.

Table 4 considers the effectiveness of focusing on

match range	1%	2%	3%-5%	6%-10%	11-20%	21-50%	more than 50%
document count	763	98	53	25	15	23	372 (6)

Table 1: Match Distribution for CMU-SCS Technical Report Data Set.

selective fingerprint match ratio (%)	full fingerprint match ratio (%)	matches/total
45	57	13206/22994
9	8.7	2005/22994
5	12	2766/22994
29	55	22541/41010
1	0.01	5/41010
1	0.01	5/41010
1	0.2	89/41010
1	0.08	34/41010
3	2.6	670/25386
1	3.0	782/25386
0	0	0/22994
0	0.3	76/22994
0	0.03	8/22994
0	0.16	38/22994
0	0.19	45/22994

Table 2: Comparison of Selective Fingerprinting and Full Fingerprinting.

	1%	2%	3%-5%	6%-10%	11-20%	21-50%	more than 50%
baseline	763	98	53	25	15	23	372 (6)
freq. check off	896	107	53	25	15	23	372 (6)
include start	2618	194	100	32	15	19	374 (8)
both disabled	17598	2415	334	35	17	19	374 (8)

Table 3: Reducing False Positives I: frequency checks (search) & document preamble.

infrequent substrings during fingerprint generation (Subsection 6.1). The first line give the baseline match ration distribution, and the second line gives the same results when substrings are selected without regard for frequency. This table indicates a reduction of false positives by more than a factor of two.

The next two tables investigate the effect of fingerprint size. In Table 5, the size of the stored fingerprint is varied from 10 to 500 (the baseline value is 100), while the search fingerprint remains constant at 1000. In Table 6, the size of the search fingerprint is varied from 100 to 5000 (the baseline value is 1000), while the search fingerprint remains constant at 100. The results indicate that our system is surprisingly insensitive to changes in fingerprint sizes. The use of 100 for storage and 1000 for searching yields a good tradeoff between search reliability and the level of positive matches.

Finally, table 7 shows the effect of changing α, the length of character subsequences, from 10 to 50 (the baseline value is 20). As expected, decreasing α has the effect of significantly increasing the number of low level matches. Increasing α has the effect of decreasing both the number of low-level and high-level matches.

8 Conclusion

We have presented a system for document comparison based on textual similarity. Target applications include related document searches and copyright/plagiarism protection. Our system uses fixed size selective fingerprints based on document substrings, and supports reliable and accurate document comparison with very small fingerprints (about 400 bytes per document). The main novelties of our work are (a) very low storage requirements (almost two orders of magnitude less than competing systems), (b) resilience to noise in documents (such as that introduced by conversion from Postscript to text), (c) security measures to the improve dependability of plagiarism searches in the context of an active adversary, and (d) significant reduction of false positives.

References

[1] C. Anderson, "Robocops: Stewart and Feder's mechanized misconduct search", *Nature*, 350(6318):454-455, April 1991.

[2] S. Brin, J. Davis and H. Garcia-Molina, "Copy Detection Mechanisms for Digital Documents", *Proceedings of the ACM SIGMOD Annual Conference*, May 1995.

[3] J. Brassil, S. Low, N. Maxemchuk and L. O'Gorman, "Electronic Marking and Identification Techniques to Discourage Document Copying", *Journal on Selected Areas in Communications*, Volume 13, Number 8, October 1995.

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

[5] U. Manber, "Finding similar files in a large file system", *Proceedings of the 1994 USENIX Conference*, pp. 1-10, January 1994.

[6] A. Parker and J. O. Hamblen, "Computer algorithms for plagiarism detection", *IEEE Transactions on Education*, 32(2):94-99, May 1989.

[7] N. Shivakumar and H. Garcia-Molina, "SCAM: A Copy Detection Mechanism for Digital Documents", *Proceedings of the 2nd International Conference on Theory and Practice of Digital Libraries*, 1995.

[8] N. Shivakumar and H. Garcia-Molina, "The SCAM Approach to Copy Detection in Digital Libraries", *Diglib Magazine*, November 1995.

[9] N. Shivakumar and H. Garcia-Molina, "Building a Scalable and Accurate Copy Detection Mechanism", *Proceedings of the 3nd International Conference on Theory and Practice of Digital Libraries*, 1996.

[10] N. R. Wagner, "Fingerprinting," *Proceedings of the 1983 Symposium on Security and Privacy*, pp. 18-22, April 1983.

[11] D. Wheeler "Computer networks are said to offer new opportunities for plagiarists", *The Chronicle of Higher Education*, pp. 17, 19, June 1993.

	1%	2%	3%-5%	6%-10%	11-20%	21-50%	more than 50%
baseline	763	98	53	25	15	23	372 (6)
random	1634	177	104	43	17	29	376 (10)

Table 4: Reducing False Positives II: use of infrequent substrings in fingerprints.

<table>
<tr><th>stored fingerprint</th><th>1%</th><th>2%</th><th>3%-5%</th><th>6%-10%</th><th>11-20%</th><th>21-50%</th><th>more than 50%</th></tr>
<tr><td>10</td><td colspan="4">134</td><td>12</td><td>17</td><td>370 (4)</td></tr>
<tr><td>20</td><td colspan="3">217</td><td>19</td><td>16</td><td>21</td><td>370 (4)</td></tr>
<tr><td>50</td><td colspan="2">442</td><td>43</td><td>39</td><td>18</td><td>21</td><td>369 (3)</td></tr>
<tr><td>100</td><td>763</td><td>98</td><td>53</td><td>25</td><td>15</td><td>23</td><td>372 (6)</td></tr>
<tr><td>200</td><td>1507</td><td>43</td><td>49</td><td>21</td><td>17</td><td>19</td><td>372 (6)</td></tr>
<tr><td>500</td><td>2162</td><td>32</td><td>38</td><td>25</td><td>14</td><td>18</td><td>372 (6)</td></tr>
</table>

Table 5: Effects of Varying Stored Fingerprint Size.

search fingerprint	1%	2%	3%-5%	6%-10%	11-20%	21-50%	more than 50%
100	364	54	36	22	20	18	374 (8)
200	520	66	50	19	17	24	371 (5)
500	777	89	52	30	17	20	372 (8)
1000	763	98	53	25	15	23	372 (6)
2000	976	91	63	21	15	20	373 (7)
5000	1064	101	62	32	16	21	372 (6)

Table 6: Effects of Varying Search Fingerprint Size.

α	1%	2%	3%-5%	6%-10%	11-20%	21-50%	more than 50%
10	6239	363	105	33	24	22	375 (9)
20	763	98	53	25	15	23	372 (6)
30	411	70	49	22	10	20	371 (5)
40	366	55	42	22	15	17	371 (5)
50	231	37	42	19	13	15	370 (4)

Table 7: Effects of Varying Substring Length.