Issues in Automatic OCR Error Classification

Jeffrey Esakov, Daniel P. Lopresti,
Jonathan S. Sandberg and Jiangying Zhou

Matsushita Information Technology Laboratory
Panasonic Technologies, Inc.
Two Research Way
Princeton, New Jersey 08540

Abstract

In this paper we present the surprising result that OCR errors are not always uniformly distributed across a page. Under certain circumstances, 30% or more of the errors incurred can be attributed to a single, avoidable phenomenon in the scanning process. This observation has important ramifications for work that explicitly or implicitly assumes a uniform error distribution. In addition, our experiments show that not just the quantity but also the nature of the errors is affected. This could have an impact on strategies used for post-process error correction.

Results such as these can be obtained only by analyzing large quantities of data in a controlled way. To this end, we also describe our algorithm for classifying OCR errors. This is based on a well-known dynamic programming approach for determining string edit distance which we have extended to handle the types of character segmentation errors inherent to OCR.

1 Introduction

Fully automated OCR promises a tremendous cost savings over the current labor-intensive process, yet 100% accuracy still remains an elusive goal. Aged books, noisy multi-generation photocopies and faxes, non-standard text layouts, damaged originals, and handwritten mark-up are topics for advanced research. Even in the case of a clean source document, problems such as accounting for white space and disambiguating between nearly-identical characters seem fundamentally hard. A large amount of research has been directed towards improving OCR accuracy.

To further improve performance, it is necessary to qualify as well as to quantify the precise nature of OCR error sources. For example, error detection/correct strategies such as dictionary hashing [4] and the Certifiable OCR idea proposed in [6] require that the post-processor have detailed knowledge about the types and the characteristics of errors likely to be made by the OCR process. Such information can be obtained only by studying a large amount of data in a systematic way. It is thus important to develop means for identifying and classifying OCR errors automatically.

In a previous paper [3] we described the nature of OCR errors generated by the OCR package for particular fonts as well as the relationship between error sets for different fonts. In this paper, we examine the

effect of scanning *process* on OCR accuracy. We find that there exists a significant difference in OCR performance under controlled conditions when taking inputs from two different types of scanners. We show that certain OCR errors are not characteristic of a particular font or OCR process, but are instead induced by physical failures in the image acquisition process. Our results suggest that to evaluate OCR systems faithfully, one must pay close consideration the scanning procedure. Such effects seem to have been overlooked in previous research.

Our algorithm for classifying OCR errors is based on a new variation of the dynamic programming algorithm described in [3]. We extend the algorithm to identify not only isolated errors in strings (i.e. single symbol substitutions, deletions and insertions), but also more complex errors involving multiple characters up to an arbitrary but pre-determined size.

Our approach to classifying OCR errors is presented in Section 2. In Section 3, we describe the procedure we followed in performing a large-scale experiment. Section 4 discusses the results of this study. Finally, we offer our conclusions in Section 5.

2 Automatic Classification of OCR Errors

Various forms of approximate string matching have been used by other researchers for the purpose of classifying OCR errors [1, 2, 7, 8, 9]. Our approach is similar in that we formulate the problem as an edit distance computation, but fundamentally different in the way we have extended the algorithm to handle character segmentation errors.

The relationship between two similar, but not necessarily identical, strings can be made mathematically precise using an edit model. In the traditional case [10], the following three operations are permitted:

a. delete a single character,

b. insert a single character,

c. substitute one character for another.

Each of these operations is assigned a cost, c_{del}, c_{ins}, and c_{sub}, and the minimum cost of any sequence of basic operations that transforms one string into the other is called the *edit distance*. This optimization problem can be solved using a well-known dynamic programming algorithm. Let $S = s_1 \ldots s_m$ be the source (original) string, and $T = t_1 \ldots t_n$ be the target (OCR) string. Define $d_{i,j}$ to be the distance between the first i characters of S and the first j characters of T. The main dynamic programming recurrence is then:

$$\mathrm{d_{i,j}} = \min \begin{cases} d_{i-1,j} + c_{del}(s_i) \\ d_{i,j-1} + c_{ins}(t_j) \\ d_{i-1,j-1} + c_{sub}(s_i, t_j) \end{cases} \quad (1)$$

for $1 \leq i \leq m$, $1 \leq j \leq n$. When Equation 1 is used as the inner loop step in an implementation, the time required is $O(mn)$ where m and n are the lengths of the two strings.

If, in addition, the choices that lead to the minimums above (i.e., the optimal decisions) are also recorded, the resulting trace-back table provides a sequence of operations that perform the transformation in question (a trace-back table is shown in Figure 1). For the OCR problem, these edits can be equated with the errors in the OCR string. For example, if all editing operations are assigned a cost of 1, the edit distance between the strings "Call me Ishmael." and "Callmc Ishma,el." is 3, and the sequence of optimal edits is "delete space," "substitute c for e," and "insert comma."

There may, in fact, be more than one optimal edit path (and hence error classification) for a given pair of strings. The computation corresponding to

"Call me Ishmael." ⇒ "Cal me Ishmael."

has two optimal paths depending on which "l" is selected for deletion. From the standpoint of having minimal cost, any such path is as plausible an explanation for what happened during the OCR process as another. Note that it is also quite conceivable that human experts will disagree on a particular classification. Some degree of ambiguity is perfectly natural and not necessarily a matter for concern.

While the editing operations (a)-(c) listed above are sufficient for describing the relationship between any two strings, there is an important phenomenon inherent to OCR that they do not capture: the notion of a segmentation error. Consider the case of

"Call me Ishmael." ⇒ "Call rne Ishmael"

The error pattern "m" ⇒ "rn" is treated as an insertion and a substitution under Equation 1, as well as by many of the other models proposed in the literature. For our purposes, however, it would be more accurate to treat this as a single error event, which we call a *multi-substitution* (or *multi-sub* for short).

We have developed a new formalism to handle multi-subs. The three traditional operations are replaced by a single generalized operation, the *p:q* substitution:

1. substitute q characters for p characters.

(a)-(c) then become 1:0, 0:1, and 1:1 substitutions, respectively. The error that motivated this discussion, "m" ⇒ "rn", is a 1:2 substitution. The cost of substituting q characters $t_j \ldots t_{j+q-1}$ for p characters $s_i \ldots s_{i+p-1}$ is:

$$c_{sub_{p:q}}(s_i \ldots s_{i+p-1}, t_j \ldots t_{j+q-1})$$

and is a function of both p and q as well as of the characters in question.

In an earlier paper [3], we described an extension of Equation 1 to handle the case $1 \leq p, q \leq 2$. This is still too limiting as multi-subs involving three characters (e.g., "m" ⇒ "ril") and four characters (e.g., "rw" ⇒ "n;cl") commonly occur in practice. Equation 2 presents a more general form of the dynamic programming recurrence capable of accounting for arbitrary multi-subs.

$$\begin{aligned} d_{i,j} = \; & \min [d_{i-g,j-h} + \\ & c_{sub_{g:h}}(s_i \ldots s_{i+g-1}, t_j \ldots t_{j+h-1}) \\ & \mid 0 \leq g \leq p, 0 \leq h \leq q] \end{aligned} \quad (2)$$

$$\text{for } 1 \leq i \leq m, 1 \leq j \leq n$$

This expanded recurrence requires time $O(mnpq)$ where m and n are the lengths of the two strings and p and q determine the maximum allowable size of a multi-sub.

As before, saving the optimal decisions as $d_{i,j}$ is computed makes it possible to enumerate the OCR errors after the edit distance phase of the algorithm completes. An example of a trace-back table is shown in Figure 1.

As written, Equations 1 and 2 allow the editing costs to be any function of the substrings in question. This makes them flexible enough to model special-case error patterns that might arise in OCR. However, from an implementation standpoint, it is difficult to make use of such generality, so fixed cost assignments are usually employed. For the results we are about to present, we use $c_{sub_{p:q}} = 1$ for all p and q, except that $c_{sub_{1:1}} = -1$ when the two characters in question match. This simplification is convenient, but results in a slight loss in classification accuracy. To illustrate this, consider the following error pattern:

$$-- \alpha\beta -- \;\Rightarrow\; -- \delta\alpha --$$

where α, β, and δ are characters from the OCR program's input/output alphabet, and the rest of the two strings match perfectly. There are at least two possible explanations for this. One is that we have

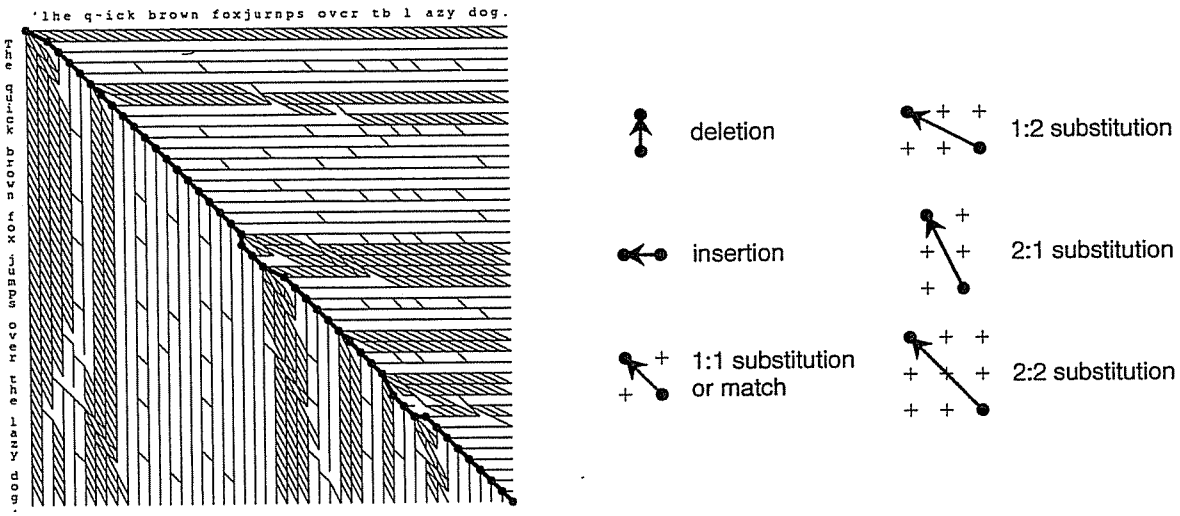


Figure 1: Example of a trace-back table.

witnessed a 2:2 substitution ($\alpha\beta \Rightarrow \delta\alpha$). Another is that a deletion ($\beta \Rightarrow$) and an insertion ($\Rightarrow \delta$) have occurred. Any edit distance computation with fixed costs must choose one or the other of these interpretations independent of the characters in question. In the case that α = "m", β = "n", and δ = "n", we have "mn" $\Rightarrow$ "nm", which seems almost certainly a 2:2 substitution. However, if α = "m", β = ".", and δ = " ", we have "m." $\Rightarrow$ " m", which seems more like a deletion and an insertion.

In the final analysis, the most obvious (and perhaps only) criteria for judging the accuracy of an OCR classification procedure is the extent to which it agrees with what a human expert would say. For the algorithm just described, we conducted an informal evaluation and found that the computer and human agreed approximately 99% of the time.

Finally, we note that our approach to classifying errors treats any *p:q* substitution as a single error event. Clearly there is a fundamental difference in the quality of the OCR output between a 1:1 and a 5:5 substitution. Hence, we need to define a measure of the damage done to a given string by an OCR error:

$$damage(\text{sub}_{p:q}) \equiv \max(p,q)$$

If S is the original string and T is the OCR string, extend $damage(S,T)$ in the obvious way to be the sum of the damages for all the errors determined when editing S into T. We can now compute OCR character recognition accuracy as:

$$accuracy(S,T) \equiv \frac{(|S| - damage(S,T))}{|S|}$$

This intuitive definition yields accuracy rates consistent with the figures published by other researchers. As an example, an original line with 100 characters that undergoes one deletion and one 2:2 substitution is said to be recognized with 97% accuracy.

3 Experimental Method

Previous large-scale OCR experiments have used documents drawn from mixed sources and qualities to study the complexity of error correction for document indexing [8] and the correlation of errors between different OCR packages [1]. We focus on character recognition errors with a clean source

document. We used a source document for which we possessed an on-line representation: Herman Melville's classic novel, *Moby-Dick*. This insures an important degree of uniformity, controllability, and reproducibility to our studies. We treat OCR as a black box and concentrate solely on the errors it generates as the result of using various scanning testbeds.

We gathered statistics on errors made by the OCR package when processing the Hendricks House edition of Herman Melville's novel *Moby-Dick*. The on-line version of the text, as prepared by Professor E. F. Irey, was obtained from the Guttenberg Project at The University of Illinois. This text was used as the data set because (1) it provides a fairly significant body of letter samples for statistics, and (2) the text is an actual work of English literature as opposed to a random assortment of letters or words and hence provides a reasonable context for OCR error analysis.

We summarize here some basic statistics for the original text. The entire text of the novel contains 1,179,194 characters and 193,550 spaces. The most frequently occurring non-space characters are "e" (113,484), "t" (84,050), and "a" (73,892). All the lower-case letters occur more than 10,000 times except for the letters "j" (815), "k" (7,751), "q" (1,202), "v" (8,313), "x" (1,186), and "z" (614). The frequency-of-occurrence for capitalized letters is in the hundreds. Fewer than 100 of each of the digits "0" through "9" occur in the text. The most frequently occurring punctuation mark is the comma (18,809) and the least frequent is the exclamation point (1,719).

In order to obtain a large quantity of uniformly formatted testing pages, we performed some minimal preprocessing of the original text. All text, including chapter numbers and titles, was placed on the page with a single blank space between each word. There were no breaks between titles and paragraphs or between two consecutive paragraphs. Tabs and multiple spaces in the text were all replaced by a single space. The output was left-justified in a single column with an average line length of 77.1 characters. Each page was printed on a 400dpi NeXT laser printer using 10-point Times and 10-point Courier fonts. There were 315 pages with exactly 48 lines on each page (except for the last which had 24 lines). Line and page breaks for the corresponding pages of each font are identical.

The scanners we used in our study were a Ricoh IS-410 flat-bed scanner and a HSD Scan-X Professional flat-bed scanner both equipped with an automatic document feeder (ADF). The Ricoh scanner generally produces a better quality image, however, it is much more expensive than the HSD scanner. We shall refer the HSD scanner as *Scanner-1* and the Ricoh scanner as *Scanner-2* in the following text.

In order to ensure that the image darkness of the two scanners is comparable, we adjusted the binarization thresholds of the two scanners to produce equally dark images. This was done by scanning 20 pages using each scanner and counting the number of black pixels generated in the images. The scanning threshold of Scanner-1 was then adjusted until the average number of black pixels per page was roughly the same ($\pm 2\%$) as that from the Scanner-2 image set.

After calculating the equivalent threshold values, the printed pages for each font were fed into both scanners using the automatic document feeders at a resolution of 300 dpi, yielding 4 sets of input for the OCR package. A primary motivation for our using automatic feeders in the test is that the use of automatic feeders is becoming more prevalent in practice. Thus it is important to study the performance of OCR systems under such environment. As a reference, two additional datasets were generated by manually placing the document pages of both fonts on the Scanner-

2 flatbed. We also generated another two datasets by scanning the document pages of the two fonts upside-down using the Scanner-1 autofeeder and rotating the resultant binary images via software. Hence, there were total 8 sets of testing data.

The scanners generated one-bit TIFF images which we used as input to OCRServant v2.03 OCR package. The OCR software produces output in Rich-Text Format (RTF) which we converted to standard ASCII using an RTF-to-ASCII filter.

It is important to point out that during the entire test, we made no modifications to the TIFF images (other than inverting the mirror image TIFF files) or the OCR-generated text. During the classification procedure, we did uncover a small number of line insertions in the OCR results. Most of the inserted lines were blank. However, in one case, a non-blank textline was introduced as the result of dirt between two textlines in the scanned image. All spurious lines were identified and removed from the text by the classification process, not by manual editing. The techniques used to perform this process are a further extension of the alignment algorithm and are discussed in [5].

4 OCR Error Distributions

In this set of experiments, the classification procedure interprets the OCR errors as up to 4-character substitutions ($0 \leq p, q \leq 4$). Figures 2, 3 show the results of the error classification for each dataset. The value at i^{th} row and j^{th} column in the tables represent the total number of i:j substitutions classified.

Figure 4 shows that the OCR performance of the Scanner-1 datasets was 0.2% lower than that of the Scanner-2 datasets. For a page of average page of 3700 characters, a 0.2% error rate accounts for roughly 7.5 more character errors. Questions then arise as to how Scanner-1 autofeeding process incurs new OCR errors. For example, are the errors random? Do the composition and distribution of the set of OCR errors change?

To answer these questions, we examined our procedure and the resulting data. We noted that the inputs to the both scanners were the same physical set of text pages, and the OCR processes were identical. Therefore, we believe that the increase in the error rate may be attributed to the hardware/firmware used in generating images. Also, we observed that the OCR process performed poorly on the Scanner-1 datasets independent of the font. Furthermore, we note from Figure 2 and Figure 3 that the OCR performance was very similar for each font when Scanner-2 was used regardless of the mode of input (ADF or manual). Therefore, we believe that the performance decline is not a characteristic of auto-feeders in general, but rather may be due to less expensive sheetfeeders.

We now examine the OCR error distribution as a function of physical location. Our interest is to find out if there exists a correlation between the error occurrence and its location on the page. In order to do so, we mark each OCR error according to its coordinates and then examine the distribution.

Our method for doing this is to logically overlay the printing area of the page with a grid of cells. Each row in the grid represents a textline. Each column of the array represents a vertical strip of width δ. We select $\delta = 40$ pixels. We associate a triplet of indices (r, c, p) to each character, where r denotes the grid row, c is the column of the grid and p is the page number. The specific column of the grid is calculated using the horizontal center of each character's bounding box, $\overline{x}$, $c = [\overline{x}/\delta]$.

Similarly, each error is also assigned with a triplet (r, c, p). where $r, c,$ and p are defined as same as above if the error

Scanner-1 ADF					
p:q	⇒ 0	⇒ 1	⇒ 2	⇒ 3	⇒ 4
0 ⇒	0	799	6	3	8
1 ⇒	202	2065	469	59	3
2 ⇒	4	433	335	64	33
3 ⇒	1	5	12	53	34
4 ⇒	0	0	0	5	18

(a)

Scanner-1 ADF Inverted					
p:q	⇒ 0	⇒ 1	⇒ 2	⇒ 3	⇒ 4
0 ⇒	0	976	6	6	10
1 ⇒	251	1478	396	28	2
2 ⇒	0	511	288	74	27
3 ⇒	0	3	13	82	45
4 ⇒	0	0	2	10	30

(b)

Scanner-2 ADF					
p:q	⇒ 0	⇒ 1	⇒ 2	⇒ 3	⇒ 4
0 ⇒	0	772	50	0	49
1 ⇒	258	634	59	3	0
2 ⇒	0	528	146	10	1
3 ⇒	0	3	4	6	0
4 ⇒	0	0	0	0	0

(c)

Scanner-2 Manual					
p:q	⇒ 0	⇒ 1	⇒ 2	⇒ 3	⇒ 4
0 ⇒	0	736	1	0	53
1 ⇒	269	733	59	4	1
2 ⇒	0	549	162	8	2
3 ⇒	0	5	5	4	2
4 ⇒	0	2	0	2	1

(d)

Figure 2: Error composition for Times font.

Scanner-1 ADF					
p:q	⇒ 0	⇒ 1	⇒ 2	⇒ 3	⇒ 4
0 ⇒	0	548	8	3	0
1 ⇒	9	2090	513	25	0
2 ⇒	4	3	151	67	20
3 ⇒	0	1	0	55	25
4 ⇒	0	0	0	0	9

(a)

Scanner-1 ADF Inverted					
p:q	⇒ 0	⇒ 1	⇒ 2	⇒ 3	⇒ 4
0 ⇒	0	312	6	2	1
1 ⇒	5	1839	325	11	2
2 ⇒	4	4	143	53	15
3 ⇒	0	0	2	82	20
4 ⇒	0	0	0	0	17

(b)

Scanner-2 ADF					
p:q	⇒ 0	⇒ 1	⇒ 2	⇒ 3	⇒ 4
0 ⇒	0	185	4	2	0
1 ⇒	4	610	23	3	0
2 ⇒	0	1	122	28	10
3 ⇒	0	0	0	31	3
4 ⇒	0	0	0	0	3

(c)

Scanner-2 Manual					
p:q	⇒ 0	⇒ 1	⇒ 2	⇒ 3	⇒ 4
0 ⇒	0	92	5	2	0
1 ⇒	11	706	39	15	1
2 ⇒	0	7	156	19	5
3 ⇒	0	0	3	23	1
4 ⇒	0	0	0	0	3

(d)

Figure 3: Error composition for Courier font.

	Scanner-1		Scanner-2	
	ADF	ADF Inv.	ADF	Manual
Courier	99.6%	99.7%	99.9%	99.9%
Times	99.4%	99.5%	99.7%	99.7%

Figure 4: OCR accuracy for all eight datasets.

involves only a single character. For burst errors involving multiple characters, c is calculated at the middle of the burst.

We calculate an average error rate per indices pair (r, c) over the 314 full pages of text in each of the datasets to produce an *error distribution map*:

$$\epsilon(r,c) = \sum_{p=1}^{314} \mathcal{N}_c(r,c,p) / \sum_{p=1}^{314} \mathcal{N}_e(r,c,p)$$

where $\mathcal{N}_c(r,c,p)$ is the number of characters at indices (r,c) in page p, and $\mathcal{N}_e(r,c,p)$ is the number of errors at indices (r,c) in page p. Figure 8 shows representative error distribution maps computed for 4 of the datasets.

The shading of the distribution maps show the error rates for each cell in the grid. The darker shades indicate a higher error rate. For this paper, we distinguished the errors per character using 3 intervals: $[0-0.5\%)$, $[0.5-2.0\%)$, and $[2.0-\infty]$.

Comparing the distribution maps of the Scanner-1 datasets (Figures 8a, 8b) with the Scanner-2 datasets (Figures 8c, 8d), one can see that the error distributions are significantly different. There exists a periodic burst of errors in the Scanner-1 ADF versions, while no such error distribution skew is seen in either version from Scanner-2. A further investigation leads us to believe that the periodicity phenomenon is linked to the mechanical movement of the auto-feeder. It is noted that during the scanning, the Scanner-1's autofeeder pauses periodically (probably to unload its buffer). The bitmap images sometimes show a distortion characterized by the "stretching" or "shrinking" of a few scanlines at these stopping positions. These stopping positions are coincident with the "burps" of the error distribution maps.

An estimate of the period of this "pause" effect can be calculated using the following approach. Determine, in pixels, the heights of the textlines and inter-line spacing regions. Using these values, construct a profile for the page showing the placement of these lines along the Y-axis of the page. Next, build a pattern based on the lines that clearly exhibit unusual amounts of damage in the sample of interest. For a range of possible periods and starting displacements, compute the lines that would be damaged in the profile and match this against the pattern. By choosing the combination of parameters that results in the best match, we can arrive at a range of possible values for the period. Figure 5 reflects the calculated period.

Examining the correlation between the textlines which correspond to calculated period and the damage histogram (Figures 9-16), we find that all the highly damaged lines fall on the calculated "pause" locations. We also note that the difference in period values correspond to the difference in scan frame width (approximately 2%).

The error distribution map for the Scanner-1/ADF Courier (Figure 8a) dataset also shows what we believe to be a CCD defect at a vertical column position.

	Scanner-1		Scanner-1 Inv.	
	Times	Courier	Times	Courier
Est. Period (pixels)	236 – 237	230 – 234	237	231

Figure 5: Estimated period of pause errors.

This defect splits many characters into two parts, yielding a visible high degree of OCR damage at the column around the dropped scanline.

From the maps, we also see a high number of OCR errors in the leftmost column in the grid. These errors are due to space insertions in the left margin (due to skew and noise).

Figures 9-16 present histograms showing the damage on a per textline basis over all 315 pages.

From these histograms we can see that the scanner "pauses" exert different degrees of damage to the datasets. Sometimes the pauses fall between the textlines so the effect is not visible. Also, different textlines exhibit different proportions of damage. For example, we see that the Scanner-1/ADF-Invert datasets have an extremely high number of errors at textline 38. We believe the effect of a pause has different impact on the OCR performance depending on the font and also on the position of the damage relative to the text baseline. Clearly, OCR accuracy will be different when there is damage in the middle of a textline as opposed to at the top of the textline.

The significance of damage induced by this scanning defect can be estimated in the following measurements: Let $damage(k)$ be the damage for text line k, $Total_D$ be the total damage across all textlines, $\overline{D}$ be the mean damage for all text lines and Δ_D be the standard deviation. Furthermore, let the set of "good" lines, G, include only those lines whose damage is less than one standard deviation from the mean:

$$G = [\text{damage}(i) | \text{damage}(i) < \overline{D} + \Delta_D]$$

We compute the mean damage over that set as $\overline{G}$. We can then calculate ratio R which reflects the significance of the scanning failure on the size of the error set:

$$\text{R} = 1 - 48 \times \overline{G}/\text{Total}_D$$

Figures 6 and 7 lists the results for the 8 test results. The data in these Figures indicate that the Scanner-1 "pause" defect introduces about 30%-40% more OCR errors. The table also shows that in general, even if we take the "pause" factor off, Scanner-1 still produces a higher number of errors than Scanner-2.

The impact of Scanner-1 defect is also evident in the characteristics of the error composition. From Figures 2 and 3, we can see that Scanner-1 causes a sharp increase of the multi-substitution errors in addition to a substantial increase of simple substitutions. The number (and proportion) of multi-substitutions is much smaller in the Scanner-2 tests. Moreover, it is noted that the 1:2 substitution errors in the Scanner-1 datasets were more than ten times higher the Scanner-2 datasets. Interestingly, the error composition of the Scanner-2 autofeeder datasets and the manual datasets remain similar in all the tests. As a matter of fact, the OCR software performed slightly better on inputs taken from the Scanner-2/ADF than on inputs from manual placement.

	Scanner-1		Scanner-2	
	ADF	ADF Inv.	ADF	Manual
$Total_D$	6,555	6,259	3,508	3,610
$Total_G$	4,334	4,512	3,508	3,610
R	0.34	0.28	0.0	0.0

Figure 6: Significance of pause errors in Times datasets.

	Scanner-1		Scanner-2	
	ADF	ADF Inv.	ADF	Manual
$Total_D$	4,674	3,790	1,355	1,449
$Total_G$	2,609	2,142	1,355	1,449
R	0.44	0.43	0.0	0.0

Figure 7: Significance of pause errors in Courier datasets.

5 Discussions and Conclusions

It has long been noted that in a "real-world" environment, page quality and font idiosyncrasies can substantially affect the number and nature of OCR errors. The data we presented in this paper reveals another important error source: the scanning system. As one of the factors to be considered when performing experimental studies, the choice of a scanner and its attendant ADF can have a major impact on the results. While some systems appear to have little or no effect on error behavior (when compared to manual page placement), others can bias the performance of an OCR package by a significant amount. This is in spite of the fact that the page images may appear comparable in both cases, with no extraordinary damage visible to the human eye.

In addition, we showed that this effect is non-random with respect to distribution across the various OCR error classes. This result is particularly important to recognition algorithms based on statistical classifiers trained using real data. The underlying assumption in such systems is that if the training set is sufficiently large, uniform coverage of the pattern space can be achieved. Our tests suggest that previously overlooked components can distort the OCR results, thereby skewing the model to the peculiarities of the training environment.

Moreover, the changes in the size and composition of error sets brought on by using different scanning hardware could color the performance evaluation of OCR systems as well as make it more difficult to reproduce reported results in some cases.

Results such as these can only be obtained by analyzing large quantities of test data in a rigorous, controlled fashion. The OCR error classification scheme we described in this paper seems to serve this purpose well. This, coupled with new,

specialized visualization tools like our two-dimensional error distribution maps, provides a powerful approach to evaluating the error behavior of OCR systems.

References

[1] R. Bradford and T. Nartker. Correlation in contemporary OCR systems. In *Proceedings of the First International Conference on Document Analysis and Recognition*, pages 516–524, October 1991.

[2] V. Concepcion and D. D'Amato. Synchronous tracking of outputs from multiple OCR systems. In *Character Recognitionx Technologies, SPIE*, volume 1906, pages 218–227, 1993.

[3] J. Esakov, D. Lopresti, and J. Sandberg. Classification and distribution of optical character recognition errors. In *IS&T/SPIE International Symposium on Electronic Imaging*, 1994.

[4] S. Kahan, T. Pavlidis, and H. Baird. On the recognition of printed characters of any font and size. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 9(2):274–288, March 1987.

[5] D. Lopresti. An algorithm for classifying optical character recognition errors. Technical Report MITL-TR-86-93, Matsushita Information Technology Laboratory, February 1994.

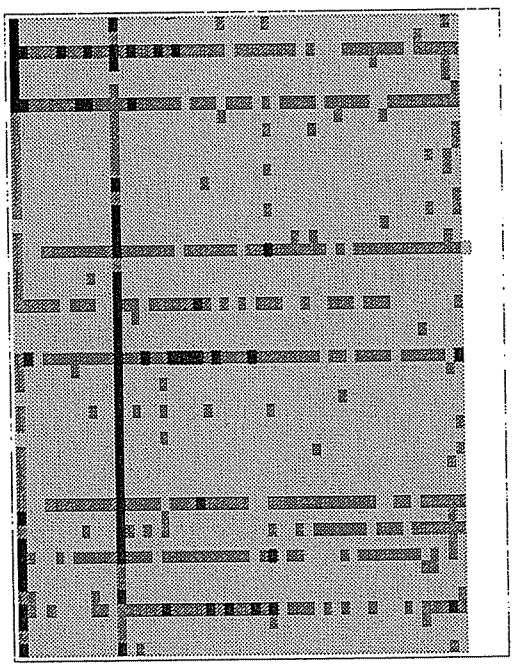
[6] D. Lopresti and J. Sandberg. Certifiable optical character recognition. In *the Second International Conference on Document Analysis and Recognition*, pages 432–435, October 1993.

[7] S. Rice, J. Kanai, and T. Nartker. A difference algorithm for OCR-generated text. In *Proceedings of the IAPR Workshop on Structural and Syntactic Pattern Recognition*, Bern,Switzerland, August 1992.

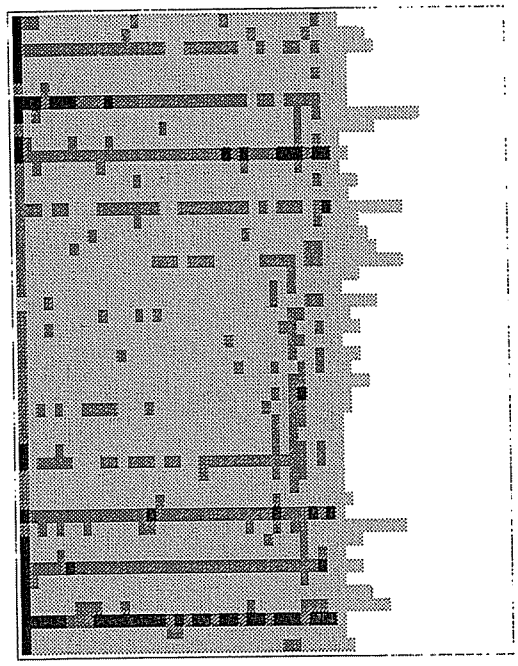
[8] S. Rice, J. Kanai, and T. Nartker. An evaluation of OCR accuracy. In *1993 Annual Report, UNLV*, pages 9–33. UNLV Information Science Research Institute, 1993.

[9] K. Taghva, J. Borsack, B. Bullard, and A. Condit. Post-editing through approximation and global correction. In *1993 Annual Report*, pages 57–68. UNLV Information Science Research Institute, 1993.

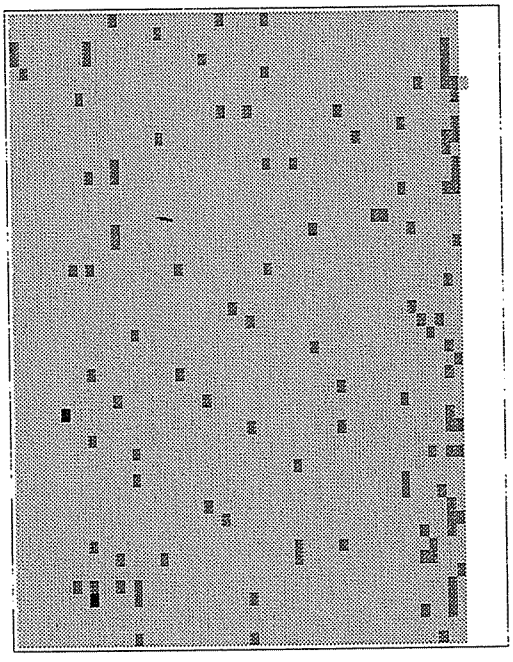
[10] R. Wagner and M. Fisher. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.



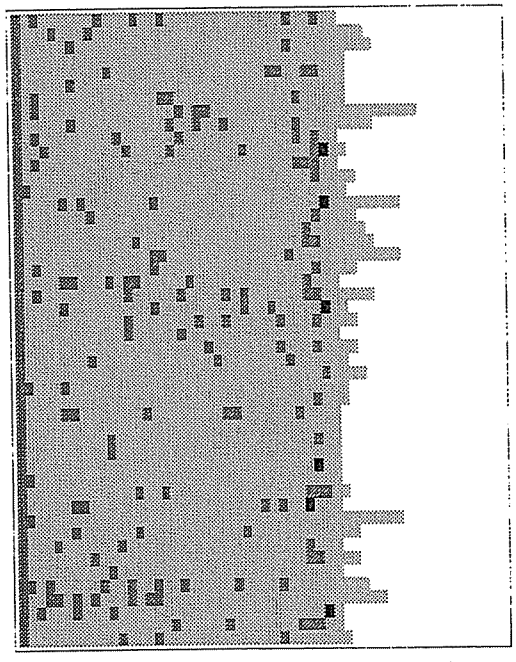
(a) Scanner-1 ADF (Courier)



(b) Scanner-1 ADF (Times)



(c) Scanner-2 Manual (Courier)



(d) Scanner-2 Manual (Times)

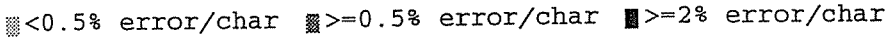


Figure 8: Representative error distribution maps.