

# Methods for Identifying Versioned and Plagiarised Documents

Timothy C. Hoad

Justin Zobel\*

School of Computer Science and Information Technology, RMIT University  
GPO Box 2476V, Melbourne 3001, Australia  
{hoad,jz}@cs.rmit.edu.au

## Abstract

The widespread use of online publishing of text promotes storage of multiple versions of documents and mirroring of documents in multiple locations, and greatly simplifies the task of plagiarising the work of others. We evaluate two families of methods for searching a collection to find documents that are co-derivative, that is, are versions or plagiarisms of each other. The first, the ranking family, uses information retrieval techniques; extending this family, we propose the identity measure, which is specifically designed for identification of co-derivative documents. The second, the fingerprinting family, uses hashing to generate a compact document description, which can then be compared to the fingerprints of the documents in the collection. We introduce a new method for evaluating the effectiveness of these techniques, and demonstrate it in practice. Using experiments on two collections, we demonstrate that the identity measure and the best fingerprinting technique are both able to accurately identify co-derivative documents. However, for fingerprinting parameters must be carefully chosen, and even so the identity measure is clearly superior.

## 1 Introduction

The Internet is the largest public repository of information ever created. Much of this information is published in more than one location. For example, an internet search using the phrase “Linux Documentation Project” results in dozens of almost identical web pages held at different locations, copied from each other and revised slightly. A related issue is that many digital documents are dynamic, continually changing and evolving. It is common practice to keep multiple versions of documents at different stages of development, so it can be necessary to determine whether two documents are different versions of the same text or are different texts altogether.

Another problem is plagiarism. Many documents that are published on the Internet are copies or plagiarisms of other documents. Since a plagiarism may not be identical to the original document, using conventional search techniques it can be difficult to distinguish plagiarised documents from those that are simply on the same topic.

The aim of this research is to develop and evaluate methods for identifying documents that originate from the same source, which we call *co-derivatives*. Existing techniques that can be used to address these problems include fingerprinting, a technique developed specifically for detecting co-derivatives [1, 2, 5, 7, 12], and ranking, a technique developed for information retrieval [18, 20]. String matching has been proposed for small collections [9, 16]. In this paper we extend ranking by developing a new *identity measure*, and explore variants of the fingerprinting method;

---

\*Contact author for all correspondence.

a contribution of this paper is a synthesis of many variants of fingerprinting into a common framework.

We evaluate and compare these techniques for detection of co-derivative documents using two document collections, a small collection seeded with test documents and a large collection of documentation. We demonstrate that both the identity measure and the anchor strategy (which is the best of the fingerprinting techniques) are suitable for identification of co-derivatives and are both able to correctly identify such documents, and greatly outperform the other techniques. Even so, the identity measure is markedly superior to the anchor strategy.

Underlying these results is the need for a way of comparing the effectiveness of co-derivative detection techniques. As our results show, the standard measures of effectiveness used in information retrieval, recall and precision, are not sufficient. We introduce two additional measures, highest false match and separation, and show in practice how they discriminate between methods.

## 2 Preliminaries

The aim of this research is to develop and evaluate techniques for identifying plagiarisms, revisions, and different versions of documents. We consider two documents to be *co-derivative* if they both originated from the same source, following a definition by Manber of plagiarism [7]. It is not required that the documents use the same words, the same structure, or the same format for them to be considered co-derivative. For example, this document is a co-derivative of all previous versions, any subsequent revisions, any substantial plagiarisms of this document, and the same document in a different format such as a web page. It is not a co-derivative of other documents on the same subject or of previous papers in the same area. As another example, in the Linux documents we discuss later, there are many instances of documents that resemble each other due to sharing of format and standardised information, but are not co-derivatives.

Although the reasons for creating co-derivatives vary, in our experience they share similar features. A student who plagiarises existing work when writing an essay will often copy verbatim long blocks of text, then may (but often does not) superficially edit the resulting document to make it look original. Then the student may add further original text to complete the work. An author revising a document will take the original version, add blocks of new material and delete obsolete material as required, and edit the remaining material to remove errors. The effect is similar: long blocks of text are largely preserved, but possibly with intermittent modifications, and some original text is added.

There are several properties that can be used by a human observer to determine whether two documents are co-derivative. Words that are common to both documents are indicative of co-derivation, especially when those words are rare or misspelled. Similarly, grammatical errors and unusual usages of words, such as the incorrect use of “that” rather than “which”, can be a good indication. In plagiarised assignments, it is common to find that some errors or atypical usages have been copied verbatim. Identical blocks of words can also be a strong indicator, especially if the blocks are of significant length, or contain unusual words.

While it is possible that formatting may be used as an indication of co-derivation, it is our experience that it is unlikely to be valuable. It is common practice in many situations to use standard document formatting for unrelated documents, for example office memos, web pages, technical reports and even some student assignments. For this reason, in common with the previous investigations of this problem, we concentrate on the use of textual content to detect co-derivation, rather than document formatting.

There are two separate problems of interest with regard to detection of co-derivatives [1]. The first involves comparison of every pair of documents in the collection to detect sets that are co-derivative, which we refer to as the *n-to-n* problem. The second is the *one-to-n* problem, which involves the comparison of a single document—the *query document*—to a collection in order to

find its co-derivatives. We focus on the one-to-n problem, and, while the algorithms used may be naively applied to the n-to-n problem, such an approach may not be the most efficient.

We investigate two approaches to the one-to-n problem. One, *ranking*, is based on information retrieval methodologies. The other, *fingerprinting*, was developed specifically for detection of co-derivatives [7]. The fingerprinting and ranking methods are very different, but have elements in common, in particular, preprocessing of documents prior to indexing. One of the simplest problems is that of query terms having different case to the terms in a document [18]. Generally, it is sufficient for two strings to have the same characters in the same order for us to consider them the same word; case-folding is therefore used in all the experiments reported in this paper.

Other preprocessing elements are stopping, stemming, and term parsing. Languages such as English make use of closed-class words—for example “the”, “of”, and “may”—that indicate the structure of a sentence and the relationships between the concepts presented, but do not have any meaning on their own. For the purpose of approximate querying, these *stopwords* can usually be ignored [18], and are typically removed from queries. The advantage of doing so is that the query processing time can be dramatically reduced, without any noticeable degradation in the quality of the results returned [18]. These stopwords, however, may be a useful indicator of co-derivation. It is plausible that they occur in similar frequencies in documents that are copied. It is also possible that they add little value, and can be removed without repercussion.

Many words in the English language have multiple variant forms, distinguished by suffix, but, in the context of querying, the form of a word is often unimportant. The suffixes from variant forms can be removed by *stemming*. Use of stemming can have several effects. The size of an inverted index will be slightly smaller, but query times may be increased marginally [18]. For ad hoc queries, stemming can improve recall, but it is not obvious what the effect is likely to be on detection of co-derivatives. The effect of stemming and stopping is measured in our experiments.

All the techniques in this paper rely on the parsing of the documents into *words* or *terms*. This is a relatively simple process, but the question of what is considered to be a word must be addressed. For this paper, a word is considered to be any string of alphanumeric characters. Words can be separated by punctuation, whitespace, or control characters [2, 5, 18]. For ad hoc querying, it is also necessary to remove document mark-up such as HTML tags, as we do in our experiments; however, whether such mark-up can be of use for detection of co-derivatives is an open question.

### 3 Ranking

The ranking techniques we use are derived from information retrieval. Ranking is used to find approximate matches to queries on text databases, and is widely used in Internet search engines and other text retrieval systems [18]. It is a two-stage process. In the first stage, the collection is indexed [18]. The second stage of the ranking process is when a query is presented to the collection. The query is used to drive computation of a similarity score for each document stored in the collection, using a function known as a *similarity measure* [18]. The documents are sorted by decreasing score and the top-ranked documents are returned to the user. This technique does not give a “yes or no” answer to the question of whether the documents are relevant to the user’s need, but orders them by estimated likelihood of relevance. Sanderson [11] used a form of this technique to identify identical documents, but does not provide any technical information.

The information required to evaluate the query depends on the similarity measure being used. By altering the similarity measure, the effectiveness of the ranking engine can be drastically altered. The following notation and statistics are used in the similarity measures we test.

- $N$  The number of documents in the collection
- $n$  The number of distinct terms in the collection

- $f_t$  The number of documents containing term  $t$
- $f_{d,t}$  The number of occurrences of term  $t$  in document  $d$
- $f_d$  The number of terms in document  $d$
- $W_d$  The weight (or length) of document  $d$
- $D$  The document collection
- $q$  The query document
- $d$  A document in collection  $D$

Two families of similarity measures were implemented for this paper: standard measures—including the inner product, a normalised inner product, and the cosine measure—and our identity measures, described in the next section.

### Inner product

The inner product similarity measure is one of the simplest practical similarity measures [18]. It is a product of two components, the term frequency and the term weight, and is calculated for each term that appears in both the query and the document. The term frequency is an expression representing the importance of each term in the document, calculated as  $1 + \log_e f_{d,t}$ . The term weight is an expression of the overall importance of the term, calculated as  $\log_e \left(1 + \frac{N}{f_t}\right)$ . The inner product is calculated by:

$$\sum_{t \in q \cap d} (1 + \log_e f_{d,t}) \cdot \log_e \left(1 + \frac{N}{f_t}\right)$$

The inner product gives high weight to documents in which the query terms appear a large number of times. The side-effect of this is that longer documents tend to be more highly ranked [18]. The advantage of the inner product measure is its speed of computation. The inner product similarity measure is not expected to perform well in detecting co-derivatives due to the fact that it favours long documents, but is an interesting baseline.

### Normalised inner product

The normalised inner product is intended to address the problem of long documents being favoured by the standard inner product measure [18]. The inner product is initially calculated as above, but the similarity is normalised after the query terms have been processed, by dividing the similarity score by the square root of the document length. The complete expression is

$$\frac{1}{\sqrt{f_d}} \cdot \sum_{t \in q \cap d} (1 + \log_e f_{d,t}) \cdot \log_e \left(1 + \frac{N}{f_t}\right)$$

The normalised inner product measure is still sensitive to variation in document length, however, and can favour short documents [18].

### Cosine measure

The cosine similarity measure is used widely in information retrieval applications for processing ad hoc queries. The core of the cosine measure is the inner product, which computes the product of vectors representing the query and document. Document and query lengths are calculated as  $W_x = \sqrt{\sum_{t \in d} w_{x,t}^2}$ , for document or query  $x$ , and are used to normalise the inner product. The query weight  $W_q$  is constant and can therefore be omitted, and thus the cosine measure can be represented as [18]:

$$\cos \theta = \frac{1}{W_d} \cdot \sum_{t \in q \cap d} (1 + \log_e f_{d,t}) \cdot \log_e \left( 1 + \frac{N}{f_t} \right)$$

While this measure is known to be effective when searching for documents using short ad hoc queries, it is not designed for detecting co-derivatives. In particular, the cosine measure is intended to compensate for differences in length, but in detection of co-derivatives the absolute lengths of query document and stored document are of importance. We developed the identity measure specifically to address this issue.

## 4 Identity measures

The similarity measures described above are intended for ad hoc querying. Our aim in developing the identity measure was to produce a similarity measure that is effective at comparing pairs of full-length documents for co-derivation. The design of the identity measure is based on the intuition that similar documents should contain similar numbers of occurrences of words. Therefore, pairs of documents with this property are given a high similarity rating, while differences in the number of occurrences are penalised.

We have explored several variations of the identity measure. All are based on the principles of document matching exemplified in the cosine measure, and all share the same asymptotic complexity as the cosine measure. The complexity is linear in the number of search terms  $t$  and number of documents  $d$ , that is,  $O(td)$ . Ranked queries can be evaluated extremely rapidly even on very large collections, as is amply demonstrated by search engines such as Google. Because evaluation time is not substantially affected by the form of the similarity measure [20], it is possible to apply the similarity measures described here to very large collections while maintaining a satisfactory execution time.

**Variation 1** The first variation makes use of the term weight, the difference between the term frequencies in query and document, and the document lengths. The term weight,  $w_t$ , is an expression of the importance of a term. Generally, the more frequent a term is in the collection, the less useful it is as a discriminator, hence the expression of the term weight used in the inner product and the cosine measure.

For measuring likelihood of co-derivation, however, a further valuable discriminator is the difference in term frequencies: for many terms  $t$  in co-derived documents,  $|f_{d,t} - f_{q,t}|$  should be small. Likewise, co-derivatives should be of similar length, so the value  $|f_d - f_q|$  should also be small. The complete measure is:

$$\frac{1}{1 + |f_d - f_q|} \cdot \sum_{t \in q \cap d} \frac{\log_e \left( \frac{N}{f_t} \right)}{1 + |f_{d,t} - f_{q,t}|}$$

**Variation 2** The first variation uses the inverse of the difference in document lengths as a discriminator. This has the potential to make the measure too sensitive to differences in document lengths. For example, consider a pair of documents of 1000 and 1005 words in length. The lengths of these documents differ by only 0.5%, but the document length discriminator above would cause the similarity score for these two documents to differ by a factor of five. To overcome this, the document length discriminator can be tempered by taking the log of the difference. The rest of the similarity measure is the same as Variation 1, and the complete formulation is:

$$\frac{1}{1 + \log_e (1 + |f_d - f_q|)} \cdot \sum_{t \in q \cap d} \frac{\log_e \left( 1 + \frac{N}{f_t} \right)}{1 + |f_{d,t} - f_{q,t}|}$$

**Variation 3** By multiplying the term weight by the sum of the frequency of the term in the document and the frequency of the term in the query, a higher rank is given to documents in which the term is rare in the collection but common in the query or the document.

$$\frac{1}{1 + \log_e(1 + |f_d - f_q|)} \cdot \sum_{t \in q \cap d} \frac{\log_e \left(1 + \frac{N}{f_t}\right) \cdot (f_{d,t} + f_{q,t})}{1 + |f_{d,t} - f_{q,t}|}$$

**Variation 4** This variation is a slight alteration to variation 2, used to assess the impact of changing the term weight discriminator.

$$\frac{1}{1 + \log_e(1 + |f_d - f_q|)} \cdot \sum_{t \in q \cap d} \frac{\log_e \left(\frac{N}{f_t}\right)}{1 + |f_{d,t} - f_{q,t}|}$$

**Variation 5** Rather than using the log of the term weight, as in all of the previous variations, in this variation the emphasis on the term weight is increased, by simply using  $N/f_t$ . This means that rare terms have a much larger weight than common terms.

$$\frac{1}{1 + \log_e(1 + |f_d - f_q|)} \cdot \sum_{t \in q \cap d} \frac{\left(\frac{N}{f_t}\right)}{1 + |f_{d,t} - f_{q,t}|}$$

## 5 Fingerprinting

Another approach to detection of similar documents is a technique known as fingerprinting, based on the work of Manber [7] and subsequent work by Garcia-Molina and Shivakumar [12]. Rather than using term occurrences and frequency information, fingerprinting aims to produce a compact description, or *fingerprint*, for each document in the collection [2, 7]. The fingerprint represents the content of the document, and, by comparing these fingerprints, it is possible to determine the likelihood that the documents are (in our terminology) co-derivatives [7].

A document fingerprint is a collection of integers that represent some key content of the document. Each of these integers is referred to as a *minutia*. Typically a fingerprint is generated by selecting substrings from the text and applying a mathematical function to each selected substring. This function, similar to a hashing function [1, 7], produces one minutia. The minutiae are then stored in an index for quick access when querying.

When a query document is compared to the collection, the fingerprint for the query is generated. For each minutia in the fingerprint, the index is queried, and a list of matching fingerprints is retrieved. The number of minutiae in common between the query fingerprint and each fingerprint in the collection determines the score of the corresponding document [2, 5, 14].

The complexity of the query process is linear in the number of documents  $d$  and the number of minutiae,  $m$ . Depending on the strategy used to select the substrings in the document, the number of minutiae can either be constant, in which case the query complexity is  $O(d)$ , or it can be proportional to the size of the document (*mat*). In this case the asymptotic complexity is the same as that in a ranked query evaluation,  $O(td)$ .

In designing a fingerprinting process, there are four areas that need consideration. The first is the function used to generate a minutia from a substring in the document. The second is the size of the substrings that are extracted from the document (the granularity). The third is the number of minutiae used to build a document fingerprint (the resolution). Fourth is the choice of the algorithm used to select substrings from the document (the selection strategy). There have been several proposed methods for fingerprinting, based on variation in these four design parameters. These are discussed below. We now discuss the four parameters.

## Fingerprint generation

The process used to convert a string to an integer (a minutia) can have a significant impact on the effectiveness of the fingerprinting method. In order to ensure that fingerprinting is fast and accurate, the fingerprint generation process must display several key properties. It needs to be reproducible [7]; every time a given string is processed, the resulting integer must be the same. The fingerprint generation function should produce as close as possible to a uniform distribution of integers [5]. The minutiae produced lie between two bounds—generally 0 and an arbitrarily high number such as  $2^{32}$ . With any fingerprint generation function or hashing function where the set of possible strings is unknown, it is inevitable that some pairs of different strings share the same integer representation, but a uniform distribution ensures that the probability of two different phrases sharing the same integer representation is as low as theoretically possible. Also, the function selected must be fast: when processing the collection to create the index, and when querying the collection, a large number of minutiae must be generated.

Any hashing function satisfies the first of these criteria, and many satisfy the first two, but few satisfy all of them [10]. Unlike many functions which use multiplication extensively, the following algorithm is efficient as well as reproducible and acceptably uniform [10].

Consider a string,  $c_0c_1 \dots c_k$ , and a function,  $A(c_i)$ , which converts a character,  $c_i$ , to an integer (by taking its ASCII value). For each character  $c_i$  in the string,

$$h(c_i) = h(c_{i-1}) \oplus (A(c_i) + h(c_{i-1}) \ll 6 + h(c_{i-1}) \gg 2) \quad (1)$$

Then, if the maximum hash value is  $m$ , the hash value for the string is given by

$$h(\text{string}) = h(c_k) \bmod m \quad (2)$$

This is the function that was used for minutiae generation throughout this paper.

## Fingerprint granularity

The size of the substring used to generate a minutia is known as the fingerprint *granularity*. This variable can have a significant impact of the accuracy of fingerprinting [13]. There are many ways that the granularity can be specified, but the difference is conceptual rather than practical. It is possible to specify the granularity by the number of characters in the string [7], or the number of sentences [12], but it is arguably easier to conceptualise as the number of words in the string [5]; the notation used in this paper.

By selecting a fine granularity, the fingerprint becomes more susceptible to false matches, but if the granularity selected is too coarse, the fingerprinting becomes too sensitive to change [5, 12]. Consider a coarse granularity of 100 words. The number of distinct 100-word phrases is much larger than the number of distinct minutiae (which are generally stored as 32-bit integers). If we compare two documents that have a minutia in common, then we must assume that these documents share the same 100-word phrase, but many 100-word phrases map to the same minutia. As the granularity becomes larger, the likelihood that two documents sharing a minutia actually share the same phrase becomes smaller. With a coarse granularity, a large proportion of matches happen by chance.

By contrast, if we consider a granularity of one word and two documents that share a minutia, it is probable that these documents do actually share the substring used to generate the minutia. However, this substring is likely to be relatively common, so the fact that the documents share this substring is not strongly indicative. In our experiments we explore granularity.

## Fingerprint resolution

The fingerprint size, or *resolution*, is the number of minutiae used to represent the document. Conceptually, the more information that we store about a pair of documents, the easier it should

be to make a decision as to whether they are co-derivatives. However, the processing required to evaluate the query, and the space required to store the index, increase proportionally with the fingerprint resolution.

The fingerprint resolution may be fixed or variable. It may, for example, be determined by the size of the document [5]. There is no algorithmic limitation that dictates that each document must have the same resolution or that the resolution of the query fingerprint must match those in the collection. A variable resolution, however, may result in longer documents being favoured during querying: a longer document may have more minutiae, and so may match more queries.

### Substring selection strategy

With the question of fingerprint resolution comes the question of the substring selection strategy. If  $r$  minutiae are to be produced, then  $r$  substrings must be selected from the document to be passed to the fingerprint generation function [5]. There are many ways that these substrings could be selected. Some of these strategies are suited to variable fingerprint resolution, and some to fixed resolution. The choice of a selection strategy can have a dramatic effect on both the accuracy and efficiency of the fingerprinting process [13].

There are four classes of selection strategies considered here: full fingerprinting, positional strategies, frequency-based strategies, and structure-based strategies.

**Full fingerprinting.** In what is perhaps the simplest selection strategy, every substring of size  $g$  in the document is selected (where  $g$  is the fingerprint granularity). This strategy produces the largest possible fingerprint resolution for the document—every minutia is included. While this is expensive for the fingerprints for stored documents, it is a valid option for the query, since only one fingerprint must be produced for the query, and this fingerprint does not need to be stored permanently. Because it uses every substring of the document, full fingerprinting could be expected to be a best-case for effectiveness.

**Positional selection.** This is a class of simple strategies that select phrases based on the offset from the beginning of the document.

*Random* substring selection is not expected to be an effective selection strategy—in fact it is expected to be a worst case. It is suited to fixed resolution fingerprinting. A fixed number of substrings are selected at random from the document.

*All substrings* selection is similar to full fingerprinting, but does not select overlapping substrings. Rather, it selects all non-overlapping substrings of size  $g$  from the document. For example, if the granularity is 3, this strategy selects the 3-word phrases that begin at position  $0, 3, 6, 9, \dots, f_d$ . This strategy is suited to variable resolution.

*First- $r$*  selection is suited to a fixed resolution. This strategy selects the first  $r$  non-overlapping substrings of length  $g$  from the document, where  $r$  is the resolution and  $g$  is the granularity. For example, if the granularity is 3, this strategy selects the 3-word phrases that begin at position  $0, 3, 6, 9, \dots, 3r$ .

*First- $r$ -sliding* selection is similar to the first- $r$  strategy, with the difference that it is based on overlapping phrases. For example, if the granularity is 3, this strategy selects the 3-word phrases that begin at position  $0, 1, 2, 3, \dots, r$ .

**Frequency-based strategies.** These select phrases based on their frequency. The intuition is that phrases that are less common are more effective discriminators when comparing documents for similarity.

*Rarest-in-document* selection chooses the substrings that produce the rarest minutiae in the document. This means that all of the minutiae must be calculated and sorted according to the

frequency in the document, then the rarest  $r$  of them selected. This strategy suffers from the problem that many of the minutiae will appear only once in any one document, resulting in only the most common substrings being eliminated; the selection would then fall to the first  $r$  substrings that are not repeated in the document. We have not tested this strategy.

*Rarest-in-collection* selection requires generation of all the minutiae for all documents. They are then sorted according to the frequency of the minutia in the collection, rather than the frequency in the document. The rarest  $r$  minutiae are selected. This strategy is intended to reduce the number of coincidental matches caused by the matching of common phrases. We have not tested this strategy.

*Rarest prefix* selection begins by finding all distinct  $p$ -character strings that form the start of a word. For each of these strings, the number of occurrences in the document is counted. The  $r$  substrings beginning with the rarest prefixes in the document are selected. This approach is suited to a fixed fingerprint resolution.

**Structure-based strategies.** These strategies use the structure of the document. This allows detection of co-derivatives after changes in word positions that can affect the positional strategies, and changes in word or minutia frequencies which can affect the frequency-based strategies.

*Anchor* selection works by locating specific, predefined strings, or *anchors*, in the text of the document. The anchors are chosen to be common enough that there is at least one in almost every document, but not so common that the fingerprint becomes very large [7]. Thirty-five 2-character anchors were used for the results in this paper. They were selected at random, but extremely common strings such as “th” and “it” were rejected. The anchor strategy is suited to variable resolution.

*Kth-in-sentence* selection chooses phrases beginning at the  $K$ th word in each sentence. This strategy is suitable for either fixed or variable fingerprint resolution.

*Kth-sentence* selection chooses phrases beginning at the start of every  $K$ th sentence in the document. It can be used for both fixed and variable fingerprint resolution.

*Kth-in-paragraph* selection chooses phrases beginning at the  $K$ th word of each paragraph in the document. It can be used for either fixed or variable resolution.

## Previous work on fingerprinting

Manber [7] applied the fingerprinting approach to the n-to-n problem using a fixed resolution fingerprint for the index to the collection, and full fingerprinting for the query document. Manber achieved impressive results in a collection of over 20,000 “readme” documents, identifying 3620 groups of identical files and 2810 groups of similar files. Manber also proposed, but did not test, the anchor selection strategy.

Manber’s work was followed up by Brin, Davis, and Garcia-Molina [1]. A system called COPS was developed, using three selection strategies, all of which produce variable-resolution fingerprints. Full fingerprinting was tested, as well as the all-phrases strategy, both of which used a granularity of one sentence. Another strategy, hashed breakpoints, was also tested. This method extends the phrase if the hash value is equal to a multiple of a constant,  $k$ . This approach produced promising results, with correct documents scoring an average of  $52.9\% \pm 25.16\%$  similarity in their experiments, and incorrect results scoring below 3% .

Garcia-Molina and Shivakumar continued this research with SCAM [12], which used a granularity of one word, as opposed to the sentence-based granularity used in the COPS system. This in effect appears to be a simple form of inner product. SCAM used a variable resolution fingerprint; it is not clear what selection strategy was employed. This approach showed some improvement over the COPS system. In later work, they focussed on the performance of various approaches to fingerprinting [13] and the problem of scaling up to multi-gigabyte databases [13].

Both SCAM and COPS differ from our work and that of Heintze [5], in that they stored not only the hash value for the phrases selected, but the actual phrases as well. This allowed reduction in hashing collisions, but required nearly one hundred times the storage space. Heintze took the approach of storing just the hash value [5], using full fingerprinting as well as the random strategy with a fixed resolution. He also used the rarest prefix strategy, and another selection strategy which produces fixed resolution fingerprints by selecting phrases producing the lowest hash values. His experiments used a resolution of 100 for the index and 1000 for the query.

Broder, Glassman, and Manasse [2] used full fingerprinting with a granularity of 10 words, and tested on a collection of 150 Gb. This produced impressive results, but it is not clear how these results were obtained. It is apparent from the results that the figures quoted are estimates.

Chowdhury et al. [3] extended this work by intelligent pre-processing of the data. Their system, called I-Match, filters out terms based on inverse document frequency. The most common terms and the rarest terms are both removed. A key advantage of the ranking method is that it weights the importance of document terms according to the frequency. I-Match provides a method to apply this to fingerprinting. Their technique was successful, but focussed on detecting near-exact copies. As each document is represented by only one hash value, the method would be unable to locate the kinds of co-derivatives we have identified in our test collections. Whether removal of terms is of value to other co-derivative matching methods is a topic for further research.

The performance of some of these methods are discussed in the context of our experimental results. We are not aware of previous work on use of ranking techniques for detecting co-derivatives, or of previous work comparing a large range of co-derivative detection techniques.

## 6 Measuring effectiveness

The similarity methods described above assign a score to each document in a collection and thus a ranking of the documents. Using human assessments of whether documents are co-derivatives, it is possible to measure the effectiveness of different similarity methods. However, it is not always satisfactory to have a list of documents in order of similarity to the query. It can be preferable to have an indication of the degree of similarity on an absolute scale [1, 12]. This objective is achievable for the task of detecting co-derivatives.

In standard querying, a query is typically a short list of words, of the same topic (hopefully) as a relevant document but not otherwise similar to it. For co-derivatives, however, there can be no document more pertinent to a query than the query document itself. When searching for co-derivatives, no document should be ranked more highly than one that is identical to the query. We refer to a similarity measure for which this property holds as *ideal*.

If the query document is scored against itself prior to the query being executed, the score given to the query by an ideal similarity measure is guaranteed to be the highest possible score for that query in the given collection. If the score for each document is divided by this highest possible score, these are normalised to a value between 0 and 1, which can be multiplied by 100 to give a percentage. This is an indicator that is simple for a human operator to understand and provides an absolute similarity score. An absolute similarity score is not possible with ordinary ranking.

For estimating system performance, *recall* and *precision* are widely used as metrics for information retrieval systems [18]. Precision is the proportion of the returned documents that are correct answers; recall is the proportion of the correct answers that are returned. Both of these metrics can be calculated after any number of documents retrieved. We measured the precision after  $s$  documents retrieved, where  $s$  is the number of correct matches for the query (known as R-precision or missed-at equivalence). We measured the recall after 20 documents retrieved, or, when less than 20 documents were returned, we measured the recall after all the documents were retrieved.

These metrics are useful for measuring the effectiveness of standard information retrieval sys-

Evaluating query 1				
Document	1	Similarity	53.57	(100.00%)
Document	5	Similarity	53.04	( 99.01%)
Document	3	Similarity	45.96	( 85.79%)
Document	6	Similarity	44.95	( 83.91%)
Document	8	Similarity	44.37	( 82.82%)
Document	4	Similarity	43.12	( 80.49%)
Document	10	Similarity	41.21	( 76.93%)
Document	2	Similarity	36.01	( 67.22%)
Document	7	Similarity	35.93	( 67.07%)
Document	9	Similarity	21.93	( 40.93%)
Document	3064	Similarity	11.86	( 22.14%)
Document	520	Similarity	11.06	( 20.64%)
Document	1298	Similarity	10.75	( 20.07%)
Document	509	Similarity	10.67	( 19.91%)
Document	2892	Similarity	10.12	( 18.89%)
Document	1578	Similarity	9.86	( 18.41%)
Document	3272	Similarity	9.54	( 17.80%)
Document	1721	Similarity	9.48	( 17.70%)
Document	30	Similarity	9.31	( 17.37%)
Document	494	Similarity	9.26	( 17.29%)

Figure 1: *Example ranking, illustrating HFM and separation.*

tems [19]; however, they are not well-suited to this task. Many of the methods used rank all of the correct documents ahead of all other documents. If we consider a query  $q$ , to which there are  $s$  similar documents in the collection and we measure precision and recall after  $s$  results, both the precision and recall are 1.00.

A better metric is the combination of *highest false match* and *separation*, both of which require an absolute score for returned documents. The highest false match (HFM) is the highest percentage given to an incorrect result. A low score is desirable for HFM. The separation is the difference between the lowest correct result and the HFM. This score gives a measurement of how much distinguishes a correct result from an incorrect one, and is only meaningful if all the correct documents have been identified, that is, if precision and recall are both 1.0.

Both the HFM and separation are useful indicators of query effectiveness, but they need to be considered together. A high HFM is acceptable if the separation is high. Similarly, low separation is acceptable if the HFM is also low. The ratio of HFM to separation provides a useful indicator of overall effectiveness.

Figure 1 is an example of the output of a query on the XML data (described below). The first row shows the query number. The following twenty rows show the results of the query in descending order of similarity. The first result is document 1. It has a similarity score of 53.57, or, as a percentage in the final column, 100.00%; it is a copy of the query document. The correct results for this query are documents 1–10. None of the first  $s$  results are incorrect, so this query achieved a precision at  $s$  results of 1.0. The twenty results shown list all of the ten correct results, so the recall at 20 is also 1.0. The first document on the list that is not a correct result is document 3064, which scored a similarity of 11.86, or 22.14%. Hence, the HFM is 22.14%. The difference between this and the lowest correct match (document 9) is the separation, in this case 18.79%. Dividing the separation by the HFM, we get the ratio, which in this case is 0.85.

Table 1: *Results of ranked queries on XML data using traditional similarity measures. All results are averages over 10 queries.*

	Precision( $s$ )	Recall(20)	HFM	Sep.	Sep./HFM
Inner product	0.39	0.73	98.82%	-34.43%	-0.35
Normalised i.p.	1.00	1.00	21.61%	46.68%	2.16
Cosine measure	1.00	1.00	18.96%	48.57%	2.56

## 7 Experiments

### Test data

Two document collections were used to evaluate the algorithms described above. The first, XML data, was a collection of 3,300 documents seeded with ten documents that were known to be similar, all modifications of a single original document. Each was produced by a different author and has significant differences, but, when compared, it is clear that they were derived from the same source. The remainder of the documents in this collection are articles from the Internet, extracted from the TREC web data [4]. The queries used for this collection were the ten documents. Both the query documents and the documents in the collection were reduced to strings of words with all formatting information removed.

The second collection, Linux data, was a larger collection known to contain multiple versions of many documents. It consists of over 80,000 documents totalling 462 megabytes, drawn from RedHat Linux distributions. Documentation in PostScript, troff, HTML, and info formats were extracted from eleven versions of RedHat Linux and were pre-processed to convert them to plain text; documents of less than 50 words were discarded. The 53 queries were chosen for their variation in size, content, document format, and release date. By manual inspection, correct matches were identified for each of these queries. As with the XML collection, all formatting information was removed from both the documents in the collection, as well as the query documents.

### XML experiments

Our initial experiments were on the XML data. Due to the small size of this collection and the fact that the correct results are clearly identified, this collection was used for much of the initial experimentation and for development of the methods, but it is artificial.

Table 1 shows the outcome of queries on the XML data using traditional ranking methods: the inner product, normalised inner product, and cosine similarity measures. The first column shows the method used. The second shows the precision, measured after  $s$  query results. For all of the queries in the XML collection,  $s$  is equal to 10. The third column shows the recall achieved for the query, measured at 20 documents retrieved. The fourth column shows the HFM, the fifth shows separation between the HFM and the lowest correct match, and the last shows the ratio between the separation and HFM.

The results for the inner product similarity measure are uninspiring. On average, the inner product achieved a precision of only 0.39, meaning that only 39% of the first 10 matches were correct. The recall value of 0.73 shows that on average only 7 to 8 of the 10 correct matches were identified in the top 20 results. For each query, separation is negative if the highest false match is above the lowest correct match, and thus may be negative overall.

For the normalised inner product and the cosine measure, the precision and recall for all queries is 1.00. This means that the ten correct documents were ranked highest, demonstrating a dramatic improvement due to normalising for length. The HFM and separation for both measures were surprisingly good. The separation score shows that not only do these measures rank the

Table 2: *Results of ranked queries on the XML data using variations of the identity measure.*

	Precision( <i>s</i> )	Recall(20)	HFM	Sep.	Sep./HFM
Var. 1	0.47	0.64	4.34%	7.12%	1.64
Var. 2	0.99	1.00	21.19%	14.25%	0.67
Var. 3	0.97	0.99	18.46%	6.70%	0.36
Var. 4	0.99	1.00	20.62%	14.88%	0.72
Var. 5	1.00	1.00	11.16%	24.64%	2.21

Table 3: *The effect of stopping and stemming on the effectiveness of ranked queries on the XML data, based on variation 5 of the identity measure.*

	Precision( <i>s</i> )	Recall(20)	HFM	Sep.	Sep./HFM
Original data	1.00	1.00	8.74%	29.44%	3.37
Stopped	1.00	1.00	8.09%	30.07%	3.72
Stemmed	1.00	1.00	11.95%	24.75%	2.07
Stopped & stemmed	1.00	1.00	11.16%	24.64%	2.26

correct documents first, but they are clearly differentiated from the incorrect documents.

Table 2 shows the results for the XML data using the variants of the identity measure. Variation 1 of the identity measure was disappointing, much inferior to the results shown by the traditional similarity measures. Analysis of the documents that were ranked highly shows that they are close in length to the query, suggesting that this variation is too sensitive to changes in document length. For variation 2, which uses the log of the difference in document length as a discriminator, results are much more encouraging, but still poor compared to the standard similarity measures. Variation 3 is less effective, suggesting that the use of the sum of term frequencies is not effective as a discriminator. Variation 4 finds all but one of the correct documents first, and is close in performance to variation 2, suggesting that the use of  $\log(N/f_t)$  rather than  $\log(1 + N/f_t)$  for the term weight discriminator does not have a significant impact on effectiveness.

Variation 5 is much the most effective. The HFM is below 12%—easily the best of any of the similarity measures tested so far—and there is high separation—nearly 25%. However, the separation-to-HFM ratio is lower than for cosine.

The final experiment using ranking on the XML data set was designed to ascertain the effect of stopping and stemming on similar document detection, with variation 5 of the identity measure. Results are shown in Table 3. This experiment shows an unexpected result. For querying via a search engine, stemming generally increases the effectiveness of query evaluation, while stopping generally only effects the efficiency [18]. These results, however, show a different result in the context of similar document detection.

The first line, where there is no stopping or stemming, shows a significant improvement over the earlier results, where stopping and stemming were both used, repeated for convenience in the last line. Just stopping, without stemming, gives even better results; overall, it is clear that stemming degrades performance, while stopping improves it. It is evident that word suffixes hold information that is useful in finding co-derivatives, while stopping reduced the incidence of false matches caused by similar frequencies of common words.

We found that fingerprinting can also be successful in identifying similar documents. The first experiments with fingerprinting were designed to evaluate the selection strategies described above. All of the queries in these experiments have been evaluated using a fingerprint granularity of 3 words. The fixed resolution selection strategies used 15 minutiae. Table 4 shows the results.

Full fingerprinting listed the correct documents first for most of the queries. The HFM averaged

Table 4: Results of queries on XML data using fingerprints generated with granularity of 3.

	Precision( <i>s</i> )	Recall(20)	HFM	Sep.	Sep./HFM
Full fingerprinting	0.94	0.99	33.80%	0.26%	0.01
<i>Positional selection</i>					
All substrings	0.73	0.91	15.72%	-5.38%	-0.34
First- <i>r</i>	0.73	0.76	6.67%	1.33%	0.20
First- <i>r</i> -sliding	1.00	1.00	0.00%	20.00%	n/a
<i>Frequency-based selection</i>					
Rarest prefix	0.32	0.40	6.72%	18.67%	2.78
<i>Structure-based selection</i>					
Anchor strategy	1.00	1.00	16.60%	13.36%	0.80
<i>K</i> th in sentence	0.97	0.98	6.84%	5.50%	0.80
<i>K</i> th sentence	0.30	0.30	13.58%	19.00%	1.40
<i>K</i> th in paragraph	0.58	0.58	5.66%	22.00%	3.89

just over 33%, but the separation was almost zero. We had expected that full fingerprinting would be the most accurate of the fingerprinting methods, but this was not the case. This may be due to the many false matches that are introduced as the resolution of the fingerprint increases.

The next strategies, all substrings and first-*r*, were not inspiring. Because the phrases that they select do not overlap, these two strategies are both very sensitive to word insertions and deletions. For example, consider the document “ABCDEF”, where each letter represents a word. With a granularity of 3, the all-substrings strategy would select phrases beginning at A and D. Consider another document “XABCDEF”, from which the phrases beginning at X, C, and F would be selected. Despite the fact that these two documents are almost identical, the all-substrings strategy would not select any common phrases.

The last line of the second block of Table 4 shows results the first-*r*-sliding strategy. This strategy is not as sensitive to insertions and deletions as the previous two, as is reflected in the better results: this is the only method that produced no false matches at all for any of the queries. This is a remarkable result, but a careful inspection of the correct documents reveals that they are all very similar at the beginning—all have the same heading, for example. If the same documents were used with different headings, this result would not occur.

Only one frequency-based strategy was explored in this paper, the rarest-prefix strategy, which we expected to perform well. However, on average, this strategy listed less than four of the correct documents in the top 10. This strategy may be poor because insertion and deletion of words can significantly alter the frequency of the prefixes, meaning that different phrases are selected. The situation may also arise when there are a large number of prefixes that occur only once in the document. If this happens, then the selection strategy becomes little better than random selection. This result contrasts with the results obtained by Heintze, who achieved much better results with this strategy [5], but the reason for this discrepancy is unclear.

The final group of selection strategies are structure-based. The results for the anchor strategy are very good, and the *K*th-in-sentence strategy was also effective. These results are based on  $K = 5$ , meaning that phrases starting at the fifth word of each sentence were selected. The two last strategies did not work as well. For the *K*th-sentence strategy, only a small number of minutiae were produced, as phrases were selected only at every fifth sentence; a document of 20 sentences would produce only four minutiae. Also, as for the “all substrings” and “first-*r*” strategies, this strategy is also sensitive to alignment problems caused by insertion and deletion. The results for the final strategy, *K*th-in-paragraph, are with  $K = 15$ . These results are also somewhat disappointing, perhaps due to alignment problems, as a single change in each paragraph in the

Table 5: *The impact of changing the fingerprint granularity on the effectiveness of fingerprint queries. All of these queries have been evaluated using the anchor selection strategy.*

Granularity	Precision(s)	Recall(20)	HFM	Sep.	Sep./HFM
1	0.85	0.93	66.06%	-3.56%	-0.05
2	0.99	1.00	30.23%	8.82%	0.29
3	0.99	1.00	19.60%	14.34%	0.73
4	0.99	1.00	16.59%	13.29%	0.80
5	0.98	1.00	15.60%	9.16%	0.59
6	0.93	0.99	16.72%	3.78%	0.23
7	0.89	0.97	17.46%	-0.61%	-0.03
8	0.86	0.93	17.13%	-1.13%	-0.07
9	0.78	0.89	18.86%	-4.45%	-0.24
10	0.68	0.85	17.51%	-5.22%	-0.30
15	0.38	0.46	16.61%	-3.42%	-0.21
20	0.27	0.31	17.50%	23.94%	1.37

document would prevent a match from being found.

The last experiments on the XML data tested the effect of changing the fingerprint granularity, shown in Table 5. The best results achieved were with a granularity of 4, but a granularity of 3 or 5 also produced a good result. By way of comparison, Manber used a granularity of 50 characters [7], Brin et al. used a granularity of one sentence [1], Shivakumar et al. had the best results with a granularity of one sentence [13], and Heintze used 30–45 characters [5]. All of these equate to around eight or nine words. Our results suggest that all of these would achieve a low HFM, but the separation would not be strong, which is reflected in the results shown by Heintze [5]. Overall, in none of our experiments did the methods tested by other researchers work well.

## Linux experiments

From the results obtained using the XML data, it is clear that some of the methods tested are not suitable for detecting co-derivatives. Only the methods found to perform well in the XML collection were tested on the Linux documentation. The size of this collection enabled an analysis of these methods in a more realistic environment.

Queries on the Linux data set were evaluated using two similarity measures and three fingerprinting techniques, as shown in Table 6. These queries were evaluated with stopping but without stemming, following the results shown in Table 3. For fingerprinting, following the results from the XML data, we used a phrase granularity of four words. We used the two most successful selection strategies from the XML queries—the anchor strategy and the first- $r$ -sliding strategy. The first- $r$ -sliding queries were evaluated using a resolution of 15. We also used a full fingerprint query against the index created using the anchor strategy.

The results for ranking show very different behaviour to that obtained on the XML data. The identity measure is far superior to the alternatives. It has the best precision, recall, HFM, and separation—that is, it shows the best performance by every metric. These results are a vindication of the design philosophy that led to the identity measure.

Of the fingerprinting methods, the anchor strategy shows the best results. All fingerprinting methods are superior to cosine, but inferior to the identity measure. Interestingly, the use of full fingerprinting for the query was not successful.

Another consideration when evaluating such techniques is consistency. Examining the results for individual queries, the identity measure and the anchor strategy are much more reliable than the alternatives. For the identity measure, there were only 7 queries with precision less than 1.00,

Table 6: *Results of queries on Linux data. These queries were evaluated with stopping but no stemming.*

	Precision( <i>s</i> )	Recall(20)	HFM	Sep.	Sep./HFM
<i>Ranking</i>					
Cosine measure	0.68	0.77	73.46%	10.92%	0.15
Identity measure var. 5	0.97	0.97	25.25%	51.75%	2.05
<i>Fingerprinting</i>					
First- <i>r</i> -sliding	0.58	0.67	67.18%	20.38%	0.30
Anchor strategy	0.92	0.92	40.14%	45.75%	1.14
Anchor strategy/full query	0.82	0.88	73.79%	16.31%	0.22

compared to 10 for the anchor strategy, 29 for the cosine measure, 35 for the first-*r*-sliding strategy, and 23 for the full-query strategy. For both the identity measure and the anchor strategy, there was only 1 case with negative separation, but 20, 5, and 13 cases respectively for the other techniques.

## 8 Further experiments

Ranking and fingerprinting are not the only approaches to detection of co-derivatives. Greedy string tiling has been applied to the task of detecting plagiarism in student programming assignments [15, 16, 17], as well as in textual assignments [16]. However, in preliminary work applying this approach to student assignments, we did not achieve good results; nor is it clear how the approach could be efficiently applied to large collections. In contrast, ranking and fingerprinting can be supported through an index.

Further experiments were run on another collection, which consisted of 145 individual essays produced by tertiary level students. The assignments were free-form and the students were permitted to choose from a variety of topics. With the identity measure, n-to-n comparison of these documents found 36 matches that were highly ranked with good separation. Manual inspection confirmed that all of these highly-ranked matches were plagiarisms. However, we believed that there were further plagiarised assignments to be found.

Another approach we have preliminarily tested is to match, not whole documents, but passages, which are fixed-sized chunks of documents [6]. We broke each of the documents in the collection into a set of sub-document chunks of 50 words, overlapping by 25 words each. We did the same to the query document, then queried the collection using each chunk, counting the number of chunks that had high similarity in each pair of documents. This led to 59 matches that were highly ranked, of which 55 were confirmed to be plagiarised: the original 36 and a further 19 that contained significant plagiarised passages but which overall had been edited sufficiently to conceal the copying. On visual inspection, the remaining 4 matches did not appear to be copies. Nonetheless, it is clear that matching of parts of documents is a powerful tool for increasing the sensitivity of these plagiarism detection techniques.

A final consideration is efficiency, in space and speed. With an efficient representation, the full-text index required for the identity measure is expected to occupy less than 10% of the space needed for the data itself [8, 18]. An index for the anchor strategy is likely to be even more compact. In our experiments, using code designed for flexibility (to test a range of co-derivative detection techniques) rather than efficiency, fingerprinting queries took substantially longer to evaluate than did ranked queries, but in a production implementation we estimate that the speeds should be similar. An area for further research is, when using the identity measure, reducing the set of terms chosen for evaluation from a query document.

## 9 Conclusions

We have explored existing techniques for identifying copies of documents, which we call co-derivatives, and have proposed a new identity measure explicitly designed for this task. Existing methods include similarity measures—designed for information retrieval, but suitable for identifying co-derivatives—and fingerprinting techniques, based on hashing. Our presentation of fingerprinting techniques synthesises a range of proposals into a single consistent framework, allowing them to be directly compared.

Our identity measure is based on the observation that identifying documents on the same topic—the standard task of information retrieval—involves measuring how much material the documents have in common; in contrast, identifying copies of documents involves measuring how much they differ. The experimental results on a large data set with 53 queries show that the identity measure is much superior to the alternatives. Only one previous technique, the anchor strategy, has acceptable performance, but is nonetheless much weaker than the identity measure. These results also showed that the key to achieving an accurate result with fingerprinting is the phrase selection strategy. Fingerprinting is sensitive to the size of the phrases selected. A granularity of between three and five words produced the best results, broadly confirming choice of parameters in earlier work.

Both the identity measure and the anchor strategy are able to identify all of the documents considered similar by a human observer in a large collection, and both clearly separate these similar documents from the rest of the collection. In contrast, the cosine measure was able to perform well when the similar documents contained subject matter that was strongly distinct from the rest of the collection, but was poor on the more realistic Linux collection. The identity measure performed well in both of these situations, and is clearly the best of the methods for identifying co-derivative documents.

### Acknowledgements

This work was supported by the Australian Research Council. We thank Hugh E. Williams and James A. Thom.

### References

- [1] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. In *Proc. ACM SIGMOD Annual Conference*, San Jose, CA, May 1995.
- [2] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Sixth International World Wide Web Conference*, April 1997.
- [3] A. Chowdhury, O. Frieder, D. Grossman, and M. McCabe. Collection statistics for fast duplicate document detection. *ACM Transactions on Information Systems*. To appear.
- [4] D. Harman. Overview of the second text retrieval conference (TREC-2). *Information Processing & Management*, 31(3):271–289, 1995.
- [5] N. Heintze. Scalable document fingerprinting (extended abstract). In *Proc. USENIX Workshop on Electronic Commerce*, November 1996.
- [6] M. Kaszkiel and J. Zobel. Effective ranking with arbitrary passages. *Journal of the American Society of Information Science and Technology*, 42(4):344–364, 2001.
- [7] U. Manber. Finding similar files in a large file system. In *1994 Winter USENIX Technical Conference*, pages 1–10, San Francisco, CA, January 1994.

- [8] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems*, 14(4):349–379, October 1996.
- [9] K. Monostori, A. Zaslavsky, and H. Schmidt. Document overlap detection system for distributed digital libraries. In *Proceedings of the fifth ACM conference on Digital Libraries*, pages 226–227, 2000.
- [10] M.V. Ramakrishna and J. Zobel. Performance in practice of string hashing functions. In *Proc. Int. Conf. on Database Systems for Advanced Applications*, pages 215–223, Melbourne, Australia, April 1997.
- [11] M. Sanderson. Duplicate detection in the Reuters collection. Technical Report TR-1997-5, Department of Computing Science, University of Glasgow, 1997.
- [12] N. Shivakumar and H. Garcia-Molina. SCAM: a copy detection mechanism for digital documents. In *Proc. International Conference on Theory and Practice of Digital Libraries*, Austin, Texas, June 1995.
- [13] N. Shivakumar and H. Garcia-Molina. Building a scalable and accurate copy detection mechanism. In *Proc. ACM Conference on Digital Libraries*, Bethesda, Maryland, March 1996.
- [14] N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents on the web. In *Proc. Workshop on Web Databases*, March 1998.
- [15] M. J. Wise. Detection of similarities in student programs: Yap’ing may be preferable to plague’ing. In *Twenty-Third SIGSCE Technical Symposium*, pages 268–271, Kansas City, USA, February 1992.
- [16] M. J. Wise. String similarity via greedy string tiling and running Karp-Rabin matching. <ftp://ftp.cs.su.oz.au/michaelw/doc/RKR GST.ps>, 1993.
- [17] M. J. Wise. YAP3: improved detection of similarities in computer programs and othertexts. In *Twenty-Seventh SIGCSE Technical Symposium*, pages 130–134, Philadelphia, USA, March 1996.
- [18] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and indexing documents and images*. Morgan Kaufmann, second edition, 1999.
- [19] J. Zobel. How reliable are the results of large-scale information retrieval experiments? In R. Wilkinson, B. Croft, K. van Rijsbergen, A. Moffat, and J. Zobel, editors, *Proc. ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314, Melbourne, Australia, July 1998.
- [20] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1):18–34, 1998.