W. S. Rosenbaum
J. J. Hilliard

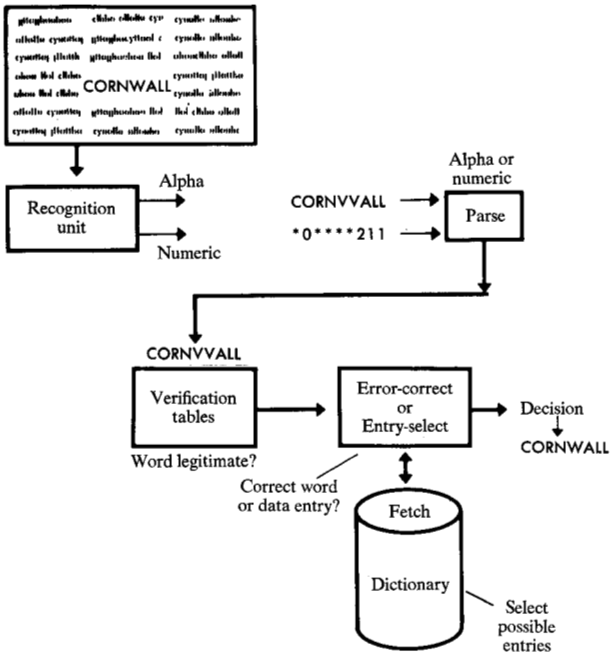
Multifont OCR Postprocessing System

Abstract: A series of techniques is being developed to postprocess noisy, multifont, nonformatted OCR data on a word basis to 1) determine if a field is alphabetic or numeric; 2) verify that an alphabetic word is legitimate; 3) fetch from a dictionary a set of potential entries using a garbled word as a key; and 4) error-correct the garbled word by selecting the most likely dictionary word. Four algorithms were developed using a technique called vector processing (representing alphabetic words as numeric vectors) and also by applying Bayes maximum likelihood solutions to correct the OCR output. The result was the development of a software simulator which processed sequential fields generated by the Advanced Optical Character Reader (in use by the U.S. Postal Service in New York City), performed the four functions indicated above, and selected the correct alphabetic word from a dictionary of 62000 entries.

1. Introduction

From its technical debut, the Optical Character Reader (OCR) has had unique potential for text processing applications. Its input processing rate far exceeds that of key punch/typewriter input and its output is in machine-readable form, unlike that of non-coded information facsimile scanning. Despite these very important attributes, OCRs have made only minor inroads to the overall text processing operation. A large part of the reluctance towards OCR utilization in publication-associated industries can be traced to the highly constrained fonts and formats in which the present generation OCRs must operate.

Figure 1 Multifont OCR postprocessing system.



When multifont nonformatted optical character recognition is attempted, problems that are not as prevalent in unifont OCRs become evident. They stem from the highly error-prone character recognition environment that is created when OCR operation is performed over many different alphabetic and numeric fonts with minimum control exercised over text conventions and typographic print quality. In scanning such text, discrimination between confusable character geometries causes a nominal five percent character recognition error rate. To cope with such character recognition reliability problems, a series of cybernetic error correction procedures, referred to as postprocessing, is required as an integral portion of the multifont nonformatted OCR system.

A postprocessing technology has been developed which has the error correction power and flexibility required to support multifont, nonformatted, OCR text processing applications. The overall cybernetics design is termed Contextual Word Recognition. The Contextual Word Recognition Postprocessor is driven by an OCR character recognition stream of the type generated by the Advanced Optical Character Reader (AOCR). The AOCR is a multifont OCR scanner which has been developed by the IBM Federal Systems and System Development Divisions under contract to the United States Postal Service. The optical character recognition capability of the system enables it to handle a wide range of type styles (fonts) and print quality. Character recognition is performed in a dual channel mode, simplifying the

Table 1 Effect of imperfect input.

Condition	*Result*	*Example*
1. Poor print quality 2. Multiple fonts	5–10% Reject or misread rate Mis-segmentation of characters	B → E W → V*
3. Unformatted information	Indeterminate alphameric fields Blank or space errors	7354 → IBSA
4. Excessive number of dictionary entries	Invalid entry selection Large computational requirement	*ALE → DALE, HALE, MALE, PALE, SALE, VALE, YALE

basic character recognition logic by allowing separate and independent alphabetic and numeric depositions to be made for each character's video image (see Section 2).

In the last two years, the OCR error correction capability of the Contextual Word Recognition Postprocessor (CWRP) has been extensively tested using actual AOCR mail character recognition tapes. From a heuristic standpoint, the results obtained from postprocessing mail address data should be readily extrapolatable to OCR nonformatted multifont operation on general text/word processing applications. The problems noted in Table 1 are germane to both applications. To substantiate these performance projections, and to possibly enhance performance characteristics of the CWRP, a program called Optical Character Word Processing Simulation (OCWPS) is now underway to apply the postprocessing algorithms to general multifont nonformatted text.

Contextual word recognition postprocessing refers to the series of cybernetic operations that are performed on the OCR data stream at one level above character recognition, called the word level (or subfield level, see Section 2). By working at the word level, certain inferences and error rectifications are possible which would not be feasible at the character level. The whole postprocessing operation can be conceptualized schematically as a procedure that answers four questions concerning each subfield of OCR data (Fig. 1). In a serial manner these questions may be posed as

1. Is the subfield alphabetic ("alpha") or numeric (detailed in Section 2)?
2. If alphabetic, does the character string contain any errors (detailed in Section 3)?
3. If garbled, what segment of the error correction dictionary should be searched to find the originally scanned version of the garbled word (detailed in Section 4)?
4. Which entry in the dictionary segment that has been fetched is the word that was scanned by the OCR and garbled into its present form (detailed in Section 5)?

The remainder of this paper explains in detail the algorithms used in the CWRP to accomplish each of the preceding OCR data manipulations. Each algorithm is discussed in terms of the functions it performs in a text processing, multifont, nonformatted OCR environment. Because the results of the previously mentioned OCWPS study are not yet complete, the statements related to specific algorithm performance capability reflect results previously obtained from those aspects of the postal problem that are the most analogous to the text processing application.

• *Frequently used abbreviations*

AOCR	Advanced Optical Character Reader
AWVR	Alpha Word Vector Representation
BOND	Bayesian Online Numeric Discriminant
BRM	Binary Reference Matrix
CRS	Crowding Segmentation
CS	Catenation Segmentation
CWRP	Contextual Word Recognition Postprocessor
DAP	Dictionary Access Point
FLP	First/Last Position (fetch discriminant)
HSS	Horizontal Splitting Segmentation
OCR	Optical Character Reader
OCWPS	Optical Character Word Processing Simulation
RCML	Regional Context Maximum Likelihood (error correction procedure)
VF	Vector Fetch
WG	Word Group

2. Alphameric subfield discrimination

The first postprocessing function performed on the raw recognition stream output is alphameric subfield discrimination. In pursuing the design of a nonformatted multifont character recognition capability, we have a rather unique "dual channel" recognition processing philosophy to maintain cost-effective recognition logic design. This implies that two separate recognition "channels" are implemented. In one channel all characters are pre-

Table 2 AOCR dual recognition streams.

	Line 1	*Line 2*	*Line 3*
Alpha	AARON BAKERS	SISO PAGE BL	SAINT LOUIS MO **11*
Numeric	4*80* 8466*5	5150 8466 8*	5*1** *001* * 0 63113

Table 3 Geometrically similar alphabetic and numeric characters.

Alpha	O	i	l	Z	S	T	B
Numeric	0	1	1	2	5	7	8

Table 4 Common OCR misinterpretations.

Alpha word	*Numeric interpretation*
SOUTH	80478 or 804TH
THIRD	781RD
FIFTH	01078 or 010TH

sumed to be alphabetic only and an appropriate alpha identification (or reject) is assigned to each character scanned (including the numerals). With respect to the second channel, all characters are presumed to be numeric and an appropriate numeric identification (or rejection) is assigned to each character scanned (including the alphabetic symbols). As a result, during the recognition stage of OCR processing, the acceptance of a character as a 0, for example, is not contingent on discerning that the scan is not that of an alphabetic O, C, or U. Typical output of the AOCR recognition unit is shown in Table 2.

In viewing Table 2, it may appear that the dual channel recognition approach has only accomplished transferring the alphameric ambiguity from the character scan level up to the subfield level. (A subfield is any set of contiguous characters preceded and followed by a blank space.) The remainder of this section addresses the resolution of this problem of alphameric subfield discrimination in a generalized omnifont text processing environment. An analytic procedure is developed and the results of its use in the complex postal address analysis problem are discussed.

Alphameric subfield discrimination procedure

Although seemingly trivial, reliable discrimination between alpha and numeric subfields in a multifont character recognition environment is a very complex process.

Discrimination between alpha and numeric characters has always been one of the most difficult problems in the design of omnifont character recognition equipment. The difficulty stems from the fact that the Roman and Arabic character sets, to which the alphas and the numerics respectively relate, were generated independently with no concern for avoiding mutual confusion. Hence, over commonly used fonts, they share many of the same basic geometric shapes as shown in Table 3.

The alphameric discrimination problem on the character recognition level is reflected at the subfield level during postprocessing. Many common alpha words can be recognized in part or in total as numeric subfields. Some common misinterpretations observed in the postal address problem are shown in Table 4. The converse also holds potentially for many numeric subfields.

The crux of the processing problem in numeric subfield discrimination is that real or aliased numeric character strings do not lend themselves to methods of direct contextual analysis. A numeric subfield is completely nonredundant, implying that any set of digits creates a meaningful data set.

Bayesian online numeric discriminant procedure

When analyzing the dual recognition streams which are comprised by the output from the AOCR, simple rules, such as determining the preponderance per subfield of alphabetic or numeric recognitions, are useful but are not sufficient for subfield alphameric genre decisions. For about 20 percent of the subfields, the recognition quality of the subfield is the same for both channels or not sufficiently different to allow reliable decision. For example,

a. Alpha channel SISO PAGE BL – indeterminate alpha subfield

b. Numeric channel 5150 8466 8* – indeterminate numeric subfield

In these instances where a recognition quality differential of at least two reject characters does not exist, the Bayesian Online Numeric Discriminant (BOND) procedure is used.

The BOND procedure seeks to achieve alphameric inference capability by associating with a numeric subfield a form of quasi-redundancy. Redundancy in a con-

textual sense means that dependencies exist between the presence of one character and the presence of another. Normally, contextual redundancy is thought of in a horizontal sense – that is, between characters on a line, within a word. An example of this concept is digram statistics. These probabilities of character juxtaposition combinations allow the projection of likely succeeding characters from knowledge of the preceding one. For example, given the alpha string SPRI-G, N would be chosen over Z to fill the blank position. Mathematically, this takes the form of the conditional probability

$$P(a_k|a_i), \tag{1}$$

where a_i is observed and a_k is projected as a possible following character. The value of (1) relates to the compatibility of the $a_i a_k$ character pair with respect to English text.

Clearly no analog-to-contextual redundancy in the form of digrams exists with respect to numeric subfields.

Although redundancy of the horizontal form does not exist for numeric subfields, redundancy of a special "vertical" nature can be induced by virtue of the AOCR recognition environment.

Alpha channel SIOUX FALLS SD S *IOI
Numeric channel 5100* 56**5 50 5 7101
} vertical redundancy.

As shown previously, for each character scanned the AOCR creates independent outputs, the attempted alpha and numeric recognitions. Characteristic of this type of dual recognition system are

1. Each legitimate numeric character is misrecognized by the alpha recognition channel as a reject or as one of a specific set of alphas. (For example, a 2 is often read in the alpha channel as a Z.)
2. Each legitimate alpha character is misrecognized by the numeric recognition channel as a reject or as one of a specific set of numerics. (For example, an S is often read in the numeric channel as a 5.)

A concept of vertical redundancy can be evolved that associates the recognition of a character in one channel with one of a set of misrecognitions possible in the other channel. This can be formulated as the conditional probability

$$P(a_i|n_j) \tag{2}$$

that, given the scanned numeric character n_j, the alpha recognition channel has misrecognized it as a_i. The converse conditional probability statement,

$$P(n_j|a_i), \tag{3}$$

is the related probability that, given the alpha character a_i, the numeric recognition channel has misrecognized it as n_j. Probabilities (2) and (3) are referred to as Channel Confusion Probabilities and are denoted formally as

$$P_{cc}(a_i|n_j) \text{ and} \tag{4}$$
$$P_{cc}(n_j|a_i). \tag{5}$$

Analysis of AOCR machine performance indicator data readily yields the complete set of channel confusion probabilities as they relate to numerics (Table 5) and to alphas (Table 6). The inference potential of these statistics is enhanced by computing them independently with respect to uppercase and lowercase alpha characters and to the various conflict and reject characters.

Using the machine performance indicator data bases, one can proceed to implement the BOND procedure. The subfields dealt with are those whose dual channel recognition output is indeterminate with respect to the reject character criterion. The BOND procedure seeks to discriminate alpha and numeric subfields on the basis of their Bayesian likelihood factors. This implies that the output of both channels is assessed from the perspective

$$P(\text{alpha read}|\text{numeric read}) \text{ and} \tag{6}$$
$$P(\text{numeric read}|\text{alpha read}). \tag{7}$$

Expression (6) is the probabilistic statement that assesses the compatibility of the alpha channel recognition output with the assumption that a numeric subfield has been scanned. Expression (7) evaluates the converse, that is, the compatibility of the numeric channel recognition output with the assumption that an alpha subfield has been scanned. Expressions (6) and (7), for computational purposes, can be expressed in terms of products of channel confusion probabilities. Hence,

$$P(\text{alpha read}|\text{numeric read}) = \prod_{j=1}^{k} P_{cc}(a_j|n_j) \tag{6a}$$

$$P(\text{numeric read}|\text{alpha read}) = \prod_{j=1}^{k} P_{cc}(n_j|a_j), \tag{7a}$$

where k is the number of characters in the subfield. In this perspective, a subfield's alpha or numeric genre stands out as the quotient of the ratio of Eq. (6a) to Eq. (7a). That is,

$$\phi = \prod_{j=1}^{k} P_{cc}(a_j|n_j) \Big/ \prod_{j=1}^{k} P_{cc}(n_j|a_j), \tag{8}$$

where $\phi \leq 1$ implies alpha and $\phi > 1$ implies numeric.

The inference inherent in the formulation of Eq. (8) results from the ratio of Bayesian likelihood factors. No syntactic or contextual information is utilized in the preceding alphameric discrimination processes. Hence, the validity of the BOND procedure with respect to general OCR application can be nominally assumed.

Using raw recognition stream data (tapes) resulting from mail processed by the AOCR, the BOND pro-

Table 5 Channel confusion statistics for numerics–values of $P_{cc}(a_j|n_i)$ in percent for each numeric read total.

Alpha read	*Numeric read*										*Numeric reject*
	0	1	2	3	4	5	6	7	8	9	*
A	0	0	1.237	0	.625	0	0	0	1.612	0	6.667
B	0	0	.336	3.553	0	.654	0	0	25.806	0	1.333
C	1.153	0	0	0	0	0	0	0	0	0	1.333
D	.576	0	0	0	0	0	0	0	0	0	0
E	0	0	.336	.508	0	.980	3.297	0	4.839	0	5.333
F	0	0	0	0	0	0	0	0	0	0	2.667
G	0	0	0	1.015	0	0	1.099	0	1.613	22.727	0
H	0	0	0	0	1.250	0	0	0	0	0	0
I	0	12.925	0	0	0	0	0	0	0	0	1.333
J	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0
L	0	61.565	0	0	.625	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	1.333
N	0	0	0	0	0	0	0	0	0	0	0
O	92.795	0	0	0	0	0	1.099	0	0	0	2.667
P	0	0	8.389	0	0	0	0	0	0	0	1.333
Q	.288	0	0	0	0	0	0	0	0	0	0
R	0	0	.671	0	0	0	0	.877	1.613	0	5.333
S	0	0	0	.508	0	74.183	1.099	0	6.452	2.273	6.667
T	0	1.361	0	0	1.250	0	0	2.632	0	0	2.667
U	.865	0	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	1.333
X	0	0	0	0	0	0	0	0	0	0	1.333
Y	0	0	0	0	0	0	0	2.632	0	0	1.333
Z	0	0	16.779	0	0	0	0	0	0	0	1.333
Alpha reject	3.746	8.503	72.148	93.909	38.750	23.856	93.407	.877	58.065	70.455	42.667
Conflict i/l	0	11.565	0	0	0	0	0	0	0	0	0
Conflict l/i	0	0	0	0	0	0	0	0	0	0	0
Conflict N/W	.576	4.082	0	.508	57.500	.327	0	92.982	0	4.545	13.333

cedure has been extensively tested. An alphameric subfield discrimination correctness rate of 99.6 percent has been achieved in these offline simulations. There is no apparent reason to assume that this same level of accuracy would not hold in a general text processing application.

Figure 2 Example of alphameric discrimination using the BOND calculation.

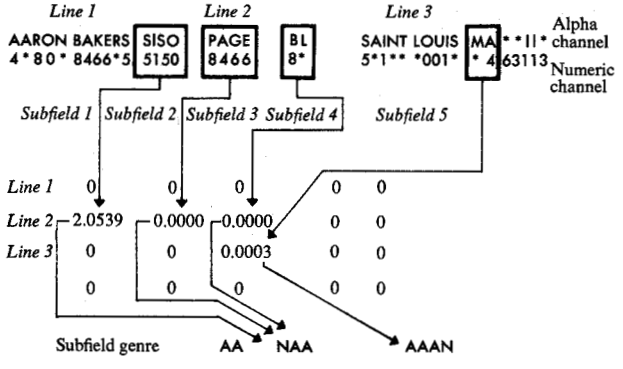


Figure 2 is a copy of the BOND procedure output for a typical AOCR read. The step-by-step calculations related to the first two BOND quotients are shown in Table 7.

3. Verification

In the Contextual Word Recognition Postprocessor, OCR word verification is formed by means of the Binary Reference Matrix (BRM). The BRM approach was conceived as a highly efficient, low-storage-requirement mode of validating whether a word scanned by the OCR has been read correctly, i.e., without misread characters. This function must be performed for each subfield that has been identified as alphabetic by the BOND procedure (Section 2). Logically, the BRM must contain a representation, in some manner, of all words that might be anticipated in documents scanned by the OCR. This "scanning vocabulary" may, at times, be even broader than the ordinary dictionary. Therefore, conventional storage, access, and search techniques against the OCR vocabulary may not be acceptable, particularly in a real-

Table 6 Channel confusion statistics for alpha-values of $P_{cc}(n_i|a_j)$ in percent for each numeric read total.

Alpha read	*Numeric read*										*Numeric reject*
	0	1	2	3	4	5	6	7	8	9	*
A	.852	2.699	3.977	0	36.932	.142	1.563	0	6.108	0	47.727
B	14.286	0	0	0	0	0	0	0	57.143	0	28.571
C	86.667	0	1.667	0	0	0	0	0	0	0	11.667
D	77.481	0	.763	.382	.763	0	.382	0	.763	0	19.466
E	.474	0	1.502	.158	.079	7.510	30.514	.079	10.119	.158	49.407
F	0	2.564	0	0	0	26.923	0	1.282	5.128	0	64.103
G	13.953	0	0	2.326	0	2.326	53.488	0	2.326	4.651	20.930
H	0	0	0	0	.515	.515	21.649	0	62.887	0	14.433
I	0	94.298	0	0	0	0	0	0	0	0	5.702
J	0	0	0	0	0	0	0	0	0	0	0
K	0	0	.361	0	7.762	0	29.061	0	1.986	0	60.830
L	0	33.898	1.695	0	1.695	0	0	0	0	0	62.712
M	0	0	0	.556	6.111	1.111	.556	0	7.778	0	83.333
N	8.353	.232	0	.077	1.392	.232	.541	.155	.619	0	88.399
O	98.222	0	0	0	.148	0	0	0	0	0	1.630
P	0	0	1.316	0	0	0	0	0	76.316	0	22.368
Q	0	0	0	0	0	0	0	0	0	50.000	50.000
R	.501	2.003	2.504	.167	0	.334	.501	.334	37.563	0	56.093
S	0	0	0	.379	0	67.803	.189	0	1.326	4.545	25.758
T	0	30.732	0	0	0	0	2.707	27.548	.478	0	38.535
U	69.444	0	0	0	0	0	0	0	0	0	30.556
V	.263	0	0	0	5.000	0	0	29.211	0	.263	65.263
W	0	0	0	0	11.015	.432	0	5.616	1.080	.216	81.641
X	0	6.897	0	0	6.897	0	0	0	0	0	86.207
Y	0	2.775	0	0	12.950	0	0	3.392	0	.103	80.781
Z	0	0	0	0	0	0	0	0	0	0	100.000
Alpha reject	12.775	10.132	1.762	.881	1.762	4.405	5.286	1.322	2.643	1.322	57.709
Conflict i/l	0	97.561	0	0	0	0	0	0	0	0	2.439
Conflict l/i	0	100.000	0	0	0	0	0	0	0	0	0
Conflict N/W	4.779	3.309	0	0	6.985	.368	1.471	3.676	3.676	.368	75.000

time application. The goal of the verification technique is to minimize storage and search time for a large dictionary associated with an OCR application. The work discussed in this paper relates to verification techniques operating with a verification word list of more than 15 000 English words of length averaging eight alphabetic characters. Under the BRM methodology, the preceding word list has been stored for verification purposes using only 10 000 bytes of storage. (Conventional alpha character EBCDIC representation would have required 120 000 bytes to store the same word list.)

• *Alpha word vector representation*

The BRM is a specialized application of the Alpha Word Vector Representation (AWVR) technique. The mechanics of this technique are shown in Table 8.

The underlying rationale of AWVR is that any word or character string can be mapped into a vector representation by assigning a unique numeric value to each letter in the alphabet. One of the most direct and intuitive assignment schemes would designate A = 1, B = 2,

Table 7 Example of the BOND calculation.

Subfield	1	2	3
Alpha channel	SISO	PAGE	BL
Numeric channel	5150	8466	8*

Subfield 1

$$\text{BOND} = \frac{P(S|5)\ \ P(I|1)\ \ P(S|5)\ \ P(O|0)}{P(5|S)\ \ P(1|I)\ \ P(5|S)\ \ P(0|O)}$$

$$= \frac{(74.2)\ \ (61.6)\ \ (74.2)\ \ (92.8)}{(67.8)\ \ (33.9)\ \ (67.8)\ \ (98.2)} = 2.0539$$

Result greater than 1 implies numeric field

Subfield 2

$$\text{BOND} = \frac{P(P|8)\ \ P(A|4)\ \ P(G|6)\ \ P(E|6)}{P(8|P)\ \ P(4|A)\ \ P(6|G)\ \ P(6|E)}$$

$$= \frac{(0.001)\ \ (0.6)\ \ (1.0)\ \ (3.3)}{(76.3)\ \ (36.9)\ \ (53.5)\ \ (30.5)} = 0.000$$

Result less than or equal to 1 implies alpha field

Table 8 Alpha word vector representation (AWVR) methodology (A = 1, B = 2, $\cdots$, Z = 26).

Step 1
Vector mapping
CORNWALL → (3, 15, 18, 14, 23, 1, 12, 12,)
Step 2
Vector attributes
(3, 15, 18, 14, 23, 1, 12, 12) → (magnitude, angle)
Magnitude = function of characters in word

$$= \sum_{N=1}^{M} L_n^2 = (3)^2 + (15)^2 + (18)^2 + (14)^2 + (23)^2 + (1)^2 + (12)^2 + (12)^2 = 1572 \equiv Y$$

Angle = function of character position

$$= \sec^{-1}\left(\frac{Y|\mathbf{R}|}{\Sigma N L_N}\right) = 83.7392 \text{ degrees}$$

Table 9 Binary reference matrix.

Angle range ⟶

Magnitude ↓

Byte number	Bit position							
	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
7403	0	1	0	0	0	0	1	0
7404	0	0	0	1	0	0	0	0
7405	0	1	0	0	0	1	0	0
9997	0	0	0	0	0	0	0	0
9998	0	0	0	0	0	0	0	0
10000	0	0	0	0	0	0	0	0

C = 3, $\cdots$, Z = 26. Any vector representation of a word so generated would, in turn, be uniquely reconstitutable in terms of the linear algebraic vector attributes of magnitude and angle, where the magnitude reflects the word character content, and the angle reflects the relative positioning of characters within the word relative to a reference vector **R**. A suitable form for the reference vector is $\mathbf{R} = (\sqrt{2}, \sqrt{3}, \sqrt{5}, \cdots)$.

It should be noted at this point that by just using the magnitude-angle representation, an alpha word of any length can be represented uniquely in only four bytes of storage.

• *Generation of the binary reference matrix (BRM)*

The ability to transform an alpha word list into its vectorial image may be looked upon as the initial phase of BRM generation. Next, it is necessary to use the vector representation in an efficient manner for verification. The BRM itself is the array that results when "legal" magnitude-angle combinations are mapped into a storage table matrix. This, in essence, allows further compaction of what in its vectorial form was already a highly compact version of the original alpha word list. The BRM is therefore a logical arrangement of storage, which associates a magnitude value and an angle segment range with each bit position. The row dimension of the BRM relates to the range of possible magnitude values that can be generated from the legal word list. Each column bit position relates to a segment of the range of angle that the same words can similarly generate. Hence, the existence of a legal word is denoted by turning on a bit position that contains the angle value of the word in the row corresponding to its magnitude. This process and the resulting core configuration are shown schematically in Table 9.

Verification of an OCR word read is accomplished by accessing the bit position in the BRM corresponding to the magnitude and angle indicated by the read. The word would be considered correct (ungarbled) if the related BRM bit position were ascertained to be in the "on" position. The computer operations required to achieve this verification can easily be accomplished within a real-time constraint, especially because the storage dimensions of the BRM make it conveniently core-storable.

Clearly, the BRM will verify the existence of any correctly read word. However, special considerations must be taken into account to allow the BRM to perform its associated task of garbled (erroneous) word discrimination. The high degree of data compaction achieved in the BRM has incurred a decrease in the uniqueness with which a word's vector mapping can be represented. It will be recalled, initially, that each vector mapping of a word–by algebraic definition–yields a unique magnitude-angle data set. The discrete integer data lend themselves well to being isomorphically mapped into the respective row designations of the BRM (Table 10). However, the angle data, which originally took the form of a continuum (nonintegral values), cannot be so directly accommodated in the BRM configuration.

To allow representation in a BRM, the angle data must be quantified into range segments compatible with the limited number of row entries offered by a bit string of reasonable length. This causes the angle part of the vector mapping scheme to have a degree of non-uniqueness associated with it in the BRM representation. Unless certain analytical safeguards are taken, the ambiguity associated with angle may compromise the BRM's error word discrimination potential. This would make the BRM unable to discern and discriminate those garbled words which have generated, by chance, a valid magnitude and come sufficiently close to a valid angle value to access the same BRM bit position as a valid word. This possibility can never be precluded entirely; it can, however, be made negligibly small by setting up the BRM to take full advantage of the sparse areas of the matrix.

Sparsity can be considered almost synonymous with BRM error-word discrimination potential. The basic idea of sparseness is to take advantage of the fact that the BRM contains many more empty (0) positions than occupied (1) positions. Logically, it follows that the greater the sparseness the less likely the false verification of error words and therefore the greater the error discrimination potential of the BRM methodology. The following strategy is used to exploit the sparseness of the BRM.

• *Specialization of the BRM vector numbering scheme* The alphameric equivalency scheme used to map the valid word list into a vector representation, which in turn is synthesized into the BRM, takes advantage of the known dictionary and OCR misread characteristics. With a properly chosen scheme, one can maximize the potential that, when an error occurs, the word falsely generated by the OCR will be rejected as invalid by the BRM. To accomplish this, there are two general restrictions which must be placed on the numbering scheme: 1) The numbering scheme must be chosen such that the density of the matrix is not uniform, and a continuous, sparse area of the matrix is identifiable; and 2) the numbering scheme must be chosen such that invalid words generate magnitude-angle representations that are located in the sparse area of the matrix.

Restriction 1 To some degree, the generation of magnitude itself produces a nonuniformity in the BRM with identifiable areas of sparsity. As an example, Fig. 3(a) shows the magnitude density function for all combinations of eight-character fields in which each of the 26 characters has an equal probability of occurrence. Magnitude values cluster toward the center of the range with sparse areas toward the low and high magnitudes. However, words in the English language do not have uniform

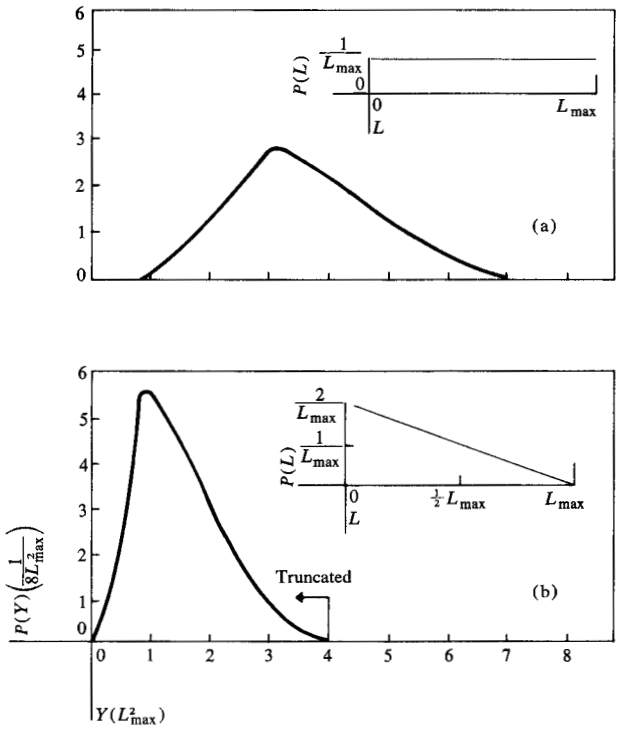


Figure 3 Density function for the magnitude of eight-character fields: (a) L is uniformly distributed $[P(L) = 1/L_{max}]$ for general fields: and (b) $P(L) = 2(1 - L/L_{max})/L_{max}$, as shown, for English words.

Table 10 Sample binary reference matrix using 10 kilobytes of storage; conventional storage would require 120 kilobytes.

Angle 0°												90°	
0	1	1	0	1	0	1	1	0	1	1	0	1	Examples
0	0	1	0	1	1	0	0	1	1	0	0	0	
1	0	0	1	1	0	1	1	0	1	1	1	1	
1	1	1	0	1	1	1	0	(1)	1	0	0	0	WALL
0	0	1	1	1	1	1	0	0	1	1	1	0	
1	0	0	1	1	1	0	0	1	1	1	1	1	
0	0	1	1	1	0	1	1	1	0	0	0	0	
0	1	0	0	1	1	0	0	0	0	(1)	1	1	
0	0	0	1	0	1	0	1	0	1	1	0	0	
1	1	1	1	0	1	0	0	0	1	0	0	0	PINE
1	0	0	0	0	0	1	1	0	0	0	1	1	
0	0	0	0	1	0	0	1	0	0	0	0	0	
0	1	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	1	0	0	0	0	0	0	0	0	1	
0	0	0	0	0	1	0	0	1	0	0	0	0	

Magnitude

character usage. Rather, character usage varies from approximately 10 percent (e) to as little as 0.1 percent (q). By assigning numerical values to characters in order

inverse to their probability of occurrence, the density function can be substantially shifted so that the lower magnitude portion of the matrix has the higher density, with the higher magnitude values becoming progressively more sparse.

For example, if the characters are ordered according to occurrence frequency and are assigned numerical values in sequence starting with 1, the resulting density function can be approximated as

$$P(L) = \frac{2}{L_{\max}}\left(1 - \frac{L}{L_{\max}}\right).$$

When this density function is transformed by the magnitude function $Y = \Sigma_{N=1}^{M} L_N^2$ for eight-character words ($M = 8$), the resulting magnitude density function [Fig. 3 (b)] is heavily populated in the lower portions of the matrix and is increasingly sparse at the higher value of magnitude. In fact, for the case of English words the probability of having an occupied matrix position above one-half the maximum possible value of magnitude ($8L_{\max}^2$) is essentially zero. In practice, the BRM is truncated for values above $4L_{\max}^2$. For the remainder of the matrix the majority (85 percent) of the legal words are represented by values below $2L_{\max}^2$, whereas the region between $2L_{\max}^2$ and $4L_{\max}^2$ has a high degree of sparsity.

To meet the first condition only, for a BRM numbering scheme the optimal solution occurs when the characters are assigned numerical values in order inverse to their probability $P(\alpha_j)$ of occurrence in the dictionary of valid words. This may be expressed as

$$\cdots < L_{k-1} < L_k < L_{k+1} < \cdots \tag{9}$$

and

$$\cdots > P(a_{k-1}) > P(a_k) > P(a_{k+1}) > \cdots. \tag{9'}$$

Restriction 2 The restriction that words garbled by the OCR generate magnitude-angle representations in the sparse area of the matrix can be satisfied by placing two conditions on the numbering scheme: (a) Because unreliable words are made up of unreliable characters, if such (easily misread) characters are assigned high values, the words which contain these characters will have high magnitude values. By this method reliable words cluster in dense areas of the matrix and unreliable words tend to be found in sparse areas. For this purpose the designation of numbers would best be made by ordering characters in accordance with their reliability and assigning the numerical values in sequence starting with 1. Stated another way, the characters should be ordered according to their unreliability and assigned numbers in inverse sequence starting with $L_{\max}$. This condition may be expressed as follows:

$$\text{Unreliability} = \sum_{\alpha_i \neq \alpha_{\text{dict}}}^{\alpha_{26}} P(\alpha_i | a_{\text{dict}}),$$

where a_{dict} is a particular input character and α_i is one of the possible output characters falsely generated by the OCR. Therefore, (9) and (9′) become

$$\cdots < L_{k-1} < L_k < L_{k+1} < \cdots \tag{10}$$

and

$$\cdots < \sum_{i \neq k-1}^{26} P(\alpha_i | a_{k-1}) < \sum_{i \neq k}^{26} P(\alpha_i | a_k) < \sum_{i \neq k+1}^{26} P(\alpha_i | a_{k+1}) < \cdots. \tag{10'}$$

(b) The condition expressed in the inequalities (10) and (10′) causes unreliable words to map into the sparse upper magnitude portions of the matrix. However, this alone is not sufficient to assure that garbled words map into sparse areas of the matrix. For example, it is possible for an unreliable character to be falsely read into a reliable character and cause the resulting false version of an unreliable word to be mapped into a lower portion of the matrix. What this probably indicates is that there are actually two measures of unreliability. One is for the dictionary word and is expressed by that portion of the character transfer function defined as

$$\sum_{\alpha_i = a_{\text{dict}}}^{\alpha_{26}} P(\alpha_i | a_{\text{dict}}).$$

The other is the unreliability associated with characters in the word as read by the OCR. This measure may be expressed by that portion of the character transfer function,

$$\sum_{a_j \neq \alpha_{\text{output}}}^{\alpha_{26}} P(a_j | \alpha_{\text{output}}),$$

in which α_{output} is a particular output character, incorrectly read by the OCR, and a_j is one of the possible input characters which caused this read. It should be noted that these two measures of unreliability are by no means equal for a particular character.

It is necessary, then, to formulate a third condition on the assignment of numerical values to characters. The purpose of this condition is to give high values to those characters in the OCR output which have a high probability of having been misread from other input characters. This condition may be expressed as follows:

$$\cdots < L_{k-1} < L_k < L_{k+1} < \cdots \tag{11}$$

and

$$\cdots < \sum_{j \neq k-1}^{26} P(a_j | \alpha_{k-1}) < \sum_{j \neq k}^{26} P(a_j | \alpha_k) < \sum_{j \neq k+1}^{26} P(a_j | \alpha_{k+1}) < \cdots. \tag{11'}$$

The condition expressed in (11) and (11′) tends to cause words, incorrectly read by the OCR, to map into higher values of magnitude than their original dictionary versions.

• *Alphameric equivalency using all assignment conditions*

The three conditions expressed in the inequalities (9) and (9′), (10) and (10′), and (11) and (11′) are not necessarily compatible with each other when based statistically on English dictionary words and normal OCR transformation characteristics. A character such as I has a relatively high occurrence rate but is also highly unreliable. The numbering scheme based on relations (9) and (9′) would be substantially different from that based on relations (10) and (10′) or (11) and (11′). It is necessary, therefore, to define some character measure that reflects the character's ranking when all three conditions are considered simultaneously. Such a ranking will not be optimal for any one condition. However, the total effect when used in word verification with the BRM should be to map incorrectly read words into a sparse region of the matrix.

Condition (9) implies that a character should have a high numerical assignment if its occurrence rate $P(a_j)$ is low. This may be restated to require that character a_j have a low numerical assignment if $1/P(a_j)$ is small.

Conditions (10) and (11) imply that a character has a high numerical assignment if its unreliability is high. This unreliability is defined differently for dictionary words than for OCR output words. It is possible to define an average measure of unreliability for a character based on both conditions. This average measure is expressed as

$$\overline{U} = \frac{1}{P(\alpha_{\text{output}}) + P(a_{\text{dict}})} \times \left[P(\alpha_{\text{output}}) \sum_{a_j \neq \alpha_{\text{output}}}^{a_{26}} P(a_j|\alpha_{\text{output}}) + P(a_{\text{dict}}) \sum_{\alpha i \neq a_{\text{dict}}}^{\alpha_{26}} P(\alpha_i|a_{\text{dict}}) \right], \quad (12)$$

where a_{dict} is a particular input character and α_{output} is the correct OCR output for this character.

For any large data sample, $P(a_{\text{dict}})$ is approximately equal to $P(\alpha_{\text{output}})$. Equation (12) may, therefore, be simplified to

$$\overline{U} = \tfrac{1}{2}\left[\sum_{a_j \neq \alpha_{\text{output}}}^{a_{26}} P(a_j|\alpha_{\text{output}}) + \sum_{\alpha_i \neq a_{\text{dict}}}^{\alpha_{26}} P(\alpha_i|a_{\text{dict}}) \right]. \quad (13)$$

Combining condition (9) with conditions (10) and (11), we see that a character should be assigned a high numerical value if both $1/P(a_j)$ and $\overline{U}$ are high and, conversely, a low value if $1/P(a_j)$ and $\overline{U}$ are low. The product of these two measures is, therefore, a meaningful condition by which to assign numerical values. The resulting expression for the assignment of numerical values could then be

$$\cdots < L_{k-1} < L_k < L_{k+1} < \cdots \quad (14)$$

and

$$\cdots < \frac{\overline{U}_{k-1}}{P(a_{k-1})} < \frac{\overline{U}_k}{P(a_k)} < \frac{\overline{U}_{k+1}}{P(a_{k+1})} < \cdots. \quad (14')$$

It should be noted that the conditions (14) and (14′) apply for any uniform numbering sequence (not just 1 to 26) which runs from $L_{\max}/Z$ to $L_{\max}$, where Z is the number of characters in the alphabet and $L_{\max}$ is the maximum numerical value in the sequence.

Also, because conditions (14) and (14′) indicate only an ordering of the characters, it is possible to select values which are not uniformly separated in the numerical sequence. This causes a deviation from the statistical model by which the conditions were derived, but in practice it permits shifting numerical assignments when empirical data indicate potential improvement in performance.

Table 11 shows the alphameric equivalency scheme that was used for a dictionary of 15 000 words. In this case $L_{\max} = 60$ and the spacing of numerical values is nonuniform.

When configured in this manner, the BRM has proved to be an effective error word discriminant tool. Extensive

Table 11 Verification numbering scheme.

Common substitutions

E → F	B → E
L → I	h → n
M → N	G → C

Number selection

A	10	N	1
B	17	O	20
C	35	P	30
D	11	Q	55
E	4	R	2
F	45	S	6
G	24	T	18
H	25	U	23
I	60	V	40
J	13	W	15
K	28	X	16
L	3	Y	21
M	50	Z	21
		b	22

testing has been conducted with the BRM occupied with a word list of approximately 15 000 street names. Recognition output of the AOCR installed in the New York City General Post Office was tested against the BRM to determine the reliability of its verification processes. An overall misverification rate of less than one percent was attainable. This is of interest in that only 10 000 bytes, or 80 000 bits, of core storage were used for the BRM. Of these 80 000 bits, nominally 15 000 were occupied (storage value = 1). Based on binomial statistics, if these occupied bit positions were randomly scattered over the matrix, with no strategy taking into account the OCR misread propensities, then, on the average, one out of every five error words should strike an occupied bit position, leading to a 20 percent erroneous verification rate. Hence, in contrast, the noted misverification rate of less than one percent stands as testimony to the strategy of building BRM to reflect the character misread propensities of the OCR and thereby effect reliable discrimination of OCR garbled words. Additional BRM error word discrimination reliability can be accrued directly by allocating additional storage to the present 10000-byte matrix.

Further, the BRM concept should not be viewed as applicable only to OCR word correctness verification. Rather, for example, the potential exists to adapt the basic techniques to perform – in a highly efficient manner – human operator keystroke verification.

4. Dictionary access

This section describes the mechanics of the dictionary access or fetch procedure. For each word that failed to verify in the BRM (Section 3), error correction processing must be entered. The strategy used to effect OCR error correction is to reference an error correction dictionary and determine from all the words listed therein which of the dictionary entries is the word that was scanned by the OCR and misread into the garbled form currently being processed. Clearly, a basic part of this operation is the ability to determine which segment of the error correction dictionary should be reviewed. Schematically this is shown in Fig. 4. The more accurately we can delineate the portion of the dictionary that contains the correct form of the garbled word, the larger the dictionary can be without compromising the efficiency and real-time nature of the OCR error correction operation.

When the verification procedure, discussed in Section 3, passes a word to error correction processing, the properties of an OCR misread make it impossible to formulate a reliable dictionary access using the normal dictionary indexing word attributes of character alphabetic properties and/or word length. The OCR error propensities can alter either or both of the word attributes in various ways. In spite of this, there is still much potential dictionary entry key information in the garbled data. To utilize a garbled word as a key to the dictionary, the character string must be analyzed in a new perspective. The vehicles for this analysis are the Vector Fetch (VF) and the Word Group (WG) file organization concepts.

• *Vector fetch methodology*

The rationale that underlies the VF dictionary accessing methodology can best be understood as a specialized application of classical statistical confidence interval theory. As normally configured, a confidence interval sets up a range of values within which the true value of a factor being estimated can be said to lie with a predetermined error tolerance.

Within the perspective of the confidence interval analysis, the VF methodology can be configured as a specialized application which uses the garbled word data to 1) estimate the dictionary location of the word that was misread by the OCR, and 2) give data-fetch relevance to the estimated Dictionary Access Point by generating around it a range of entries wherein the required word information lies with a predetermined certainty. The description of the analytical mechanics involved in the implementation of the preceding dictionary access/data fetch methodology is logically broken into three portions:

1. Estimation of the Dictionary Access Point;
2. Determination of the fetch width constraints; and
3. Dictionary organization.

Estimation of dictionary access point

The Dictionary Access Point (DAP) is the initial estimate of where the correct form of the OCR garbled word lies in the error correction dictionary. The vehicle for this initial estimation process is a specialized Hashing transformation applied to the garbled alpha character string. Underlying the Hashing transformation is a specially developed numeric code, in which each character in the recognition alphabet has a numeric designation that reflects its absolute and its relative OCR recognition reliabilities. The particulars of the alphameric assignment scheme are elaborated later. It presently suffices to say that the numeric magnitude assignment is related to reliability of recognition of the alpha character. In its simplest form this implies that the more reliable an alpha character recognition, the more weight is put upon it in the Hashing calculation.

Given this alphameric assignment scheme, the DAP follows as a summation of positive integers:

$$\mathrm{DAP} = \sum_{N=1}^{M} L_N, \tag{15}$$

where L_N is the numeric value assigned to the character in the Nth position of the garbled word, and M is the number of character positions in the garbled word. The key to this technique is the derivation of the appropriate alphameric assignment scheme. Dual and seemingly conflicting constraints have to be accommodated in the assignment scheme. Essentially, the alphameric assignment used to compute the DAP has to

1. minimize the effect on the DAP of intercharacter aliasing resulting from OCR misreads and
2. map the dictionary word into a relatively uniform spread over the range of DAPs.

The first constraint reflects the desire that Eq. (15), the Hashing formulation, be as insensitive as possible to the expected result of OCR substitution and segmentation misreads. The second constraint seeks to avoid a trivial solution evolving as a result of the first constraint. Such a solution would be the collapsing of the dictionary so that all entries occupy a single DAP or a very narrow band of DAPs. If this were the case, nearly the entire dictionary would be brought down in each fetch. This, in terms of real-time processing constraints, would be an unacceptable situation and would defeat the intent of the VF algorithm.

The optimal alphameric assignment scheme for the VF can be derived by a mathematical approach using linear programming, which is based on expressing the OCR intercharacter aliasing propensities as linear relations. This implies, for every non-null event in the OCR confusion matrix, a normed distance (i.e., an absolute value relationship) of the form

$$\|X_\alpha - X_\beta\| \leq \text{constant}, \tag{16}$$

where X_α and X_β are the numeric designates of the alphabetic characters denoted in the general case by α and β. Existing OCR confusion statistics, when reconstituted in the above form, yielded 437 separate expressions of the form (16). Standard linear optimization formulation, however, is not able to directly accommodate a normed distance as a base variable in the system of constraints or in its objective function.

To allow the programming optimization of the VF alphameric assignment scheme to reflect a normed analog of the OCR misread characteristics, a mixed-integer linear programming formulation was adopted. Each constraining relation of the form (16) is reconstituted as a set of mixed-integer linear programming constraints of the form

$$K + Z_{\alpha\beta} \geq X_\alpha - X_\beta + 2KI_{\alpha\beta} \geq K - Z_{\alpha\beta}, \tag{17}$$

where $I_{\alpha\beta}$ represents the set of integer variables constrained to take on the values one or zero; $Z_{\alpha\beta}$ is the variable over which the objective function optimization, min $(\Sigma P_{\alpha\beta} Z_{\alpha\beta})$, is performed; $P_{\alpha\beta}$ is the relative weight or importance value associated with the respective constraint; and K is the fetch error tolerance in units of magnitude. In the present analysis, $P_{\alpha\beta}$ has been set equal to the cumulative occurrence rate of the respective α, β characters. Up to this point, the system of optimization equations has injected into the analysis only constraints consistent with the first goal above.

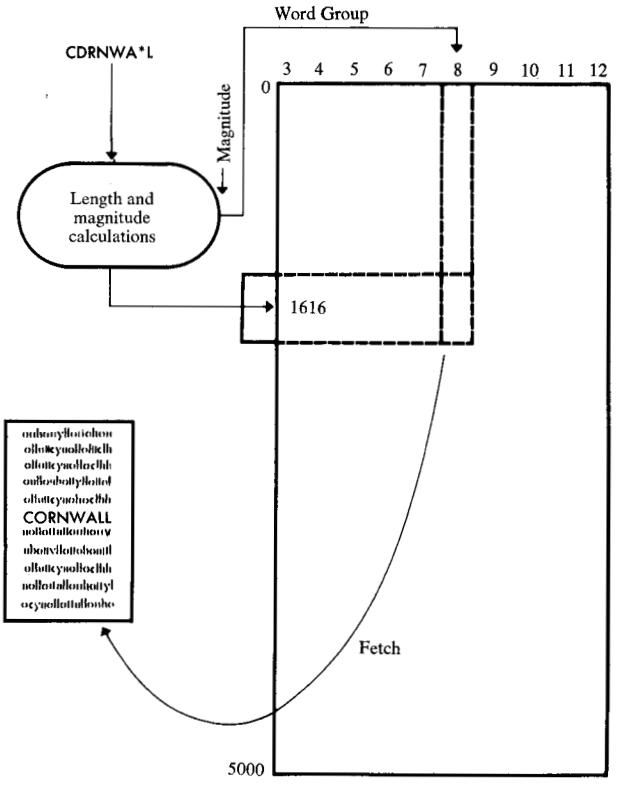


Figure 4 Partial fetch process.

The second goal, the avoidance of inordinate degrees of clustering of dictionary entries in any range of magnitude, is accomplished by appending to the system of OCR misread inter-relationships (17) a series of constraints which reflects a suitable dictionary infra-structure that maintains salutary entry-distribution characteristics with respect to all segments of the dictionary. These latter constraints are set up by randomly selected legal entries from the dictionary word list and specify that a predetermined normed distance be maintained between them in the final dictionary vector structure. For example, the entries CORNWALL and SHERWOOD can be used to yield a vector dictionary infra-structure constraint of the form

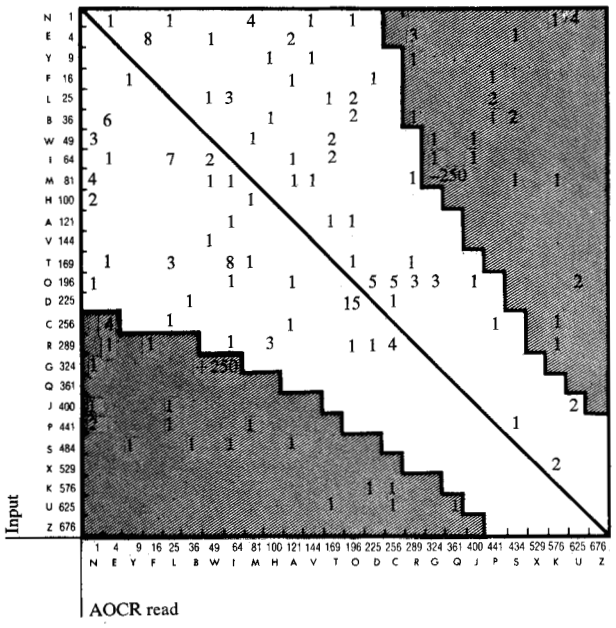


Figure 5 Numeric assignment scheme and substitution matrix. The off-diagonal numerals are the relative occurrence rates.

$$X_C + X_O + X_R + X_N + X_W + X_A + X_L + X_L - (X_S + X_H + X_E + X_R + X_W + X_O + X_O + X_D) = X_C + X_N + X_A + 2X_L - X_S - X_H - X_E - X_O - X_D > D_1. \quad (18)$$

The value of D_1 represents the normed distance between entries CORNWALL and SHERWOOD in the dictionary, where an alphameric assignment scheme has been used which yields good dictionary word-list-spread characteristics while not necessarily meeting all the OCR minimal Hashing distortion constraints as given by (17). One of the suitable modes for derivation of this initial alphameric assignment scheme is examined below. The optimization-programming array of constraints is completed by adding the additional infra-structure constraints consistent with the simple linear format described by the CORNWALL, SHERWOOD example in (18).

The initial alphameric assignment scheme used to generate relations of the form (18) was obtained by treating Eq. (15) as a vector magnitude computation; that is,

$$Y = \sum_{N=1}^{M} L_N^2, \quad (19)$$

and by assigning 1 through 26 (L_N^2, 1 through 676) to the characters in the alphabet.

Figure 5 indicates how the numeric assignments are made in a manner that is semiconsistent with that required by the OCR-misread magnitude-distortion minimization constraints posed by (17). The horizontal set represents the OCR recognition decision. All correct recognitions are indicated by the diagnoal of the matrix. All substitutions or rejects are off the diagonal. For example, if an H and an M are given values of 10 and 9, respectively, and if an H is misread as an M, the difference of magnitude is 100 minus 81, or 19. This would be an appropriate selection because H-M substitution is common.

If the OCR misread distortion tolerance is set at ±250 units [i.e., the nominal value of the factor K on the right side of the system of equations generated from (17)], a relatively simple yet meaningful initial assignment of alpha characters to the numeric designations indicated on the axes of the confusion matrix can be derived; then a large number of common recognition errors are contained within these ±250-unit Hashing file-address distortion tolerance boundaries.

The initial numeric assignment scheme is shown in Fig. 5, where the shaded portion of the figure has those misreads for which the initial scheme cannot compensate (the numbers within the matrix relate to the relative occurrence rates of the specific misread errors). Empirical analysis with this numbering scheme showed that although it did not satisfy all constraints of the form (16), it did transform a word list into a suitably distributed dictionary that did not contain high-density ranges with inordinate clustering of dictionary entries For this reason, this numbering scheme was used to define the normed distances between the randomly selected entries used to formulate the dictionary infra-structure constraints, as given by (18).

Other numbering schemes could have been successfully used for the bases of these infra-structure constraints. The vector magnitude scheme was used because of its simplicity and our experience with it from initial investigations.

The resulting formulation of mixed-integer linear programming constraints and objective functions was solved using the IBM Mathematical Programming System [1]. The final output of the programming solution yielded a set of alphameric assignments which minimized Hashing distortions due to OCR misread, while maintaining a relatively uniform spread of entries over the dictionary. The alphameric assignment scheme is shown in Table 12.

Determination of fetch width constraints

If the garbled word data were transformed into a magnitude value using the alphameric assignment scheme shown in Table 12, it could be assumed that the garbled and correct forms of the same word would map into fairly similar (close) magnitude values. If the correct form

of each word had been stored in the error correction dictionary with respect to its magnitude, then the DAP yielded by (15) would approach the vicinity of the correct word entry required for completion of error correction processing. However, to successfully perform the decision process that underlies the Regional Context Maximum Likelihood (RCML) error correction procedure (Section 5), it is a prerequisite that the garbled form of the word be compared in a conditional probabilistic format with the correct version of that word. Hence, the DAP, in itself, is not sufficient for fetching the data required for the latter phases of OCR error correction. However, the proximity of the DAP to the correct dictionary entry makes it a natural axis point for the construction of a confidence interval that will act as the delimiter of a dictionary fetch range. If properly configured, the fetch range will bring into core storage a block of address entries which contains within it, with a predetermined error tolerance, the correct version of the garbled word. As in the preceding example, the selection of ±250 units as a fetch width implies an error tolerance, i.e., the possibility of the correct version of the garbled word being outside the fetch range.

The three major OCR error sources that must be compensated for in the construction of the dictionary fetch range are 1) reject characters, 2) substitution errors, and 3) segmentation errors. The fetch is most effective for the reject and substitution errors. Segmentation errors are statistically less predictable and therefore not as readily overcome. A garbled word can become unretrievable using the VF if successive misreads within the word additively reinforce each other until a magnitude difference greater than ±250 units is achieved. This situation is comparatively rare, in that successive misreads tend to cancel randomly, to some degree, the magnitude deviation that each has added.

Word group file organization

Lengthwise organization of dictionary files is used to complement and reinforce the discrimination potential of the VF methodology. The VF is a vehicle that enables garbled alpha data to be given relevance as a file key. There is, however, another powerful discriminant, namely, word length.

Figure 4 shows a schematic of the fetch process for the garbled word. The magnitude of the error word is calculated using (15). For the word shown the magnitude is 1 616. The word length is also used to reduce the number of entries in the fetch. For OCR garbled data, length cannot be used as an absolute discriminant because segmentation errors may artificially increase or decrease the word length. A common approach to this problem is to include in the fetch not only words of the same length as the error word, but also all words of adjacent (±1 character) length and even those that differ by as much as two characters. This is done according to rules which themselves are length-dependent. The problem with this approach is that it leads to unacceptably large fetch sizes (on the average, approximately 20 percent of the dictionary).

Table 12 Final fetch vector alphameric assignment scheme—values generated using mixed-integer linear programming.

A = 250	G = 190	M = 217	S = 429	Y = 110
B = 213	H = 303	N = 55	T = 141	Z = 429
C = 248	I = 75	O = 340	U = 110	* = 200
D = 321	J = 235	P = 121	V = 17	Conflict I/l = 75
E = 95	K = 470	Q = 284	W = 115	Conflict N/W = 225
F = 213	L = 76	R = 275	X = 429	

It is again possible to utilize OCR error statistics to improve the word length discrimination. Because word length changes are caused by some type of segmentation (splitting or catenation), only the words prone to be mis-segmented by virtue of their composition are entered in more than one of the word-length-oriented dictionary subdivisions. This leads to the Word Group concept. In a Word Group, all words of a designated length are included, as well as words of all other lengths that have a significant probability of being mis-segmented to the basic length.

The implementation of Word Group file organization is dependent on the determination of objective criteria by which a word and its character composition may be evaluated for degree of mis-segmentation propensity and consequent multiple Word Group entry requirements. To allow objective assessment of a dictionary entry word group candidacy, the following statistical segmentation threshold calculation is performed.

The probability of word segmentation is described functionally by Eq. (20):

$$P(\text{word}_{\text{seg}}) = 1 - P(\text{word}_{\overline{\text{seg}}}) \equiv 1 - P(W_{\overline{\text{seg}}}), \qquad (20)$$

where the bar notation indicates the complement of the segmentation event, that is, the non-occurrence of segmentation. From empirical data averaged over all word lengths, the value of the right side of Eq. (20) can be assessed as 0.6 percent. It is reasonable, therefore, to take as a threshold for Word Group duplicative entry, any word whose cumulative character segmentation probability surpasses this nominal value or, in other words,

$$P(W_{\text{seg}}) > T = 0.006. \qquad (21)$$

The relationship in Eq. (20) can be made more meaningful by posing it in terms of constituent character events as

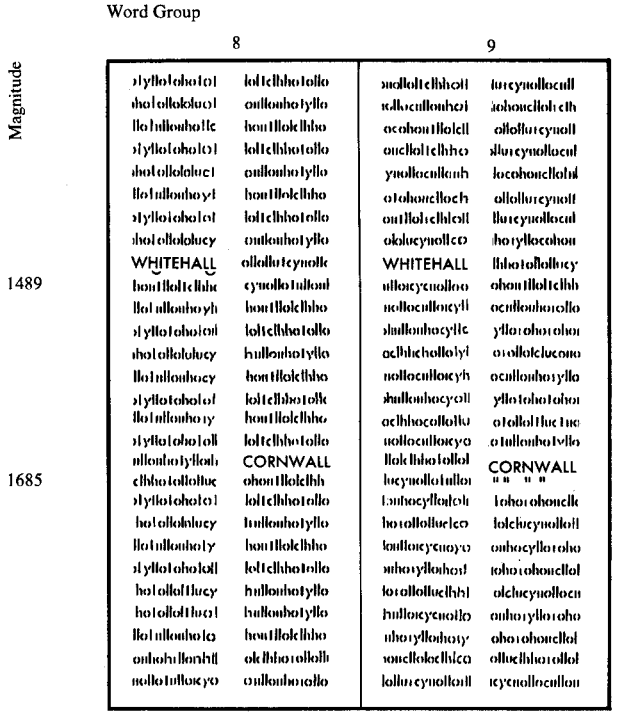


Figure 6 Dictionary organization.

$$P(W_{\text{seg}}) = 1 - P(\alpha_{1\overline{\text{seg}}}) \cdot P(\alpha_{2\overline{\text{seg}}}) \cdots P(\alpha_{N\overline{\text{seg}}}). \qquad (22)$$

By substituting Eq. (22) into Eq. (21) we obtain

$$P(\alpha_{1\overline{\text{seg}}}) \cdot P(\alpha_{2\overline{\text{seg}}}) \cdots P(\alpha_{N\overline{\text{seg}}}) < 1 - T.$$

In terms of logarithms, this finally results in a general threshold relationship for Word Group candidacy, namely,

$$\|\log P(\alpha_{1\overline{\text{seg}}}) + \log P(\alpha_{2\overline{\text{seg}}}) + \cdots + \log P(\alpha_{N\overline{\text{seg}}})\| > \|\log (1 - T)\|. \qquad (23)$$

By relating (23) to the binomial model which underlies its application, we can readily solve for the levels of mis-segmentation propensity (probability) which make a word a candidate for duplicative entry in one Word Group, two Word Groups, etc. This is performed as follows.

Threshold for one segmentation event:

$$\sum_{j=1}^{L} \|\log P(\alpha_{j\overline{\text{seg}}})\| > \|\log (1 - T)\|,$$

where L is the number of characters in a word. Threshold for two segmentation events:

$$P(\text{word}_{\text{seg-2}}) = C_2^{\,L}\, \overline{P(\alpha_{\text{seg}})}^{\,2}\, \overline{P(\alpha_{\overline{\text{seg}}})}^{\,L-2} = \frac{L!}{2!(L-2)!}\,[1 - \overline{P(\alpha_{\overline{\text{seg}}})}]^2\, \overline{P(\alpha_{\overline{\text{seg}}})}^{\,L-2}.$$

where $\overline{P}$ means the average value of P. Hence, the word mis-segmentation threshold for a dictionary entry to be entered in two adjacent Word Groups becomes

$$P(\text{word}_{\text{seg-2}}) \approx \frac{L!}{2!(L-2)!}\,[1 - \overline{P(\alpha_{\text{seg}})}]^2 > T.$$

This expression can be put in convenient computational form for a particular length, e.g., 8:

$$\|\log \overline{P(\alpha_{\overline{\text{seg}}})}\| > \|\log [1 - \sqrt{T(2!)(6!)/(8!)}]\|.$$

Analgous analytical procedures can be applied to obtain the complete spectrum of Word Group thresholds, i.e., for single entry, double entry, triple entry, etc., for each respective word length.

In a Word Group using the previously derived mis-segmentation propensity thresholds, all words of the designated length are included, as well as words of other lengths that have a significant probability of being mis-segmented to that length. Therefore, a single word may appear in several Word Groups, based on its character composition. For example, in Fig. 6 the word CORNWALL appears in Word Group 8, its correct length. CORNWALL, however, has four characters that are prone to splitting segmentation (one character segmented into two). These are C, O, N, and W. It has been determined that there is a significant probability of CORNWALL being mis-read as a nine-character word, such as CORNVVALL, or a ten-character word such as CIJRNVVALL. Therefore, the word is also included in Word Groups 9 and 10. Similarly, WHITEHALL is initially in Word Group 9. However, it is also included in Word Group 8 because it has two character pairs, either of which is likely to catenate into a single character; these are HI and LL.

In summary, the dictionary organization takes the form of autonomous Word Groups based on alpha-field length. This implies that all N-character dictionary entries are listed together, where $N = 1, 2, 3, \cdots$ up to the length the longest set of dictionary words being considered. Appended to each of these error correction dictionary entry subsets are dictionary words of a different length but whose alphabetic composition makes their segmentation propensity exceed a threshold so that they are likely candidates for OCR length distortion effects.

The number of entries in the fetch produced by using both magnitude and Word Group discriminants has been shown in simulation to be between six and seven percent of the number of unique entries in the total dictionary. This reduction in fetch size is achieved while having only a small effect on fetch accuracy.

• *Further fetch discrimination*

With respect to reduction in dictionary fetch size, Vector Fetch and Word Groups are passive discriminants. This implies that their fetch reduction potential is

achieved by virtue of dictionary organization. No significant computational activity must be expended to utilize the discriminatory potential of these fetch constraints. In conjunction with VF and WG, a third very effective passive fetch discriminant can be incorporated into the fetch operation. The third discriminant requires that either the first- or the last-position character match between the garbled word field and any dictionary entry to be fetched into core storage. The First/Last Position (FLP) fetch constraint is of particular significance in terms of its reliability, its discrimination potential, and the OCR recognition anomaly that underlies its application.

The FLP fetch constraint results from the observed fact that even when a word is badly garbled, rarely are both the first and last letters misread. In referring to a word as garbled, we specifically mean to differentiate between a subfield that contains misreads and one that is a "wipe out." The "wipe out," which normally results from interference or a nonreadable font, is a misread that has lost all character content relevance and is basically uncorrectable. A garbled subfield, on the other hand, contains recognition errors normally resulting from substitution or mis-segmentation. It is extremely unlikely that mis-segmentation will exist strongly enough at both the beginning and the end of a word to create misreads in both positions, if the word is not a wipe out. Analysis of OCR data has shown occurrence of simultaneous first- and last-character misreads in a subfield that has some degree of character content to be much less than one percent. Thus, it is a working assumption that at least the first or the last character is valid in any word that is to enter and complete the error correction process.

Implementation of the FLP constraint, shown in Fig. 7, is achieved by the following file organization: The dictionary is double-stored, for example, using separate spools of the disk storage facility. In the first dictionary copy, within each alphabetic grouping (e.g., first character A, first character B, etc.), the entries are organized in Word Groups and then, within each Word Group, division is by entry magnitude. The second dictionary copy is identical to the first except that it is alphabetized according to the last character in each entry. Not only does this configuration achieve, in a passive manner, the FLP fetch discrimination but also it greatly increases the I/O efficiency. It allows more potential for data (or processing) overlap and reduces disk latency to a minimum.

The most important factor in the FLP utilization is the reduction of fetch size it achieves. Both analytically and in fetch simulation, the FLP discrimination has been shown to reduce fetch size by an average factor slightly greater than six. This means that by using in concert all

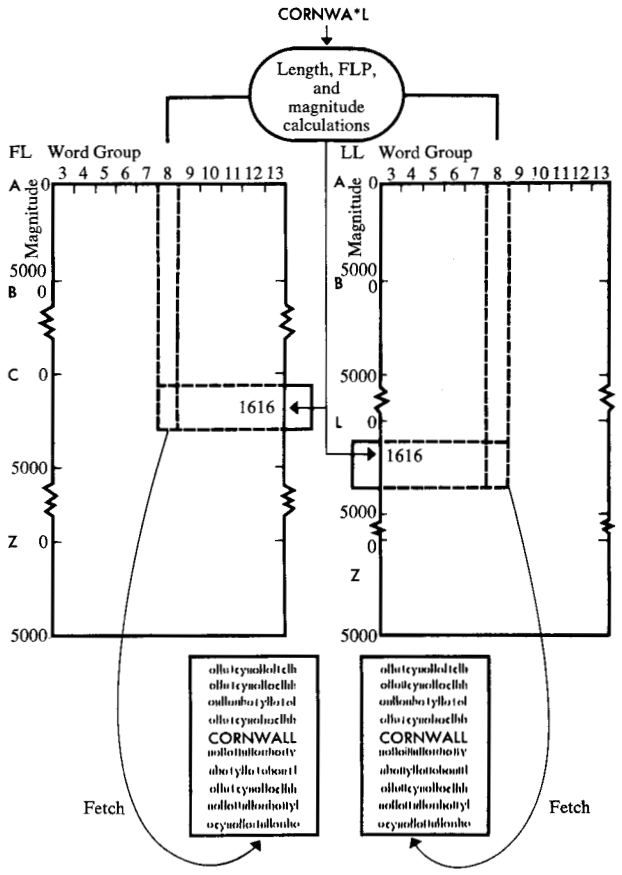


Figure 7 Complete fetch process from double-stored dictionary.

three fetch discriminants, the average block of data that enters the error correction phase is only about one percent of the dictionary. (Use of simple word length, i.e., not Word Group, would lead to an average fetch size of 26 percent.) This is particularly significant considering that no overt computational effort has been expended while achieving this reduction. Table 13 shows the approximate fetch performance resulting from simulation when the three fetch discriminants are used in concert.

In summary, the OCR file organization results in the ability to greatly scale down the computing potential required to support the OCR operation without sacrificing performance accuracy or real-time operating characteristics.

5. Error correction

The final procedure performed by contextual word recognition postprocessing is Regional Context Maximum Likelihood (RCML) error correction. The RCML procedure provides highly reliable OCR error correction of alpha words that have been garbled during recognition processing by character substitution, character rejection,

Table 13 Fetch accuracy with 0.75 percent[a] of file fetched.

	Error type					
				Segmentation		
	Rejection	*Substitution*	*Crowding*	*Splitting*	*Cate-nation*	*Average recovery rate*
Relative occurrence (%)	57.11	24.53	5.70	6.72	5.94	
Recovery rate (%)	98.9	97.4	90.4	76.7	56.5	94.05

[a]This is the average percentage of the error correction dictionary that meets the triple physical constraint of being at the intersection of
1. Magnitude of the data fetch range;
2. Dictionary word group containing the entries related to the garbled word's alpha character string length; and
3. Matching the garbled word in at least the first or the last character.

horizontal splitting segmentation, catenation segmentation, or crowding segmentation.

• *RCML procedure*

RCML error correction operates on the garbled word that failed verification (Section 3) and on the set of words fetched from the OCR error correction dictionary (Section 4). A typical problem is shown in Table 14. The output of the RCML procedure is the determination of which of these dictionary entries corresponds to the word that has been scanned by the OCR and garbled into its present incorrect form. Because the OCR error correction operation must often be performed against a dictionary word list as comprehensive as an ordinary dictionary, the RCML algorithm must be capable of discriminating among contending dictionary entries that may differ by as little as one alpha character. The RCML procedure must therefore utilize more than just the count of matching characters between the garbled word and a dictionary entry in order to achieve reliable error correction.

Error correction by the RCML method is done by means of a conditional probabilistic analysis. This approach evaluates the likelihood that each of the respective dictionary entries being considered could have been mapped into the garbled character string by means of the OCR device's error misread propensities. The analysis uses all data available in formulating this probability. Physically, the likelihood analysis corresponds to the computation of an analog distance between a dictionary word and the garbled data, weighted by the *a priori* probability that the dictionary entry would have occurred in the alpha fields being OCR scanned. Mathematically, this analysis is formulated by the conditional probabilistic statement

$$P(\text{dictionary entry}|\text{garbled alpha string}) = \frac{P(\text{dictionary entry, garbled alpha string})}{P(\text{garbled alpha string})}. \quad (24)$$

The denominator of Eq. (24) is essentially a scaling factor and has the same value for all the entries being compared with the garbled alpha string. Hence, the relative ranking of each entry (i.e., the probability of each entry mapping into the garbled alpha string) is based on the value of the numerator in Eq. (24). Therefore, for the rest of the error correction analysis, the focus is on what maximizes the numerator.

By applying Bayes' theorem, we can reformulate the numerator in Eq. (24) as

$$P(\text{dictionary entry, garbled alpha string}) = P(\text{garbled alpha string}|\text{dictionary entry}) \times P(\text{dictionary entry}). \quad (25)$$

• *The a priori factor*

The probability factor P(dictionary entry) is called the *a priori* probability of the event. For text processing, it is the probability that the dictionary entry being compared to the garbled character string appears in the data being scanned. *A priori* data for word occurrence rates in general English text may be obtained from prior analysis [2].

For more specialized text input, other sources of *a priori* data are available. In addition, an adaptive self-teaching approach can be utilized which allows the dictionary word entries to dynamically attain their appropriate *a priori* values. Detailed discussion of such self-teaching algorithms is, however, outside the scope of this paper.

• *Evaluation of the likelihood factor*

The probability factor

$$P(\text{garbled alpha string}|\text{dictionary entry}) \quad (26)$$

is called the likelihood factor. The major computational effort of the RCML error correction procedure centers around the evaluation of this expression.

In the evaluation of the likelihood factor one must capture, in a probabilistic form, the misread propensities of the subject OCR. The conditional format of (26) poses the likelihood as follows: Given a dictionary entry, what is the probability of the OCR misread propersities having mapped it into the garbled alpha string? Since OCRs recognize an alpha field on a character-by-character basis (i.e., they do not directly recognize words as single entities), (26) is really the product of a series of independent probabilistic events. In this perspective, there are two categories of OCR misrecognition that must be addressed: They are substitution and segmentation.

Substitution maximum likelihood analysis

OCR substitution manifests itself in two ways. The first is character substitution. The recognition unit captures the video image of a single character, but the features required for alpha determination are aliased as another character. Logically, this can occur only if there is some degree of similarity in shape of the alpha characters involved. Examples of such letter combinations are B, D; D, O; O, C; l, i; etc. The second form of substitution is character rejection. As with character substitution, the recognition unit captures a single character. However, rejection occurs because of the inability of the recognition logic to relate it to any character or because more than one set of alpha determination logic is satisfied by the character features isolated. In this discussion, all rejects are denoted by an asterisk (*).

From a probability standpoint, both of the misread effects can be posed as simple, independent, conditional probabilities. Respectively, character substitution and reject substitution enter (25) as

$$P_c(L_j|L_i) \text{ and} \quad (27)$$

$$P_c(*|L_i). \quad (28)$$

These represent the probability that the alpha character L_i is scanned by the OCR and that L_j or * is the output. This probability datum is derived from a character confusion matrix and is prestored, requiring no computation time. The character confusion statistics are compiled separately relative to uppercase and lowercase alpha characters.

An example indicates how expressions of the forms (27) and (28) can be applied in this Bayesian decision process. The garbled word is CDRNWA*L and the entry from the dictionary fetch which is being tested is CORNWALL. The likelihood factor is given by the probabilistic series of independent events as shown in this example.

Table 14 Basic error correction/dictionary match problem.

Garbled word
CDRNWA*L
Dictionary candidates
CROMWELL
CHRISTIAN
CLARIDGE
COLONIAL
CORNWALL
TOWNHALL
HOSPITAL
GLENFALL
NATIONAL
WHITEHALL

Example

Garbled word = CDRNWA*L

Dictionary word = CORNWALL

Likelihood factor = $P(\text{CDRNWA*L}|\text{CORNWALL})$

$$= P_c(\text{C}|\text{C}) \cdot P_c(\text{D}|\text{O}) \cdot P_c(\text{R}|\text{R}) \cdot P_c(\text{N}|\text{N}) \cdots P_c(*|\text{L}) \cdot P_c(\text{L}|\text{L}).$$

The likelihood factor is the product of a number of independent character confusion probabilities, which results in a relative value that can be compared with that generated by the other words under test. The entry word which has the highest probability of being the original word is chosen, provided it meets certain reasonableness criteria.

Segmentation maximum likelihood analysis

Segmentation differs from substitution in that its independent events correspond to groupings of at least two characters. Nominally, there are three types of segmentation error. They are horizontal splitting segmentation, catenation segmentation, and crowding segmentation. The underlying mechanical factor, which all of these segmentation types have in common, is that they are generated by the improper discernment of character beginning and end points.

Horizontal Splitting Segmentation (HSS) is prone to broad (wide) uppercase characters, such as W, M, N, U, O, and C. The HSS effect occurs when the recognition unit is misled into cutting one of these characters into two parts. Each portion is in turn reviewed by the recognition logic as if it were a legal character. This results in

Table 15 Horizontal splitting segmentation.

H	I H	N	I L	U	LI
	P L		NO		TI
	P I		NT		T*
	* I		R*	W	I W
M	I *		RI		**
	ME		T N		
	MR		* I		
	NN	O	OD		
	N*		OE		
	* C		OK		
	* I		OS		
	* M		O*		
	* N		* I		
	* *		OY		

several patterns of character and/or rejection misrecognition. Several of the more common forms are indicated in Table 15.

From a probabiltistic standpoint, the segmentation misread effect can be expressed as a dual aliasing effect conditioned on the occurrence of one of the set of uppercase letters noted previously. Functionally, this is indicated as

$$P_c(L_jL_{j+1}|L_i).$$

Obviously, the evaluation of (26) becomes more complicated when the HSS effect must be considered. The control logic of the calculation must consider three possible conditions when one of the segmentation prone characters, $L_{i_{seg}}$, is encountered. The conditions are

1. $L_{i_{seg}}$ has given rise to a simple substitution effect of the form

$$P_c(L_j|L_{i_{seg}}) \text{ or} \tag{29}$$

$$P_c(*|L_{i_{seg}}). \tag{30}$$

2. $L_{i_{seg}}$ has been improperly segmented, giving rise to an HSS effect of the form

$$P_c(L_jL_{j+1}|L_{i_{seg}}). \tag{31}$$

3. $L_{i_{seg}}$ has been properly recognized and outputed, giving rise to

$$P_c(L_{i_{seg}}|L_{i_{seg}}). \tag{32}$$

The presence of this last possibility is especially difficult to discern correctly because the most common type of character HSS (Table 15) recreates itself along with an additional spurious character. The analytic details of the inclusion of HSS in the evaluation of the likelihood factor (25) are discussed later so that it can be elaborated upon in the perspective of catenation and crowding segmentation errors.

Catenation Segmentation (CS) is nearly the mirror image of HSS. It occurs principally among closely spaced lowercase characters. Mechanically, CS evolves when the recognition unit is unable to discern in the scan the presence of two individual characters. Hence, the AOCR recognition logic proceeds to process the characters in a logically catenated manner.

This effect occurs mainly due to characters printed in a stylized manner or by crowded typewriter slugs. Table 16 contains several of the most CS prone letter combinations. In a probabilistic format the CS event can be posed as

$$P_c(L_j|L_iL_{i+1}) \text{ and} \tag{33}$$

$$P_c(*|L_iL_{i+1}). \tag{34}$$

The latter event may be particularly difficult to isolate while evaluating (26) because L_i itself may have a high propensity for mapping into an asterisk (i.e., being rejected) and is therefore suggestive of a plain substitution instead of a CS.

Crowding Segmentation (CRS) differs from HSS and CS error types by not affecting word length. The causative factors related to CRS are character spacing and juxtaposition. A potential CRS event occurs when the recognition unit isolates two characters but, because of their proximity to each other, misassigns the segmentation point. This effectively segments portions of one character into the video representation of the other. A misread results if the addition of the neighboring character segment either 1) creates a composite character that triggers the recognition logic of a different character or 2) interferes with the recognition analysis and leads to a reject (*) output.

The overriding factor behind CRS is character geometry. Only relatively few of the 676 possible diagrams are prone to "snowballing" a print-crowding effect into a character misread, as described previously. An example of such a character pair and the evolution of a CRS event is shown in Fig. 8, where the re digram maps into an n* combination. It should be noted that the observed video image would not have evolved if the subject digram was er or ri. The appropriate confusion data related to the CRS events can be quantified in the form

$$P_c(L_jL_{j+1}|L_iL_{i+1}). \tag{35}$$

To structure an effective and efficient evaluation methodology for the likelihood factor (26), one must stress the commonality of its possible constituents. Essentially, each of the candidate aliasing effects can be represented as a confusion probability. The only additional factor that must be accommodated in the analysis is that, unlike the treatment of simple substitution shown in the example, a predictable one-to-one correspondence

Table 16 Catenation segmentation.

br		el		fr	
do		en		ir	
en		er	e	la	
ff		es		mr	
fr		ne		or	r
gu	*	ue		ra	
la		mr	k	rg	
ok		ok		ro	
or		el		rs	
rv		kl	e	sa	
sa		la		ry— y	
tn					
ja		kl		re— v	
sa	a	ra	m		
ta		rt			
ck		ne— n			
ch	c				
ci		io			
		jo	o		
dy— d		or			
		em— p			

Machine recognition → n

Machine recognition → * (rejection)

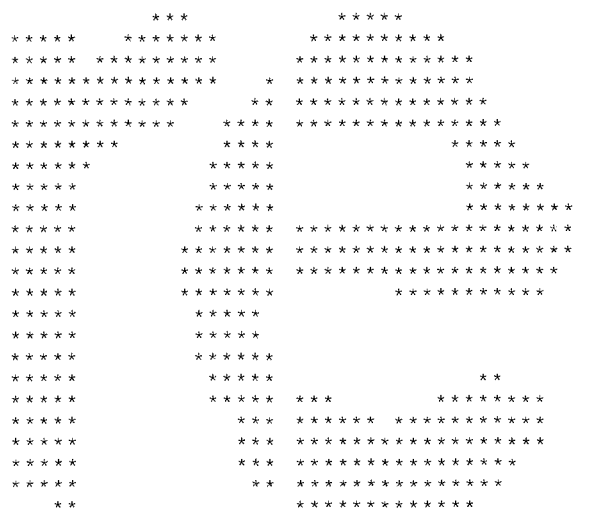


Figure 8 Binary video scan of a character pair that resulted in crowding segmentation.

between characters in a dictionary entry and the garbled data field no longer strictly holds. This, of course, follows because the occurrence of an HSS error in one character of a dictionary word creates two characters in its garbled representation. The converse holds for CS error. Implicit in each of the previous mis-segmentation possibilities is the requirement to realign the remainder of the garbled address data to compensate for the character misalignment effect incurred due to the presence of a segmentation error.

To configure a reliable algorithm that accommodates the segmentation considerations, two innovations must be appended to the standard procedures, as applied in (26), when evaluating the likelihood factor. The innovations are exception character-pair flagging and the use of regional context. These innovations are explained as follows.

Exception character and character-pair flagging

There are about six HSS prone characters and 30 to 50 CS prone character pairs. By themselves, they constitute only a small part of the alpha composition of the dictionary. Unless a flag is encountered, the likelihood factor analysis proceeds as if character substitution were the only garbling factor to be considered. Only when a flag is encountered does the routine branch into the special logic for possible segmentation error occurrences.

Special characters can be inserted into each word where its segmentation prone characters or character pairs exist. This, however, has the drawback of increasing the average word length and destroying the compactness of the dictionary, which is important for I/O efficiency. Hence, to accommodate the flagging and storage requirements, a special alpha character storage convention is adopted. Each alpha character is stored using only five of the eight bits usually used to store a character. The other three bits are then used to provide eight flag-code combinations, two of which are delegated for HSS character and CS character pairs.

If, for display purposes, the HSS code is denoted by "!", then, for example, the word WALSTON, which contains both HSS and CS occurrences, would be stored in the dictionary as

|W!| |A| |L| |S?| |T?| |O!| |N!|.

Use of regional context

The key to HSS and CS flags being used effectively in the likelihood factor computation is regional context.

Unless an alpha character in a dictionary entry is preceded by a flag, it is assumed that it enters into the likelihood factor analysis as an event of the form $P_c(L_i|L_j)$, where $i = j$ is among the possibilities. This implies that only the possibility of simple substitution is being assumed. If a flag is encountered in the dictionary entry, then the analysis associated with the likelihood factor must address, in addition to simple substitution, the possibility of segmentation. The use of regional context now enters as follows.

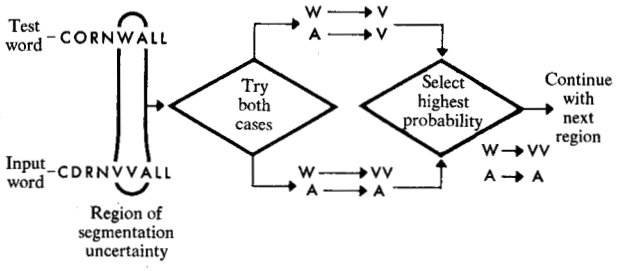


Figure 9 Example of regional context.

Assume that the flag indicates the possibility of HSS. At this point, three possibilities exist; they are

1 and 2: $P_c(L_j|L_{i_{seg}})$;

3: $P_c(L_jL_{j+1}|L_{i_{seg}})$,

where $j = i_{seg}$ is included. To proceed with the evaluation of the likelihood factor, a decision must be made between events 1 and 2 and the HSS event posed by 3. The decision mechanism rests on the use of regional context.

If condition 3 is correct, then the remainder of the garbled character string must be left-adjusted one position. This changes the existing correspondence between the characters of the garbled alpha string and the dictionary entry. The change (shift) in regional context is reflected in terms of the likelihood factor constituents as

$$P_c(L_jL_{j+1}|L_{i_{seg}}) \cdot P_c(L_{j+2}|L_{i+1}). \tag{36}$$

If condition 1 or 2 is correct, then the regional context is not disturbed and the likelihood factor constituents corresponding to those in (36) are

$$P_c(L_j|L_{i_{seg}}) \cdot P_c(L_{j+1}|L_{i+1}). \tag{37}$$

The decision concerning the presence or absence of HSS then follows by whichever formulation, (36) or (37), yields the larger probability value.

Similarly, if a flag denotes the presence of a character pair that is prone to CS, then

$$P_c(L_j|L_iL_{i+1}) \cdot P_c(L_{j+1}|L_{i+2}) \tag{38}$$

would denote the related constituents of the likelihood factor under that supposition. This expression would be evaluated relative to

$$P_c(L_j|L_i) \cdot P_c(L_{j+1}|L_{i+1}), \tag{39}$$

which is the likelihood factor evaluation progression that would exist in the absence of a CS misread. The decision criterion, as with HSS misread, would be based on the relative probabilities of the respective expressions. Figure 9 further illustrates the implementation of regional context in segmentation type error correction.

For crowding segmentation, the evaluation of the likelihood factor in (26) follows by also denoting the possible CRS prone character digrams in the dictionary entries by a special character. The evaluation progression at this point then considers the two possibilities,

$P_c(L_{j+1}|L_{i+1}) \cdot P_c(L_j|L_i)$ and

$P_c(L_{j+1}L_j|L_{i+1}L_i)$.

Because, unlike the HSS and CSS evaluations, no change in character string length must be taken into account, the choice of how to treat and include the digrams $L_{i+1}L_i$ in the likelihood calculation follows from whichever of the above expressions yields the larger probability.

• *Programming expediencies for decreasing computing requirements*

The RCML procedure may have to be evoked many times during the OCR processing of an average document. It therefore becomes important to minimize the related computation. A substantial decrease in computing requirement is accrued by appending to the basic error correction algorithm a comprehensive series of dictionary candidate screening processes. In concert, these logical procedures actually increase algorithm reliability while decreasing computing time.

The package of logical screening processes includes a premature termination threshold and a Go/No Go threshold.

Premature termination threshold

Premature termination effects a major decrease in computation performed by the error correction function by terminating the consideration of a dictionary entry as soon as its likelihood factor drops below a fixed percentage of the largest likelihood factor obtained in the analysis so far. Recall that the likelihood factor (26) measures, in a probabilistic fashion, the degree of match or mismatch between a garbled word and a dictionary-fetch entry. The evaluation format of the likelihood factor lends itself naturally to this type of thresholding; it can be evaluated as a series of multiplications of confusion probabilities (values between 0 and 1). As with any multiplicative series of terms less than one, each successive multiplication decreases the value of the existing product.

The normal tolerance level is taken to be 10 percent of the largest likelihood computed so far in the analysis. This threshold markedly decreases computation, once the RCML procedure has encountered the correct dictionary entry.

Following is an example of the thresholding implementation. If it is assumed that an 80-percent likelihood factor is the largest so far in the analysis, then the toler-

ance level is eight percent. Hence, for the error correction routine to continue, consideration of any forthcoming entry requires that the entry maintain a likelihood value of at least 72 percent. Most dictionary entries show a sufficient incompatibility within one or two characters to drop below the tolerance level and are therefore terminated.

Go/No Go threshold

This thresholding operation is akin to the premature termination criterion. It focuses, however, on the other end of the probability spectrum. It allows consideration of a dictionary candidate to be terminated as soon as it drops below an absolute minimum threshold. Its value follows from the fact that no matter how dissimilar a garbled word and a dictionary entry are, the likelihood factor is computable.

Fortunately, such a likelihood computation quickly converges toward zero. By placing a lower limit on the acceptable likelihood values, the term-by-term evaluation of an only casually related dictionary entry can be terminated if it drops below the threshold.

Performance curve

In concert, the Go/No Go and the premature termination thresholds make a significant difference in the number of operations that, on the average, are performed during an error correction operation. Figure 10 shows the cumulative operations performed for the maximum likelihood computation with and without thresholding. This graph reflects an average case with the correct entry encountered halfway through the search. It should be noted that, of the entries which enter error processing, only about one in 20 is processed beyond the Go/No Go and premature termination thresholds.

Although the discussion and analysis of the evaluation of the likelihood factor have been posed in terms of a series of multiplicative operations, in reality, for further computational efficiency, they are performed in the computer as an addition of prestored logarithmic values (logs) of probabilities. This procedure replaces the relatively slow operation of multiplication with the high-speed add instruction. In itself, the use of the addition of logs of probabilities, instead of direct multiplication, decreases the computational requirements by a factor of about six. By virtue of all the computational expediencies, the complete Contextual Word Recognition Postprocessor can be accommodated for many text processing applications on relatively small processors.

• *Reasonability criterion*

At the completion of RCML processing two possibilities exist: Either

1. no candidate entry has passed the threshold criterion or

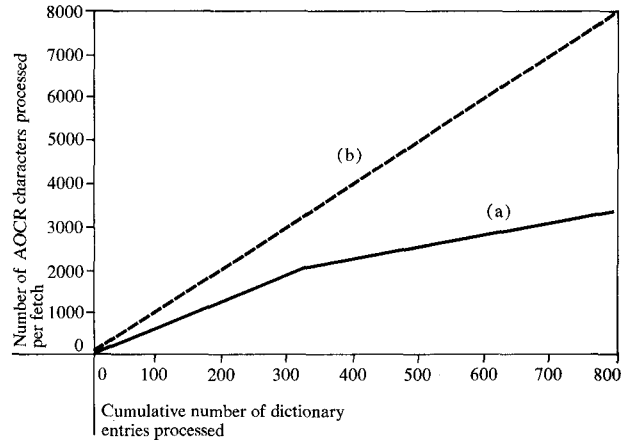


Figure 10 Comparison of computational performance, at 10 characters per entry, (a) with and (b) without thresholding.

2. one or more dictionary entries have completed RCML processing.

If condition 1 exists, then no error correction is possible and a reject is indicated. In most cases, this situation is the result of the correct form of the garbled word not being present in the data block fetched. Condition 2 is the more common case. It indicates, from a likelihood-of-match standpoint, that one or more of the dictionary entries reviewed might have been misread into the garbled word.

The correct choice may or may not be in the final set of dictionary candidates. To control, at this point, the potential for miscorrection, a reasonability criterion requires a match in at least 50 percent of the character positions for a final candidate to be accepted as the correct form of the garbled word. The output of the RCML procedure is reviewed in the order of high-to-low probability score, and the first to pass is accepted. If none of the final candidates passes, a reject is indicated.

The accuracy and reliability of RCML error correction has been assessed by offline simulation using recognition tapes from the AOCR. The following was observed in a run against a dictionary of 62000 entries when the correct word was present in the fetch:

a. 97.7 percent correct;
b. 0.7 percent rejected (rejection implies no dictionary candidate was accepted);
c. 1.6 percent incorrect (incorrect selection implies the wrong word was associated with the garbled word).

• *RCML error correction technique simulation results*

As mentioned previously, the RCML error correction procedure was applied to actual AOCR garbled mail address data. Via this technique, substitution, character

Table 17 AOCR garbled words corrected by Bayesian technique.

Garbled word	*Comment*	*Correct word chosen*
LONISVE*L	Substitution	LOUISVILLE
NEWI*RK	Substitution	NEW YORK
MICHIAAN*	Splitting segmentation	MICHIGAN
COL*BUS	Substitution	COLUMBUS
*OWAM	Rejection	IOWA
TH*RD	Rejection	THIRD
ATLAN*IE	Substitution	ATLANTIC
*O**ERCE	Rejection	COMMERCE
ERONX	Substitution	BRONX
AVENUE OF AMRICAS	Catenation segmentation	AVENUE OF THE AMERICAS
GRATNECTK	Catenation segmentation	GREATNECK
BROAD**NY	Splitting segmentation	BROADWAY
H*IDSON	Splitting segmentation	HUDSON
RANKLIN	No read	FRANKLIN
MONI*MENT	Splitting segmentation	MONUMENT

Performance on error words
Correct selection – 97.7%
Rejection – 0.7%
Incorrect selection – 1.6%

rejection, horizontal splitting segmentation, catenation segmentation, and crowding segmentation errors were treated. The results contained in Table 17 demonstrate the effectiveness and flexibility of the technique.

6. Conclusions

The Contextual Word Recognition Postprocessor technology provides a flexible, real-time, reliable system for performing multifont OCR error correction. Integrated into a multifont OCR architecture, these cybernetic procedures offer a new avenue for overcoming the reliability problems that have plagued operational implementation of omnifont recognition processing.

Tests conducted over a period of a year have shown that the Contextual Word Recognition Postprocessor is a significant advance in the state of the art in automatic error correction. In light of the published results of Damereau [3], Vossler [4], the IBM Research Division [5], Szanser [6], and Hahn [7], the error correction technology discussed in this paper offers two unique advantages.

First, the Contextual Word Recognition Postprocessor offers a computationally practical and easily implementable method of coupling a large, almost open-ended, error correction dictionary with later phases of the automatic error recovery process. This is achieved using the Binary Reference Matrix for verification purposes and the Vector Fetch/Word Group methodology for accessing the error correcting dictionary. The error correction dictionary used in our tests was more than six times larger than the largest such file used by any of the referenced experimenters.

Second, the Regional Context Maximum Likelihood error correction procedure yielded by far the highest error recovery reliability of any of the above techniques. The mis-association rate of the RCML correction was less than half the rate reported by any of the preceding investigators.

Similar enhancements in the state of the art of automatic error correction can be assessed relative to published results related to digram and trigram inference techniques for effecting garbled word error recovery. Direct "apples-to-apples" comparison of contextual word recognition is more difficult, however, because work in this vein is mainly posed as recognition enhancement instead of error recovery and no dictionary word list as such is used. Published results related to digram and trigram methods, however, clearly indicate compromised reliability and deterioration of performance when large vocabularies must be used. The major published results in digram and trigram techniques are Vossler [4], Carlson [8], Raviv [9], Cornew [10], and Riseman [11].

Acknowledgments

The authors gratefully acknowledge the cooperation extended to them by J. E. Hoel and J. Jancin in getting this paper ready for publication; the contributions of

A. M. Chaires and J. M. Ciconte in the early stages of this work; and their indebtedness to D. B. Convis, C. J. Houneycutt, S. Klein, R. K. Lewis, J. L. Peace, M. A. Reese, and J. T. Stringer for the teamwork which made these results possible.

References

1. *IBM Mathematical Programming System,* Program 5734-XM-4, IBM Data Processing Division, White Plains, NY.
2. G. Dewey, *Relative Frequency of English Speech Sounds,* Harvard University Press, Cambridge, MA, 1923.
3. F. J. Damerau, "A Technique for Computer Detection and Correction of Spelling Errors," *Comm. ACM* **7,** 171 (1964).
4. C. M. Vossler and J. J. Branston, "The Use of Context for Correcting Garbled English Text," *Proceedings of the 19th National Conference of the ACM,* Philadelphia, PA, August 1964, p. D2.4-1.
5. "Advanced Electro Optical Recognition Techniques," report of contract RE 140-68 between the U.S. Post Office Department and the IBM Research Division, Yorktown Heights, NY, June 1968–June 1969.
6. A. J. Szanser, *Automatic Error-Correction in Natural Languages,* Pergammon Press, London, 1970.
7. P. M. Hahn, "A Best Match File Search Procedure for Postal Dictionaries," *Proceedings of the International Conference on Cybernetics and Society,* Washington, DC, October 1972, p. 512.
8. G. Carlson, "Techniques for Replacing Characters That Are Garbled on Input," *Proceedings of the AFIPS Spring Joint Computer Conference,* Boston, MA, 1966, p. 189.
9. J. Raviv, "Decision Making in Markov Chains Applied to the Problem of Pattern Recognition," *IEEE Trans. Information Theory* **IT-13,** 536 (1967).
10. R. W. Cornew, "A Statistical Method of Spelling Correction," *Information and Control* **12,** 79 (1968).
11. E. M. Riseman and R. W. Ehrich, "Contextual Word Recognition Using Binary Diagrams," IEEE Trans. Computers **C-20,** 397 (1971).

Received May 17, 1974; revised October 15, 1974

The authors are located at the IBM Federal Systems Division, 18100 Frederick Pike, Gaithersburg, Maryland 20760.