J Supercomput (2008) 45: 255–276
DOI 10.1007/s11227-007-0171-z

# Lexicon randomization for near-duplicate detection with I-Match

**A. Kołcz · A. Chowdhury**

Published online: 26 January 2008

**Abstract** Detection of near duplicate documents is an important problem in many data mining and information filtering applications. When faced with massive quantities of data, traditional techniques relying on direct inter-document similarity computation are often not feasible given the time and memory performance constraints. On the other hand, fingerprint-based methods, such as I-Match, while very attractive computationally, can be unstable even to small perturbations of document content, which causes signature fragmentation. We focus on I-Match and present a randomization-based technique of increasing its signature stability, with the proposed method consistently outperforming traditional I-Match by as high as 40–60% in terms of the relative improvement in near-duplicate recall. Importantly, the large gains in detection accuracy are offset by only small increases in computational requirements. We also address the complimentary problem of spurious matches, which is particularly important for I-Match when fingerprinting long documents. Our discussion is supported by experiments involving large web-page and email datasets.

**Keywords** Information retrieval efficiency · Spam detection

## 1 Introduction

In recent years, large dynamic document repositories have become commonplace, owing to the Internet phenomenon and to rapid advances in the storage technology. Due to a variety of factors, including redundancy/mirroring, spam and plagiarism, such repositories may contain more than one copy of some documents, where

A. Kołcz
Microsoft Live Labs, 1 Microsoft Way, Redmonnd, WA 98052, USA

A. Chowdhury (✉)
University of Bremen TZI, Bremen, Germany
e-mail: abdur@ir.iit.edu

sometimes the multiple copies are not exactly identical, but are similar enough to be considered as near duplicates. Near-duplicate proliferation is often undesirable, with potential problems including increased storage requirements, decrease in the quality of search engine performance and spam. Large concentrations of duplicate documents may also skew the content distribution statistics with potentially harmful consequences to machine learning applications [27]. Aside from identifying near-duplicates within document repositories, a related important problem is to verify if an outside document may have a near-duplicate in a repository. For example, in email spam filtering, a system may want to examine the stream of incoming messages and verify that none of them can be considered a variation of already known spam.

A number of duplicate-detection schemes have been proposed in the literature (e.g., [5, 6, 9]). Their focus varies from providing high detection rates to minimizing the amounts of computational and storage resources needed by the detection process. With massive document repositories, run-time performance tends to be critical, which makes relatively simple single-hash techniques such as I-Match [9], particularly attractive. Unfortunately, document signatures produced by such techniques are potentially unstable in the presence of even small changes to document content. In applications such as spam filtering, where the adversary often purposely randomizes the content of individual messages to avoid detection [16, 17], such instability is clearly undesirable.

In this work, we propose an extension of the I-Match technique (but also applicable to other single-signature schemes) that significantly increases its robustness to small document changes. Our approach is based on randomization of the I-Match lexicon. The improvements in detection accuracy come at the cost of increased signature count, with an easily controllable trade-off. In a number of experiments, we show consistent superiority of our approach over the original scheme and demonstrate its attractiveness for the target applications of information retrieval and spam filtering.

In addition to tackling the problem of I-Match stability, we also address certain deficiencies of the standard algorithm which may result in spurious matches, particularly for longer document. We suggest an extension that forces the algorithm to either abstain from producing a signature or create a more precise one whenever there is an indication that a regular signature might be unreliable.

The paper is organized as follows: Sect. 2 provides an overview of prior work in the area of near duplicate document detection. Section 3 focuses on the I-Match algorithm and suggests a modification improving its reliability for long documents. Section 4 introduces the randomized lexicon technique and provides a rationale for its effectiveness. In Sect. 5, we discuss applications of near-duplicate detection in filtering of spam and in preprocessing web-page collections prior to retrieval. In Sect. 6, the experimental setup is outlined with the results presented in Sect. 7. The paper is concluded in Sect. 8.

## 2 Near-duplicate detection: prior and related work

The problem of finding duplicate, albeit nonidentical documents has been the subject of research in the text-retrieval and web-search communities, with application focus ranging from plagiarism detection in web publishing to redundancy reduction in web search and database storage. Generally, one distinguishes between the problems of assessing the resemblance and containment of documents [6], although the techniques used to address them are often closely related [6, 35]. In particular, in plagiarism-detection applications, it is often of interest if one document contains the whole or at least the major components (e.g., sentences or paragraphs) of another document. However, in the context of this work, the problem of document containment is less of an issue.

A number of approaches to near-duplicate detection have been proposed, which can roughly be classified as [23]:

- Similarity based: where two documents are considered identical if their distance, according to a measure such as the cosine similarity or the Jaccard distance falls below a certain threshold;
- Signature based: where two documents are considered identical if their projections onto a set of attributes are the same.

Operationally, we can also distinguish between detecting near duplicates from within the scope of a particular document collection and establishing the near-duplicate relationship of documents between an arbitrary source and a particular document collection. The latter view is relevant in applications such as spam filtering or plagiarism detection, where it is desired to find out if new documents are sufficiently similar to any of the existing ones. Similarity-based techniques can be further subcategorized according to the document representation chosen. On one end of the spectrum, the standard information-retrieval bag-of-words representation is used. This approach was taken, for example, in [8, 23, 34]. Alternatively, document decomposition into units larger than words (e.g., word n-grams, sentences, or paragraphs) leads to partial retention of positional information. In particular, shingle-oriented fingerprinting techniques such as COPS [5], KOALA [21], and Document Syntactic Clustering (DSC) [6, 7] view a document as a stream of tokens (e.g., words or characters), which is broken into overlapping or nonoverlapping segments referred to as shingles. Thus, a document is represented as a bag of shingles, with the bag-of-words representation viewed as a special case.

In principle, similarity-based techniques require that all pairs of documents are compared, i.e., each document is compared to every other document and similarity is calculated. For massive data sets, brute-force implementations can be computationally prohibitive, with the theoretical $O(d^2)$ runtime, where $d$ is the number of documents. In practice, only documents with nonzero shingle (or word) overlap need to be considered which, depending on the actual representation, reduces the amount of computation needed. Also, efficient data structures, such as the inverted index [29, 31, 33, 36] are typically deployed.

Several optimization techniques were proposed to reduce the number of comparisons made. In [11], only documents whose sizes are sufficiently close are compared against each other. In the fingerprinting context, frequently occurring shingles may be eliminated [21]. Also, instead of performing similarity computation over the complete sets of shingles, random sampling (e.g., implemented deterministically via min-wise independent hashing [6]) retains only a small subset of shingles per document.

Such simplifications, however, do hinder the accuracy. Since no semantic premise is used to reduce the volume of data, a random degree of fuzziness is introduced to the matching process increasing the rate of false positives, i.e., resulting in relatively nonsimilar documents being identified as potential duplicates.

Even with computational shortcuts, however, similarity based approaches to duplicate detection in large datasets tend to be computationally expensive. For example, with the performance-improving technique of removing shingles occurring in over 1,000 documents and keeping only every twenty-fifth shingle, the implementation of the DSC algorithm is still $O(s \cdot d)$, where $s$ is the average number of shingles per document and $d$ is the number of documents [7]. A more efficient alternative, DSC Super Shingles (DSC-SS), uses concatenations of several shingles called super shingles. Here, instead of measuring resemblance as a ratio of matching shingles, resemblance is defined as matching a single super shingle in two documents, which is much more efficient because it no longer requires the calculation of shingle overlap. A similar approach is used in [18], under the name of Locality Sensitive Hashing (LSH). Here, instead of shingles, a number of $k$-tuples of words are selected from each document via min-wise independent hashing and if two documents have at least one such tuple in common they are considered to be near duplicates.

The super-shingle approach to document comparison is close to the one taken by I-Match [9], where each document is characterized as a filtered set of terms conveniently mapped into a single hash value, with two documents considered as near duplicates if their sets of terms are the same. I-Match overcomes the short-document reliability problem of DSC-SS and is very efficient to implement since document comparison involves just a single hash-table lookup. It does suffer, however, from potential instability of the generated signatures to even small changes in document content [28]. The shortcomings of the super-shingling technique were also addressed in [14] under the name of mega-shingling, which can be interpreted as a special case of LSH.

Lastly, we note the similarity between the problem of near-duplicate detection of text documents and the problems of database deduplication/cleaning [22] and record linkage [38] addressed by the data mining community. The main difference lies in the fact that database records represent structured or semistructured information and are typically much shorter than text documents. Also, the notions of similarity may be data specific (e.g., matching street addresses vs. matching of citations or references [30]), which encourages development of specialized distance metrics [3].

## 3 I-Match and its extensions

Similarity-based duplicate detection approaches inherently map each document to one or more clusters of possible duplicates, depending on the choice of the similarity threshold. While that is appropriate when detecting the similarity of documents or detecting plagiarism, such techniques produce high overhead when large collections are evaluated. The I-Match [9] approach produces a single hash representation of a document, thus guaranteeing that any document will map to one and only one cluster, while still providing fuzziness of nonexact matching. An I-Match signature is determined by the set of unique terms shared by a document and the I-Match lexicon. The signature generation process can be described as follows:

(1) The collection statistics of a large document corpus are used to define an I-Match lexicon, $L$, to be used in signature generation.
(2) For each document, $d$, the set of unique terms $U$ contained in $d$ is identified.
(3) I-Match signature is defined as a hashed representation of the intersection $S = (L \cap U)$, where the signature is rejected if is $|S|$ below a user-defined threshold.

The effectiveness of I-Match relies on the appropriate choice of lexicon $L$. Experimental data suggest that one effective strategy of lexicon selection is to impose an upper and lower limit on the inverted document frequency (*idf*) for words in the document collection (i.e., a variant of Zipfian filtering), since terms that are very frequent (i.e., with very low *idf*), and terms which are very infrequent (i.e., with very high *idf*), tend to be less useful in duplicate detection than terms with mid-range *idf* values [9]. Note that high-*idf* terms may be very effective in pinpointing a particular document, but they also capture misspelled words and other spurious strings, which reduce their value in identifying near rather than exact duplicates. This is especially important in applications such as spam filtering, where much of the document randomization is created on purpose, precisely to make copy detection more difficult. There are no firm guidelines with regards to the choice of *idf* cutoff points, so the selection of a lexicon typically involves a degree of trial and error. Also, criteria not strictly *idf*-based have been suggested. In [11], the *idf* information is combined with in-document term frequency and other statistics to arrive at an importance score called IQ, which is then used in term ranking. In another scheme [24], it was proposed to estimate the stability of each candidate word to small changes in document content and to retain the top-$N$ candidates that are most stable and, at the same time, are relatively frequent in the collection. The description provided in [24] provides few details, however.

I-Match may result in false positives matches if a large document has a very small intersection with $L$. In other words, I-Match signature of a document may become unreliable when $\frac{|S|}{|U|}$ becomes too small, which is illustrated in Fig. 1. Here, the algorithm is applied to two different email messages containing the same banner. Although the relevant content of these messages is contained not in the banner itself, the I-Match signatures for the two messages are identical since the main contents fail to intersect with the I-Match lexicon. In the example shown, the failure is caused by the foreign language of the message payloads in the context of the I-Match lexicon being derived from a corpus of primarily English documents.

Here, we propose an extension of the I-Match technique that addresses the small-intersection problem. Note that the collection statistics define an ordering over the set of all possible lexicon terms, with term *idf* used to determine the sorting order. Assuming that the primary lexicon, $L$, corresponds to a range of *idf* values, we reject all terms with lower *idf* values and define a secondary lexicon, $B$, as one containing the remaining terms ranked according to their increasing *idf* values.

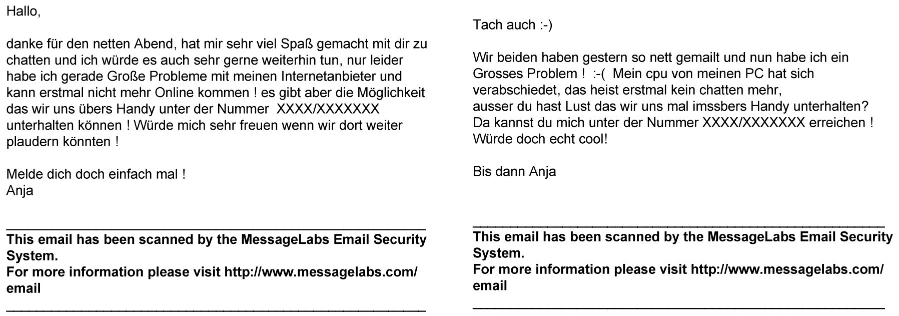
In the modified I-Match procedure, whenever the primary lexicon fails to intersect with sufficiently many terms in $U$, the secondary lexicon is used to supply extra terms, until the number of elements in $S$ exceeds a user-defined threshold. Since the secondary I-Match lexicon may be much bigger than the primary one, and since it may contain many potentially "noisy" terms, the extra terms are added according to their rank order, i.e., the more frequent terms are considered first. This insures that the auxiliary terms contributing to the signature will have *idf* values as close as possible to the primary ones and thus, assuming that the *idf*-ranking of terms is indeed meaningful, the signature will consist of the most relevant terms. In other words, lexicon $L$ defines a $[idf_{\min}, idf_{\max}]$ window in the document frequency range for terms present in the collection. The terms for which $idf < idf_{\min}$ correspond to stop-words and very frequent words, which are not likely to correlate with the gist of any particular document. These ones remain ignored, but we turn our attention to terms for which $idf > idf_{\max}$. These words are less frequent than those found in $L$ and the higher their *idf* value, the more likely they are to result in a uniqueness of a signature to which they contribute. Therefore, to the extent that these terms need to be used to address the problem of the intersection of a document with $L$ being too small, the more frequent ones are considered first to decrease the likelihood of the resulting signature being exact.

The steps involved in signature generation according to the modified I-Match procedure are given below:

(1) The collection statistics of a large document corpus are used to define the primary lexicon, $L$, and the secondary lexicon, $B$, to be used in signature generation; e.g., if $L$ is derived using Zipfian filtering contains terms for which inverse document frequencies are in $[idf_{\min}, idf_{\max}]$, $B$ would contains terms for which $idf > idf_{\max}$ (the size $B$ is controlled by a practical limit the maximum size of a lexicon) ranked according to their *idf* values, with the more frequent ones considered more relevant.
(2) For each document, $d$, the set of unique terms $U$ in $d$ is found.
(3) I-Match signature is defined as a hashed representation of the intersection $S = (L \cap U)$.
(4) If $\frac{|S|}{|U|}$ falls below a user-defined threshold, $S$ is expanded by the minimum number of top-ranking terms $(L \cap U)$ necessary to satisfy the $\frac{|S|}{|U|}$ threshold requirement. The threshold values are determined empirically and may be domain specific (e.g., different for emails and Web documents).
(5) The signature is rejected if either $|S|$ or $\frac{|S|}{|U|}$ fall below their respective thresholds.

In the example provided in Fig. 1, the signatures of the two documents would be either null, or different, provided sufficient vocabulary existed in the secondary lexicon. It is usually more useful to provide an exact signature than none. For example, in the spam filtering application, a spammer may split a single campaign into a number of discrete versions but then send multiple copies of each version, which could be identified by an exact match. To insure that few NULL signatures are generated, the size of the secondary lexicon might have to be very large. In practice, one is likely to be faced with memory constraint limitation. The choice of size of the secondary lexicon is, therefore, likely to represent a trade-off between the available resources (RAM) and the desire to avoid generating NULL signatures.

The proposed modification to I-Match is different from one of the I-Match variants investigated in [9], where a fixed fraction of high-*idf* terms per document was used



Hallo,

danke für den netten Abend, hat mir sehr viel Spaß gemacht mit dir zu chatten und ich würde es auch sehr gerne weiterhin tun, nur leider habe ich gerade Große Probleme mit meinen Internetanbieter und kann erstmal nicht mehr Online kommen ! es gibt aber die Möglichkeit das wir uns übers Handy unter der Nummer XXXX/XXXXXXX unterhalten können ! Würde mich sehr freuen wenn wir dort weiter plaudern könnten !

Melde dich doch einfach mal !
Anja

**This email has been scanned by the MessageLabs Email Security System.**
**For more information please visit http://www.messagelabs.com/email**

Tach auch :-)

Wir beiden haben gestern so nett gemailt und nun habe ich ein Grosses Problem ! :-( Mein cpu von meinen PC hat sich verabschiedet, das heist erstmal kein chatten mehr,
ausser du hast Lust das wir uns mal imssbers Handy unterhalten?
Da kannst du mich unter der Nummer XXXX/XXXXXXX erreichen !
Würde doch echt cool!

Bis dann Anja

**This email has been scanned by the MessageLabs Email Security System.**
**For more information please visit http://www.messagelabs.com/email**

**Fig. 1** Example of failure in the original I-Match to account for longer documents with insufficient overlap with the I-Match lexicons. For a lexicon derived from a predominantly English corpus, the signature will be formed based on the identical English banner in the bottom of the messages. The signatures will thus be identical (assuming that words such as *hat*, *cpu*, *cool* are not part of the lexicon) even though they do not account for the actual relevant content of the messages at all. Note that in this case, the German messages are actually related, but we cannot attribute detection of this correspondence to the strength of the I-Match algorithm itself

in signature generation. Although the other scheme also insures adequate representation of large documents, it may lead to under-representation of short documents, for which the signatures tend to be more brittle than ones based on selecting all terms intersecting with the primary lexicon [9]. Our modification addresses the reliability of the algorithm in terms of avoiding spurious matches. In the next section, we approach the complementary problem of increasing the stability of signatures to document perturbations.

## 4 Decreasing the fragility of I-Match signatures

Ideally, the signature of a document should be insensitive to small changes in document content. For example, in the context of the spam-filtering application, these include changing the order of words in the document, as well as inserting or removing a small number of words. Unlike signature-generation algorithms relying on positional information of words, I-Match is inherently insensitive to changes in the word order, but inserting or deleting a word from the active lexicon will change the value of the signature. Signature brittleness is particularly undesirable given the adversarial nature of spam filtering, where an attacker might attempt to guess the composition of the lexicon and purposely randomize messages with respect to the lexicon's vocabulary.

Let us reverse the roles of the document and the lexicon, however. We can reasonably expect that if a lexicon is modified by small number of additions/deletions, this is unlikely to significantly change the stability of I-Match signatures with respect to the modified lexicon. Moreover, as experimental data suggest [9], similar levels of duplicate detection accuracy can often be obtained by largely nonoverlapping lexicons. A small modification to document content may thus change an I-Match signature due to a particular lexicon, but at the same time, there may exist a number of alternative lexicons for which I-Match performs with equivalent accuracy and for which the signatures may be unaffected by such a change.

The latter observation suggests the benefits of creating multiple signatures per document, which seems to require the presence of multiple different lexicons, selection of which could be nontrivial. We note, however, that such lexicons can be related to one another. In particular, let us consider a setup where a suitable lexicon is chosen and then $K$ different copies of the original lexicon are derived by randomly eliminating a fraction $p$ of terms in the original, i.e., the $K$ extra lexicons are proper subsets of the original. Assuming that $p$ is small, we expect the quality of signatures due to the additional lexicons to be similar to the original. Using the arguments presented above, an extended I-Match signature of a document in the randomized lexicon scheme is defined as a $(K+1)$-tuple, consisting of I-Match signatures due to the original lexicon and its $K$ perturbations. Any two documents are considered to be near duplicates if their extended signatures overlap on at least one of the $K+1$ coordinates.

Let us take a document and modify it by randomly removing or adding a word from the original lexicon, with $n$ such changes in total (note that changes involving vocabulary outside of the original lexicon cannot affect the extended I-Match signature). Each such change will necessarily alter the signature according to the original lexicon, whereas the probability that at least one of the $K$ additional signatures will be unaffected by such a change can be estimated as:

$$1 - \left(1 - p^n\right)^K . \tag{1}$$

This is derived as follows: For a particular perturbation of the original lexicon, a change to the document contents will not affect the signature as long as the change occurs within the subset of the original lexicon that is missing in the perturbation, which occurs with the probability of $p$. Assuming that $n$ is much smaller than the size of the missing subset, the probability that $n$ such changes will preserve the signature can be approximated as $p^n$. Since the $K$ additional lexicons were generated independently from one another, the process in which a number of the signatures is changed in response to modifications to the document can be modeled as $K$ Bernoulli trials. Accordingly, the probability that all $K$ signatures will change is equal to $(1-p^n)^K$ and, conversely, the probability that at least one of them will be unaffected is given by (1).

Equation (1) can be seen as the stability of the extended I-Match signature to changes that are *guaranteed* to affect the I-Match signature according to the original lexicon alone. As illustrated in Fig. 2, at the cost of using a few extra lexicons, the stability of I-Match signatures can be increased significantly. Equation (1) guides the process of choosing the value of $K$ is practice, and there is a natural trade-off between the increase in reliability and the extra computational resources needed to maintain additional lexicons and to compute multiple signatures instead of just one.

Our approach is related to the supershingling technique of [6] and the locality sensitive hashing (LSH) [15, 18]. As in those techniques, a number of signatures is generated per document and if a pair of documents has at least one of them in common, the documents are considered to be near-duplicates. The differences lie in how the individual signatures are generated. The supershingle approach is based explicitly on the concept of syntactic similarity using word n-grams as the basic unit

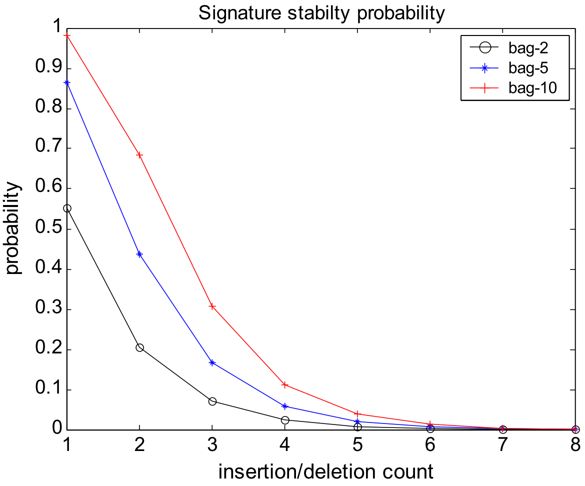


**Fig. 2** Stability of bagged I-Match signatures under random insertion/deletion of words in the original document for the case of $p = 0.33$. The y-axis corresponds to the probability that the extended I-Match signature will not be affected by a change to the document contents. *Bag-n* signified that $n$ randomized lexicons were used

and is thus more vulnerable to radical changes to word ordering (e.g., not uncommon in spam). Unlike lexicon randomization and LSH, the number of signatures (i.e., supershingles) generated per document depends on document size, and as noted in [6], the supershingle approach tends to lose reliability in the case of short documents, which was one of the factors motivating the development of I-Match [9].

Locality sensitive hashing creates a fixed number of signatures per document, where each signature can be seen as the result of a specific randomization of word order. The LSH approach does not explicitly rely on a lexicon, so any word can potentially contribute, although very frequent words are typically ignored. The technique is based on the use of a number of different hash functions, where a particular hash function imposes a random ordering over the set of all possible words, and the top $k$ words[1] from the intersection between the document and the ordered word list are concatenated and hashed to produce a signature. To generate alternative signatures, a number of different hash functions from the same family are used, which leads to different random word orderings. LSH explicitly mandates the number of terms that constitute a document signature. Given that little word filtering is applied and that the ranking of words is random, the choice of $k$ may be critical, especially for longer documents, where a small subset of $k$ words may be insufficient to guarantee reasonable signature uniqueness. The megashingling technique of [14] is similar to LSH, but it selects subsets of $k$ shingles instead of $k$ words. It can be seen that unlike our scheme, the stability of LSH signatures to random additions/deletions of words depends on document length, with longer documents being more immune and shorter ones being more vulnerable. In the web-clustering experiments described in [18], 125 signatures were generated per document to ensure adequate near-duplicate recall. Also, in order to limit the number of false-positive matches, a two-phase setup

[1] Alternatively, one could use $k$ different hash functions to generate a single signature. In such a case, the intersection of a random word ranking with a document is done $k$ times, each time with the topmost element being appended to the buffer, which is eventually used to produce the signature hash.

was deployed, whereby once potential duplicate clusters were identified; their elements were filtered using an explicit word-overlap distance metric.

Lastly, we note a degree of similarity between lexicon randomization and the use of bagging [4] in machine learning. For learners sensitive to small perturbations of the training set, the variance component of the prediction error can be reduced when instead of a single model, an ensemble of multiple models is learned, each based on a bagged version of the original training data. Similarly, I-Match can be viewed as being sensitive to changes to the contents of a lexicon, as well as to the contents of a document. By using multiple perturbed versions of the original lexicon, obtained by bagging or a similar randomization process (as described above), the stability of the extended signature (which can be viewed as an ensemble of individual signatures) with respect to random changes to a document is increased. The process of extended signature matching can be seen as a special case of voting, but where instead of the usual majority vote used in bagging of classifiers, the decision-making process is asymmetric, as it takes just one member of the voting ensemble to say yes for a match decision to be made.

The multiple signature method presented in this section addresses the problem of signature fragmentation, or signature stability, resulting from document perturbation. This problem is complementary to one of reliability, where two unrelated documents receive the same signature, which was addressed by the usage a secondary lexicon proposed in Sect. 3. Note that if lexicon randomization is applied to a version of I-Match employing a secondary lexicon $B$, sampling with replacement is applied to both the primary lexicon, $L$, and the secondary lexicon, $B$.

## 5 Applications

### 5.1 Web search

From the usability perspective, it is important that results of web search queries do not contain multiple references to the same information source. Unfortunately, due to mirroring and free copying, it is fairly common that the same web-page content may be available in multiple locations and under different names. Such documents often differ slightly, but the changes tend to be irrelevant. Another example is given by news stories, where often the same news article with minor modifications is posted multiple times and in different locations. Apart from contributing to information clutter, duplicates and near-duplicates increase the size of the document index, with negative consequences to indexing and retrieval performance. For example, it has been reported that by deduplicating the results of a typical web crawl, the size of an inverted index can be reduced by as much as $1/3$ [7, 14]. Finding near replicas of web-pages has been one of the key motivations for researching scalable deduplication techniques [7, 9, 18, 24], and given the exponential growth of the web, it continues to be an important application.

### 5.2 Spam filtering

Duplicate detection systems often operate in a batch mode, where the objective is to find all duplicates in a large existing collection. Alternatively, given a duplicate free collection and a stream of documents, one may wish to filter all documents in the stream that can be considered as near duplicates of some elements in the collection. This latter on-line view is of particular interest to spam filtering, where the document collection might correspond to exemplars of known spam, while the document stream might correspond to incoming messages of unknown category. Recently, there has been growing interest in applications of machine learning and data mining techniques to the problem of filtering Unsolicited Commercial Email (UCE) [13]. Although the unsolicited nature of an email message cannot always be determined from the contents of the message alone, content based approaches have been by far the most widely used [1, 12, 32]. In particular, since an email can be seen as semi-structured text (with the possibility of multimedia extensions), text categorization techniques are readily applicable. Unlike many other text-classification problems, spam filtering tends to be challenging due to its adversarial nature. Spammers actively deploy techniques designed to confuse filtering techniques, thus making the characteristics of spam very dynamic. This complicates feature selection and often leads to sample selection bias, where the statistics of the data used to train a classifier are different from the ones the classifier will encounter in practice. One also needs to consider the asymmetric nature of the misclassification costs [26], whereby misclassifying a legitimate message as spam is potentially much more expensive than the opposite error.

Nevertheless, many filtering systems (e.g., Distributed Checksum Clearinghouse (DCC) or Vipul's razor[2]) try to exploit one key characteristic of spam, i.e., its tendency to be sent in high volume. Following the blanket-marketing approach, many spammers send essentially the same message to as many users as possible. The messages are rarely identical, however, precisely to avoid template-based detection schemes. Strategies employed to foil detection schemes include appending random character strings to the subject line of the message, inserting neutral text passages into the body of a message, changing the order of paragraphs and randomizing the nonalphanumeric content just to name a few [16]. As pointed out in [17], spammers may easily produce a number of alternative campaign messages and send them out in such as way so as to reduce the volume of each campaign handled by any particular filter (i.e., list splitting). It is also true, however, that spam campaigns tend to be quite repetitive and oriented about largely the same topics, e.g., pornography or certain pharmaceuticals, which limits their expressiveness to some extent. Also, many commercial and public systems affect (and rely on feedback from) very large communities of users, which reduces the effectiveness of list-splitting and content-randomization spamming tactics.

## 6 Experimental setup

In the following, we evaluate the near-duplicate detection accuracy of the modified and extended I-Match using web-page and email document collections, where detection using single and multiple randomized signatures is compared.

[2] http://razor.sourceforge.net/.

### 6.1 Web page data

The WT10G [19, 20] dataset from NIST was used for our web page similarity experiments. While this 10 GB 1.7 million document collection is synthetic in nature, it was developed to possess characteristics of the larger web for text retrieval effectiveness research [2]. Additionally, the WT10G corpus has been examined for similarity of the collection to the web as a whole (e.g., using metrics such as link connectivity) and was found to be representative [37]. These factors make it attractive for use in duplicate similarity experiments in which web pages are being examined.

### 6.2 Email data

We considered the following datasets:

- *The Legitimate email collection* consisted of 18,555 messages collected from 4,607 volunteers as examples of nonspam. To account for the possibility of class noise, the messages were hand-labeled and confirmed to be legitimate, with questionable messages removed from the set. The data was considered duplicate free. In our experiments, this dataset was used primarily in evaluation to assess if near-duplicate detection of spam may lead to any false-positives among legitimate emails.
- *The Honeypot-Spam collection* consisted of 10,039 messages collected by accounts set up to attract spam (i.e., they should not be receiving any email at all). These data were known to contain many highly similar messages.
- *The Cluster-Spam collection* consisted of spam messages grouped in 28 clusters. These data were obtained by interactively querying a large database of verified spam messages with the explicit goal of finding near duplicates or highly similar/related messages. A cluster contained messages extracted via queries employing a few different combinations of keywords. During the collection processes, efforts were made to focus on messages likely to be difficult to identify via content matching, i.e., showing clear attempts at randomization.

### 6.3 Document preprocessing

After removing HTML markup, each document was mapped onto the set of unique words, where a word was defined as a sequence of alphanumeric characters delimited by white space. In the case of email messages, only the text contained in the subject line and the message body was considered. Words were converted to lower case and the ones containing more than one digit as well as those having fewer than four characters were removed. Additionally, email messages having fewer than five unique word features were considered as too short and were ignored.

In the case of the email data, trivial duplicates were removed. For the Honeypot and Cluster spam datasets, this resulted in a reduction in the number of documents from 10,039 to 5,328 and from 8,703 to 6,389, respectively. Note the high duplicate rate in the Honeypot dataset. The message-count statistics of the preprocessed datasets are shown in Table 1.

**Table 1** Message count statistics of the email datasets used in the experiments

| Dataset | Message count |
|---|---|
| Legitimate | 18,555 |
| Honeypot-spam | 5,328 |
| Cluster-spam | 6,389 |

### 6.4 The evaluation process

The exact point at which two documents cease to be near-duplicates and become just highly similar is difficult to define, which is related to the general difficulty of comparing the quality of different clusterings of the same data. To avoid the ambiguity in the near-duplicate judgments, we chose the traditional cosine similarity measure as a benchmark metric against which the accuracy of the signature-based techniques was compared. For documents $i$ and $j$, their cosine similarity is defined as

$$\text{cosine}(i, j) = \frac{|\text{common unique features}(i, j)|}{\sqrt{d(i)d(j)}}, \tag{2}$$

where $d(j)$ is the number of unique features in document $j$. The definition ignores in-document term frequency often used with the cosine-similarity measure. This is due to the interpretation of near-sameness used by I-Match, i.e., order independent sufficient overlap of unique terms between any two documents. Our experience suggested that two documents can safely be considered as near-duplicates if their cosine similarity is greater than 0.9. In the presence of severe randomization, this does not guarantee that all duplicates of a particular template document will be recovered, but it is desired that a good duplicate-detection technique identify a large fraction of the same documents as the cosine-similarity approach.

Given a query $i$, and a document collection, we define the recall of a signature-based detection technique as the ratio of the number of documents flagged as duplicates of $i$ to the corresponding number identified by the cosine measure, when using $i$ as a template,

$$\text{recall}(i) = \frac{|\text{duplicates found for } i|}{|\text{documents } j \text{ such that cosine}(i, j) \geq 0.9|}. \tag{3}$$

Similarly, precision is defined as

$$\text{precision}(i) = \frac{|\text{duplicates found for } i| \cap |\text{documents} j \text{ such that cosine}(i, j) \geq 0.9|}{|\text{duplicates found for } i|}. \tag{4}$$

Note that from the spam-filtering perspective, we often do not care if the copy-detection systems flags just similar spam messages as near-duplicates of the template set, as long as there are no false-positive errors in which legitimate messages are classified as spam, which makes the use of precision less relevant. To account for the performance of a copy-detection technique from the classification standpoint, we define a utility function, such that when using a spam message $i$ as a query the utility is given by

$$\text{utility}(i) = |\text{spam}(i)| - \cos t \cdot |\text{legit}(i)| \tag{5}$$

where the $|\text{spam}(i)|$ and $\text{legit}(i)|$ are the numbers of spam and nonspam messaged extracted, with $\cos t$ defined as the relative cost of misclassifying a legitimate message as spam. The values of $\cos t$ reported in the literature typically vary between 1 and 1,000 [1, 32]. In our experiments, we used the setting of $\cos t = 100$. The utility function combines the benefit due to elimination of spam with the significant cost of false positives.

## 6.5 I-Match signature algorithm settings

In applying I-Match and its extensions, one important question is the choice of the I-Match lexicon. In previous studies, the collection statistics of a large document set were used to find near-duplicates within that same collection, but it was also suggested that a large diverse *training* collection could be effective in detecting duplicates in a *different* collection. This is of particular importance, since the content distribution may be constantly changing [25], and as is the case for email data in our experiments, one often does not have a large enough target document collection to derive a stable lexicon. As recently shown in [10], there might be disadvantages to constantly updating the collection statistics to track the target distribution of content since it tends to reduce the time-validity of signatures while adding little in terms of deduplication accuracy. To evaluate the effect a lexicon choice might have, in the experiments with the web-page collection, we considered two lexicons: one derived from the WT10G dataset itself and one derived from a large collection of news stories (here referred to as SGML), which was also used in [9]. In the case of the email data, just the SGML-based lexicon was applied. The SGML dataset corresponded to TREC disks [4, 5],[3] which is a compilation of Financial Times, LA Times, and other news collections. This dataset provides good statistics for the English language because it has high quality data, in terms of formatting noise and use of the language.

### *6.5.1 WT10G lexicon*

The WG10G collection contained 1,679,076 documents and 5.8 million unique terms, out of which 411 thousand terms were selected by retaining those words for which normalized *idf* (*nidf*) resided in the interval $nidf \in [0.2, 0.8]$. The choice of this particular interval was motivated by our past experience with I-Match applied to web data and the results reported in [9].

### *6.5.2 SGML lexicon*

The SGML collection contained 556,000 documents, with an average length of 662 terms and a total number of unique terms of 488,000. The interval $nidf \in [0.2, 0.8]$ contained the lexicon terms used in our experiments with the web data. For experiments with the email data, cross-validation suggested a different choice of the *nidf* range: $[0.2, 0.3]$.

[3] http://trec.nist.gov/data/qa/T8_QAdata/disks4_5.html.

### 6.6 Lexicon bagging

In experiments with lexicon randomization, the original copy of the lexicon was augmented with $K$ copies (with $K$ in $\{1, \ldots, 10\}$, i.e., we considered experiment runs using original + 1, original + 2, ... lexicons) obtained by bootstrap sampling from the original and ensuring that each term was selected at most once. Thus, each randomized lexicon shared approximately 67% of terms with the original.

## 7 Results

### 7.1 Web-page data

We used a random sample of 115,342 documents (approximately 7% of the total) as queries against the full WT10G collection. For each query, a set of documents for which the cosine similarity exceeded the 0.9 threshold was identified to be used for measuring the recall of hash-based techniques. Only 26,491 of the queries had a cosine neighborhood containing documents other than themselves and only those were used in the evaluation of I-Match. Note that computation of cosine similarity is quite expensive and would generally be unsuitable in operational settings.

I-Match with the WT10G and SGML lexicon choices was then applied to the document corpus and for the near-duplicate clusters containing the documents from the query set; the recall of the I-Match technique was measured. The recall results are shown in Fig. 3. The use of lexicon randomization clearly provides a dramatic increase in the near-duplicate recall, although there is a saturation effect and beyond a certain point there is little benefit of introducing additional perturbed lexicons.[4]

Interestingly, the SGML lexicon led to higher detection accuracy than the lexicon derived from the target collection. We suspect this is because the SGML collection was much cleaner (it contained professionally edited articles), and thus more representative of the content on which we usually wish to focus when deduplicating. Conversely, the WT10G collection contained both proper and misspelled versions of words, as well as various types of formatting noise (even with a good parser, noise due to formatting, and logic, such as javascript, still exists), which would contribute to signature brittleness.

This seems to strengthen the arguments advanced in [10] that deriving a lexicon from large stable document collection may be preferred. Of course, in practice, this might depend on other factors, such as the language in which the target documents are written.

Figure 4 shows the dependence of the precision on the number of I-Match lexicons used when using the SGML lexicon. Interestingly, the precision obtained with

[4] We have not performed rigorous comparison of the different variants of I-Match with alternative single signature algorithms. However, to give the reader some idea of the comparative performance, a run of the megashingling algorithm with its own document parsing and other settings have led to the recall of 0.668 and precision of 0.758, which was roughly equivalent to I-Match utilizing the SGML collection with 5 randomized lexicons. The megashingling code used was corresponded to "Shingleprinting code for estimating document similarity" available at: http://research.microsoft.com/research/downloads/default.aspx.

**Fig. 3** Near-duplicate web-page recall of I-Match as a function of the number of randomized lexicons used. Usage of multiple randomized lexicons is generally beneficial, but I-Match appears to perform best with lexicons derived from the SGML corpus. Near-duplicate spam recall of I-Match as a function of the number of randomized lexicons used. The absolute values of recall depend on the dataset (i.e., true amount of duplication present), but the benefits of using multiple lexicons are clear

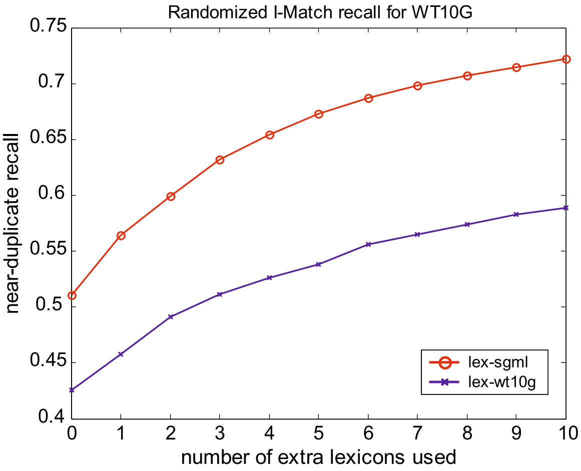
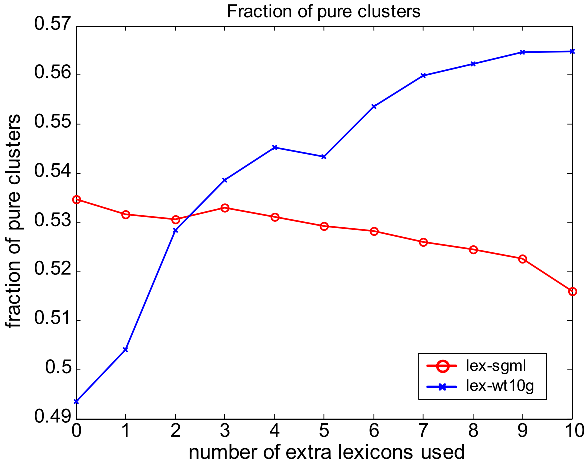


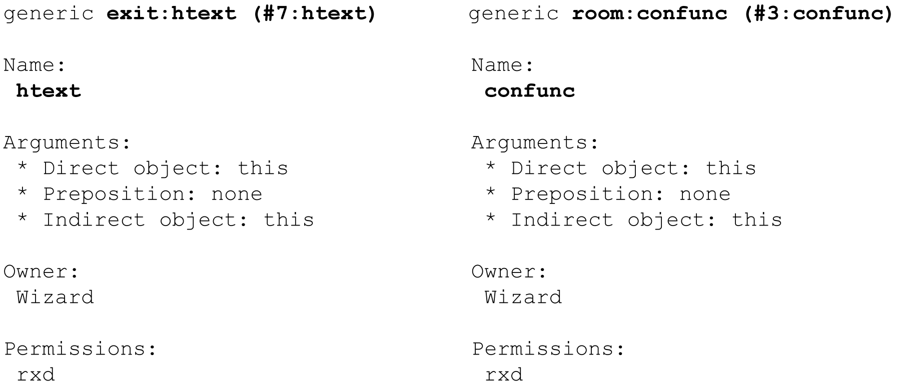
**Fig. 4** Fraction of pure clusters for the two lexicon choices. Note that with more randomized lexicons the fraction decreases only slightly for one lexicon choice (*sgml*) and actually increases for the other (*w10g*)



a native lexicon, i.e., one derived from the same collection, appears to be higher. We suspect this is because many of the low frequency terms can be specific to a particular collection and much harder to find in other collections.

### *7.1.1 Formatting artifacts*

Exact measurements of precision and recall in near-duplicate detection are impossible due to the vague definition of "near-sameness." Our choice of cosine-similarity threshold is reasonable, but arguably a slightly lower or slightly higher value could be used as well. In particular, documents with lower cosine similarity values can often be considered as near duplicates and are correctly identified as such by I-Match, but this leads to the lowering of precision as defined by (3). As such, the current definitions of precision and recall can be loosely interpreted as their lower bounds.



```
generic exit:htext (#7:htext)

Name:
 htext

Arguments:
 * Direct object: this
 * Preposition: none
 * Indirect object: this

Owner:
 Wizard

Permissions:
 rxd
```

```
generic room:confunc (#3:confunc)

Name:
 confunc

Arguments:
 * Direct object: this
 * Preposition: none
 * Indirect object: this

Owner:
 Wizard

Permissions:
 rxd
```

**Fig. 5** An example of a spurious match within the WT10G collection. The two documents are almost identical from the pure word overlap point of view. However, due to their nature (help pages corresponding to distinct functions) they might be considered as different

There are also situations, however, where matches produced by I-Match are truly spurious. In particular, the WT10G collection contains numerous software manual pages, where the same template is used to describe the functionality of different routines. Figure 5 provides an example. In such cases, most of the content between any two help pages is actually the same. However, the differentiating content uses specialized vocabulary, which is unlikely to be selected as part of the I-Match lexicon. Note that even the cosine-similarity definition of near-sameness can lead to such a mistake if the fraction of template-induced formatting text is large enough. This again points at the inherent ambiguity in the definition of a near-duplicate. The nature of help pages forces us to ignore the template-induced similarities and focus instead on the small differences, i.e., function names. However, for different types of content (e.g., news) such small differences may be irrelevant.

When the number of randomized lexicons is increased, the precision of I-Match decreases as shown in Fig. 4. However, when the number of "pure" clusters for which I-Match achieves 100% precision is accounted for, it is apparent that this fraction is quite stable or actually shows an increasing trend, depending on the choice of the lexicon. This is indicative of the effect of template-induced clusters accumulating more and more "spurious" matches, accounting for a large number of documents, while at the same time randomization leads to the discovery of new correct clusters that had been missed when using fewer lexicons.

## 7.2 Honeypot-spam vs. legitimate email

In this experiment, a random 10% of the Honeypot-spam data was used as queries against the Honeypot-spam and the legitimate-email datasets. The resulting average values of the recall and utility metrics are given in Table 2. None of the near-duplicate detection configurations produced any false-positive matches against the legitimate email collection. Lexicon randomization provided a clear benefit, both in terms of duplicate detection and spam detection metrics.

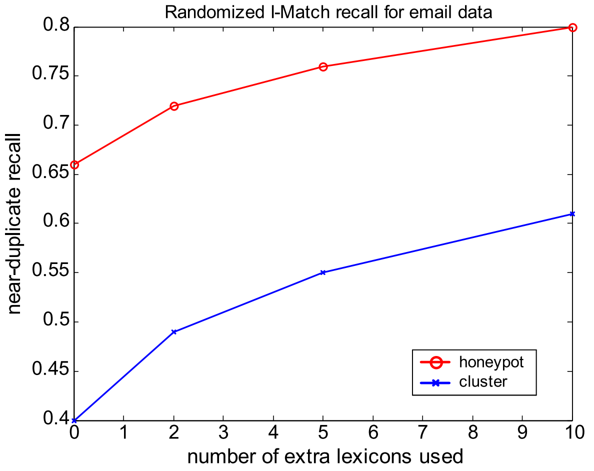


**Fig. 6** Near-duplicate spam recall of I-Match as a function of the number of randomized lexicons used. The absolute values of recall depend on the dataset (i.e., true amount of duplication present), but the benefits of using multiple lexicons are clear

**Table 2** Duplicate detection accuracy (reported as recall, relative increase in recall and utility) in the Honeypot-spam vs. legitimate-email experiment. $K$-bag signifies that $K$ auxiliary lexicons were used

| Lexicon count | I-Match | | |
|---|---|---|---|
| | Recall | Relative Increase | Utility |
| 1 | 0.66 | 0% | 30.17 |
| 1 + 2-bag | 0.72 | 9% | 36.32 |
| 1 + 5-bag | 0.76 | 15% | 43.34 |
| 1 + 10-bag | 0.80 | 21% | 56.93 |

**Table 3** Duplicate detection accuracy (reported as recall, relative increase in recall and utility) in the Cluster-spam vs. legitimate-email experiment, where the original SGML-based I-Match lexicon was extended by its random perturbations ($K$-bag signifies that $K$ auxiliary lexicons were used)

| Lexicon count | I-Match | | |
|---|---|---|---|
| | Recall | Relative Increase | Utility |
| 1 | 0.40 | 0% | 46.34 |
| 1 + 2-bag | 0.49 | 23% | 81.25 |
| 1 + 5-bag | 0.55 | 38% | 96.24 |
| 1 + 10-bag | 0.61 | 52% | 113.38 |

### 7.3 Cluster-spam vs. legitimate email

In this experiment a random 10% of the cluster-spam was used as queries against the cluster-spam and legitimate-email datasets. The average values of the recall and utility metrics are given in Table 3. As in the Honeypot experiment, there were no false-positive matches against the legitimate email collection. Figure 6 illustrates the benefit of lexicon randomized for both email datasets.

### 7.4 Discussion

Given that the SGML lexicon was derived using a large collection of news articles, it is interesting to observe its good generalization performance in the application considered, since web-page and email documents are generally different from news stories. This supports the claim that once a large diverse document collection is used, little in terms of copy-detection accuracy can be gained by tracking the changes to content distribution to fine tune the algorithm to the collection to which it is actually applied.

The results shown in Figs. 3 and 6, and in Tables 2 and 3 indicate that by using even a few extra-randomized lexicons, both the recall and utility (in the case of email) metrics can be improved significantly. Given that the computational cost of computing each additional signature is bounded by the cost of computing the original one (since it involves a subset of the same words) and considering that the memory/storage requirements grow linearly with the number of lexicons, one can argue that a practical system should be able to benefit from incorporating a small number of randomized lexicons without compromising its computational resources. A precise cost-benefit analysis would of course need to consider, among other things, the amount of RAM consumed by the original lexicon vis a vis the total amount of RAM available to the system.

One interesting question is whether perfect spam recall is possible with enough lexicons and, if so, how many lexicons are needed to satisfy this requirement. Let $M = p \cdot L$ be the number of terms that are eliminated from the original lexicon to create an auxiliary one. Assuming that the perturbation of the original spam message involves $n$ lexicon terms, it will alter both the original and the secondary signature if $n > M$. In such a case, perfect recall will not be possible for any number of additional lexicons, but this situation is unlikely since in most cases $M$ is expected to be much larger than the number of words in any email message. If perfect recall is possible, the probability that the signature due to any particular auxiliary lexicon remains unchanged is:

$$q = \frac{\binom{L-n}{M}}{\binom{L}{M}} = \frac{(L-n)^{\underline{M}}}{L^{\underline{M}}} \approx \left(1 - \frac{M}{L}\right)^{n} \tag{6}$$

i.e., to the ratio of the number of times an auxiliary lexicon can be selected such that it excludes the perturbation words to the overall number of possibilities. One can then envision generating auxiliary lexicons till one is found for which the success defined as unaltered signature is achieved. The probability that the number of trials till success is $K$ is governed by the geometric distribution, i.e.,

$$P(K) = (1-q)^{K-1} q \tag{7}$$

with the expected number of trials till success of $1/q$. Table 4 tabulates $P(K)$ for several values of $L$ and $n$, assuming that $M = 0.33 \cdot L$, which is the case for lexicon bagging.

It can be seen that as indicated by (6), the expected number of trials depends primarily on the size of the perturbation and not the absolute lexicon size. Due to the exponential dependence of $q$ on the amount of perturbation, the expectation of perfect recall becomes unrealistic for large perturbations, although one has to keep in mind that given the selectivity of an I-Match lexicon, the value of $n$ will be typically smaller than the actual alternation to the original message in terms of the number of words added/inserted.

**Table 4** The expected number of auxiliary lexicons required to achieve perfect recall for the specified amount of perturbation, $n$. Different rows show the dependence on the absolute size of the original lexicon, $L$

| | $n=2$ | $n=5$ | $n=10$ | $n=20$ | $n=50$ |
|---|---|---|---|---|---|
| $L=1000$ | 2.23 | 7.44 | 56.10 | 3,309.92 | 932,403,182.29 |
| $L=10{,}000$ | 2.23 | 7.41 | 54.98 | 3,037.91 | 527,924,300.92 |
| $L=100{,}000$ | 2.23 | 7.41 | 54.87 | 3,012.39 | 499,897,879.50 |

## 8 Conclusions

We considered the problem of improving the stability of I-Match signatures with respect to small modifications to document content. The proposed solution involves the use of several I-Match signatures, rather than just one, all derived from randomized versions of the original lexicon. Despite utilizing multiple fingerprints, the proposed scheme does not involve direct computation of document overlap, which makes signature comparison only marginally slower than in the case of single-valued fingerprints. Additionally, clear improvements in signature stability can be seen when adding just one extra signature component, with more gains to be made as more are added.

The original I-Match algorithm was modified to improve its reliability for very long documents by insisting that a certain fraction of terms need participate in the creation of a signature. This decreases the chance of any two markedly different documents being projected onto the same signature. Also, we demonstrated that lexicons for I-Match can be successfully derived from a collection different from the target one, which in fact may be preferable if the target collection is noisy.

The proposed extended I-Match signature scheme does indeed provide greater robustness to term additions and deletions. Its effectiveness as a countermeasure to word substitutions is smaller, however, since a substitution is equivalent to an addition-deletion combination. In future work, we intend to investigate ways to improve signature stability in the presence of term substitutions.

## References

1. Androutsopoulos I, Koutsias J, Chandrinos K, Paliouras G, Spyropoulos C (2000) An evaluation of Naive Bayesian anti-spam filtering. In: Potamias, G, Moustakis, V, van Someren, M (eds) Proceedings of the workshop on machine learning in the new information age: 11th European conference on machine learning (ECML2000), pp 9–17
2. Bailey P, Craswell N, Hawking D (2003) Engineering a multi-purpose test collection for web retrieval experiments. Inf Process Manag 39:853–871
3. Bilenko M, Mooney RJ (2002) Learning to combine trained distance metrics for duplicate detection in databases. Technical report AI 02-296, Artificial Intelligence Lab, University of Texas at Austin
4. Breiman L (1996) Bagging predictors. Mach Lear 24:123–140
5. Brin S, Davis J, Garcia-Molina H (1995) Copy detection mechanisms for digital documents. In: Proceeding of SIGMOD, pp 398–409
6. Broder A (1997) On the resemblance and containment of documents. In: Proceedings of complexity and compression of sequences (SEQUENCES '97), pp 21–29

7. Broder A, Glassman S, Manasse M, Zweig G (1997) Syntactic clustering of the Web. In: Proceedings of the sixth international world wide web conference
8. Buckley C, Cardie C, Mardisa S, Mitra M, Pierce D, Wagsta K, Walz J (2000) The smart/empire tipster IR system. In: TIPSTER phase III proceedings. Morgan Kaufmann
9. Chowdhury A, Frieder O, Grossman DA, McCabe MC (2002) Collection statistics for fast duplicate document detection. ACM Trans Inf Syst 20(2):171–191
10. Conrad J, Guo X, Schriber C (2003) Online duplicate document detection: signature reliability in a dynamic retrieval environment. In: CIKM, pp 443–452
11. Cooper J, Coden A, Brown E (2002) A novel method for detecting similar documents. In: Proceedings of the 35th Hawaii international conference on system sciences
12. Drucker H, Wu D, Vapnik V (1999) Support vector machines for spam categorization. IEEE Trans Neur Netw 10(5):1048–1054
13. Fawcett T (2003) "In vivo" spam filtering: a challenge problem for data mining. KDD Explor 5(2):203–231
14. Fetterly D, Manasse M, Najork M (2003) On the evolution of clusters of near-duplicate web pages. In: Proceedings of the 1st Latin American web congress, pp 37–45
15. Gionis A, Indyk P, Motwani R (1997) Similarity search in high dimensions via hashing. In: Proceedings of the 25th international conference on very large databases (VLDB)
16. Graham-Cummings J (2003) The spammers' compendium. In: Proceedings of the spam conference
17. Hall RJ (1999) A countermeasure to duplicate-detecting anti-spam techniques. Technical report 99.9.1, AT&T Labs Research
18. Haveliwala T, Gionis A, Indyk P (2000) Scalable techniques for clustering the web. In: Proceedings of WebDB-2000
19. Hawking D (2000) Overview of the TREC-9 web track. In: TREC-9 NIST
20. Hawking D, Craswell N (2001) Overview of the TREC-2001 web track. In: TREC-10 NIST
21. Heintze N (1996) Scalable document fingerprinting. In: 1996 USENIX workshop on electronic commerce, November 1996
22. Hernandez M, Stolfo S (1995) The merge/purge problem for large databases. In: Proceedings of the SIGMOD conference
23. Hoad TC, Zobel J (2002) Methods for identifying versioned and plagiarised documents. J Am Soc Inf Sci Technol
24. Ilyinsky S, Kuzmin M, Melkov A, Segalovich I (2002) An efficient method to detect duplicates of web documents with the use of inverted index. In: Proceedings of the eleventh international world wide web conference
25. Kleinberg J (2002) Bursty and hierarchical structure in streams. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining (KDD-2002)
26. Kołcz A, Alspector J (2001) SVM-based filtering of e-mail spam with content-specific misclassification costs. In: Proceedings of the workshop on text mining (TextDM'2001)
27. Kołcz A, Chowdhury A, Alspector J (2003) Data duplication: an imbalance problem? In: Proceedings of the ICML'2003 workshop on learning from imbalanced datasets (II)
28. Kołcz A, Chowdhury A, Alspector J (2004) Improved robustness of signature-based near-replica detection via lexicon randomization. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining (KDD-2004)
29. Kwok K (1996) Relevance feedback in information retrieval. In: Proceedings of the nineteenth annual international ACM SIGIR conference on research and development in information retrieval
30. McCallum A, Nigam K, Ungar L (2000) Efficient clustering of high-dimensional data sets with application to reference matching. In: Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining (KDD-2000)
31. Robertson S, Walker S, Beaulieu M (1998) Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive. In: Proceedings of the 7th text retrieval conference
32. Sahami M, Dumais S, Heckerman D, Horvitz E (1998) A Bayesian approach to filtering junk e-mail. In: Proceedings of the AAAI-98 workshop on learning for text categorization
33. Salton G, Yang C, Wong A (1975) A vector-space model for information retrieval. Commun ACM 18
34. Sanderson M (1997) Duplicate detection in the Reuters collection. Technical report TR-1997-5, Department of Computing Science, University of Glasgow
35. Shivakumar N, Garcia-Molina H (1999) Finding near-replicas of documents on the web. In: WEBDB: international workshop on the world wide web and databases, WebDB. LNCS

36. Singhal A, Buckley C, Mitra M (1996) Pivoted document length normalization. In: Proceedings of the nineteenth annual international ACM SIGIR conference on research and development in information retrieval
37. Soboroff I (2002) Does wt10g look like the web? In: SIGIR 2002, pp 423–424
38. Winkler WE (1999) The state of record linkage and current research problems. Technical report, Statistical Research Division, US Bureau of Census, Washington, DC, 1999