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

November 1, 1995

Abstract

We present experimental results suggesting that between 20% and 50% of the errors caused by a single OCR package can be eliminated by simply scanning a page three times and running a "consensus sequence" voting procedure. This technique, which originates from molecular biology, takes exponential time in general, but can be specialized to a fast heuristic guaranteed to be optimal for the cases of interest. The improvement in recognition accuracy is achieved without making *a priori* assumptions about the distribution of OCR errors (*i.e.*, no "training" is required).

Keywords: *OCR error correction*, *voting algorithm*, *classifier combination*, *edit distance*, *consensus sequence*, *repeated sampling*.

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 Σ (for English, Σ 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.

While in the most abstract sense OCR is 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.

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

*Appears in *Computer Vision and Image Understanding*, 67(1): 39-47, July 1997.

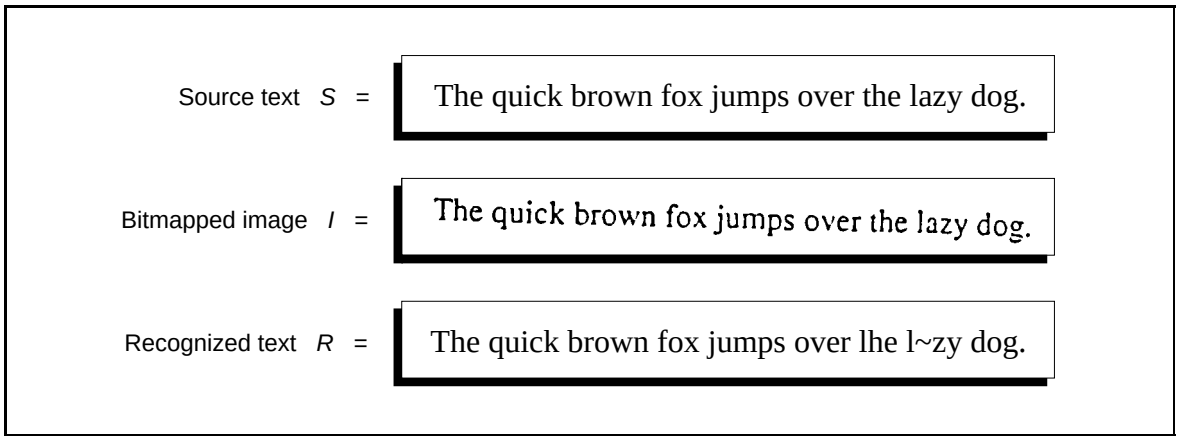

Figure 1: An OCR example (Caere OmniPage Professional, ver. 2.1).

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.`

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, 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 another package, may well exhibit an overlapping but 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.

In this paper, we describe a method of correcting OCR errors based on computing a "consensus sequence" from the outputs obtained by repeatedly scanning and OCR'ing the

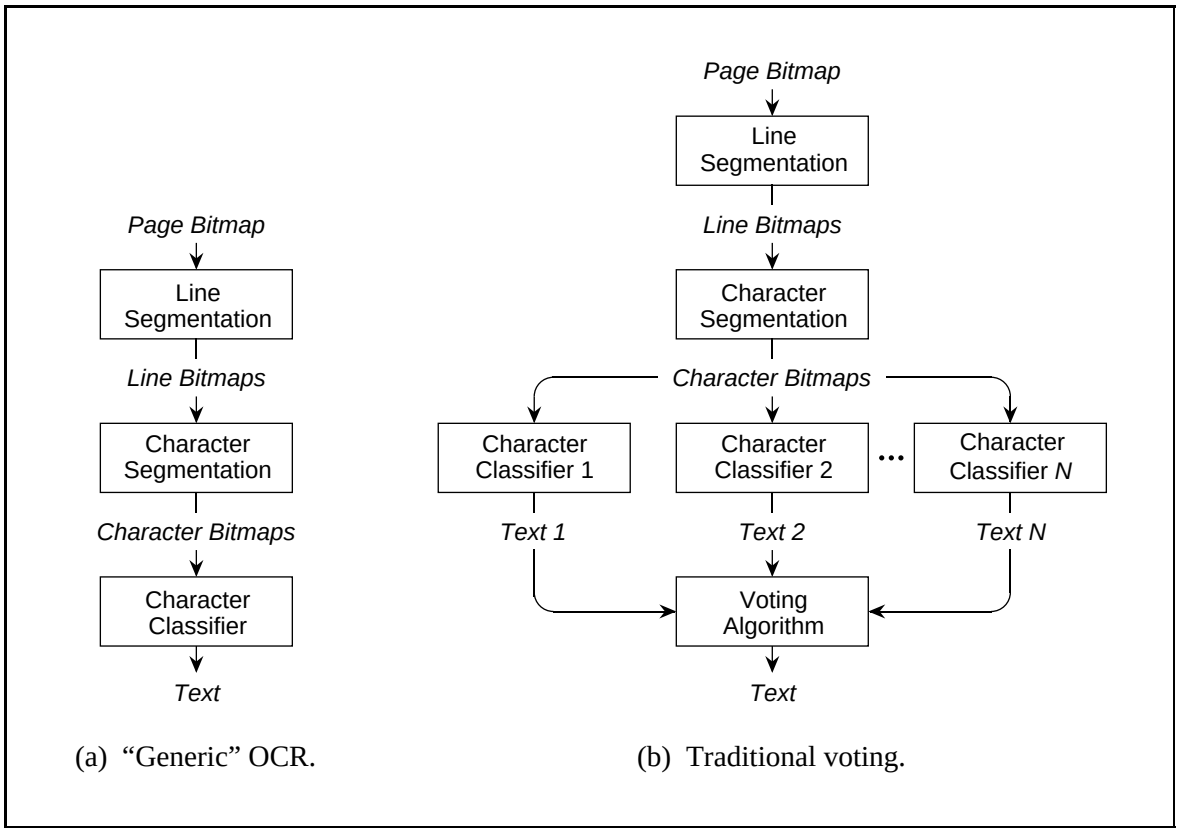

Figure 2: Overview of "generic" OCR and traditional voting procedures.

page in question. In one set of experiments, we found that this process eliminates between 20% and 50% of the errors that would have otherwise arisen.

In the next section we discuss related work. We then present our approach to the OCR voting problem. Section 4 gives empirical 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 common approach is to take directly a set of classifier outputs 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 use 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, 5, 6], etc. These approaches often require a comparable and consistent representation of the decisions produced by individual

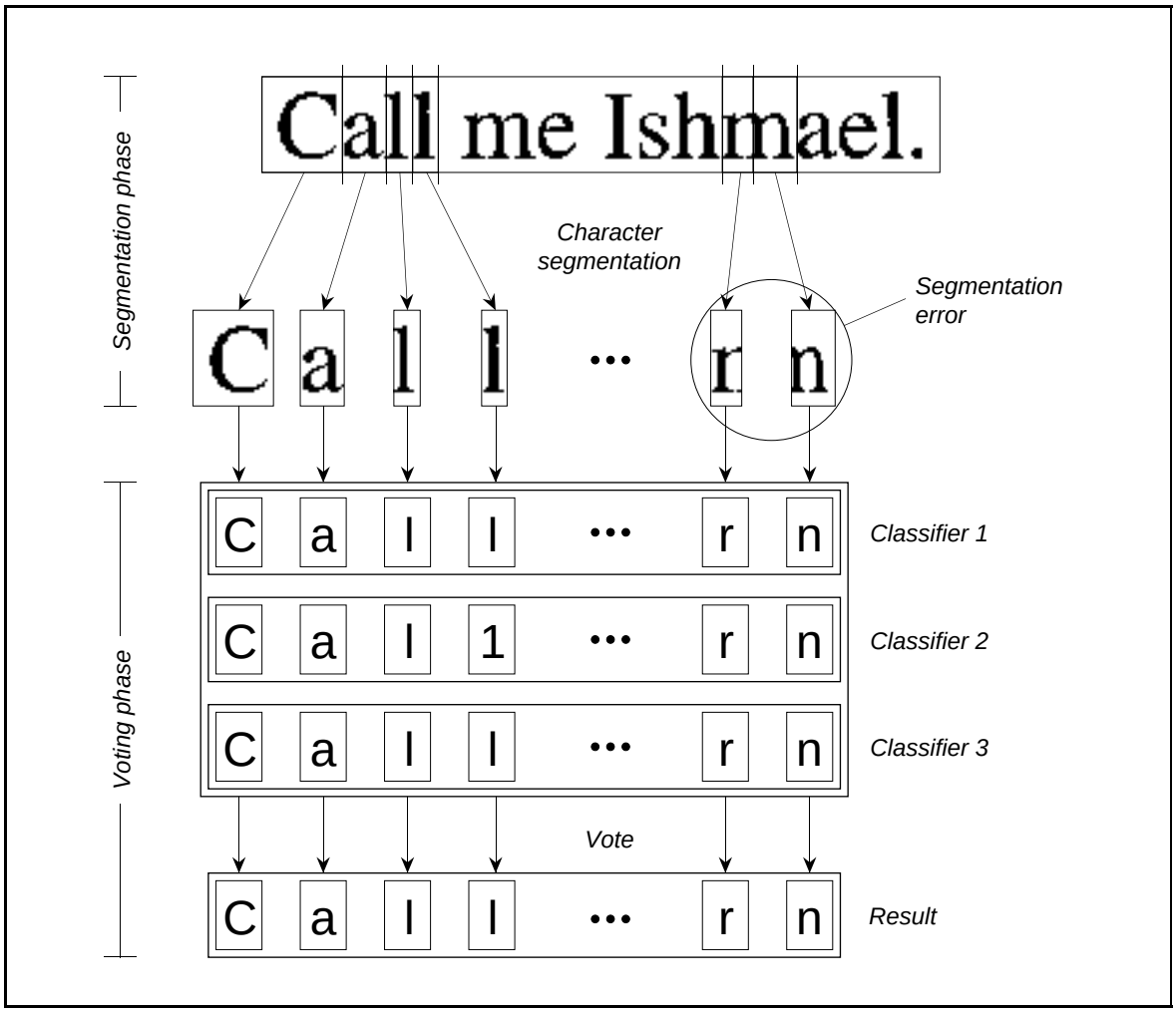

Figure 3: Cases where traditional voting succeeds and fails.

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 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]. Similarly, Handley and Hickey consider the case of merging the outputs generated by three separate OCR packages [12]. These last two approaches are close in spirit to ours, which can be viewed as a generalization of the algorithm and its heuristic to larger numbers of OCR voters, and its application to the outputs produced by a single OCR package when the same page is scanned repeatedly.

3 Consensus Sequence Voting

It has been noted that a large percentage of all OCR errors are due to failures in character segmentation [13]. 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.

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 [14], 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.

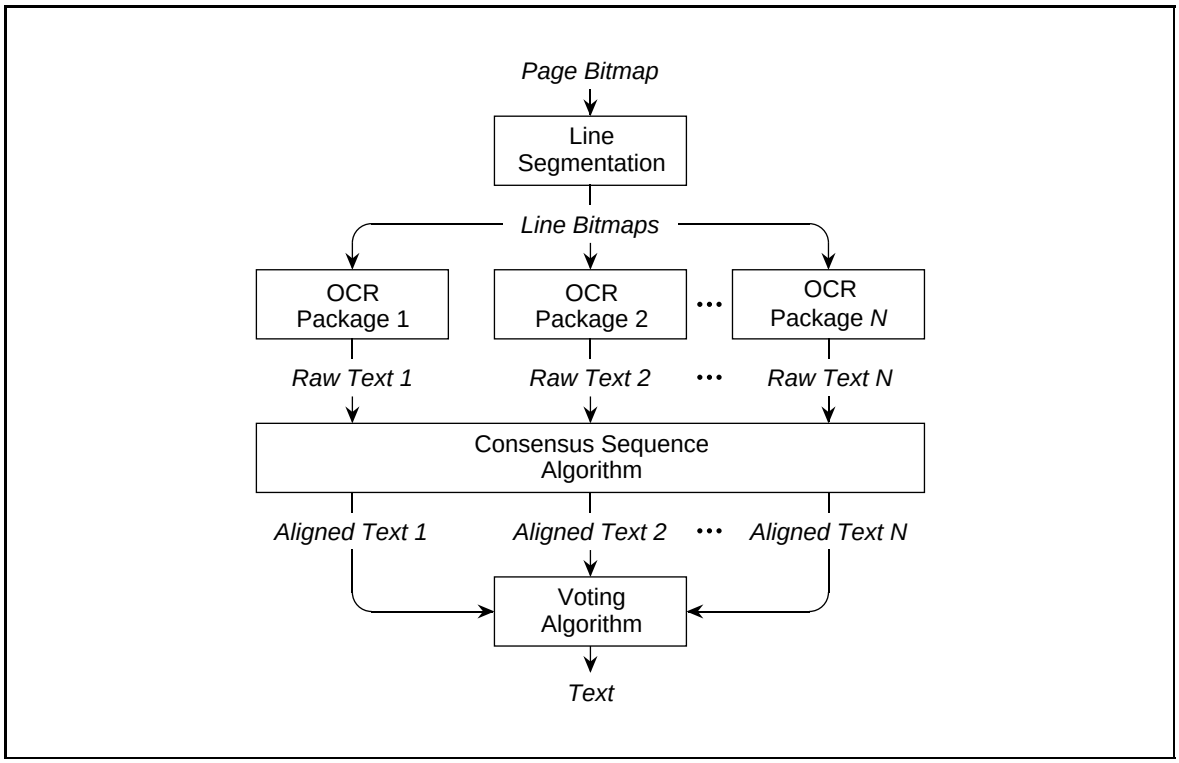

Figure 4: Consensus sequence voting.

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 [15, 16]. 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 ϕ 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 δ 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 Σ', 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 [17]. 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 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 Δ. Then we have:

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

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

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 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. This is illustrated in Figure 5.

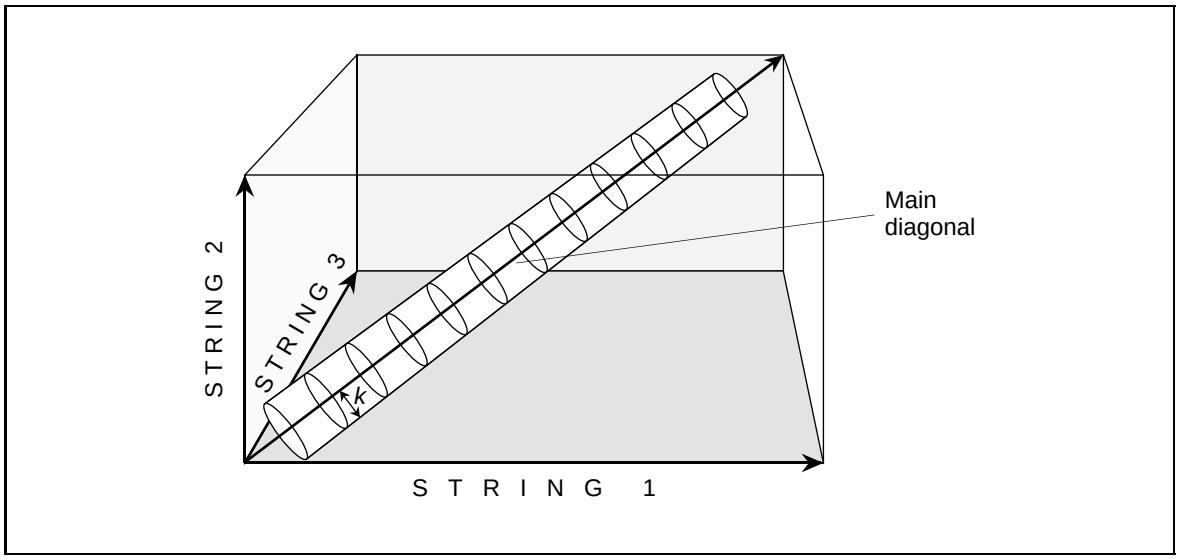

Figure 5: Search space of the heuristic for the case $N = 3$.

This basic concept has been exploited to speed up the two-dimensional version of the problem (*i.e.*, Eq. (1)) in the fields of speech recognition [18] and molecular biology [19]. The extension to multi-dimensional spaces is described in a general way by Carrillo and

Lipman [20], who also present a detailed discussion of a number of the computational issues involved.

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

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

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:

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

When the sequences in question are all similar, a performance guarantee can be made for the heuristic. With some reasonable assumptions about the cost functions ($c_{del} = c_{ins} = 1$ and c_{sub} non-negative), a straightforward result follows immediately from the previous work in the area (*e.g.*, [18, 19, 20]):

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

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

Sketch of Proof: If $\mathcal{D}(R_1, R_2, \ldots, R_N) \leq k$, then we know there can be a total of at most k deletions and insertions in any optimal path (since each such operation has cost 1). These are the only operations that can cause an editing path to stray from the main diagonal. Hence, all optimal paths must lie within a radius k of the diagonal. This corresponds precisely to the portion of the table built by the heuristic. By the data dependencies of the dynamic programming recurrence, all of the intermediate values associated with any optimal path are computed the same way in the heuristic as they are in the original algorithm. Hence, $\widehat{\mathcal{D}}(R_1, R_2, \ldots, R_N) = \mathcal{D}(R_1, R_2, \ldots, R_N)$. □

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 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.

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 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 6. 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 6: Voting example using real OCR data.

4 Experimental Results

In this section we describe two sets of experiments which illustrate 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 distributions 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 50% of the OCR errors.

This point is illustrated by the two images in Figure 7. These nearly identical bitmaps were obtained by scanning the same page twice in rapid succession. Despite their close similarity, one OCR package yielded "part," for the left image and "pan," for the right. This apparent "randomness" in the distribution of OCR errors can be exploited by voting procedures like the one just described.

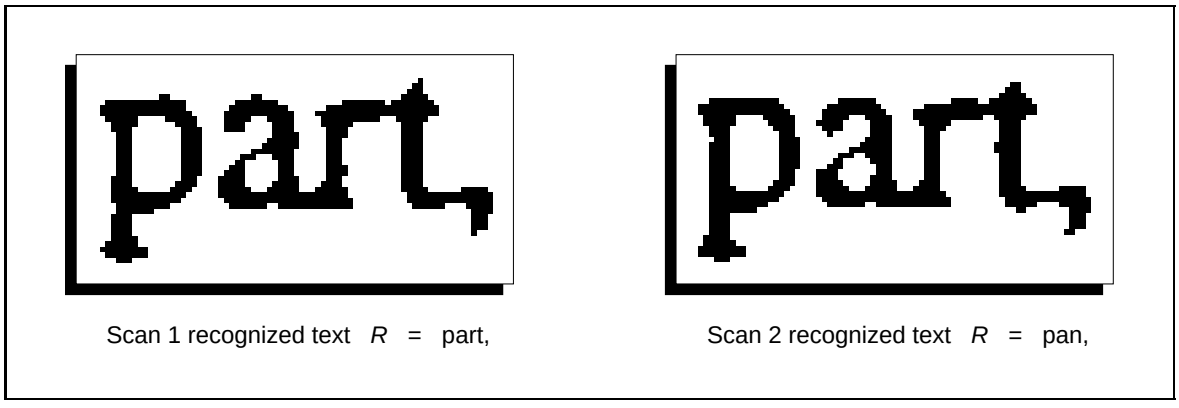

Figure 7: Perturbed inputs yield different results ("part," vs. "pan,").

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. 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, as we make no *a priori* assumptions about the voters.

To test the effectiveness of our voting scheme, we performed two parallel sets of experiments: one using Times font and one using Courier. 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 or Courier and printed on a 400 dpi NeXT laserprinter.

So that we could examine datasets of different qualities, we successively photocopied the source documents a number of times using a Panasonic FP 6070 copier. For the Times experiment, 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. For Courier, we were limited to five generations of photocopies as the sixth was so badly degraded that it caused our OCR package to crash. These 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 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. A default value of $k = 6$ was used. 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, 16].

We begin with the results of our Times experiment. Table 1 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 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 is presented in Table 2. Here we give error counts for the three original Times 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 this data, 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

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

Table 1: Consensus sequence voting – accuracy improvement (Times).

amendable to correction by this method. Still, it is clear that a significant "baseline" error rate can be eliminated through consensus sequence voting.

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

Table 2: Consensus sequence voting – OCR error counts (Times).

We then repeated the same experiment using 10-point Courier. As noted previously, the quality of the sixth generation photocopy was so bad that we were unable to OCR it. However, we observed a comparable improvement due to voting for the remainder of the test data. Table 3 gives recognition accuracies for the three scanned versions, as well as for the corresponding voting results.

Error counts for the three original OCR runs and for the consensus sequence votes are presented in Table 4. As in the Times experiment, 3-way consensus sequence voting performed better than any of the original OCR runs. Note that the voting process was able to correct more than 50% of the errors in the first- and second-generation datasets.

We conclude this section by emphasizing that repeated sampling and consensus sequence voting are two separate notions. The former provides a convenient way for us to evaluate the voting algorithm that is the main topic of this paper. Consensus sequence voting could, of course, be used in any application where the outputs of multiple classifiers must be combined.

Test dataset	OCR1 accuracy	OCR2 accuracy	OCR3 accuracy	Average accuracy	Vote accuracy
g_0	99.8	99.9	99.9	99.9	99.9
g_1	99.6	99.8	99.8	99.7	99.9
g_2	99.3	99.4	99.3	99.4	99.6
g_3	98.6	98.6	98.5	98.6	99.0
g_4	96.2	96.6	96.6	96.4	97.5
g_5	91.8	92.0	91.9	91.9	92.9

Table 3: Consensus sequence voting – accuracy improvement (Courier).

Test dataset	OCR1 errors	OCR2 errors	OCR3 errors	Average errors	Vote errors	Improvement rate
g_0	111	93	102	102	50	0.510
g_1	283	161	175	206	97	0.529
g_2	508	430	498	479	329	0.313
g_3	1,066	1,032	1,115	1,071	777	0.275
g_4	2,806	2,561	2,539	2,635	1,840	0.301
g_5	6,064	5,910	6,016	5,997	5,294	0.117

Table 4: Consensus sequence voting – OCR error counts (Courier).

5 Conclusions

In this paper we have presented a new approach to the problem of voting to correct OCR errors. Based on an algorithm 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 results shows that this technique can correct between 20% and 50% of OCR errors over a range of document qualities.

Consensus sequence voting has some obvious limits. Unlike certain other approaches to post-processing error correction, it is not possible to "synthesize" characters that are completely missing or that have been mis-recognized by all of the voters. While a dictionary-based scheme may be able to determine that "l~zy" should be "lazy" (recall Figure 1), if none of the OCR runs returns an *a* in the proper location, voting cannot correct this error.

Similarly, there was an implicit assumption throughout our presentation that the proper correspondence between the many lines of separate OCR runs is known in advance. If, for example, a run were to add an extra line due to noise in the scanned bitmap, or break one line into two because of document skew, then the input to the consensus sequence algorithm would not be consistent and, hence, the results would be unpredictable. This particular issue can be addressed by incorporating another level of matching in the process, one which corresponds lines in the same way the algorithm we have described corresponds

characters [12].

The notion that sampling the same input repeatedly can lead to better recognition results is interesting in its own right. In another paper, we proved that by relaxing a basic assumption in statistical pattern recognition – that the input be sampled only once – it becomes possible to build classifiers that "beat" the Bayes error bound [21]. This result is quite general and appears to have applications beyond optical character recognition.

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.

6 Acknowledgements

The authors would like to thank Dr. Henry Baird for his helpful comments on an earlier version of this paper. The *Moby-Dick* text we used in our experiments was obtained from the Guttenberg Project at the University of Illinois, as prepared by Professor E. F. Irey from the Hendricks House edition. The trademarks mentioned in this paper are the property of their respective companies.

References

[1] C. Nadal, R. Legault, and C. 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, Atlantic City, New Jersey, June 1990.

[2] M. Sabourin, A. Mitiche, D. Thomas, and G. Nagy. Classifier combination for hand-printed digit recognition. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 163–166, Tsukuba Science City, Japan, October 1993.

[3] T. K. 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, Tsukuba Science City, Japan, October 1993.

[4] T. K. Ho, J. J. Hull, and S. N. Srihari. On multiple classifier systems for pattern recognition. In *Proceedings of the 11th International Conference on Pattern Recognition*, pages 84–87, The 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 C. Y. Suen. Combination of multiple classifiers with measurement values. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 598–601, Tsukuba Science City, Japan, October 1993.

[7] T. K. Ho, J. J. Hull, and S. N. Srihari. Word recognition with multi-level contextual knowledge. In *Proceedings of the First International Conference on Document Analysis and Recognition*, pages 905–915, St. Malo, France, October 1991.

[8] B. Plessis, A. Sicsu, L. Heutte, E. Menu, E. Lecolinet, O. Debon, and J. V. Moreau. 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, Tsukuba Science City, Japan, October 1993.

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

[10] S. V. Rice, J. Kanai, and T. 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] V. P. Concepcion and D. P. D'Amoto. Synchronous tracking of outputs from multiple OCR systems. *SPIE Character Recognition Technologies*, 1906, 1993.

[12] J. C. Handley and T. B. Hickey. Merging optical character recognition outputs for improved accuracy. In *Proceedings of the RIAO Conference on Intelligent Text and Image Handling*, pages 160–174, Barcelona, Spain, April 1991.

[13] 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, Tsukuba Science City, Japan, October 1993.

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

[15] J. Esakov, D. P. Lopresti, and J. 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, San Jose, CA, February 1994.

[16] J. Esakov, D. P. Lopresti, J. S. Sandberg, and J. Zhou. Issues in automatic OCR error classification. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, pages 401–412, Las Vegas, NV, April 1994.

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

[18] L. R. Rabiner, A. E. Rosenberg, and S. E. Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-26(6):575–582, December 1978.

[19] J. W. Fickett. Fast optimal alignment. *Nucleic Acids Research*, 12(1):175–179, 1984.

[20] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082, October 1988.

[21] J. Zhou and D. Lopresti. Repeated sampling to improve classifier accuracy. In *Proceedings of the IAPR Workshop on Machine Vision Applications*, pages 346–351, Kawasaki, Japan, December 1994.