# Using Consensus Sequence Voting to Correct OCR Errors

**Daniel Lopresti** **Jiangying Zhou**
*Matsushita Information Technology Laboratory*
*Panasonic Technologies, Inc.*
*Two Research Way*
*Princeton, NJ 08540, USA*
*[dpl,jz]@mitl.research.panasonic.com*

**ABSTRACT**

In this paper we present an approach to the OCR voting problem based on a result from molecular biology. Our technique uses a practical heuristic modification of an exponential-time algorithm that is guaranteed to be optimal for all cases of interest. Experimental results demonstrate that between 20% and 40% of the errors caused by a single OCR package can be corrected by simply scanning a page three times and running our voting procedure. This level of performance is achieved without making *a priori* assumptions about the distribution of OCR errors (i.e., no "training" is required).

## 1 Introduction

Optical character recognition (OCR) is a process that maps a page image $I$ representing a source text string $S$ into a recognized string $R$. Typically, both $S$ and $R$ are defined over the same fixed alphabet $\Sigma$ (for English, $\Sigma$ is usually ASCII). An example is shown in Figure 1. In this case, $R$ contains two errors introduced by the OCR software: a $t$ was misread as an $l$ (in the second "the"), and a mapping for the $a$ in "lazy" could not be determined with reasonable certainty, hence a "don't know" character (~) was output.

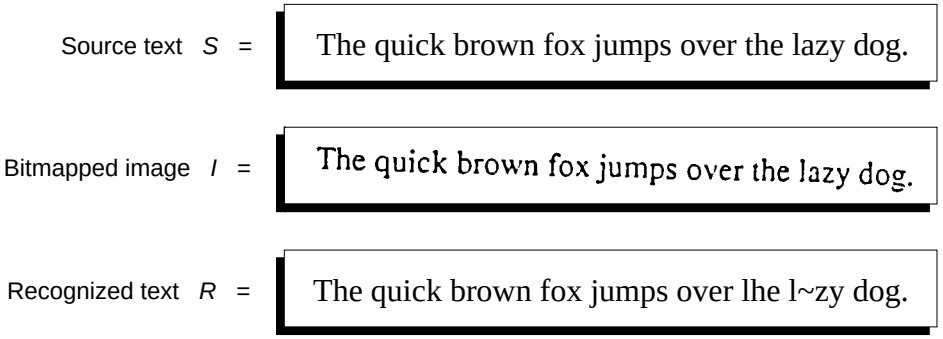


**Figure 1: An OCR example (Caere OmniPage Professional, ver. 2.1).**

While the most abstract view of OCR is as a "black box" recognizer accepting bitmaps and generating text, in truth the process consists of multiple levels of segmentation followed by a final recognition (i.e., pattern matching) step. A conceptual overview of the "generic" OCR process is shown in Figure 2(a). Page images are segmented into lines, lines into characters, and then finally characters are matched against pre-defined prototypes to determine their translations.

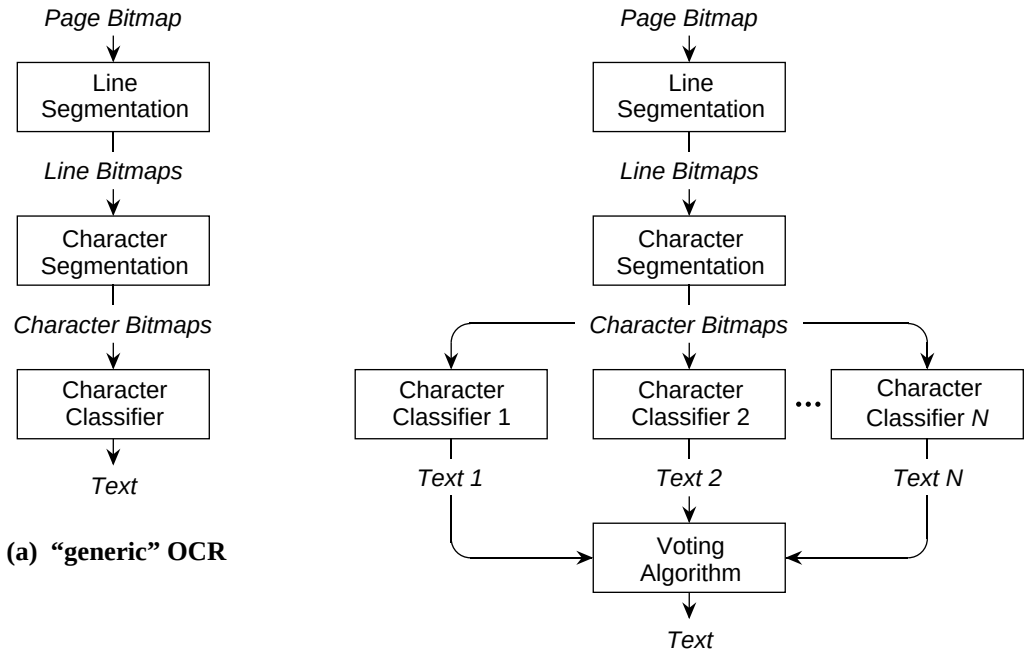


**Figure 2: Overview of "generic" OCR and traditional voting procedures.**

Current OCR is far from perfect, as Figure 1 indicated. Mistakes can be made at any level in Figure 2(a), and error rates typically range from 0.2% (for clean, first-generation copy) to 20% or worse (for multi-generation photocopies and faxes). The following example illustrates some common OCR errors:

**Source** `The quick brown fox jumps over the lazy dog.`
**Recognized** `'lhe q~ick brown foxjurnps ovcr tb l azy dog.`

Intuitively, we can classify the errors as: simple substitutions ($e \rightarrow c$), multi-substitutions ($T \rightarrow 'l$, $m \rightarrow rn$, $he \rightarrow b$), space deletions and insertions, and unrecognized characters ($u \rightarrow \tilde{}$). Without knowing the internals of the OCR package, it is not possible to state with certainty what has caused a given error. It is, however, fairly safe to say that some errors arise in the classification step (e.g., $e \rightarrow c$), while others are due to one of the segmentation phases (e.g., $m \rightarrow rn$).

It is well known that OCR classifiers (and non-linear classifiers in general) are sensitive to slight perturbations in their input. As a result, OCR error behavior can vary; the same page re-scanned and re-OCR'ed, or OCR'ed using a different package, may well exhibit an overlapping but essentially different set of errors. Motivated by this phenomenon, a number of researchers have suggested combining the outputs of multiple classifiers through *voting* procedures, with the goal of producing results better than those that could be obtained using any single classifier.

The basic voting process is illustrated in Figure 2(b). While there are important differences between the schemes described in the literature, this same overall approach is used in

most if not all cases. Instead of a single classifier as the last phase in OCR, the outputs of $N$ classifiers are combined using any of a number of voting algorithms.

A specific example of this is shown in Figure 3. Here we also illustrate how such a scheme might fail. The voting procedure (simple majority voting in this case) corrects the $l \rightarrow 1$ error in the output of Classifier 2 (i.e., "Cal1"). But because of a segmentation mistake, all three classifiers are mislead into reading $m$ as $rn$.

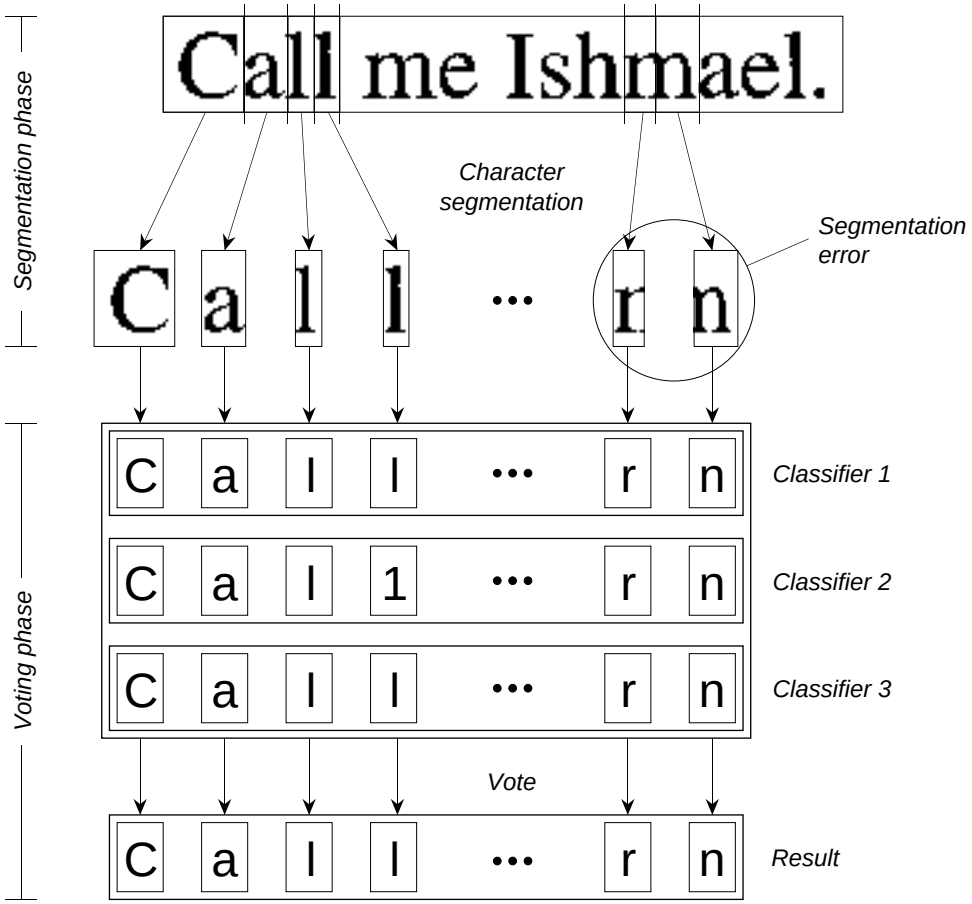


**Figure 3: Cases where traditional voting succeeds and fails.**

In the next section we discuss related work. We then present our new approach to the OCR voting problem. Section 4 gives experimental results that demonstrate the promise of this method. Finally, we draw some conclusions and present areas for future research in Section 5.

## 2 Related Research

Existing methods for combining OCR results for error detection and correction mainly deal with isolated characters. A very common approach is to take a set of classifier outputs directly and determine the input class with the highest aggregate score[1]. An alternative strategy is to explicitly determine the classifier most likely to produce the correct decision[2].

Another approach is to measure how well individual classifiers recognize members of a class and uses this measure to build discriminant functions[3]. Other methods have explored various weighted-voting schemes including the ranking-order method, the confidence accumulation method, the Dempster-Shafer evidence method[4–6], etc. These approaches often require a comparable and consistent representation of the decisions produced by individual classifiers.

An alternative to the isolated-character-based approaches is to combine the outputs of OCR processes using a word-level scheme. In a paper by Ho, Hull, and Srihari, a decision combination strategy computes confidence scores for a lexicon produced by a collection of word recognition algorithms and derives a consensus ranking[7]. Another word-based approach uses a similar Borda count method[8]. In both cases the decision is made on words, not on characters within words. Hence, a legitimate but incorrect word may pass as the final decision. Moreover, in practice word boundaries are sometimes difficult to identify reliably. For example, punctuation marks are often taken as word delimiters in addition to spaces[9]. Punctuation marks, however, are very error-prone due to their small size.

Recently, the idea of combining $N$ strings generated by multiple OCR systems has also received attention. For example, a method for finding a common substring is proposed for use in an OCR voting scheme[10]. The approach described is not guaranteed to find the *longest* common substring, however, which makes a rigorous analysis difficult. In a paper by Concepcion and D'Amoto, a dynamic programming algorithm is used to synchronize two OCR outputs[11]. This approach is close in spirit to ours, which can be viewed as a generalization of a related algorithm and its heuristic to larger numbers of OCR voters.

## 3 Consensus Sequence Voting

It has been noted that a large percentage of all OCR errors are due to failures in character segmentation[12]. Unfortunately, this step is not covered by the voting procedure in Figure 2(b). Ideally, we would like to be able to combine the results of multiple segmentation attempts as well as multiple character classifications. This does, however, introduce an additional complication, as shown below:

**OCR 1** `Cal1 me Ishmael.`
**OCR 2** `Call me Ishrnael.`
**OCR 3** `Call mc Ishmael.`

Each OCR line contains one error. Simple character-by-character voting is able to correct two of the errors ($l \rightarrow 1$ and $e \rightarrow c$). The other error, however, involves a segmentation mistake ($m \rightarrow rn$), making a character-by-character vote impossible without first taking explicit action to determine a correspondence between the various characters. Somehow we must recognize that $m$, $rn$, and $m$ should all vote together in the same "election."

This problem bears a strong resemblance to one from the field of molecular biology, where it is not uncommon to be confronted by three or more related DNA sequences with a need to determine their most plausible shared ancestor. By analogy, we can view OCR candidate lines as having "evolved" from a single source line as the result of errors in the OCR process. Adopting this model from molecular biology, along with a dynamic programming algorithm for its solution and a heuristic to speed the computation (both to be described shortly), allows us to structure the OCR voting problem as shown in Figure 4.

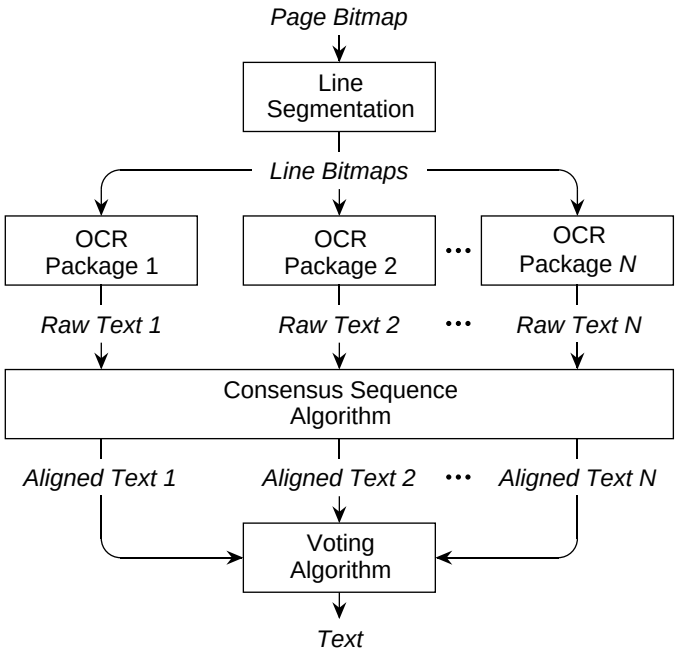


**Figure 4: Consensus sequence voting.**

The problem of determining a consensus sequence given multiple candidate sequences is related to the well-known, mathematically rigorous computation for determining the *edit distance* between two sequences $S$ and $R$. In the traditional case[13], the following three operations are permitted:

1. delete a character,
2. insert a character,
3. substitute one character for another.

Each of these is assigned a cost, $c_{del}$, $c_{ins}$, and $c_{sub}$, and the edit distance, $d(S,R)$, is defined as the minimum cost of any sequence of basic operations that transforms $S$ into $R$. This optimization problem can be solved using a well-known dynamic programming algorithm. Let $S = s_1 s_2 \ldots s_m$, $R = r_1 r_2 \ldots r_n$, and define $d_{i,j}$ to be the distance between the first $i$ characters of $S$ and the first $j$ characters of $R$. Note that $d(S,R) = d_{m,n}$. The main dynamic programming recurrence is then:

$$d_{i,j} = \min \begin{cases} d_{i-1,j-1} + c_{sub}(s_i, r_j) \\ d_{i-1,j} + c_{del}(s_i) \\ d_{i,j-1} + c_{ins}(r_j) \end{cases} \qquad \text{for } 1 \le i \le m, 1 \le j \le n \tag{1}$$

When Eq. (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. By letting $S$ be the source (original) text and $R$ be the recognized (OCR) text, and recording the optimal decision(s) made at each step, this procedure can be used for the classification of OCR errors[14,15]. More complicated variations can handle more sophisticated errors, including multi-character substitutions (i.e., segmentation errors).

The *consensus sequence* problem for candidate lines $R_1, R_2, \ldots, R_N$ is defined in terms of edit distance: determine a sequence $C$ such that the combined cost of editing $C$ into each of the $R_i$ is minimized. That is, if $\mathcal{D}(R_1, R_2, \ldots, R_N)$ represents the cost of the consensus sequence alignment, then:

$$\mathcal{D}(R_1, R_2, \ldots, R_N) \equiv \min_{C \in \Sigma^*} \sum_{i=1}^{N} d(C, R_i)$$

and the set of all possible consensus sequences is:

$$Cons(R_1, R_2, \ldots, R_N) \equiv \{C \in \Sigma^* \mid \sum_{i=1}^{N} d(C, R_i) = \mathcal{D}(R_1, R_2, \ldots, R_N)\} \tag{2}$$

While in general there may be more than one consensus sequence, we are usually only interested in finding a representative $C \in Cons(R_1, R_2, \ldots, R_N)$.

Before discussing the consensus sequence computation, it is helpful to define several new quantities. Let $\phi$ be the empty string, $\Sigma' = \Sigma \cup \{\phi\}$, $c_{sub}(c, \phi) = c_{del}(c)$, and $c_{sub}(\phi, c) = c_{ins}(c)$. Now define:

$$\delta(r_1, r_2, \ldots, r_N) \equiv \min_{c \in \Sigma'} [c_{sub}(c, r_1) + c_{sub}(c, r_2) + \ldots + c_{sub}(c, r_N)] \tag{3}$$

We can view $\delta$ as the voting function, as it determines the "best" value $c$ at a given stage in the computation. Although finding this $c$ might appear to require an exhaustive search over $\Sigma'$, in practice the cost function $c_{sub}$ is usually simple enough that the choice of the optimal $c$ is obvious.

An algorithm for computing $\mathcal{D}$ (and hence for determining a consensus sequence) is given in a paper by Kruskal[16]. It involves constructing $C$ as we build a multi-dimensional distance table. For three candidate sequences $R_1 = r_{1_1} r_{1_2} \ldots r_{1_l}$, $R_2 = r_{2_1} r_{2_2} \ldots r_{2_m}$, and $R_3 = r_{3_1} r_{3_2} \ldots r_{3_n}$, the recurrence is:

$$\mathcal{D}_{i,j,k} = \min \begin{cases} \mathcal{D}_{i-1,j-1,k-1} + \delta(r_{1_i}, r_{2_j}, r_{3_k}) \\ \mathcal{D}_{i-1,j-1,k} + \delta(r_{1_i}, r_{2_j}, \phi) \\ \mathcal{D}_{i-1,j,k-1} + \delta(r_{1_i}, \phi, r_{3_k}) \\ \mathcal{D}_{i-1,j,k} + \delta(r_{1_i}, \phi, \phi) \\ \mathcal{D}_{i,j-1,k-1} + \delta(\phi, r_{2_j}, r_{3_k}) \\ \mathcal{D}_{i,j-1,k} + \delta(\phi, r_{2_j}, \phi) \\ \mathcal{D}_{i,j,k-1} + \delta(\phi, \phi, r_{3_k}) \end{cases} \quad \text{for } 1 \le i \le l, 1 \le j \le m, 1 \le k \le n \tag{4}$$

This computation requires time $O(lmn)$.

Consider for a moment the first term in Eq. (4) (the one involving $\mathcal{D}_{i-1,j-1,k-1}$). A natural cost assignment to use is:

$$c_{sub}(c, r) = \begin{cases} 0 & \text{if } c = r \\ 1 & \text{if } c \neq r \end{cases}$$

Under these circumstances, it should be quite clear that the character $c \in \Sigma'$ that yields the minimum combined cost is precisely the majority "vote" among the three characters $r_{1_i}$, $r_{2_j}$, and $r_{3_k}$. If there is no majority (i.e., all three characters are different), then any

one can be chosen arbitrarily. The other cases in Eq. (4) handle possible scenarios involving one or more of the sequences "missing" the character in question and hence voting for a deletion.

To generalize the consensus sequence computation to higher dimensions (i.e., more voters), let $\Delta \in \{0, 1\}$ and define:

$$r_{i:\Delta} \equiv \begin{cases} \phi & \text{if } \Delta = 0 \\ r_i & \text{if } \Delta = 1 \end{cases}$$

We view $r_{i:\Delta}$ as the substring starting at position $i$ of length $\Delta$. Then we have:

$$\mathcal{D}_{i_1,i_2,\ldots,i_N} = \min_{\Delta_i} [\ \mathcal{D}_{i_1-\Delta_1,i_2-\Delta_2,\ldots,i_N-\Delta_N} + \delta(r_{1_{i_1:\Delta_1}}, r_{2_{i_2:\Delta_2}}, \ldots, r_{N_{i_N:\Delta_N}})\ ] \qquad (5)$$

$$\text{for } \ 1 \le i_j \le |R_j|, \ \ j = 1, 2, \ldots, N$$

The number of basic steps to evaluate one iteration of Eq. (5) is $2^N - 1$, hence the total for the computation is:

$$(2^N - 1) \prod_{i=1}^{N} |R_i|$$

If the sequences all have length $n$, the time is $O(n^N)$. That is, the time is exponential in the number of sequences (i.e., voters).

This fact has lead a number of researchers to regard this approach as impractical despite its mathematical elegance. Fortunately, though, there is a simple heuristic that works in all cases of interest (i.e., we can prove strict bounds on its performance), and that makes the computation quite feasible from the standpoint of its run-time.

The heuristic is based on the observation that if the sequences in a collection are all quite similar (as should be the case with voting OCR lines), then the optimal dynamic programming path must remain close to the main diagonal for Eqs. (1), (4), and (5). As a result, we need only compute those distance values in a region near the diagonal, bounded by a small constant, $k$, which is determined in advance.

Let $\widehat{\mathcal{D}}(R_1, R_2, \ldots, R_N)$ be the value computed by the heuristic. As before, the recurrence is:

$$\widehat{\mathcal{D}}_{i_1,i_2,\ldots,i_N} = \min_{\Delta_i} [\ \widehat{\mathcal{D}}_{i_1-\Delta_1,i_2-\Delta_2,\ldots,i_N-\Delta_N} + \delta(r_{1_{i_1:\Delta_1}}, r_{2_{i_2:\Delta_2}}, \ldots, r_{N_{i_N:\Delta_N}})\ ] \qquad (6)$$

However, instead of allowing the indices to range freely as in Eq. (5), we only compute a particular $\widehat{\mathcal{D}}_{i_1,i_2,\ldots,i_N}$ when:

$$-k \le i_\alpha - i_\beta \le k \quad \text{for } \ 1 \le \alpha, \beta \le N$$

When the sequences in question are all similar, a performance guarantee can be made for the heuristic. The following straightforward result has been stated independently by researchers working in several different disciplines; we repeat it here without proof:

**Theorem 1** *If* $\mathcal{D}(R_1, R_2, \ldots, R_N) \le k$, *then*

$$\widehat{\mathcal{D}}(R_1, R_2, \ldots, R_N) = \mathcal{D}(R_1, R_2, \ldots, R_N).$$

Of course, not only does the heuristic compute the same cost function as the optimal algorithm under these circumstances, it also determines the same consensus sequence.

The precise number of steps used by the heuristic is somewhat more difficult to analyze than the original algorithm, but the asymptotic time complexity is fairly easy to determine. Again we assume that all $N$ sequences have the same length $n$. If we allow one index, say $i_1$, to range freely $1 \leq i_1 \leq n$, then each of the remaining $N-1$ indices is constrained to take on between $k+1$ and $2k+1$ values. Hence, the asymptotic time complexity is $O(nk^N)$. While this is still exponential, $k$ is limited by the maximum number of OCR errors we expect on a line, and is typically much smaller than $n$, the total length of a line. Table 1 compares the times required for the optimal algorithm and the heuristic assuming that $n = 80$ and $k = 8$. The improvement offered by the heuristic is dramatic.

**Table 1: Approximate number of computation steps (optimal vs. heuristic).**

| $N$ (voters) | Optimal | Heuristic | Speed-up factor |
|---|---|---|---|
| 1 | 80 | 80 | 1 |
| 2 | 6,400 | 640 | 10 |
| 3 | 512,000 | 5,120 | 100 |
| 4 | 40,960,000 | 40,960 | 1,000 |
| 5 | 3,276,800,000 | 327,680 | 10,000 |
| 6 | 262,144,000,000 | 2,621,440 | 100,000 |
| 7 | $2.10 \times 10^{13}$ | 20,971,520 | 1,000,000 |
| 8 | $1.68 \times 10^{15}$ | 167,772,160 | 10,000,000 |
| 9 | $1.34 \times 10^{17}$ | 1,342,177,280 | 100,000,000 |
| 10 | $1.07 \times 10^{19}$ | 10,737,418,240 | 1,000,000,000 |

In an era of 100-200 MIPS workstations, optimal voting is not feasible for more than four voters, whereas the heuristic remains practical for up to eight voters. For the case of four voters, the heuristic is three orders of magnitude faster than the original algorithm. Further improvements in the heuristic seem quite possible and are a subject for future research.

An example of the performance of our approach on a line of real OCR test data is shown in Figure 5. The computation took 0.1 seconds on a DECstation 5000/200 workstation.

```
OCR 1   the circulation.  Whenever I ~nd myself growing grim about the mouth;
OCR 2   the circulao'on.  Whenever I find myself growing grim about the mou~;
OCR 3   the circulation.  Whenever I find myself growinp ~rim about the mouth;
Vote    the circulation.  Whenever I find myself growing grim about the mouth;
```

**Figure 5: Voting example using real OCR data.**

# 4 Experimental Results

In this section we describe an experiment which illustrates how "random" perturbations in the OCR input image can be used to alter the distribution of OCR errors and how this

result can be subsequently exploited by our voting process to improve OCR performance.

Many researchers have explored the use of multiple OCR classifiers to generate independent "guesses" for a given input. We take an alternate approach that uses only a single OCR process to generate multiple candidates. Our idea is to create small variations in the input image *prior* to OCR. The rationale behind this is that OCR processes are sensitive to slight perturbations in their input. For example, error distributionserror distribution can vary from scan to scan (i.e., the same page re-scanned may well exhibit a different error profile). Motivated by this phenomenon, we investigated using the natural variation caused by the scanning process to generate multiple OCR input images for consensus sequence voting. We found that by using the results from three scanned versions of the same page as input, our voting procedure could correct between 20% and 40% of the OCR errors.

This point is illustrated by the two images in Figure 6. These nearly identical bitmaps were obtained by binarizing the same gray-level image using slightly different thresholds (190/195). Despite their close similarity, one OCR package yielded "concemment" for the left image and "concernment" for the right. The inherent non-linear nature of OCR classifiers induces a certain degree of "randomness" in the distribution of OCR errors which can be exploited by voting procedures like the one just described.

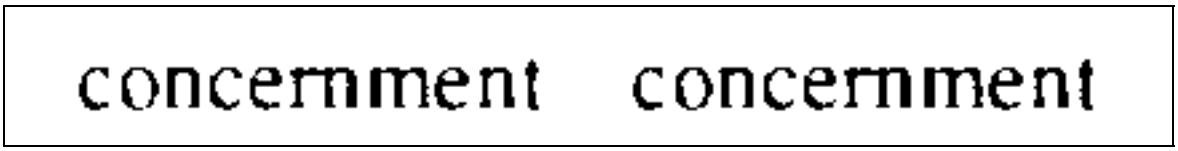


**Figure 6: Perturbed inputs yield different results ("concemment" vs. "concernment").**

An appealing characteristic of using only one OCR package for our tests is that it provides a controlled environment for analyzing the voting process; all the inputs are consistent and comparable. It has long been noted that combined classifiers perform better than a single classifier. Yet how exactly this improvement comes about is not always clear. Viewed from this perspective, our experimental approach lays a uniform foundation for the analysis of various voting strategies. More importantly, we do not make any *a priori* assumptions about the voters. Multiple-classifier voting schemes that rely on weights require training data that necessarily limits their generality. A well-known weakness of such systems is that their solutions are often training-set-specific. The results we are about to present avoid this pitfall.

To test the effectiveness of our voting scheme, we generated results on seven datasets representing a range of document qualities. The source document in all cases was derived from the first 20 pages of Herman Melville's novel *Moby-Dick*. The text of the novel was pre-processed to ensure uniform page formatting. In particular, all text was placed on the page with a single space between words. There were no explicit paragraph breaks (i.e., all lines were made nearly the same length, approximately 77 characters). The output was left-justified in a single column, with exactly 48 lines on each page. The pages were set in 10-point Times and printed on a 400dpi NeXT laserprinter.

So that we could examine datasets of different qualities, we photocopied the source document six times successively using a Panasonic FP 6070 copier. This yielded seven sets of test pages $g_0, g_1, g_2, \ldots, g_6$, where $g_0$ corresponds to the original source document and $g_i$ to the $i^{th}$ generation photocopy. These seven test sets were then fed through a Ricoh IS-410 flat-bed scanner using the automatic document feeder and scanned at a resolution of 300

**Table 2: Consensus sequence voting – accuracy improvement.**

| Test dataset | OCR1 accuracy | OCR2 accuracy | OCR3 accuracy | Average accuracy | Vote accuracy |
|---|---|---|---|---|---|
| $g_0$ | 99.7 | 99.7 | 99.7 | 99.7 | 99.8 |
| $g_1$ | 99.4 | 99.3 | 99.4 | 99.4 | 99.6 |
| $g_2$ | 98.7 | 98.7 | 98.7 | 98.7 | 99.2 |
| $g_3$ | 97.5 | 97.2 | 97.5 | 97.4 | 98.2 |
| $g_4$ | 95.4 | 95.4 | 95.4 | 95.4 | 96.3 |
| $g_5$ | 93.8 | 93.7 | 94.1 | 93.9 | 95.1 |
| $g_6$ | 91.9 | 91.9 | 91.9 | 91.9 | 93.3 |

dpi. Each dataset $g_i$ was scanned three times successively under the same settings.

The scanner generated one-bit TIFF images which we used as input to OCRServant v2.03 running on a NeXT workstation. For each test dataset, the voting process took the OCR results for the three scanned versions and produced a consensus sequence output. We refer to the "raw" OCR results as *OCR1*, *OCR2*, and *OCR3*, and to the voting result as *Vote*. To analyze the various outputs, we applied our automatic OCR error classification procedure[15]. Table 2 gives recognition accuracies for the three scanned versions of each dataset, as well as the accuracies for the corresponding voting result. In all cases, 3-way consensus sequence voting performed better than any of the original OCR runs. Note that the voting process shows a consistent improvement in spite of the fact that the original three datasets for a given generation are all quite close in terms of accuracy. The strength of the consensus sequence voting process comes from exploiting the differences in their error distributions.

A closer examination of the effects of voting on OCR error behavior in each case is presented in Table 3. Here we give error counts for the three original OCR runs as well as for the consensus sequence vote. These figures illustrate the impact of voting relative to the size of the error set. From Table 3, it is evident that by simply scanning each page three times, voting was able to correct up to 38% of the errors caused by OCR. Interestingly, voting corrects an increasing percentage of errors as the document quality degrades slightly, but then its net effect starts to dissipate as the copy quality deteriorates further. This suggests that certain errors are inherent in the document (i.e., the physical medium) and hence not amendable to voting, a fairly intuitive result. Still, it is quite clear that a significant "baseline" error rate can be eliminated through consensus sequence voting.

# 5 Conclusions

In this paper we have presented a new approach to the problem of voting to correct OCR errors. Based on a result from molecular biology, the consensus sequence heuristic we described is fast enough to be practical, yet it is still guaranteed to be optimal for all cases of interest. Our experimental data shows that this technique can correct between 20% and 40% of OCR errors over a range of document qualities.

The notion that repeatedly sampling an input can lead to higher recognition accuracies is also a fundamental result. In another paper, we show that by relaxing a basic assumption in statistical pattern recognition, that the input be sampled only once, it becomes possible

**Table 3: Consensus sequence voting – OCR error counts.**

| Test dataset | OCR1 errors | OCR2 errors | OCR3 errors | Average errors | Vote errors | Improvement rate |
|---|---|---|---|---|---|---|
| $g_0$ | 228 | 231 | 249 | 236 | 157 | 0.335 |
| $g_1$ | 472 | 480 | 505 | 485 | 312 | 0.358 |
| $g_2$ | 945 | 957 | 970 | 957 | 598 | 0.375 |
| $g_3$ | 1,870 | 1,886 | 2,048 | 1,934 | 1,359 | 0.298 |
| $g_4$ | 3,403 | 3,430 | 3,431 | 3,421 | 2,742 | 0.199 |
| $g_5$ | 4,350 | 4,608 | 4,637 | 4,531 | 3,651 | 0.194 |
| $g_6$ | 6,025 | 6,031 | 6,046 | 6,034 | 4,963 | 0.177 |

to build classifiers that beat the Bayes error bound[17].

In the future, we plan to examine ways of further speeding the voting process. We also intend to evaluate the consensus sequence model from the standpoint of weighted-voting. Finally, we hope to extend this approach to more general image recognition problems.

## References


[1] C. Nadal, R. Legault, and Ching Y. Suen. Complementary algorithms for the recognition of totally unconstrained handwritten numerals. In *Proceedings of International Conference on Pattern Recognition*, volume 1, pages 443–449, New Jersey, 1990.

[2] Michael Sabourin, Amar Mitiche, Danny Thomas, and George Nagy. Classifier combination for hand-printed digit recognition. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 163–166, Japan, October 1993.

[3] Tin Kam Ho. Recognition of handwritten digits by combining independent learning vector quantizations. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 818–821, Japan, October 1993.

[4] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. On multiple classifier systems for pattern recognition. In *Proceedings of the 11th International Conference on Pattern Recognition*, pages 84–87, Netherlands, September 1992.

[5] E. Mandler and J. Schurmann. Combining the classification results of independent classifiers based on Dempster-Shafer theory of evidence. In E. S. Felsema and L. N. Kanai, editors, *Pattern Recognition and Artificial Intelligence*. Elseiver Science, North Holland, 1988.

[6] Y. S. Huang and Ching Y. Suen. Combination of multiple classifiers with measurement values. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 598–601, Japan, October 1993.

[7] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Word recognition with multi-level contextual knowledge. In *Proceedings of the First International Conference on Document Analysis and Recognition*, pages 905–915, October 1991.

[8] Brigitte Plessis, Anne Sicsu, and Laurent Heutte et al. A multi-classifier combination strategy for the recognition of handwritten cursive words. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 642–645, Japan, October 1993.

[9] R. M. K. Sinha and Birendra Prasada. Visual text recognition through contextual processing. *Pattern Recognition*, 21(5), 1988.

[10] Stephen V. Rice, Junichi Kanai, and Thomas A. Nartker. A difference algorithm for OCR-generated text. In *Proceedings of the IAPR Workshop on Structural and Syntactic Pattern Recognition*, Bern, Switzerland, August 1992.

[11] Vicente P. Concepcion and Donald P. D'Amoto. Synchronous tracking of outputs from multiple OCR systems. *SPIE Character Recognition Technologies*, 1906, 1993.

[12] C. H. Chen and J. L. DeCurtins. Word recognition in a segmentation-free approach to OCR. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 573–576, October 1993.

[13] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

[14] Jeffrey Esakov, Daniel P. Lopresti, and Jonathan S. Sandberg. Classification and distribution of optical character recognition errors. In Luc M. Vincent and Theo Pavlidis, editors, *Proceedings of the IS&T/SPIE International Symposium on Electronic Imaging*, volume 2181, pages 204–216, February 1994.

[15] Jeffrey Esakov, Daniel P. Lopresti, Jonathan S. Sandberg, and Jiangying Zhou. Issues in automatic OCR error classification. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, pages 401–412, April 1994.

[16] Joseph B. Kruskal. An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM Review*, 25(2):201–237, April 1983.

[17] Jiangying Zhou and Daniel Lopresti. Transcending the Bayes Limit through repeated sampling. In *Proceedings of the IAPR Workshop on Machine Vision Applications (to appear)*, Kawasaki, Japan, December 1994.

# Index