A Comparison of Text-Based Methods for Detecting Duplication in Document Image Databases*

Daniel P. Lopresti

Bell Labs, Lucent Technologies Inc.
600 Mountain Avenue, Room 2D-447
Murray Hill, NJ 07974, USA
dpl@research.bell-labs.com

ABSTRACT

This paper presents an experimental evaluation of several text-based methods for detecting duplication in document image databases using uncorrected OCR output. This task is challenging because of both the wide range of degradations printed documents can suffer, and conflicting interpretations of what it means to be a "duplicate." We report results for five sets of experiments exploring various aspects of the problem space. While the techniques studied are generally robust in the face of most types of OCR errors, there are nonetheless important differences which we identify and discuss in detail.

Keywords: duplicate detection, approximate string matching, information retrieval, optical character recognition, document analysis

1. INTRODUCTION

An important problem facing document management systems is determining whether duplicates already exist in the database when a new document arrives for processing. For example, one particular collection activity for the U.S. Government's Gulf War Declassification Project accumulated 564,000 pages, the majority of which (292,000 pages) were later found to be duplicates of documents already on hand.[2] Clearly, techniques for detecting duplicates could prove extremely valuable, both in terms of cost savings as well as deepening our understanding of the relationship between documents.

Although this task might seem straightforward at first glance, it becomes quite challenging when one considers the different possible interpretations of what it means to be a "duplicate," and the many types of damage that can be inflicted on a physical medium such as the printed page. In previous papers,[6,7] we introduced a formalism based on approximate string matching for categorizing duplicates into four classes as illustrated in Figure 1:

Full-layout duplicates are visually identical (*e.g.*, one is a photocopy or fax of the other).

Full-content duplicates have identical textual content, but not necessarily the same layout of text lines (*e.g.*, a Web page printed twice using different margin settings).

Partial-layout duplicates share significant content and have the same layout (*e.g.*, the copy-and-pasting of whole paragraphs from one document to another).

Partial-content duplicates share significant content but not necessarily the same layout (*e.g.*, the copy-and-pasting of individual sentences).

In addition, we presented a family of algorithms, one for each of the four classes, which operate on the "raw" (*i.e.*, uncorrected) output of an optical character recognition (OCR) process. We presented exerimental results showing that these methods were robust in the presence of real-world noise, and that the models could successfully distinguish the various types of duplication.

There are, of course, a number of other well-known approaches to searching textual databases. While schemes predicated on finding long strings of perfect similarity will likely fail when noisy documents are included in the

* Presented at *Document Recognition and Retrieval VII (IS&T/SPIE Electronic Imaging 2000)*, January 2000, San Jose, CA.

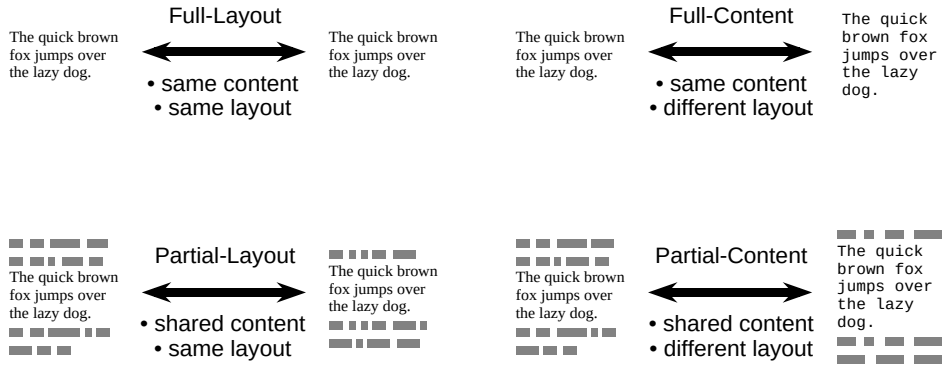

Figure 1. The four duplicate classes.

database, the traditional vector space metric from information retrieval, for example, has been found to be relatively immune to OCR errors.[14] Identifying plagiarism and copyright violations, where some degree of difference between the documents in question can naturally be expected, is also a related problem.[11] Such methods could be run on OCR output in an attempt to identify duplicates in document image databases.

The present paper makes use of this work and builds on our previous results in duplicate detection in several ways. We present an experimental evaluation comparing other text-based methods to the string matching techniques mentioned earlier. We also examine three new "hard" scenarios: cases where the query is severely degraded (35% OCR error rate), the reading order is incorrectly determined, and the database contains numerous false duplicates. These highlight the strengths and weaknesses of the different approaches.

The remainder of this paper is organized as follows. In Section 2, we briefly review related work. The algorithms to be studied are summarized in Section 3. Experimental results are presented in Section 4. Finally, Section 5 gives our conclusions and discusses possible future work.

2. RELATED WORK

A number of researchers have begun to examine the problem of detecting duplicates in the context of document image databases.[1,3–5,8,9,13] Still, most previous work on this subject has concentrated on identifying which features to extract and not so much on the different ways they might be compared. This step is typically handled using one or another of the techniques from the literature.

Broadly speaking, these approaches can be classified depending on whether they operate on low-level image features[3,4,8,9] or on the output of a symbolic recognition process such as OCR.[1,5,13] The former are more general in the sense they can be applied to non-textual input (*e.g.*, drawings, photographs), but more limiting in that they can only be used to find full-layout duplicates. Our primary interest lies in techniques from the latter category.

Spitz, for example, employs character shape codes as features and compares them using a standard string matching algorithm.[13] In the taxonomy presented in Section 1, this corresponds to the full-content problem. Doermann, et al., also use shape codes, but extract n-grams for a specific text line to index into a table of document pointers.[1] Since this signature is computed from a single line, it does not explicitly measure the similarity of complete pages. The intention, though, is that this is a method for addressing the full-layout problem. Hull, et al., describe three techniques: one based on decomposing the page into a grid and counting connected components within each cell, another using word lengths as a hash key, and one comparing image features (pass codes arising from fax compression) under a Hausdorff distance measure.[4] More details on the last method appear in later paper.[3] The first and third of these fall in the full-layout category, while the second can be classified as searching for full-content duplicates. In a recent paper, Lee and Hull describe a method for performing duplicate detection on symbolically compressed images by solving the text deciphering problem through the use of Hidden Markov Models.[5] They then apply n-gram indexing with term weighting to detect duplicates, addressing the full-content problem.

Elsewhere, Taghva, et al., observed that traditional vector-space techniques from the field of information retrieval (IR) are for the most part unaffected by OCR errors when the input is relatively clean.[14] Also seemingly related is

the general copy detection problem. Shivakumar and Garcia-Molina have developed efficient methods for searching large online databases for signs of copyright infringement.[11] A later paper of theirs considers the task of identifying near-replicas on the World Wide Web (WWW) to improve the performance of Web crawlers, archivers, and search engines.[12] All of these approaches are text-based, employing character or word n-grams or longer syntactic entities (sentences, paragraphs, etc.), and must allow for the fact that two documents need not be identical for the results of their comparison to qualify as "interesting." There are, however, significant differences between a typical IR query and a complete document, and between the kinds of errors that arise during OCR and the steps taken to conceal an attempt at plagiarism. One question we seek to answer in this paper is how well these methods work for detecting duplicates in the presence of large amounts of "noise" and under the different models described in the Introduction.

3. ALGORITHMS

In this section, we briefly summarize the algorithms under study. The reader is referred to the original works in question for more details.

3.1. Approximate String Matching Applied to Duplicate Detection

The string measures are based on the well-known concept of *edit distance*. In the simplest case, the following three operations are permitted: (1) delete a symbol, (2) insert a symbol, (3) substitute one symbol for another. Each of these is assigned a cost, c_{del}, c_{ins}, and c_{sub}, and the edit distance is defined as the minimum cost of any sequence of basic operations that transforms one string into the other.

As it relates to full-content duplicates, this optimization problem can be solved using a dynamic programming algorithm.[15] Let $Q = q_1 q_2 \ldots q_m$ be the query document, $D = d_1 d_2 \ldots d_n$ be the database document, and define $dist1_{i,j}$ to be the distance between the first i symbols of Q and the first j symbols of D. The initial conditions are:

$$\begin{array}{lcll} dist1_{0,0} & = & 0 & \\ dist1_{i,0} & = & dist1_{i-1,0} + c_{del}(q_i) & 1 \le i \le m,\ 1 \le j \le n \\ dist1_{0,j} & = & dist1_{0,j-1} + c_{ins}(d_j) & \end{array} \tag{1}$$

and the main dynamic programming recurrence is:

$$dist1_{i,j} = \min \left\{ \begin{array}{lcl} dist1_{i-1,j} & + & c_{del}(q_i) \\ dist1_{i,j-1} & + & c_{ins}(d_j) \\ dist1_{i-1,j-1} & + & c_{sub}(q_i, d_j) \end{array} \right. \qquad 1 \le i \le m,\ 1 \le j \le n \tag{2}$$

The computation builds a matrix of distance values working from the upper left corner ($dist1_{0,0}$) to the lower right ($dist1_{m,n}$).

The other three string algorithms, *sdist1* for partial-content duplicates, *dist2* for full-layout duplicates, and *sdist2* for partial-layout duplicates, reflect various adaptations of this approach to allow for partial matchings and 2-D document structure.[6,7] For the partial duplicate problem, what is needed is the best match between any two substrings of Q and D. The edit distance is made 0 along the first row and column of the matrix, so the initial conditions become:

$$sdist1_{0,0} = sdist1_{i,0} = sdist1_{0,j} = 0 \qquad 1 \le i \le m,\ 1 \le j \le n \tag{3}$$

In addition, another term is added to the inner-loop recurrence capping the maximum distance at any cell to be 0. This has the effect of allowing a match to begin at any position between the two strings. The recurrence is:

$$sdist1_{i,j} = \min \left\{ \begin{array}{lcl} 0 & & \\ sdist1_{i-1,j} & + & c_{del}(q_i) \\ sdist1_{i,j-1} & + & c_{ins}(d_j) \\ sdist1_{i-1,j-1} & + & c_{sub}(q_i, d_j) \end{array} \right. \qquad 1 \le i \le m,\ 1 \le j \le n \tag{4}$$

Finally, the resulting distance matrix is searched for its smallest value. This reflects the end-point of the best substring match. The starting point can be found by tracing back the sequence of optimal editing decisions.

For the 2-D models (*i.e.*, layout duplicates), another level is added to the optimization. The problem is still one of editing, but at the higher level the basic entities are now strings (lines). Say that $Q = Q^1 Q^2 \ldots Q^k$ and

$D = D^1 D^2 \ldots D^l$, where each Q^i and D^j is itself a string. For full-layout duplicates, the inner-loop recurrence takes the same general form as the 1-D case:

$$dist2_{i,j} = \min \begin{cases} dist2_{i-1,j} & + \; C_{del}(Q^i) \\ dist2_{i,j-1} & + \; C_{ins}(D^j) \\ dist2_{i-1,j-1} & + \; C_{sub}(Q^i, D^j) \end{cases} \qquad 1 \le i \le k,\ 1 \le j \le l \tag{5}$$

where C_{del}, C_{ins}, and C_{sub} are the costs of deleting, inserting, and substituting whole lines, respectively. The initial conditions are defined analogously to Equation 1.

Since the basic editing operations now involve full strings, it is natural to define the new costs as:

$$\begin{array}{lcl} C_{del}(Q^i) & \equiv & dist1(Q^i, \phi) \\ C_{ins}(D^j) & \equiv & dist1(\phi, D^j) \\ C_{sub}(Q^i, D^j) & \equiv & dist1(Q^i, D^j) \end{array} \qquad 1 \le i \le k,\ 1 \le j \le l \tag{6}$$

where ϕ is the null string. Hence, the 2-D computation is defined in terms of the 1-D computation.

Lastly, the extension for partial-layout duplicates combines the modifications for the partial (Equation 4) and layout (Equation 5) problems:

$$sdist2_{i,j} = \min \begin{cases} 0 \\ sdist2_{i-1,j} & + \; C_{del}(Q^i) \\ sdist2_{i,j-1} & + \; C_{ins}(D^j) \\ sdist2_{i-1,j-1} & + \; C_{sub}(Q^i, D^j) \end{cases} \qquad 1 \le i \le k,\ 1 \le j \le l \tag{7}$$

Note that C_{del}, C_{ins}, and C_{sub} are defined as before in terms of *dist1* (*i.e.*, Equation 6).

At this point four different algorithms have been presented, one for each of the models described in the Introduction. For exact duplicates, the distance returned by any of the algorithms will either be 0 or a negative number that grows smaller as the lengths of the documents increase. For dissimilar documents, the maximum distance grows larger as the lengths increase. It is always the case that, for a given query, a smaller distance corresponds to a better match. In order for the results for different queries to be comparable, however, it is necessary to normalize the distances.

If the target interval is $[0, 1]$, where 0 represents a perfect match and 1 a complete mismatch, then the following formula provides an appropriate mapping:

$$EditSim(Q, D) = \frac{dist - mindist}{maxdist - mindist} \tag{8}$$

where *dist* is one of the four edit distance measures and *mindist* and *maxdist* are, respectively, the minimum and maximum possible distances for the comparison in question.

Assuming a full-duplicate computation, and making certain reasonable assumptions about the cost functions, the minimum is obtained when all of the characters in the query match the database document and there are no extra, unmatched characters. If the query is $Q = q_1 q_2 \ldots q_m$, then:

$$mindist = \sum_{i=1}^{m} c_{sub}(q_i, q_i) \tag{9}$$

Or, more simply, $mindist = m \cdot c_{mat}$ when the costs are constant and c_{mat} is the cost of an exact match.

The maximum distance, on the other hand, is determined by the query and the set of all strings with the same length as the database document. If the cost functions are unconstrained, this in itself becomes an optimization problem. Fortunately, for constant costs there is a simple closed-form solution. Without loss of generality, let the query be the shorter of the two strings (*i.e.*, $m \le n$). There are two possible "worst-case" scenarios: either all of the symbols of the query are substituted and the remaining symbols of the database string are inserted, or all of the query symbols are deleted and the entire database string is inserted. Thus:

$$maxdist = \min \begin{cases} m \cdot c_{sub} + (n - m) \cdot c_{ins} \\ m \cdot c_{del} + n \cdot c_{ins} \end{cases} \tag{10}$$

The partial-duplicate computations are normalized similarly.

3.2. The Vector Space IR Method

The vector space model first proposed by Salton is extremely popular in the field of information retrieval.[10] This approach assigns large weights to terms that occur frequently in a given document but rarely in others because such terms are able to distinguish the document in question from the rest of the database. Let tf_{ik} be the frequency of term t_k in document D_i, n_k be the number of documents containing term t_k, T be the total number of terms, and N be the size of the database. Then a common weighting scheme ($tf \times idf$) defines w_{ik}, the weight of term t_k in document D_i, to be:

$$w_{ik} = \frac{tf_{ik} \cdot log(N/n_k)}{\sqrt{\sum_{j=1}^{T} (tf_{ij})^2 \cdot (log(N/n_j))^2}} \tag{11}$$

The summation in the denominator normalizes the length of the vector so that all documents have an equal chance of being retrieved.

Given query and document vectors $Q_i = (w_{i1}, w_{i2}, \ldots, w_{iT})$ and $D_j = (w_{j1}, w_{j2}, \ldots, w_{jT})$, a vector dot product is computed to quantify the similarity between the two:

$$VSsim(Q_i, D_j) = \sum_{k=1}^{T} w_{ik} w_{jk} \tag{12}$$

3.3. The SCAM Algorithm for Plagiarism Detection

SCAM, the work of Shivakumar and Garcia-Molina, attempts to overcome some of the limitations of the vector space model when the lengths of the documents differ significantly.[11] They define a *closeness set* for two documents Q_i and D_j, $c(Q_i, D_j)$, to contain those terms that have a similar number of occurrences in both documents. That is, a term t_k is in $c(Q_i, D_j)$ if it satisfies the following condition:

$$\epsilon - \left(\frac{tf_{ik}}{tf_{jk}} + \frac{tf_{jk}}{tf_{ik}} \right) > 0 \tag{13}$$

where $\epsilon = (2^+, \infty)$ is a user-settable parameter.

Next, they define an asymmetric measure:

$$subset(Q_i, D_j) = \frac{\sum_{t_k \in c(Q_i, D_j)} \alpha_k^2 \cdot tf_{ik} \cdot tf_{jk}}{\sum_{k=1}^{T} \alpha_k^2 \cdot (tf_{ik})^2} \tag{14}$$

that reflects the degree to which Q_i is contained within (*i.e.*, is a partial-content duplicate of) D_j. The similarity between two documents is then computed to be:

$$SCAMsim(Q_i, D_j) = \max \{subset(Q_i, D_j), subset(D_j, Q_i)\} \tag{15}$$

4. EXPERIMENTAL RESULTS

To investigate the performance of the algorithms described in this paper, five sets of experiments were designed. The first three studied various real-world noise sources and their effects, the next examined the four duplicate models and how they relate, and the last looked at the impact of including multiple false duplicates in the database.

The base test collection consisted of 1,000 professionally written news articles gathered from Usenet and was used as-is (*i.e.*, no attempt was made to inject OCR errors, either real or synthetic). The query documents, however, and the intended duplicates were "authentic," pages that had been printed, scanned, and OCR'ed.

We analyzed the four string algorithms, vector space using both word unigram tokens (with stopword removal) and character trigram tokens (without stopword removal), and SCAM. All of the algorithms were coded in C and run on an SGI O2 workstation. To make the results directly comparable, the outputs were normalized to the interval $[0, 1]$, with 0 corresponding to a perfect match (this entailed subtracting the vector space results from 1). For the full-duplicate computations, the edit costs were set to be $c_{del} = c_{ins} = c_{sub} = 1$ and $c_{mat} = 0$. For the partial-duplicate computations, the match cost was changed to $c_{mat} = -1$. In our experiments using SCAM, we set $\alpha = 1$ and $\epsilon = 3$.

4.1. Experiment 1

The goal of this experiment was to study duplicate detection under realistic, moderate noise conditions. The source document was 1,395 characters long (26 lines, 203 words). Two sets of six pages were created, one set to be inserted into the database as the duplicates, and the other to serve as the queries. Within each set, one page was used as-is and the others were subjected to one of five different degradations: faxing, noticeably light or dark or third generation photocopying, or handwritten markup (annotations) that obscured five of the lines on the page. The pages were then scanned and OCR'ed. In addition, the original ASCII text for the query document was left in the database. Hence, each of six queries was run against a database of 1,000 documents containing seven intended duplicates, six that had been OCR'ed plus the original. For the present discussion, however, we focus on one query in particular, the faxed document.

Table 1 below shows the OCR accuracies. Note that the rates range widely, dropping as low as 75.6%. As expected, a large variety of OCR errors were encountered. Beyond this, other complications arose as well. For example, the standard headers prepended to faxes were transcribed (albeit with numerous mistakes), and the lines that had been crossed-out were completely missing from the annotated pages.

Table 1. OCR accuracies for Experiment 1.

	OCR Accuracy	
Document Type	Database	Query
Printed	96.2%	–
Faxed	77.7%	83.9%
3rd Generation	95.9%	–
Light	86.1%	–
Dark	94.0%	–
Annotated	75.6%	–

We then ran the *dist2* approximate string matching algorithm, as well as the vector space measure using single-word tokens and character trigrams and the SCAM method. The charts in Figure 2 plot, for the faxed query, the normalized distance computed by each of the four approaches for every document in the database. Note that there is usually a clear distinction between true duplicates and everything else. This demonstrates that all the techniques are robust when faced with the sorts of OCR errors seen in practice. (In the case of *dist2*, results for the rest of the queries can be found elsewhere.[7])

Certain other interesting effects can be seen in the charts. The annotated duplicate is ranked quite differently by the four methods. Recall that approximately 20% of the content of this document was completely obscured. SCAM tolerates this kind of damage quite well, as it is tuned to handle partial matchings. To be fair, however, the *dist2* algorithm is not designed to capture this form of similarity; *sdist2* or *sdist1* would be a more appropriate choice in this case. Note also that the faxed duplicate is a poorer match under all of the vector space methods (including SCAM) than under the string technique. This may be because the standard fax header includes terms that are relatively unique in the collection, but printed at such coarse resolution that they cannot be recognized reliably by OCR. Lastly, several non-duplicates are assigned relatively low distance scores by the vector space measures (the hollow circles plotted towards the right side of the charts). These were investigated and found to be follow-up postings to the original article chosen as the query. While sharing some of the same unique terminology (*e.g.*, proper names), they were by no means true duplicates.

4.2. Experiment 2

The purpose of this experiment was to study the impact of using a more severely degraded input. The query was copied at an extremely light setting on a photocopier, with the resulting OCR accuracy estimated to be 65.0%. Outputs for the same four algorithms used in the previous experiment are presented in Figure 3.

Note that these results are in some ways similar to those for the first experiment, with the values for the duplicates raised by a significant but fairly uniform amount in all cases. The light duplicates appear to be slightly better matches than before (relative to the other duplicates). This is probably because the same sorts of errors arose in the query and database documents (light input tends to induce segmentation errors: individual characters broken into two or

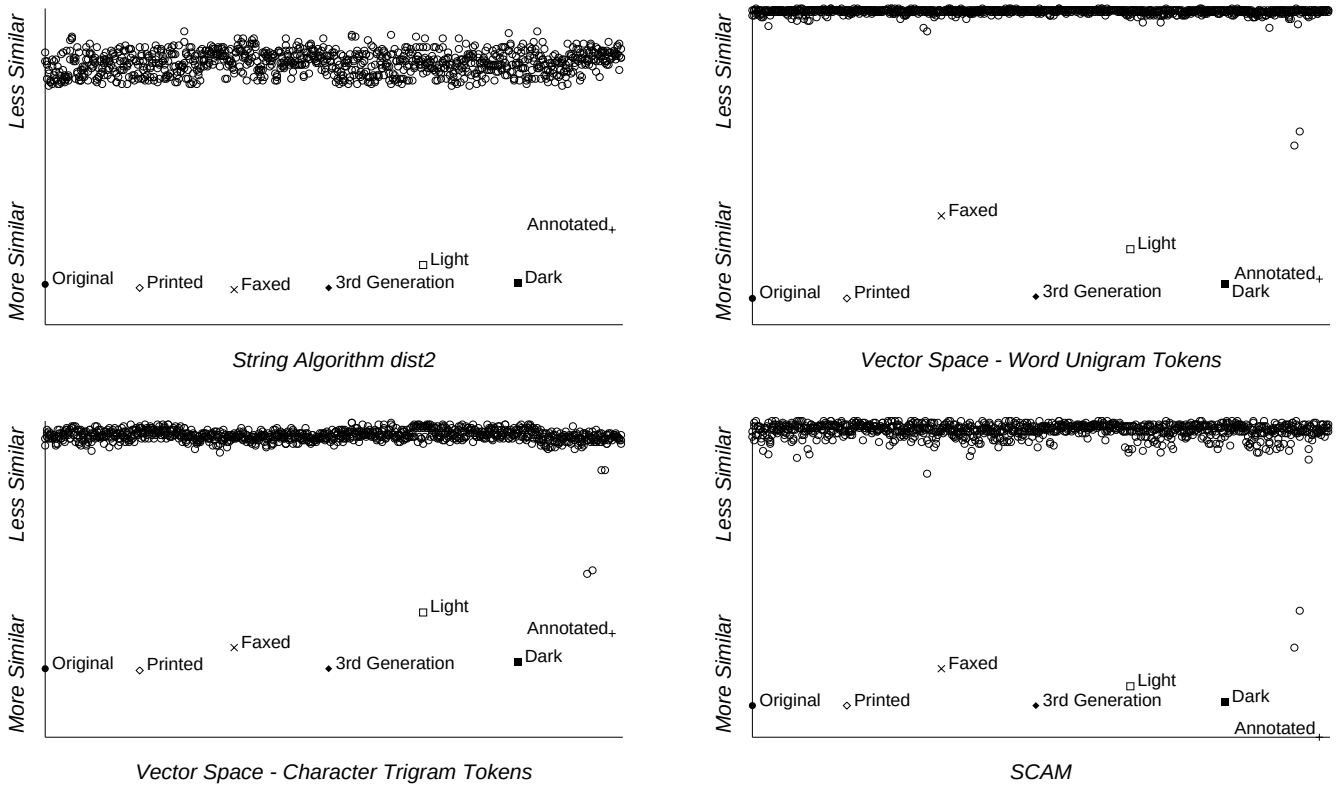

Figure 2. Results for Experiment 1 (faxed query).

more pieces). Closer study of the charts suggests that the vector space techniques may be more seriously affected by the additional noise than the string algorithm. An examination of the minimum and average separations between the duplicate/non-duplicate classes bears this out. Even so, all of the methods are still quite capable of detecting duplicates at this level of damage.

4.3. Experiment 3

A fundamental difference between the string-based models and those derived from the vector space approach is the dependence of the former on determining a canonical reading order for the text. (This reading order need not conform to the way a human would interpret the text, it need only be consistent from one document to the next.) While finding a reading order is generally not hard for single column documents, it can become challenging for complex, multicolumn layouts.

The intent of this experiment was to examine duplicate detection when reading order is a problem. We took the query from the previous experiments and formatted it in two-column mode, forcing the inter-column spacing to be so tight that the column break was lost on some (but not all) lines. As before, the page was photocopied light and then scanned and OCR'ed. In this case it was impossible to compute a value for the OCR accuracy, but visual inspection suggested it was probably somewhat higher than in Experiment 2. Figure 4 gives the results for this test.

As expected, the string technique nearly loses its ability to distinguish duplicates from non-duplicates. This is the penalty paid for not maintaining the same reading order. The vector space charts, on the other hand, look much like those for the first two experiments, lying somewhere between the earlier sets of results. Hence, they are immune to this sort of failure in the document analysis process.

4.4. Experiment 4

The purpose of this experiment was to determine how the different duplicate models and comparison measures relate empirically (recall Figure 1). The same source document was used as in the previous experiments. Duplicates were constructed from the query by changing the line breaks and/or deleting roughly half of the text from the beginning of the document and appending an equal amount of unrelated text to the end.

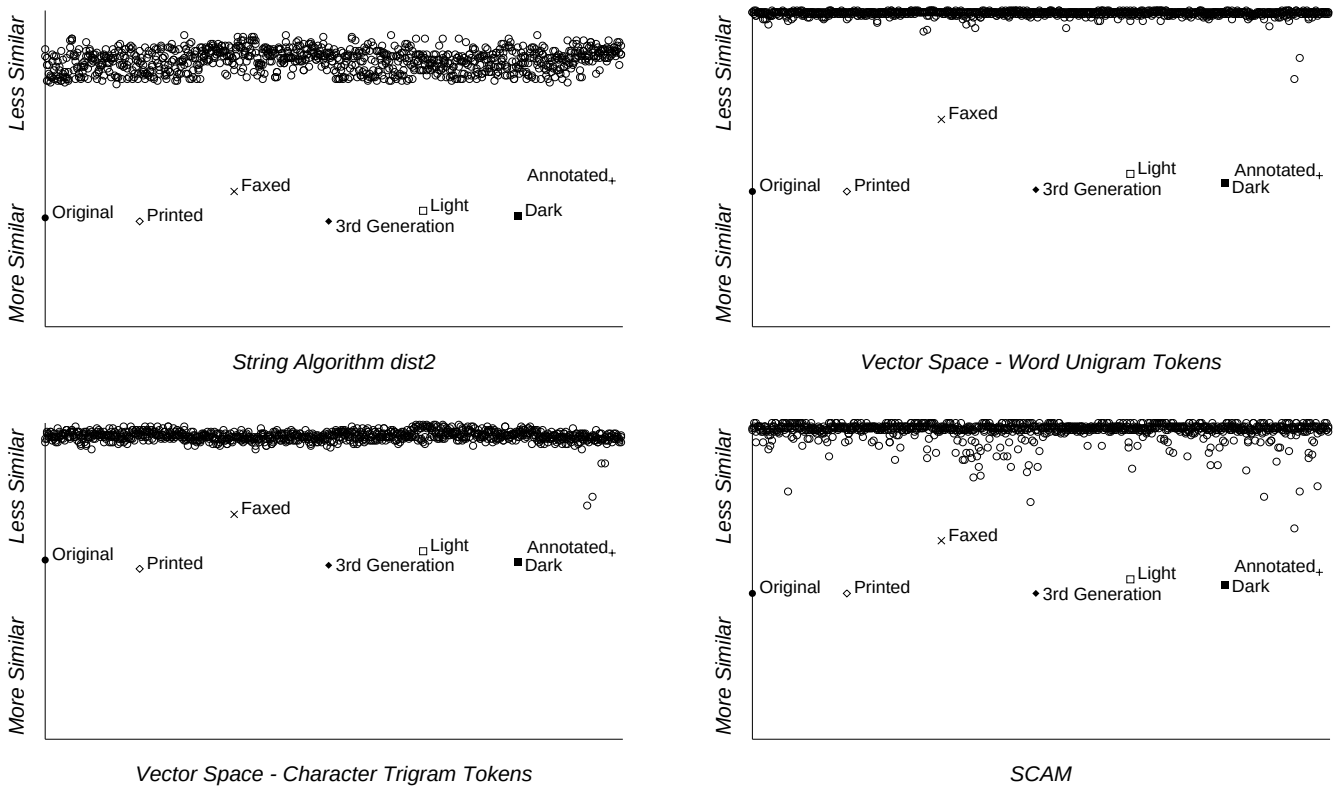

Figure 3. Results for Experiment 2 (light photocopied query).

The pages were then printed, scanned, and OCR'ed. In this case, the OCR accuracies were all fairly close, ranging from 94.9% to 96.1%, as indicated in Table 2. As before, the original source text was left in the database to serve as a second full-layout duplicate of the query. Thus, there were between two and five duplicates in the database, depending on the model.

Table 2. OCR accuracies for Experiment 4.

	OCR Accuracy	
Document Type	Database	Query
Full-Layout	96.0%	95.9%
Full-Content	96.1%	–
Partial-Layout	94.9%	–
Partial-Content	96.0%	–

The results for this experiment are shown in Figure 5 for the string techniques, and in Figure 6 for the vector space and SCAM methods. Note that various string matching algorithms are capable of distinguishing different types of duplicates, while the vector space and SCAM measures are all quite similar, producing results most like the *sdist1* algorithm. This suggests that, in certain applications, the string-based approaches may yield higher precision (*e.g.*, locating only duplicates that are photocopies of the document in question and not other types also present in the database).

4.5. Experiment 5

While the vector space and SCAM techniques are more robust with respect to variations in reading order (as demonstrated in Experiment 3), this same attribute could prove to be a disadvantage when the precise ordering of terms is crucial to identifying true duplicates.

The goal of this final experiment was to study duplicate detection using a database that also contained a number of false (*i.e.*, near) duplicates. We injected 49 issues of a daily electronic newsletter into the same database of 1,000

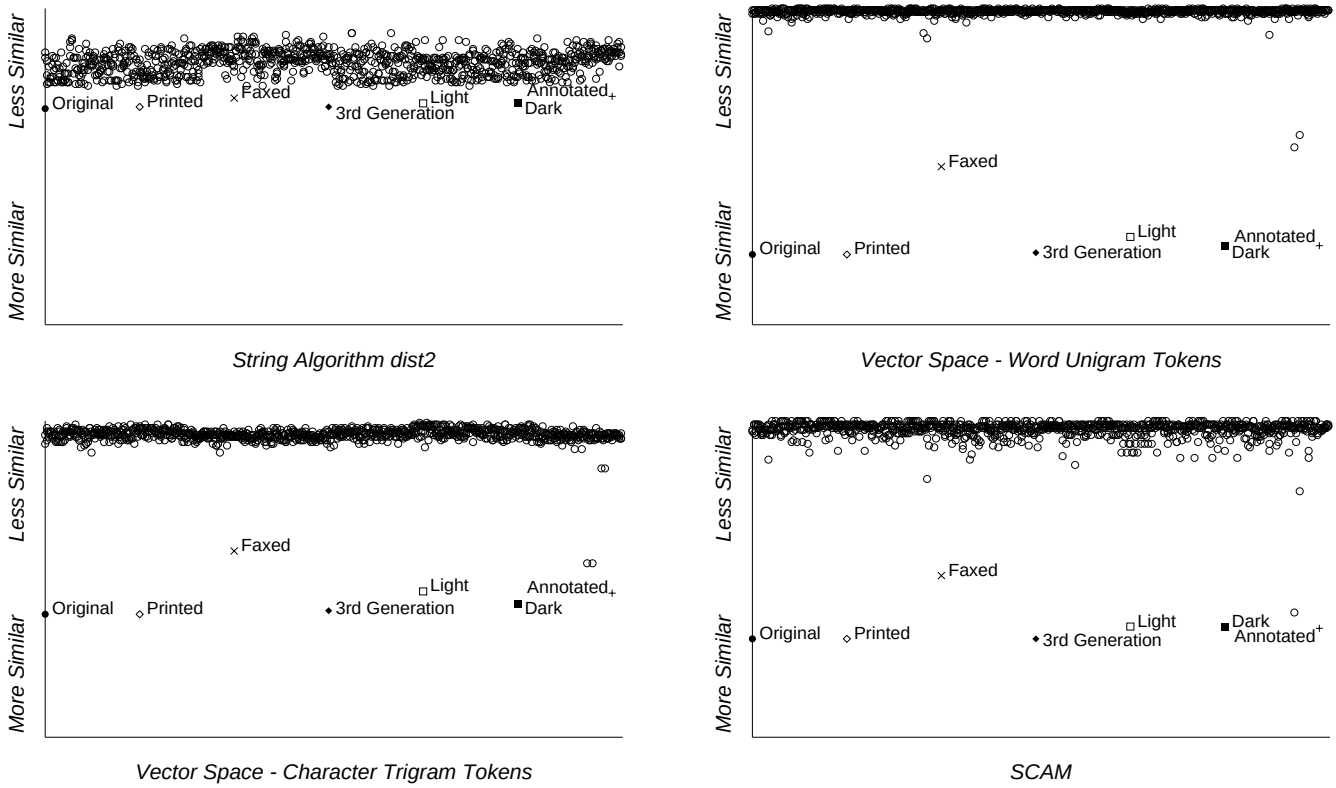

Figure 4. Results for Experiment 3 (two-column query with improper reading order).

documents used in our previous tests. While the content of the newsletter varies significantly from day to day, most issues share common section headings as well as company- and person-specific references (*e.g.*, the names of executives, products, partners, competitors). One issue was selected for use as the query, printed in 12-point Times using Microsoft Word, photocopied light, and then scanned and OCR'ed. The OCR accuracy for the query was estimated to be 74.2%. All of the documents in the database were "perfect" (*i.e.*, electronic) text, including both the intended match and the 48 false duplicates. Figure 7 presents the results for this experiment.

As can be seen in the charts, the vector space methods and SCAM produce less separation between the intended duplicate and the remainder of the documents than does the *dist2* string algorithm. Indeed, the presence of so many false duplicates significantly impairs the ability of the character trigram implementation to rank the true match appropriately. This confirms that the performance of a given technique depends not only on the condition of the query and the duplicates in the database, but on the existence of potential near (but otherwise uninteresting) matches as well.

5. CONCLUSIONS

In this paper we have presented an experimental evaluation of several text-based methods for detecting duplication in document image databases. All were tested using uncorrected OCR output for documents that had been subjected to a variety of real-world degradations. While the techniques under study are generally robust in the face of most types of OCR errors, there are nonetheless important differences, as we have shown. It appears likely that the most effective, efficient solution to this problem will be to combine several of the methods discussed as well as perhaps approaches based on lower-level image features not considered in the present paper.

Table 3 summarizes the algorithms once again. Here a solid bullet (•) indicates the broadest class for which a given method will work, while a hollow bullet (◦) indicates more restricted kinds of duplicates it will also locate. Since some of the models subsume others, an obvious question is "Why bother with the less general ones?" The answer lies in increased precision for those situations where admitting a larger class of duplicates is undesirable (*e.g.*, when the targeted duplicates are known to be photocopies). Special cases may also make it possible to develop more efficient algorithms.

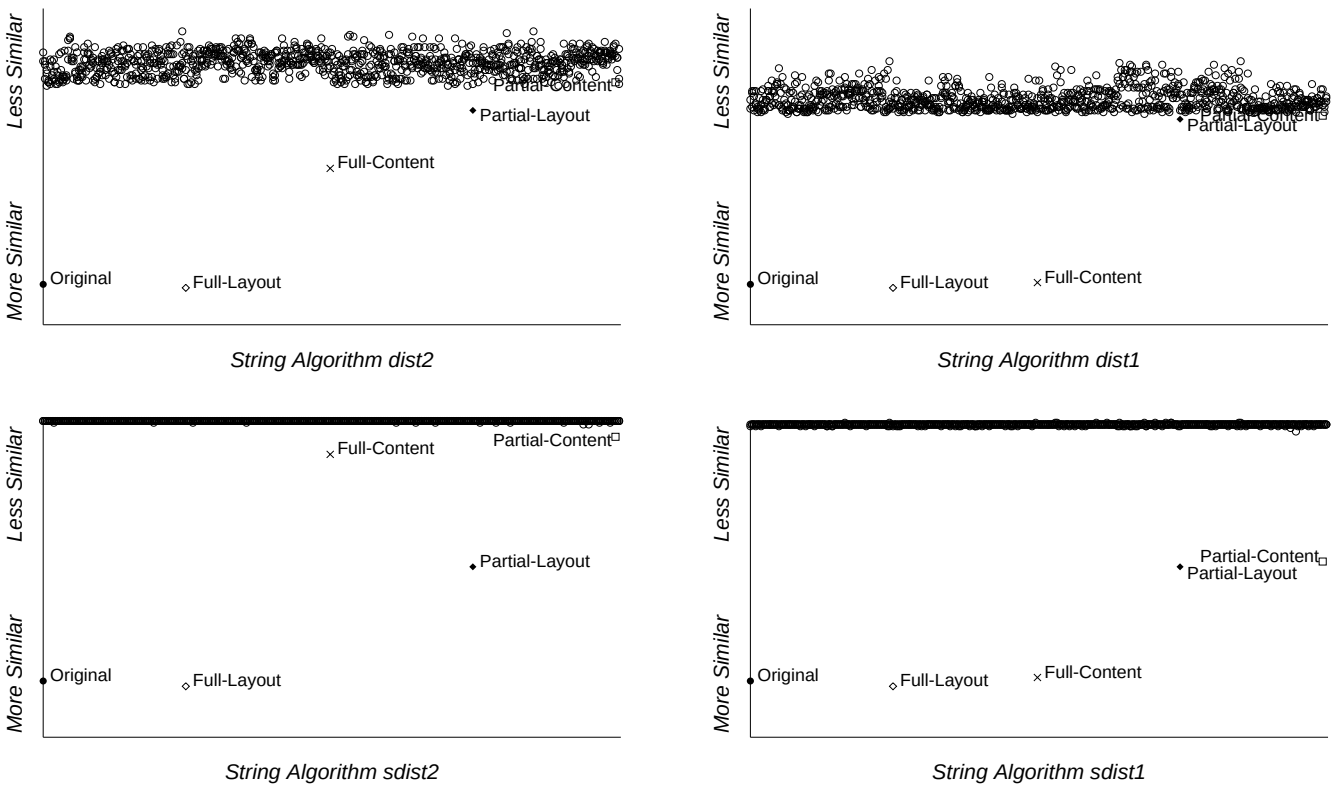

Figure 5. Approximate string matching results for Experiment 4 (the various duplicate models).

Table 3. Comparison of the methods studied in this paper.

Measure	Duplicate Type			
	Full-Layout	Full-Content	Partial-Layout	Partial-Content
dist2	●			
dist1	○	●		
sdist2	○		●	
sdist1	○	○	○	●
VS word unigram	○	○	○	●
VS char trigram	○	○	○	●
SCAM	○	○	○	●

Possible topics for future research include examining the resource requirements (time, space) for each of the implementations since this may prove more important than small differences in ranking ability. The test collection used in the current study was small, and the range of document analysis errors, while suggestive, was by no means comprehensive. Hence, the question of handling much larger, more realistic databases is open and will undoubtedly raise new issues as well as further opportunities for model and algorithm development.

REFERENCES

1. D. Doermann, H. Li, and O. Kia. The detection of duplicates in document image databases. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, pages 314–318, Ulm, Germany, August 1997.
2. G. G. Gilmore. Former Army operations officers assist DoD's search. *Army Link News*, December 1997. http://www.dtic.mil/armylink/news/Dec1997/a19971209moredata.html.
3. J. J. Hull. Document image similarity and equivalence detection. *International Journal on Document Analysis and Recognition*, 1(1):37–42, February 1998.

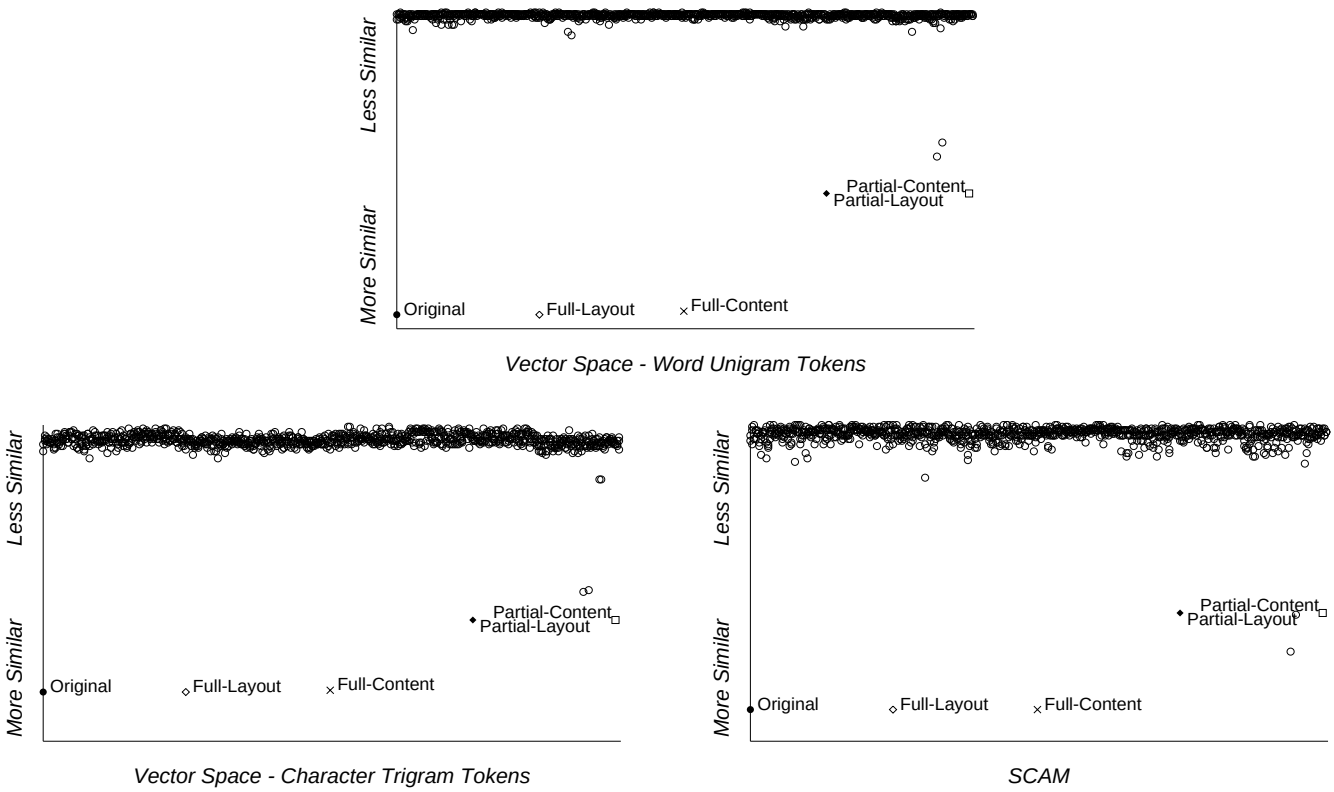

Figure 6. Vector space and SCAM results for Experiment 4 (the various duplicate models).

4. J. J. Hull, J. Cullen, and M. Peairs. Document image matching and retrieval techniques. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 31–35, Annapolis, MD, April-May 1997.
5. D.-S. Lee and J. J. Hull. Duplicate detection for symbolically compressed documents. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pages 305–308, Bangalore, India, September 1999.
6. D. Lopresti. Models and algorithms for duplicate document detection. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pages 297–300, Bangalore, India, September 1999.
7. D. Lopresti. String techniques for duplicate document detection. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 101–112, Annapolis, MD, April 1999.
8. F. Prokoski. Database partitioning and duplicate document detection based on optical correlation. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 86–97, Annapolis, MD, April 1999.
9. R. Rogers, V. Chalana, G. Marchisio, T. Nguyen, and A. Bruce. Duplicate document detection in DocBrowse. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 119–127, Annapolis, MD, April 1999.
10. G. Salton and J. Allan. Text retrieval using the vector processing model. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, pages 9–22, Las Vegas, NV, April 1994.
11. N. Shivakumar and H. Garcia-Molina. SCAM: A copy detection mechanism for digital documents. In *Proceedings of the Second International Conference on Theory and Practice of Digital Libraries*, Austin, TX, 1995. http://www.csdl.tamu.edu/DL95/papers/shivakumar.ps.
12. N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents on the Web. In *Proceedings of the Workshop on Web Databases*, March 1998.
13. A. L. Spitz. Duplicate document detection. In *Proceedings of Document Recognition IV (IS&T/SPIE Electronic Imaging)*, pages 88–94, San Jose, CA, February 1997.
14. K. Taghva, J. Borsack, A. Condit, and P. Inaparthy. Effects of OCR errors on short documents. In *Annual Report of UNLV Information Science Research Institute*, pages 99–105, Las Vegas, NV, 1995.

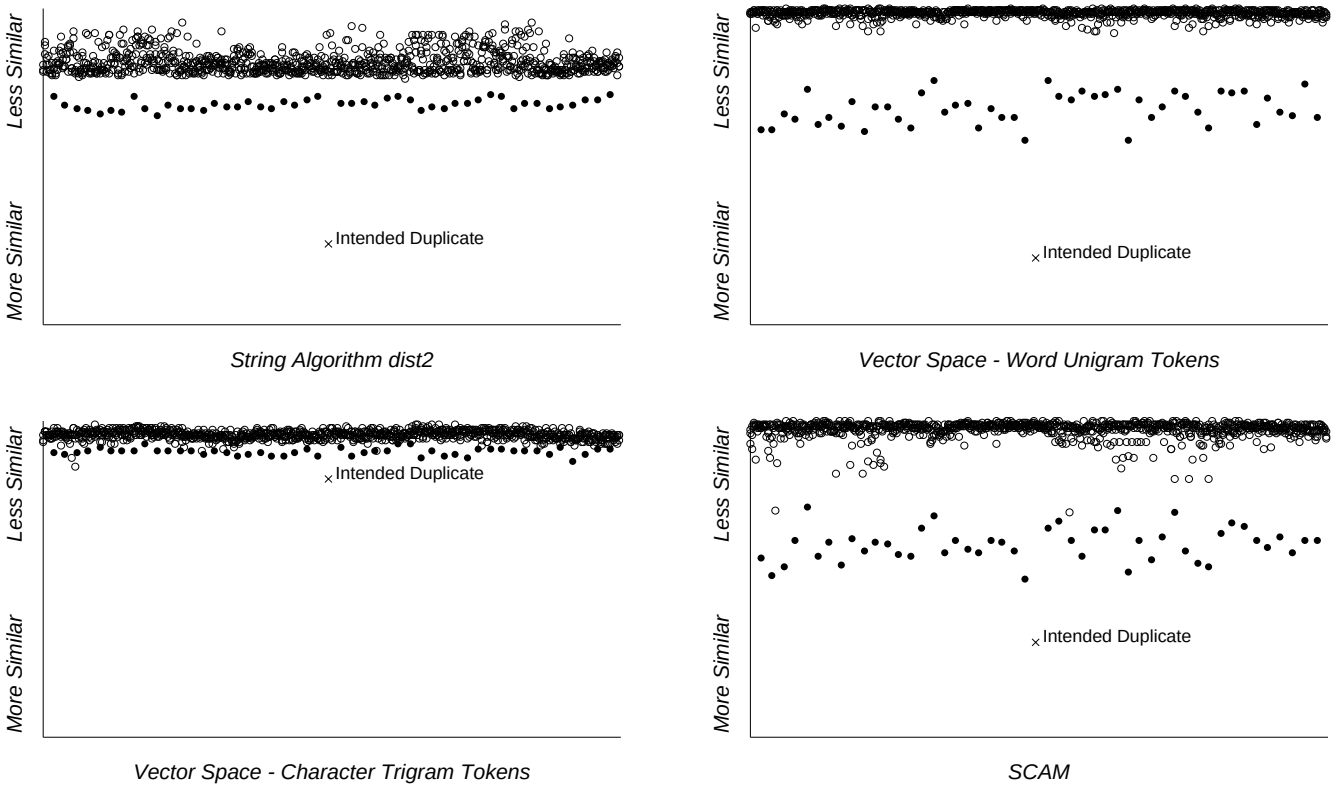

Figure 7. Results for Experiment 5 (database containing numerous false duplicates).

15. R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.