Models and Algorithms for Duplicate Document Detection*

Daniel P. Lopresti
Bell Labs, Lucent Technologies Inc.
600 Mountain Avenue, Room 2C-552
Murray Hill, NJ 07974 USA
dpl@research.bell-labs.com

Abstract

This paper introduces a framework for clarifying and formalizing the duplicate document detection problem. Four distinct models are presented, each with a corresponding algorithm for its solution derived from the realm of approximate string matching. The robustness of these techniques is demonstrated through a set of experiments using data reflecting real-world degradation effects.

1. Introduction

As information management and networking technologies continue to proliferate, document image databases are growing rapidly in size and importance. A key problem facing such systems is determining whether duplicates already exist in the database when a new document arrives. This is challenging both because of the various ways a document can become degraded and because of the many possible interpretations of what it means to be a "duplicate."

For example, one document might be a photocopy of another, or a fax. The copies could be visually identical or one might have additional handwritten notes appended to it. If the original document was generated on-line, a duplicate could contain exactly the same text, only formatted in a different way (changes in font, line spacings and lengths, etc.). A duplicate might possess substantially the same content, but with minor alterations due to editing (*i.e.,* earlier or later versions of the same document). Of course, in any of these cases, the image of either or both of the documents may contain significant "noise" due to the way the printed page was handled or anomalies in the scanning process.

Whatever the definition, the process of determining whether one document is a duplicate of another involves two steps: (1) extracting appropriate information (features) from the incoming document image, and (2) comparing the features against those previously extracted from documents in the database. Previous work on detecting duplicates (*e.g.*, [1, 2, 4]) has concentrated mostly on exploring the first of these, turning to more traditional measures when it comes to the second. Here the emphasis is on the models and algorithms associated with comparing document representations (*i.e.,* the second step), with features taken to be the uncorrected text output from a commercial OCR package.

The remainder of this paper is organized as follows. Section 2 presents four distinct but related models for the duplicate detection problem. Each of these is solved optimally using a dynamic programming algorithm, as discussed in Section 3. Section 4 describes experimental results that demonstrate the robustness of these techniques. Finally, conclusions are given in Section 5.

2. Models

The focus in this work is on document pages that, while in image form, are primarily textual in content. Viewed abstractly, such a page is a series of lines, each consisting of a sequence of symbols. In this *string-of-strings* viewpoint, the term "symbol" can be defined quite liberally. It could be interpreted as meaning characters, but higher- and lower-level representations are also possible.

The problem can be partitioned along two dimensions: whether the duplication is full or partial, and whether the layout of text lines is maintained or not:

1. If two documents are visually identical, one is a photocopy or a fax of the other, say, they are *full-layout* duplicates.
2. If two documents have identical textual content, but not necessarily the same layout (*i.e.*, line breaks), they are *full-content* duplicates.
3. If two documents share significant content with the same layout, they are *partial-layout* duplicates.

*Presented at the *Fifth International Conference on Document Analysis and Recognition*, Bangalore, India, September 1999.

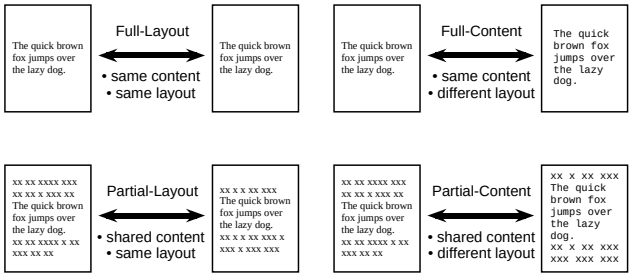

Figure 1. The four duplicate classes.

4. If two documents share content but their layout is not necessarily the same, they are *partial-content* duplicates.

These various types of duplication are shown in Figure 1. Note that although the text used to illustrate the figure is "clean," in general it will be necessary to handle the full range of potential document recognition errors.

3. Algorithms

The literature on approximate string matching is rich with techniques for dealing with the sorts of errors that arise when OCR'ing "difficult" documents. Beginning with some definitions, a *string*, $D = d_1 d_2 \ldots d_n$, is a finite sequence of symbols chosen from a finite alphabet, $d_i \in \Sigma$. String $S = s_1 s_2 \ldots s_m$ is a *substring* of string $D = d_1 d_2 \ldots d_n$ if $m \leq n$ and there exists an integer k in the range $[0, n-m]$ such that $s_i = d_{i+k}$ for $i = 1, 2, \ldots, m$. In the 1-D case (*i.e.*, content duplicates), a *document* is simply a string. In the 2-D case (*i.e.*, layout duplicates), a document is a sequence of strings, $D = D^1 D^2 \ldots D^m$ where $D^i = d_1^i d_2^i \ldots d_n^i$.

A standard measure for approximate string matching is provided by *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.

3.1. The Full-Content Duplicate Problem

As it relates to full-content duplicates, this optimization problem can be solved using a well-known dynamic programming algorithm [5]. 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 characters of Q and the first j characters of D. The initial conditions are:

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

and the main dynamic programming recurrence is:

$$dist1_{i,j} = \min \begin{cases} dist1_{i-1,j} + c_{del}(q_i) \\ dist1_{i,j-1} + c_{ins}(d_j) \\ dist1_{i-1,j-1} + c_{sub}(q_i, d_j) \end{cases} \quad (2)$$

for $1 \leq i \leq m$, $1 \leq j \leq n$. The computation builds a matrix of distance values working from the upper left corner ($dist1_{0,0}$) to the lower right ($dist1_{m,n}$).

3.2. The Partial-Content Duplicate Problem

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:

$$sdist1_{0,0} = sdist1_{i,0} = sdist1_{0,j} = 0 \quad (3)$$

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:

$$sdist1_{i,j} = \min \begin{cases} 0 \\ sdist1_{i-1,j} + c_{del}(q_i) \\ sdist1_{i,j-1} + c_{ins}(d_j) \\ sdist1_{i-1,j-1} + c_{sub}(q_i, d_j) \end{cases} \quad (4)$$

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.

3.3. The Full-Layout Duplicate Problem

For the 2-D models (*i.e.*, layout duplicates), another level is added to the optimization. The problem is still

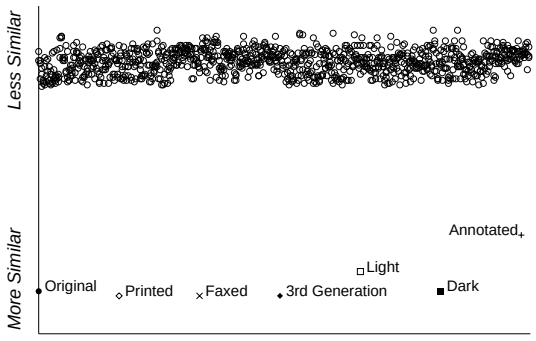

Figure 2. Full-layout duplicate detection.

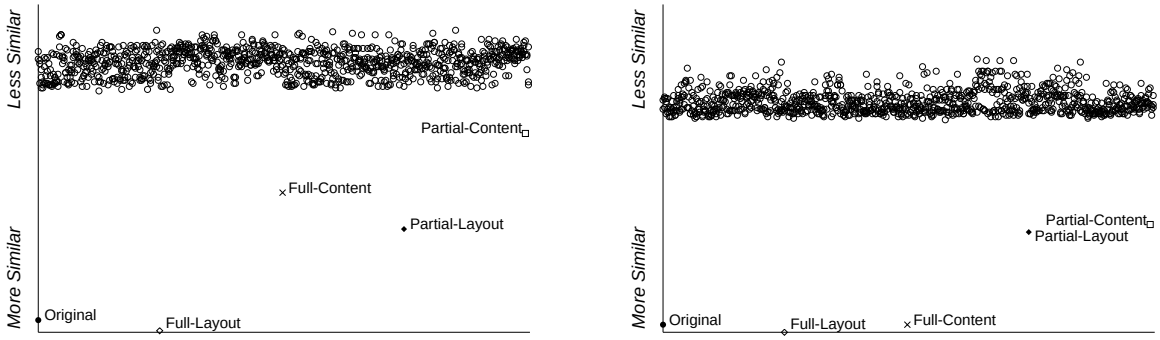

Figure 3. Duplicate detection using algorithms *dist2* (left) and *dist1* (right).

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:

$$\textit{dist2}_{i,j} = \min \begin{cases} \textit{dist2}_{i-1,j} & + \; C_{del}(Q^i) \\ \textit{dist2}_{i,j-1} & + \; C_{ins}(D^j) \\ \textit{dist2}_{i-1,j-1} & + \; C_{sub}(Q^i, D^j) \end{cases} \quad (5)$$

for $1 \leq i \leq k$, $1 \leq j \leq l$, 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:

$$\begin{aligned} C_{del}(Q^i) &\equiv \textit{dist1}(Q^i, \phi) \\ C_{ins}(D^j) &\equiv \textit{dist1}(\phi, D^j) \\ C_{sub}(Q^i, D^j) &\equiv \textit{dist1}(Q^i, D^j) \end{aligned} \quad (6)$$

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

3.4. The Partial-Layout Duplicate Problem

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

$$\textit{sdist2}_{i,j} = \min \begin{cases} 0 & \\ \textit{sdist2}_{i-1,j} & + \; C_{del}(Q^i) \\ \textit{sdist2}_{i,j-1} & + \; C_{ins}(D^j) \\ \textit{sdist2}_{i-1,j-1} & + \; C_{sub}(Q^i, D^j) \end{cases} \quad (7)$$

for $1 \leq i \leq k$, $1 \leq j \leq l$. Note that C_{del}, C_{ins}, and C_{sub} are defined as before in terms of *dist1* (*i.e.*, Equation 6).

4. Experimental Results

To investigate the performance of the algorithms described in this paper, two sets of experiments were designed. The first examined duplicate detection in the presence of various degradation effects, while the second studied the four duplicate models and algorithms and how they relate.

The test database consisted of 1,000 professionally written news articles collected 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 all "authentic" (pages that had been printed, scanned, and OCR'ed).

4.1. Experiment 1

The goal of this experiment was to study duplicate detection under realistic 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 intended 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, excessively light or dark or 3rd generation photocopying, or handwritten markup (annotations) that completely 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).

The OCR accuracies were found to range from 73.5% to 96.2%. As expected, a large variety of OCR errors were encountered, as well as other problems (*e.g.*, fax headers were transcribed and crossed-out lines were completely missed).

Since the query documents and their intended matches have the same layout, this is a full-layout duplicate detection problem and the *dist2* measure is appropriate. The chart in Figures 2 plots, for the faxed query, the normalized edit distance for every document in the database. Note that there is always a clear distinction between true duplicates and everything else. This demonstrates that the technique is robust when faced with the sorts of OCR errors seen in practice.

The charts for the remainder of the queries are similar

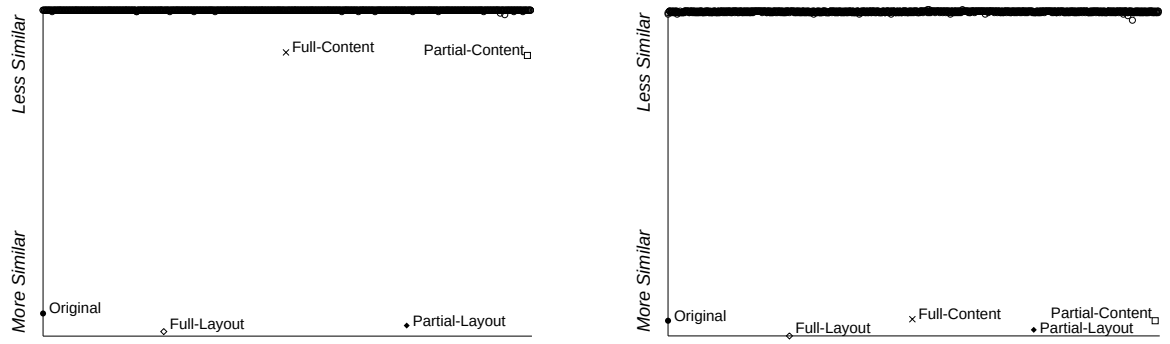

Figure 4. Duplicate detection using algorithms *sdist2* (left) and *sdist1* (right).

and can be found in another paper [3].

4.2. Experiment 2

The purpose of this experiment was to determine how the different duplicate models relate empirically. The four algorithms described in Section 3 were run using the same source document as in the previous experiment. Duplicates were constructed from the query by changing the line breaks and/or appending roughly equal amounts of unrelated text to the beginning and end of the document.

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 before, the original source text was left in the database to serve as a second full-layout duplicate of the query. Hence, there were between two and five duplicates in the database, depending on the model.

The results for this experiment are shown in Figures 3 and 4. Since there is a fair amount of residual similarity even in the non-matching cases, the normalized edit distances are lower than for purely random documents. Note that, as expected, algorithm *dist2* works best for full-layout duplicates, and *dist1* adds to this full-content duplicates (Figure 3). The partial-layout algorithm *sdist2* can detect full- and partial-layout duplicates, while *sdist1* covers all four duplicate classes (Figure 4).

5. Conclusions

This paper has examined a number of issues related to the detection of duplicates in document image databases using uncorrected OCR output. Four distinct models for formalizing the problem were presented, along with algorithms for determining the optimal solution in each case. Table 1 enumerates these classes one last time. A solid dot (•) highlights the algorithm most suited to a particular problem, while a hollow dot (○) indicates that the algorithm will find not only such duplicates but other types as well.

Since some of the problems seem to 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.

	dist2	*dist1*	*sdist2*	*sdist1*
Full-layout	•	○	○	○
Full-content		•		○
Partial-layout			•	○
Partial-content				•

Table 1. The algorithms and where they apply.

References

[1] D. Doermann, H. Li, and O. Kia. The detection of duplicates in document image databases. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 314–318, Ulm, Germany, Aug. 1997.

[2] 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, Apr. 1997.

[3] D. Lopresti. String techniques for duplicate document detection. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 101–112, Annapolis, MD, Apr. 1999.

[4] A. L. Spitz. Duplicate document detection. In *Proceedings of Document Recognition IV (IS&T/SPIE Electronic Imaging)*, pages 88–94, San Jose, CA, Feb. 1997.

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