Comparing the Utility of Optical Character Recognition and Character Shape Coding in Duplicate Document Detection*

Daniel Lopresti[1] and A. Lawrence Spitz[2]

[1] Bell Labs, Lucent Technologies Inc.
600 Mountain Avenue, Room 2D-447
Murray Hill, NJ 07974
dlopresti@lucent.com
[2] Document Recognition Technologies, Inc.
616 Ramona Street, Suite 20
Palo Alto, CA 94301
spitz@docrec.com

Abstract. In this paper we study the performance of optical character recognition (OCR) and character shape coding (CSC) in support of duplicate document detection. We employ a public domain database supplemented by several common types of image degradation and apply two fundamentally different approaches for comparing documents, string edit distance and vector space (cosine) similarity. We show that CSC duplicate detection can be just as effective as OCR, at considerably lower computational cost.

1 Introduction

In the development and maintenance of document image databases, it is important to be able to detect duplicate documents in order to protect against having multiple instances of the same document in the database, or at least to flag such occurrences so the user can be informed. The presence of duplicates may not only result in inefficiencies of storage, database maintenance, and search times, but could introduce ambiguity because two seemingly identical copies of a document might be marked-up differently.

For certain applications, e.g., document declassification, duplicate detection is so crucial that a "false hit" (returning a non-duplicate) is far more tolerable than a "miss" (not finding one of the duplicates present in the database).

On large databases, those having hundreds of thousands or millions of documents, computational efficiency becomes critical to minimize the time necessary to decide whether or not a new document should be added to the database. To this end, we examine two competing technologies for transforming document

* Presented at the *Fourth International Workshop on Document Analysis Systems*, Rio de Janeiro, Brazil, December 2000.

images into representations that are easier to compare: optical character recognition (OCR) and character shape coding (CSC). While the former is the better known technology, the computational overhead for CSC is much lower, and it offers some additional advantages in terms of handling poor quality images.

Previous work on the topic has taken place at differing levels of abstraction of the document image. This can be broken into two broad classes: comparison of images or image-related features, and comparison of character-based transformations of the image. In the least abstract case, Prokowski describes a method based on optical correlation requiring special hardware [9]. Rogers et al. compare images in software by matching power spectra of projection profiles [10]. Hull et al. have described the direct comparison of document images using Hausdorf distances between sets of CCITT Group IV pass codes [3, 4]. This is a computationally intensive method, so much so that they compare only a one-square-inch region of image data for each document pair, that square inch having been located by the database provider during the segmentation process.

Hull has also based duplicate detection on hashed word-length sequence information [2]. Spitz subjected images to character shape coding before performing a string edit distance calculation [15]. Lopresti expanded on the string matching paradigm emphasizing its applicability to differing levels of abstraction such as words, characters and n-grams, and allowing for full and partial matches to documents with the same and differing text layouts [5–7].

In this paper, we unite our previously separate work involving character shape coding on the one hand, and algorithms for duplicate detection on the other, extending it in several important ways. We evaluate the effectiveness of CSC and OCR for two fundamentally different comparison measures, string edit distance and vector space (cosine) similarity, using a standard database of scanned document images supplemented with newly-created duplicates exhibiting a wide range of real-world degradations.

In Sect. 2, we describe the selection and modification of the page image data upon which our study is based. Sect. 3 details the CSC and OCR transformations applied to yield a document representation suitable for performing duplicate detection. The two comparison measures we use, one based on approximate string matching and the other the vector space metric from information retrieval, are presented in Sect. 4. Experimental results are given in Sect. 5. Finally, we present our conclusions and discuss possible topics for future research in Sect. 6.

2 Test Corpus

We used as a starting basis the University of Washington I database (UW1) [8]. We selected the 340 page-images that comprise the documents prefixed by the letters e, i, and s. The s and e sets contain 125 duplicate document images with those from the s set scanned from original journal pages, and the e set scanned from first generation photocopies. The i set adds another 90 document pages scanned from first and later generation photocopies.

We then chose 10 images from the *s* set for special treatment: *s007*, *s00c*, *s00f*, *s00g*, *s00h*, *s011*, *s013*, *s015*, *s01c*, and *s01e*. While all originating from journals, these exhibit a fairly broad range of document layout styles, content, typography, formatting conventions, etc. The images were printed on an HP LaserJet 8000 DN laserprinter, the resulting hardcopy was subjected to real-world degradation (e.g., by faxing the page, copying it, or marking sections of the text with an orange highlighter pen), and then the pages were rescanned, in most cases at 300 dpi, using a UMAX Astra 1200S scanner. The new prefixes and their corresponding interpretations are listed in Table 1.

Table 1. Duplicate documents used in this study.

Prefix	*Interpretation*
e	UW1 photocopy
s	UW1 original
f	Fax (standard mode)
g	Orange highlighter
o	Rescanned at 75 dpi
q	Rescanned at 100 dpi

Prefix	*Interpretation*
r	Rescanned at 150 dpi
u	Crumpled and flattened out
w	Very light photocopy
y	Very dark photocopy
z	Coffee-stained

While none of these various forms of damage changed the content of the page or made it unreadable from a human standpoint, they do create challenges for recognition software. Light and dark photocopies, for example, can result in numerous character-level segmentation errors (splits and merges, respectively). Fax images are often regarded as among the most difficult to handle. Note also that rescanning a page at a lower resolution presents an interesting tradeoff; recognition accuracy suffers, but processing time as well as storage requirements also drop (potentially by a factor of four when the resolution is cut in half).

Because we wish to concentrate on symbol-level effects, we sought to minimize differences in the higher-level stages of document analysis. To achieve independence with regard to layout segmentation, and to guarantee consistency in submitting exactly the same images to both CSC and OCR, we used the information in the zone attribute files supplied with the UW1 dataset. We cropped each image to include all zones labeled as "upright text." By restricting the input in this way, we eliminated "text-with-special-symbols," which usually implies mathematical content, as well as the following kinds of zones: "advertisement," "announcement," "drawing," "halftone," "logo," "map," "math," "not-clear," "ruling," "seal," and "table."

For the unmodified UW1 images (*e*, *i*, and *s*), we applied the coordinates found in the UW1 zone box files. In the case of the printed/rescanned images, where zone coordinates have been subject to translation, skew and (possibly anamorphic) scaling, the zoning was performed manually based on the UW1 zone attributes. In the end, we prepared a total of 430 page images, of which 110 ($= 10 \times 11$) corresponded to the duplicates of interest.

3 Image-to-Symbol Transformations

We transformed the character data in image zones into two symbolic representations: ASCII, using optical character recognition, and character shape codes.

3.1 Optical Character Recognition

We used Caere (now ScanSoft) OmniPage 9.0 as the OCR engine in our tests [1]. This is a commercial product which has benefited from considerable engineering effort directed at handling noisy data and optimizing performance. We configured the software to accept TIF files as input and to produce flat text, preserving line breaks, as output. Since OmniPage apparently has a minimum page size, small images were surrounded with white margins to make them acceptable. No manual error correction was performed on the output.

3.2 Character Shape Coding

The generation of character shape codes (CSC's) has been described extensively elsewhere [12–14]. Character shape coding is a robust, computationally inexpensive process based on the gross shape and location of character images with respect to their text lines. The page image is first converted into a list of connected components. Components which are too large to be text, and all components that are contained within the bounding boxes of these large components, are removed from consideration since they are presumed to represent graphical, rather than textual, information. After page skew detection and correction, as well as correction for any baseline curvature which often arises from photocopying, two important fiducial levels are defined for each text line: the baseline and the height of an x-height character. Fig. 1 shows the relationship of a text fragment to its text line.

Fig. 1. Text line parameter positions.

The mapping of character image locations to the set of shape codes is given in Table 2. Note that for codes i and j, the positions are for the lower of the two components. Since character shape coding relies on gross features, it is robust in the presence of font and size changes, scanning resolution, and most image quality degradations such as skew and multiple generation photocopying [16]. The simplicity of character shape coding makes it an inherently fast process. In some applications, shape coding runs more than an order of magnitude faster than traditional OCR processes.

Table 2. Character shape coding.

CSC	*Character Images*	*Components*		
		Number	*Top Position*	*Bottom Position*
A	A-Zbdfhklt	1	above x-line	baseline
x	acemnorsuvwxz	1	x-line	baseline
g	gpqy	1	x-line	below baseline
i	i	2	x-line	baseline
j	j	2	x-line	below baseline

However, this simplicity comes at a cost of missing some typographic complexity. For the corpus of documents processed for this paper, the most important shortfall is the inability to handle subscripts and superscripts correctly. In small numbers, these characters will just add noise to the CSC-encoded output. In large numbers, the statistical process of determining the baseline and x-line might be corrupted leading to erroneous encoding of the entire line. We did not observe this latter effect.

An additional problem arises when an all-upper-case text line is encountered. Because there is no actual x-line to be found, the CSC process will erroneously encode the text line as containing only x-height characters. However, since in the duplicate detection application consistency is more important than absolute accuracy, and since duplicates would contain the same all-upper-case text lines, this limitation is not an issue.

3.3 Relative Accuracy

While consistency is more important than accuracy in duplicate detection, the latter can serve as an indicator of potential utility. Measuring the performance of OCR processes via string edit distance [17] is an established technique. We used this same approach for quantifying the accuracy of CSC relative to the transliterated ground truth.

As can be seen in Fig. 2, OCR accuracy is somewhat higher than CSC accuracy for the best documents. However, as the ability of OCR to transform the image accurately degrades, its advantage over CSC disappears. Note that the 110 documents represented in the figure comprise the 20 scans supplied in the UW1 dataset, plus the 90 duplicates created for this study.

4 Algorithms for Duplicate Detection

In this section, we summarize the two algorithms for duplicate detection used in our experiments. The reader is referred to the original works in question for more details.

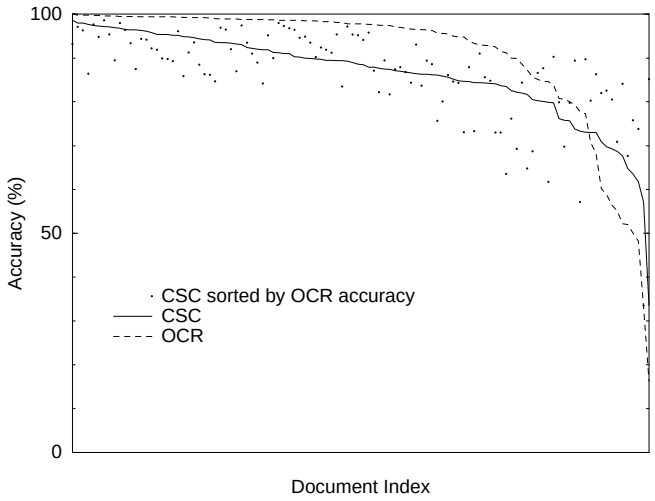

Fig. 2. CSC and OCR accuracies as a function of decreasing image quality.

4.1 Approximate String Matching

Approximate string matching is 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.

This optimization problem can be solved using a dynamic programming algorithm [17]. 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:

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

and the main dynamic programming recurrence is:

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

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

For the present work, we follow [5–7] and add another dimension to the optimization. The problem is still one of editing, but at the higher level the basic

entities are now strings (i.e., text 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. The inner-loop recurrence takes the same general form as the 1-D case:

$$dist2_{i,j} = \min \begin{cases} dist2_{i-1,j} & + C_{del}(Q^i) \\ dist2_{i,j-1} & + C_{ins}(D^j) \\ dist2_{i-1,j-1} & + C_{sub}(Q^i, D^j) \end{cases} \qquad 1 \le i \le k,\ 1 \le j \le l \qquad (3)$$

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 Eq. 1.

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

$$\begin{array}{ll} C_{del}(Q^i) & \equiv dist1(Q^i, \phi) \\ C_{ins}(D^j) & \equiv dist1(\phi, D^j) \\ C_{sub}(Q^i, D^j) & \equiv dist1(Q^i, D^j) \end{array} \qquad 1 \le i \le k,\ 1 \le j \le l \qquad (4)$$

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

For exact duplicates, the distance returned by 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. To make the results for different queries 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:

$$EditSim(Q, D) = \frac{dist - mindist}{maxdist - mindist} \qquad (5)$$

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

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:

$$mindist = \sum_{i=1}^{m} c_{sub}(q_i, q_i) \qquad (6)$$

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:

$$maxdist = \min \begin{cases} m \cdot c_{sub} + (n - m) \cdot c_{ins} \\ m \cdot c_{del} + n \cdot c_{ins} \end{cases} \tag{7}$$

4.2 The Vector Space Measure

The vector space model first proposed by Salton is extremely popular in the field of information retrieval [11]. 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:

$$w_{ik} = \frac{tf_{ik} \cdot log(N/n_k)}{\sqrt{\sum_{j=1}^{T} (tf_{ij})^2 \cdot (log(N/n_j))^2}} \tag{8}$$

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

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

$$VSsim(Q_i, D_j) = \sum_{k=1}^{T} w_{ik} w_{jk} \tag{9}$$

5 Experimental Results

For the experiments reported in this paper, we employed the entire database of 430 pages processed through either CSC or OCR. The appropriate output for the 10 *s*-prefixed documents listed in Sect. 2 were used as queries. Since the database contained one duplicate for each category depicted in Table 1, there were a total of 11 intended "hits" for each query (including one exact match).

For our edit distance measure, we ran algorithm *dist2* (i.e., Eq. 3) with the cost assignments $c_{del} = c_{ins} = 1$, $c_{sub} = 2$, and $c_{mat} = -1$. The distance was normalized to between 0 and 1 as described in the previous section. For the vector space computation, we employed word-unigram tokens. In the case of the OCR output, we mapped all characters to their lower-case equivalents and followed the standard IR custom of filtering out stopwords. This particular step was not applied to the CSC output. We made no attempt at stemming, another common practice in IR.

Table 3 presents the string matching results for comparing each of the 10 queries against the database of 430 documents under the CSC and OCR transformations. Note that there is always significant separation between the average distances for the "duplicate" and "non-duplicate" classes. The values shown for precision at 100% recall indicate the number of non-duplicates one must examine to be sure of finding all of the duplicates; 1.0 is the best possible score (i.e., there are no false hits mixed in with the true duplicates). In almost every instance, the string technique is perfect for both the CSC and OCR representations. The one aberration is query *s013*. Interestingly, CSC did badly for *y013* (the dark photocopy) because of excessive black "speckle" noise, whereas OCR produced poor quality output for a different document, *o013* (the 75 dpi scan). We note that, in general, OCR was less robust at lower scanning resolutions than CSC. Excluding these worst-case duplicates for *s013* yields precision measures of 1.0 for both CSC and OCR.

Table 3. Performance of string matching for duplicate detection (CSC left, OCR right).

	CSC			*Prec.*	*OCR*			*Prec.*
	Average Distance			*@100%*	*Average Distance*			*@100%*
Query	*Dupes*	*Non-Dupes*	*Sep.*	*Recall*	*Dupes*	*Non-Dupes*	*Sep.*	*Recall*
s007	0.0964	0.7610	0.6646	1.00	0.0772	0.8747	0.7975	1.00
s00c	0.1036	0.7061	0.6025	1.00	0.0860	0.8400	0.7540	1.00
s00f	0.1132	0.8040	0.6909	1.00	0.1316	0.8987	0.7672	1.00
s00g	0.2137	0.7023	0.4886	1.00	0.2154	0.8356	0.6202	1.00
s00h	0.2539	0.6496	0.3958	1.00	0.2147	0.8270	0.6123	1.00
s011	0.1702	0.7497	0.5795	1.00	0.1134	0.8761	0.7627	1.00
s013	0.1442	0.6478	0.5036	0.07	0.1216	0.8249	0.7033	0.04
s015	0.1174	0.6655	0.5481	1.00	0.0438	0.8281	0.7842	1.00
s01c	0.1590	0.6669	0.5079	1.00	0.1260	0.8346	0.7085	1.00
s01e	0.1374	0.7370	0.5996	1.00	0.0924	0.8680	0.7756	1.00
Avg.	0.1509	0.7090	0.5581	0.91	0.1222	0.8508	0.7286	0.90

Analogous figures for the vector space measure are given in Table 4. The obvious conclusion here is that the string algorithm is more robust than vector space, at least with respect to the dataset under study. It is important to keep in mind that a precision of 0.35 or 0.49, while far from perfect, is not as catastrophic as a recognition accuracy of the same amount. It simply means that the user must wade through roughly equal numbers of true and false hits.

Nevertheless, these results differ considerably from those reported in an earlier study where the performance of the string and vector space techniques was found to be comparable [6]. In looking at the reasons for this, we note that several of the query documents in the present dataset are quite short (e.g., the OCR'ed version of *s00f* is 502 bytes) – far shorter than the query used in [6] which was 1,395 bytes. Also, for each of our *s* queries, there are typically some number of other pages from the same journal article (using the same distinctive terminol-

ogy) present in the database. These pages may be returned by vector space as matches, even though they are not really duplicates of the query. Lastly, it is conceivable that word-size tokens are wrong for use in this application. We examined character-trigrams, but found their performance even worse, especially for the CSC representation. Still, some other size may produce better results than what is seen here.

Table 4. Performance of vector space for duplicate detection (CSC left, OCR right).

	CSC			*Prec.*	*OCR*			*Prec.*
	Average Distance			*@100%*	*Average Distance*			*@100%*
Query	*Dupes*	*Non-Dupes*	*Sep.*	*Recall*	*Dupes*	*Non-Dupes*	*Sep.*	*Recall*
s007	0.4292	0.9786	0.5495	0.73	0.2352	0.9840	0.7488	0.55
s00c	0.4509	0.9608	0.5099	0.41	0.2103	0.9654	0.7550	0.31
s00f	0.5204	0.9781	0.4577	0.11	0.3486	0.9756	0.6271	0.24
s00g	0.4765	0.9644	0.4879	0.38	0.2743	0.9673	0.6930	0.17
s00h	0.5943	0.9677	0.3733	0.08	0.2997	0.9838	0.6841	0.48
s011	0.5936	0.9560	0.3624	0.09	0.3127	0.9815	0.6688	0.16
s013	0.4696	0.9760	0.5064	0.48	0.1896	0.9914	0.8018	0.85
s015	0.5953	0.9796	0.3843	0.73	0.2103	0.9928	0.7826	1.00
s01c	0.5538	0.9570	0.4032	0.42	0.2738	0.9854	0.7116	0.58
s01e	0.6262	0.9875	0.3612	0.03	0.3159	0.9936	0.6777	0.58
Avg.	0.5310	0.9706	0.4396	0.35	0.2670	0.9821	0.7150	0.49

Detailed results for one specific query, *s00c*, are presented in Figs. 3 and 4. Here all 430 distance values are graphed for each representation/comparison combination, with the intended duplicates marked by their appropriate prefix letter. Both OCR and character shape codes work well for determining the presence or absence of duplicates under the string matching measure. Though the "cloud" of non-duplicates is more diffuse in the case of CSC's, these is almost always a significant gap between the cloud and the closest actual duplicate. The presence of this gap leads to high values of precision at 100% recall, minimizing the number of non-duplicates that need to be examined. The failure of vector space to distinguish cleanly the two classes is also apparent in the figures.

Table 5 provides another view of the results for the string matching algorithm, averaging the distances for all duplicates of a given type. Clearly, the most difficult degradations for CSC are dark photocopies and faxes. This is likely a function of speckle noise – a more aggressive approach to noise reduction might ameliorate the situation. On the other hand, OCR has increasing trouble as the scanning resolution drops. Evidently OmniPage is optimized for a specific input resolution, and performs poorly for anything outside that range. This may be an area where CSC has an inherent advantage, since the fine typographic detail necessary for full-scale OCR is largely irrelevant when computing shape codes.

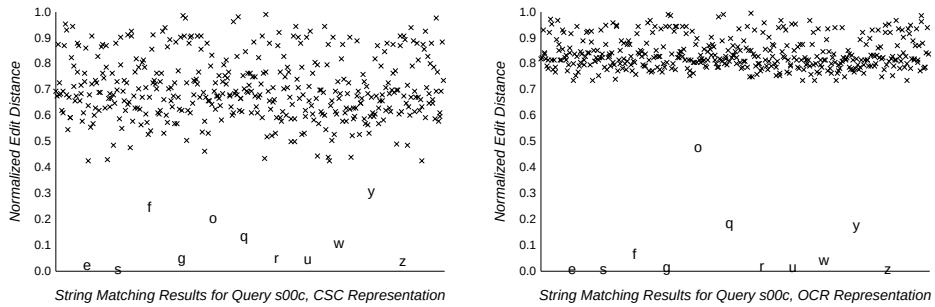

Fig. 3. String matching results for query *s00c* (CSC left, OCR right).

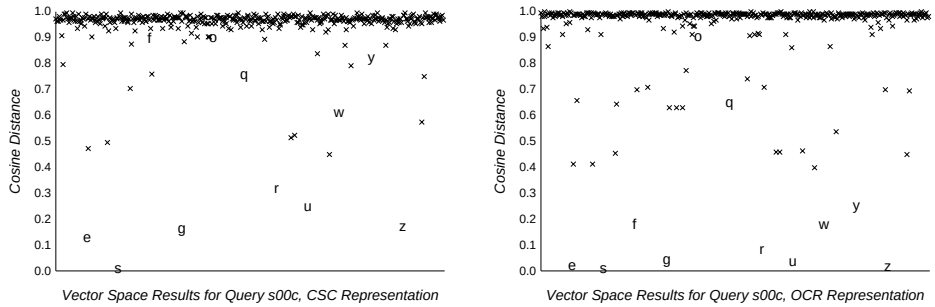

Fig. 4. Vector space results for query *s00c* (CSC left, OCR right).

6 Conclusions and Future Research

In this paper, we have compared the performance of two image-to-symbol transformations, optical character recognition and character shape coding, in support of duplicate document detection. We found that CSC duplicate detection can be just as effective as OCR, at considerably lower computational cost. We also discovered that, for the dataset under study, approximate string matching is more robust than vector space retrieval.

To model more accurately the document declassification application where duplicate detection and removal is essential, we intend to apply, and possibly

Table 5. Performance by degradation type (string matching measure).

Prefix	*Interpretation*	*Average Distance*	
		CSC	*OCR*
e	UW1 copy	0.039789	0.005448
s	UW1 original	0.000000	0.000000
f	Fax	0.287316	0.151144
g	Highlighter	0.098236	0.054466
o	75 dpi	0.244531	0.511369
q	100 dpi	0.183039	0.235255

Prefix	*Interpretation*	*Average Distance*	
		CSC	*OCR*
r	150 dpi	0.101373	0.060560
u	Crumpled	0.103709	0.052978
w	Very light copy	0.126319	0.080887
y	Very dark copy	0.332226	0.142019
z	Coffee-stained	0.084438	0.046130

adapt, the techniques described in this paper to a representative corpus of typewritten documents.

We also plan to try using the vector space measure as a pre-filter to eliminate, at low cost, those documents that cannot possibly be duplicates by virtue of their dissimilar vocabularies. It might also be interesting to pursue the additional computational advantage to had in string comparison when the alphabet is relatively small ($\sim$ 10 symbols in the case of CSC vs. $\sim$ 100 symbols for OCR).

References

1. Caere OmniPage Pro. http://www.caere.com/products/omnipage/pro/.
2. J. J. Hull. Document image matching and retrieval with multiple distortion-invariant descriptors. In A. L. Spitz and A. Dengel, editors, *Document Analysis Systems*, pages 379–396. World Scientific, Singapore, 1995.
3. J. J. Hull. Document image similarity and equivalence detection. *International Journal on Document Analysis and Recognition*, 1(1):37–42, February 1998.
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. 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.
6. D. Lopresti. A comparison of text-based methods for detecting duplication in document image databases. In *Proceedings of Document Recognition and Retrieval VII (IS&T/SPIE Electronic Imaging)*, volume 3967, pages 210–221, San Jose, CA, January 2000.
7. D. Lopresti. String techniques for detecting duplicates in document databases. *International Journal on Document Analysis and Recognition*, 2(4):186–199, June 2000.
8. I. T. Phillips and J. Ha. The implementation methodology for a CD-ROM English document database. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 484–487, Tsukuba Science City, Japan, October 1993.
9. 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.
10. 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.
11. 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.
12. A. L. Spitz. Generalized line, word and character finding. In S. Impedovo, editor, *Progress in Image Analysis and Processing III*, pages 377–383. World Scientific, Singapore, 1994.
13. A. L. Spitz. Text line characterization by connected component transformations. In *Proceedings of Document Recognition I (IS&T/SPIE Electronic Imaging)*, volume 2181, pages 97–105, San Jose, CA, February 1994.

14. A. L. Spitz. Using character shape codes for word spotting in document images. In D. Dori and A. Bruckstein, editors, *Shape, Structure and Pattern Recognition*, pages 382–389. World Scientific, Singapore, 1995.
15. 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.
16. A. L. Spitz and J. P. Marks. Measuring the robustness of character shape coding. In *Proceedings of the IAPR Workshop on Document Analysis Systems*, pages 19–28, Nagano, Japan, 1998.
17. R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.