Since each AND operation is changed to an OR operation subjected to an odd number of complementations, it is equivalent to the original AND operation. Q.E.D.

Example 8:

$$f = (A + B\bar{C} + D(E + F)) = (A + (B\bar{C}) + (D \mid E + F \mid)) \quad (27a)$$

$$= ((A + ((B)' + (C')')' + ((D)' + (E + F)')')')'. \quad (27b)$$

Equation (27b) is the same as (25); its circuit is shown as Fig. 3. The remarks after Example 6 apply in a similar manner to NOR transformations.

V. CONCLUSION

A generalized DeMorgan's theorem presented in Section II simplifies considerably the transformation of logic equations with complemented groups to the sum of product form.

The inverse transformations, from basic logic equations to the NAND and NOR forms, are generalized by the theorems given in Sections III and IV.

The theorems can be used for transformations by computers.

ACKNOWLEDGMENT

The author wishes to thank Dr. W. D. Little and A. S. Corbett for numerous suggestions.

REFERENCES

[1] W. E. Wickes, *Logic Design with Integrated Circuits*. New York: Wiley, 1968.
[2] D. D. Givone, *Introduction to Switching Circuit Theory*. New York: McGraw-Hill, 1970.
[3] S. H. Caldwell, *Switching Circuits and Logic Design*. New York: McGraw-Hill, 1958.
[4] G. A. Maley and J. Earle, *The Logic Design of Transistor Digital Computers*. Englewood Cliffs, N. J.: Prentice-Hall, 1963.
[5] P. M. Kintner, *Electronic Digital Techniques*. New York: McGraw-Hill, 1968.
[6] E. J. McCluskey, *Introduction to the Theory of Switching Circuits*. New York: McGraw-Hill, 1965.

H. Christian Wehrfritz (S'64–M'66) was born in Hetzelsdorf, Germany, on September 14, 1930. He received the B.Sc. degree from the University of Alberta, Edmonton, Alta., Canada, in 1965, and the M.A.Sc. degree from the University of Waterloo, Waterloo, Ont., Canada, in 1970, both in electrical engineering. He did graduate work at Utah State University, Logan, in 1968.

From 1965 to 1966 he was an Electrical Engineer at Canadian General Electric, Toronto, Ont., Canada, where he worked on the installation of electronic control equipment for steel mills. In 1966 he joined Mohawk College of Applied Arts and Technology, Hamilton, Ont., Canada, as a Lecturer of Electrical Engineering. In 1969 he became Supervisor in the Department of Control Systems and Instrumentation at Mohawk College. He specializes in digital controls and computer electronics.

Mr. Wehrfritz is a Registered Professional Engineer in the Province of Ontario, Canada. From 1972 to 1973 he was chairman of the Chapter on Computers and Automation of the Hamilton Section of the IEEE.

A Contextual Postprocessing System for Error Correction Using Binary *n*-Grams

EDWARD M. RISEMAN, MEMBER, IEEE, AND ALLEN R. HANSON, MEMBER, IEEE

***Abstract*—The effectiveness of various forms of contextual information in a postprocessing system for detection and correction of errors in words is examined. Various algorithms utilizing context are considered, from a dictionary algorithm which has available the maximum amount of information, to a set of contextual algorithms utilizing positional binary *n*-gram statistics. The latter information differs from the usual *n*-gram letter statistics in that the probabilities are position-dependent and each is quantized to 1 or 0, depending upon whether or not it is nonzero. This type of information is extremely compact and the computation for error correction is orders of magnitude less than that required by the dictionary algorithm.**

The technique described can allow relatively poor classifiers to become reliable systems by drastically cutting error rates with only modest reject rates. Experimental results are presented on the error, correction, and reject rates that are achievable as a function of the type of contextual information employed, and the size of the data base from which this information is obtained. The most powerful algorithms detect almost all errors and correct between 60 and 95 percent of them. As an example, a set of binary positional trigrams operating on a set of 800 6-letter words can reduce a word error rate of *47 percent* to a reject rate of 8.9 percent and an error rate of

Manuscript received October 26, 1972; revised July 5, 1973. This work was supported by the Office of Naval Research under Grant ONR 049-332.

E. M. Riseman is with the Department of Computer and Information Science, University of Massachusetts, Amherst, Mass. 01002.

A. R. Hanson is with the School of Language and Communication, Hampshire College, Amherst, Mass. 01002.

only 0.38 percent. Several modifications to produce more powerful systems are discussed.

***Index Terms*—Character recognition, context, contextual postprocessor, dictionary, error correction, error detection, pattern classification, pattern recognition, positional binary n-grams.**

I. INTRODUCTION

DESPITE the vast research effort in the field of pattern recognition [1]–[3], there has been relatively little utilization of contextual information [4]–[26]. Humans use context in reading so extensively that quite often they do not even realize they have read a misspelled word correctly. In fact, the recognition rate on isolated characters where there is no contextual information is only 96 percent [27]. In the area of character recognition, the potential aid of contextual techniques appears great.

Generally, contextual techniques have been incorporated in character recognition systems by following a classifier with an independently operating contextual postprocessor for detection and/or correction of errors. Most of the contextual analyses are restricted to processing a single word on the basis of the characters within that word.[1] It has been estimated that the English language is over 50 percent redundant [28]; consequently, the characters surrounding those in error might be sufficient to detect the errors and suggest the corrections. Two approaches that will be examined utilize somewhat different types of information: 1) a dictionary of words, and 2) a set of binary matrices that approximate the structure of the dictionary. The use of a dictionary for error detection and correction can be quite straightforward. An error is detected when a word is not in the dictionary; the error word can be changed to the most similar word in the dictionary in an attempt at correction.

Several researchers [7], [9]–[14], [18] attempted to overcome the problems associated with using a dictionary by extracting structural information about the words that would have appeared in the dictionary. The probability of digrams and trigrams (letter pairs and triplets) is reasonably constant across a large sample of text and, therefore, can be used to characterize the language under consideration. One can view this information as a first- and second-order Markov process approximating the word structure of the language under consideration. The use of digram, trigram, and even quadgram probabilities has occurred in both character recognition and error correction. The difficulty here, once again, is whether the resultant error-reject rates and improvement in computational efficiency justifies the amount of storage required if the technique uses the 26^3 (17576) trigram probabilities. The utilization of n-grams for n greater than 3 is usually prohibitive due to the vast storage required ($26^4 \cong$ 456 000 words).

Techniques exist for compacting much of the useful information in n-grams. Almost 50 percent of digram probabilities are 0; a much lower density of nonzero entries occurs among trigram probabilities [7]. One can greatly reduce the required storage by quantizing the probability to 0 or 1, depending on whether it is 0 or nonzero, respectively [18]; the matrix of 26^n bits is referred to as a binary n-gram. Thus much of the information is retained, but its compactness makes feasible the storage and use of the much vaster amounts of positional contextual statistics.

The second type of information that we will utilize is a set of positional binary n-grams. As an example, a positional binary *di*gram D_{ij} is a 26×26 binary matrix denoting which letter *pairs* have a nonzero probability of occurrence in positions i and j in the allowable input words. The set of all positional binary digrams includes a binary matrix for every pair of positions. Positional binary n-grams can be used to detect words in error, fix the position(s) of the error, and often determine the characters that will correct the word. For comparative purposes we will also include nonpositional binary n-grams in this category.

Generally, dictionary methods have proven to be the most effective for error detection and correction. The difficulty with the use of the dictionary is that searches and comparisons for error correction are time consuming and increase with the size of the dictionary. This does not imply that there are not many applications where a modest dictionary could be economically utilized in a special purpose recognition system [18]. The positional binary n-grams may require more storage than a dictionary, but are much more compact than the storage of the probabilities of n-grams used by previous researchers. Positional binary n-grams exhibit strong advantages over the use of the usual probabilistic information or the dictionary; the process of collecting and storing contextual statistical information is simplified and the computational complexity is reduced. Algorithms using this information can approach the effectiveness of the dictionary for error detection and correction.

This paper examines the relative effectiveness of the various forms of contextual information in a postprocessing system for error detection and correction. The algorithms will be compared on the basis of the resulting error-reject rates after processing, as well as their computational efficiency and storage requirements. Errors that are undetectable and uncorrectable are discussed with experimental results determining the effectiveness of the procedures as a function of the size of the dictionary from which the input words are selected.

II. USE OF DICTIONARY

Let us consider the manner in which a dictionary could be used to correct errors in samples of words from the dictionary. Only the characters within the word will be

[1] The syntax and semantics of the surrounding words, sentences, and paragraphs might also contain valuable information. However, only a few systems have employed this type of context, usually in programming languages where the syntax and semantics are well defined [19]–[21].

used to correct the characters in error; thus the dictionary itself is assumed to be all the contextual information available and, in this sense, will be considered complete information.[2]

Given a sample in which it is unknown whether or not an error is present, the most straightforward procedure is to look up the word in the dictionary. If the word is in the dictionary, it does not necessarily mean that no error occurred. An error might have transformed one dictionary word into another dictionary word, a case in which the error is *inherently* undetectable without the wider use of context previously mentioned. One has no choice but to assume the word is correct.

If the word is not in the dictionary, yet it is known to be a sample of a dictionary word, then an error has been detected. If the word is to be corrected, one might search the dictionary for words that are very similar. If one is utilizing a character recognition system with a reasonably high initial recognition rate, most of the words in error will contain only one error. A few will contain two errors and very few will contain more than two errors. For example, assuming character error rates of 5 and 10 percent only 0.8 and 3.4 percent, respectively, of 6 letter words would have more than 2 errors. If character errors are independent, then it is clear that the bulk of words in error involve few symbol errors.

Suppose we assume that a word in error involves only one misrecognized character. Then, if the most similar word in the dictionary differs by a single character from the sample and no other word differs by one character, it must be the correct version of the word. If this assumption of the single character error is not valid (i.e., there are 2 or more errors), the word in error might be improperly transformed into another incorrect word. In general, a word is corrected to the dictionary word that it is most similar to if there is no other candidate of equal similarity; otherwise, the word in error is rejected. In those cases where there are no words differing by a single character *and* only one differing by two characters, correction is made to this latter word. Note that the entire dictionary must be searched to determine the set of dictionary words that are *most* similar to the word in error. We will say it is *inherently uncorrectable* given the dictionary information if there are several words that are equally most similar. One could choose the word with the highest *a priori* probability if this information were available, but more often the word would be rejected because the expected probability of error would be too great. The set of words still in error after this entire process would consist of inherently undetectable errors and words in error that were improperly corrected.

As the dictionary is enlarged, one can see that the rates of inherently undetectable and uncorrectable errors increase since it is more likely: 1) that a random error will produce some other word in the dictionary, and 2) that there will be more words similar to the one in error, making it less likely to be able to correct the word. As the dictionary grows, the amount of storage required increases; the amount of computation also increases, since one must first look up the unknown word in the dictionary and then search the entire dictionary if it is absent. Some of these disadvantages will be overcome by the use of positional binary n-gram statistics.

[2] The only further nonsemantic information that could be available would be the word probabilities.

III. POSITIONAL BINARY n-GRAMS

Clearly, it would be advantageous to make the amount of storage and the computation time required independent of the dictionary. On the other hand, once storage is fixed and the dictionary grows sufficiently large, the performance of any error detection and correction algorithm will begin to suffer since all of the information in the dictionary is not retained. The evaluation of the method then becomes an evaluation of the tradeoff between the savings of reduced computation and/or storage with the increase in the error rate.

Most of the contextual algorithms that have performed well in the past did so at the expense of long computation times or a large amount of storage. They made use of a complete dictionary or else extracted the information in a probabilistic form in terms of the probability of trigrams and quadgrams. However, the following technique, developed by Riseman and Ehrich [18], is an effective means of extracting large amounts of the information from the dictionary in a readily retrievable form at a relatively modest cost of storage.

The context algorithms utilize information extracted from the dictionary. This information (called the *syntax* of the dictionary) is in the form of a data base of quantized n-grams [18]. These n-grams, for any n, may be either positional or nonpositional, depending upon the amount of information stored and the method by which the information is extracted from the dictionary. For example, corresponding to the commonly used nonpositional letter pair (digram) probabilities is a 27×27 binary matrix whose entries are 1 if and only if that corresponding letter pair appears in some word in the dictionary.[3] Positional digram matrices, on the other hand, are constructed so as to take into consideration the relative positions of the letters within words. One binary matrix exists for each distinct pair of positions, and for 6-letter words, 15 $(6 \times 5/2)$ of them are required to contain all pairwise positional information contained in the dictionary. For illustrative purposes, the construction and use of binary digrams will be described first.

The dictionary may be partitioned by word length. For each of these subdictionaries, a pair of letter positions i and j can be used to define a 26×26 binary matrix D_{ij} called a *binary digram*. The (k,l)th position of D_{ij} is defined to be 1 only if some word in the dictionary contains letter k in the ith position and letter l in the jth

[3] Note that blanks must be considered in nonpositional information requiring 27 characters; positional n-grams using 26 characters automatically include this information since a blank always precedes the first position and follows the last position.

position; otherwise, it has a value 0. Thus the probabilities of letter pairs in all pairs of positions have been quantized to binary values of 1 or 0.

The set of all positional binary digrams allows one to obtain the same information that would be obtained from an associative memory that could only be asked the following type of question: is there some word in the dictionary that has letters α and β in positions i and j, respectively? This set of binary matrices contains a great deal of information due to the fact that they tend to be fairly sparse. In fact, it has been pointed out [7] that over 40 percent of the 676 letter pairs never occur in contiguous positions in any words in the English language. This leads one to believe that the positional binary digrams are even sparser. Binary digrams involving adjacent positions are generally less dense than others, while binary digrams involving the first two and last two positions are the sparsest. The first statement is intuitive if one considers, for example, that only u can immediately follow q but that many choices are available for possible letters in widely separated positions. The second statement means that there are fewer word beginning and ending pairs than pairs in other positions. To our knowledge, these positional statistics have never been collected across the entire English language, although a fairly large subset of the common six-letter words in English was used in experiments described later in this paper.

The extension to binary trigrams or a set of positional binary trigrams is straightforward. In fact, the dictionary of words of length l can be viewed as a binary l-gram organized in a different fashion. Since very few of the 26^l possible l-letter words actually are in the dictionary, rather than construct an exceptionally large and sparse matrix, the dictionary itself is a list of the nonzero entries. In essence, the techniques that will be described below involve approximating the information in l-grams with binary n-grams, $1 < n < l$.

A. Error Detection

Given the set of positional binary digrams, one can determine that an error has occurred if the dictionary syntax in the form of binary digrams has been violated. Assume a sample word consists of the following characters: $\alpha_{i_1},\alpha_{i_2},\cdots,\alpha_{i_l}$ where α_{i_k} is a member of the alphabet. Then we must have

$$D_{jk}(\alpha_{i_j},\alpha_{i_k}) = 1 \qquad \text{for all } 1 \leq j < k \leq l.$$

We are simply ensuring that, *independently* for each pair of positions in the sample, the characters that do appear there also appear in some dictionary word. There may be some errors that are detectable if the dictionary was available (inherently detectable) but that are *undetectable* by the dictionary syntax. An example that illustrates this very simply is the following.

Dictionary	SUT is an undetectable error
SAT	D_{12}: entry for SU is 1 due to SUN
CUT	D_{13}: entry for ST is 1 due to SAT
SUN	D_{23}: entry for UT is 1 due to CUT.

Error Correction

Among those errors that are detectable, the set of binary digrams can be used to attempt correction. Consider a six-letter word, $\alpha_{i_1},\alpha_{i_2},\alpha_{i_3},\alpha_{i_4},\alpha_{i_5},\alpha_{i_6}$, with α_{i_3} the only character that is incorrect. If the binary digrams used are sufficiently sparse, several of them may detect an error. In this example there are five chances to detect an error: $D_{13}(\alpha_{i_1},\alpha_{i_3})$, $D_{23}(\alpha_{i_2},\alpha_{i_3})$, $D_{34}(\alpha_{i_3},\alpha_{i_4})$, $D_{35}(\alpha_{i_3},\alpha_{i_5})$, and $D_{36}(\alpha_{i_3},\alpha_{i_6})$. If at least 2 of them detect an error (say D_{23} and D_{35}) at a minimum the position of the single error is fixed by noting that only position 3 appears more than once in the detection of the error. If some character is to replace α_{i_3}, then each of the five digrams contains information about the admissibility of the various choices. The row $D_{13}(\alpha_{i_1},*)$, where the * varies across all choices, contains a 1 in those places in which a character is allowable (by the dictionary syntax) in position 3, given that α_{i_1} appears in position 1. If we logically intersect the proper rows and columns in the five digrams, we will have utilized all the available information:

$$D_{13}(\alpha_{i_1},*) \wedge D_{23}(\alpha_{i_2},*) \wedge D_{34}'(\alpha_{i_4},*) \wedge D_{35}'(\alpha_{i_5},*) \wedge D_{36}'(\alpha_{i_6},*)$$

where D_{ij}' denotes the transpose of D_{ij}. If the jth entry in this vector is 1, then the substitution of α_j for α_{i_3} will produce a word which is admissible by the dictionary syntax because it will satisfy all the 5 digrams involving position 3. If there is only one entry having a value 1, the choice is clear and the word will be corrected properly (still assuming a single error). In those cases where there are several entries having a value 1, there is ambiguity in the correction process and the word is rejected.

The process that has been described can take place only after the position of the error is fixed. An attempt to fix the position is made by examining the positions of the set of digrams that detect an error. However, it often will be unclear whether one error or multiple errors have occurred in the word. If only one character is in error, then only one position can occur more than once because only a digram involving this position and one other can detect an error. The violation of this condition implies that multiple errors have taken place. No attempt will be made to correct multiple error words using positional binary digrams. Thus we will first describe the simpler process by which the position of a single error is fixed utilizing binary digrams and then generalize this process to the use of positional binary trigrams correcting words with one or two errors.

Let us assume that R of the digrams detect an error; for the rth such digram D_{ij}, construct the subset $\beta_r = \{i,j\}$. This condition implies that an error must have occurred in position i or j or both. If only a single symbol is in error, then there must be some position i that occurs in every β_r. Thus we need only the intersection of these subsets $\beta = \wedge_{r=1}^{l} \beta_r$. If β is empty, then more than one error must have occurred. If there is one entry, then we have certainly fixed the position of the single error and we employ the correction process. Note that actually

there may be more than one error and if the correction process reduces to a single alternative it will improperly correct the word thereby reintroducing an error; fortunately, this event rarely occurred in the experiments conducted.

The final possibility is that β contains two elements; this occurs only in those cases where a single digram detects the error. Again, we assume that only a single error is involved, but now we do not know the position. However, when there is a choice of two positions, the correction algorithm can be applied to each position while assuming the other is correct. If one position has only one possible letter substitution and the other has none, the location of a single error is clear and the correction can be made. This process could be called *position determination by elimination.* In any other case with the use of digrams, the word is considered to be uncorrectable and rejected.

C. Using Positional Binary Trigram

Now the use of the set of positional binary trigrams T_{ijk} will be described. The trigram information is a far more effective approximation of the dictionary information; consequently, the algorithm will be developed to correct both one- and two-error words. If we intersect the set of β_r constructed from each T_{ijk} that detects an error and it is empty, once again, there must be an error in more than one position. However, in any other case, one or more errors may have occurred. To handle the larger number of possibilities we will construct a table denoting the positions involved. For example, assume that D_{123}, D_{134}, D_{135}, and D_{145} all detect an error. This is illustrated in Table I with each row marking the positions of one of the trigrams that detects an error. It should be clear that the set of positions in error must account for at least one X in each row; otherwise, the trigram for that row could not have detected an error. Thus, if a single error occurred, it could only have occurred in position 1. However, there are seven pairs of positions that could have errors and still account for the trigrams that have detected the error. This can easily be determined by enumerating the $\binom{6}{2} = 15$ pairs of positions that could involve 2 errors and checking whether all rows are covered. Table II is a case where 6 of the 20 available positional trigrams detects an error. No single errors are possible and there is only one choice for a two-error word; of course there may be more than two errors.

The correction algorithm is now summarized. First, an attempt will be made to correct the word, assuming only one error has occurred. The intersection of the proper 26-bit rows and columns of the 10 positional trigrams involving the error position yields the alternatives that are admissible substitutions. If there is only one alternative the word is corrected; if there is more than one alternative, the word is rejected. If there are none, then the hypothesis that there is a single error in the current position is invalid. If there are no other possible positions

TABLE I
EXAMPLE OF FIXING THE POSITION OF ERRORS VIA POSITIONAL BINARY TRIGRAMS

1	2	3	4	5	6
x	x	x			
x		x	x		
x		x		x	
x			x	x	

Note: Possibilities for one error—{1}. Possibilities for two errors—{1,2}, {1,3}, {1,4}, {1,5}, {1,6}, {3,4}, {3,5}.

TABLE II
EXAMPLE OF FIXING THE POSITION OF ERRORS VIA POSITIONAL BINARY TRIGRAMS

1	2	3	4	5	6
x	x		x		
x		x		x	
x				x	x
	x	x			x
		x	x	x	
			x	x	x

Note: Possibilities for two errors—{2,5}.

for a single error, an attempt is made to correct the word assuming two errors. Whenever the position of one or two errors is ambiguous, position determination by elimination is tried. Correction is carried out only when it appears that the position of both errors are fixed *and* only one alternative for each error exists. Words that cannot be corrected or are found to contain more than two errors by the position determination process will be rejected. The process of correcting words of other length with n-grams is easily generalized.

Although it appears to be more difficult to fix the position and correct an error using trigrams, this is not necessarily so. In 6-letter words, for example, there are more trigrams (10) than digrams (5) involving each position of a word, and each of these trigrams is much sparser. Consequently, more of them, on the average, will detect an error, fix the position, and develop fewer alternative letters for each position in error.

IV. PREVIOUS EXPERIMENTAL RESULTS

Sitar [7] developed a postprocessing error detection and correction system utilizing digram and trigram statistics from the English language as well as statistics on the type of errors made by the classifier. The latter information can aid the system greatly but must be extracted from each classifier. Low probabilities of letter pairs and triplets were used to detect and fix the positions of errors in text. Digram and trigram information were able to detect 40 and 60 percent of the errors, respectively. However, only 80–90 percent of the positions of the errors could be fixed, and then only 75 percent of these were correctable. Accounting for new errors introduced by the processing, the error rate was reduced by about one-third using trigrams.

Damerau [26] used a dictionary comparison procedure

for word correction. This involved comparing an encoding of each word of the dictionary to an encoding of the sample word and, if there was no match, correcting the sample to a word differing in a single position. Using the potential misspellings detected by the encoding and comparison procedures, Damerau also looked for words that had two adjacent characters interchanged as well as one deleted or inserted character. For single character errors only, the process corrected 95 percent of the errors in words from a 1600 word dictionary. However, this procedure might deteriorate rapidly for multiple error words.

Vossler and Branston [11] compared the use of a dictionary to digram information in an error correction system. Both methods employed the confusion matrix statistics from the classifier. Given the classifier output, the word in the dictionary that has the maximum likelihood of having been input is selected. This procedure requires much storage and computation as well as the *a priori* probability of the occurrence of each dictionary word. The latter information is not usually available and varies with the subject of the text. The second method used digram statistics to approximate the probability of occurrence of the dictionary words and once again choosing that word with the maximum likelihood of having been input. The performance of the dictionary was far superior to digram information. Using a 364-word dictionary, the dictionary correction rate was 93 percent, while the digram correction rate was 45 percent. A procedure combining the 2 processes was used with a 3700-word dictionary. The input was newspaper text, and about 25 percent of these words did not appear in the dictionary; in those cases digram probabilities were used to attempt correction. In this experiment the correction rate was about 44 percent.

Carlson [13] used a very large amount of information—all contiguous positional (positions 1,2,3; positions 2,3,4, etc.) trigram probabilities—to correct errors in English first names. In addition, the position of the character in error is fed to the postprocessor, bypassing the problem of detecting and fixing the position of the error. The error correction rate was about 95 percent.

Finally, Raviv [14] developed a Bayes' decision procedure for classification of a character which balances the information from contextual considerations. Extensive amounts of empirical results were presented. In some experiments, only a slight improvement in the error rate was achieved by going from digram probabilities to trigram probabilities. Since this information is integrated into the classification process and there were not any experiments run using no contextual information, the effect of context in the improvement in the error rate is not available.

V. STORAGE AND COMPUTATIONAL ANALYSIS

Before one can properly compare the performance of the contextual algorithms with the dictionary method in this paper, one must take into consideration their relative cost, namely, the amount of storage and computation necessary for these algorithms. For comparison purposes, it will be assumed that the dictionary consists of six-letter words.

The space required to store the dictionary is directly proportional to the length of the dictionary whereas the storage for n-grams is independent of the dictionary and remains constant. For dictionaries under 12 000 words, the 20 positional trigrams require more storage ($\cong$350K bits). The difference can be an order of magnitude in modest-sized dictionaries (1–3K words); further details can be found in [33]. We feel that there are strong reasons to ignore this disadvantage in the use of positional n-grams and the set of positional trigrams in particular. The total storage required for positional n-grams, $n \leq 3$, is not very great, the cost of high speed storage is plumetting, and since it need only be read-only memory, the cost is even less. On this basis, it is assumed that the difference in storage requirements is not significant and can be ignored.

The computation required is another matter. One can envision vast amounts of information being processed by any character recognition system. If, in addition to this system, one adds a postprocessor, it is clear that speed would be at a premium. The time required for error detection by the dictionary algorithm and the contextual algorithms are of the same order of magnitude. A lexicographic binary search requires $\log_2 n$ dictionary accesses[4] (this process must be time sequential) to determine whether or not a word is in the dictionary. On the other hand, to detect an error, the full set of trigrams may require up to 20 bit accesses or 10 on the average (which may be done in parallel), and fewer for the systems using a subset of trigrams or digrams.

It is in the correction process that the dictionary algorithm suffers greatly. To choose the most similar word to one in error, the entire dictionary must be searched. One can determine a lower bound on the amount of computation for a one-error word that is correctable (i.e., there is only one dictionary word that differs by one character from the incorrect sample). We will assume that all but one of the dictionary words differ from the error sample in the first two character positions and do not have to be examined further. The remaining dictionary word (the correct word) differs in only one position and the whole word must be examined. This means that as a *lower bound* m dictionary accesses and $2(m-1)+6$ comparisons are required, or $3m+4$ operations as a crude measure; similarly, one can derive $4m+3$ operations as a lower bound on the two-error correction process. Table III summarizes the comparison of computational requirements. To correct a word with one error using positional trigrams, one must carry out the following procedure: using the ten trigrams which involve the single position

[4] Note that $\log_2 300 = 8.12$ and $\log_2 2755 = 11.8$.

of the character in error, one must access 10 26-bit rows (or columns), intersect them, and then examine each of the 26 bits (if only one of the bits is 1, correction takes place) for a total of 46 operations. Assuming it takes 30 operations to determine the position of the error (by counting the number of times a position occurs in the set of trigrams detecting an error), one obtains a crude estimate of 76 operations. This computation is independent of the dictionary length. The other computational estimates are obtained in a similar fashion and although they are rough, they clearly point our the orders of magnitude difference in speed.

It is interesting to note that the computational advantages of binary n-grams would diminish only a little for words of greater length. The computation required by the error correction process is only dependent upon the number of n-grams employed. The number of positional n-grams available increases with the length of the word—words of length 9 and 10 yield 84 and 120 positional trigrams, respectively. However, any single position is only covered by 28 and 36 trigrams, respectively. It is pointed out later in the paper that it would probably only be desirable or necessary to employ a much smaller subset of the trigrams. Therefore, if only half of these positional trigrams were employed, the computation would increase by less than a factor of 2, still leaving a huge savings over the dictionary. For words of length shorter than 6, the same type of argument shows that the computational advantage over the dictionary increases. It can be concluded that great savings in computation result independently of the length of the word.

TABLE III
COMPARISON OF COMPUTATIONAL REQUIREMENTS FOR ERROR CORRECTION

size of word set	dictionary - lower bound on operations required		n-grams		
m	1-error correction	2-error correction		1-error correction	2-error correction
300	904	1203			
			15 pos. dig.	53	--
800	2404	3203	1 non-pos. trig.	38	76
			4 pos. trig.	40	75
1300	3904	5203	6 pos. trig.	47	84
			20 pos. trig.	76	114
2755	8269	11023	1 non-pos. quad.	42	84

VI. EXPERIMENTAL RESULTS

The experiments to be described all were carried out on six-letter words as representative of words in the English language. The procedures employed would work in a similar fashion for dictionaries and n-grams constructed for words of other lengths.

A. The Data Base

A list of 2755 6-letter words was compiled and used for all of the following experiments; this list comprised the largest word set employed. In order to determine the effectiveness of the algorithms as a function of the size of the set of input words, subsets of 300, 800, and 1300 words were created from the 2755 words. The 300-word subset consisted of arbitrary 6-letter words compiled by the authors. The remaining 2455 words were obtained from Thorndike–Lorge [31] by selecting all 6-letter words from categories AA-10, inclusive. The 800 and 1300 word subsets consisted of the 300 word subset and words randomly selected from the Thorndike–Lorge list. The full 2755 word set contains a large portion of 6-letter words commonly used. From each of these four subsets, the syntax was extracted and stored in the binary n-grams.

The data base of n-grams chosen for this experiment is:

- 1ND one nonpositional digram requiring $27^2 = 729$ bits of storage;
- 15PD entire set of 15 positional digrams requiring $15 \times 26^2 = 10\,140$ bits;
- 1NT one nonpositional trigram requiring $27^3 = 19\,683$ bits;
- 20PT the entire set of 20 positional trigrams requiring $20 \times 26^3 = 351\,520$ bits;
- 1NQ one nonpositional quadgram requiring $27^4 = 531\,444$ bits.

In addition, two subsets of the set of 20 positional trigrams were chosen:

- 4PT subset of 4 of the 20 positional trigrams requiring $4 \times 26^3 = 70\,304$ bits;
- 6PT subset of 6 of the 20 positional trigrams requiring $6 \times 26^3 = 105\,456$ bits.

Experiment 1—Detection of Errors: The object of this experiment was to test the effectiveness of the context algorithm in detecting errors occurring in the input stream: the effectiveness was to be determined as a function of the type of n-gram utilized and as a function of the size of the word set from which the syntax had been extracted.

A test set of 600 1-, 2-, and 3-error words (for a total of 1800 words) was generated from the word data base by randomly generating the position(s) of the error(s) and the substituted letter(s) according to a uniform distribution. The results of applying the context algorithm to the test set are shown in Table IV. Although the dictionary algorithm is more effective than the set of all positional trigrams (20PT) for error detection, the detectability rates of the 20PT are so high already that the difference is negligible. Depending on the size of the word set, detect-

TABLE IV
ERROR DETECTION RATES (IN PERCENT)

	Size of word set	DICT.	1NQ	20PT	6PT	4PT	1NT	15PD	1ND
1 error words	300	99.8	99.5	99.8	99.3	98.3	96.1	95.0	63.4
	800	99.8	99.2	99.7	98.3	96.0	88.0	84.6	48.4
	1300	99.8	98.0	99.7	97.2	94.0	85.4	78.9	44.2
	2755	99.7	96.9	98.6	93.8	88.2	78.0	69.6	32.5
2 error words	300	100	100	100	100	100	98.5	98.5	81.1
	800	100	100	100	100	99.8	97.0	96.4	68.5
	1300	100	99.8	100	99.7	99.3	95.5	94.0	63.4
	2755	100	99.5	99.8	98.8	98.2	91.6	87.6	49.4
3 error words	300	100	100	100	100	100	99.8	99.8	89.6
	800	100	100	100	100	100	99.3	98.8	81.7
	1300	99.8	99.8	99.8	99.7	99.7	98.5	96.8	76.6
	2755	99.8	99.7	99.7	99.5	99.5	96.2	94.9	63.8

ability rates for 1-error words vary from 98.6 to 99.8 percent and yield almost perfect detection for words with 2 or more errors.

A comparison of the various contextual algorithms shows that the only types of contextual information which do not cause error detectability rates to drop significantly as the dictionary size increases is the 20PT and 1NQ. There is a surprising payoff from the use of 1ND—30–65 percent of 1-error words with the use of just 729 bits. All other types detect at least 95 percent of errors for small word sets. The detection rates of the various n-gram algorithms decreases as the size of the word set increases. This effect is due to the increasing density of the n-gram matrix, which grows as a function of the logarithm of the size of the word set as shown in Fig. 1.

It might be surprising to some to note that the set of all positional trigrams is more effective than 1NQ in error detection (and in error correction, as shown in Experiment 2), despite the far smaller amount of storage that is employed as well as the higher density of 20PT matrices. This seems to point to the importance of retaining positional information.

Experiment 2—Correctability: The object of this experiment was to determine the conditional correction rate of detectable errors; that is, given that an error has been detected, what is the probability of correcting it? Table V indicates conditional correction rates obtained on one- and two-error words. No attempt was made to correct three-error words with any n-grams, nor two-error words via digrams; corrections were actually made *only* when one letter was possible for the positions in error.

Once again, the dictionary and 20PT algorithms were best. The contextual algorithm employing the entire set of trigrams yielded 1-error correction rates varying from 61 to 95 percent of the errors as a function of the size of the word set. The correction rate for 2-error words drops to the 34–83 percent range. In the case of the largest word sets, the dictionary algorithm corrects about 20 percent and 12 percent more of the 1- and 2-error words, respectively. The dictionary algorithm showed a sizable improvement over the 20PT only in the case of the larger word sets. All the other types of contextual information produce significantly worse results.

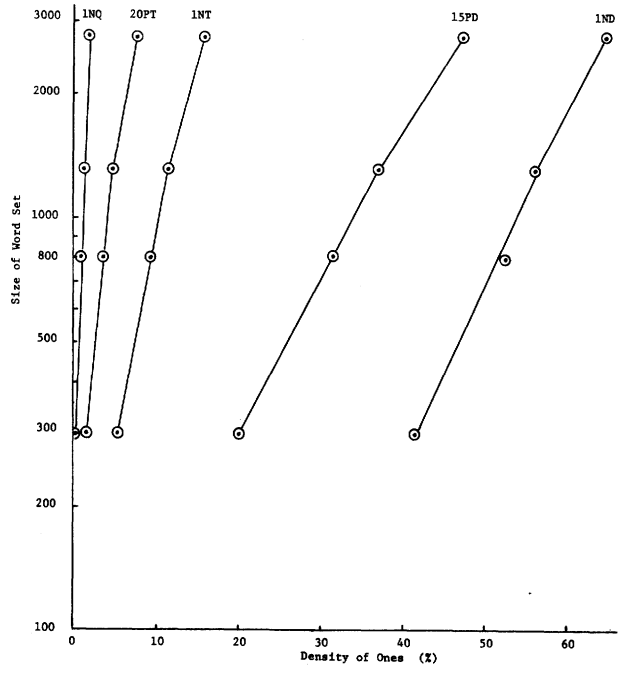


Fig. 1. Density of 1's in binary n-grams as a functions of the size of the word set.

Experiment 3—Analysis of Uncorrectable Errors: A deeper analysis of the errors that were detectable but uncorrectable by the 20PT algorithm was performed. In this experiment correction was attempted on a total of 45 000 1-error words using all positional trigrams and the cause of uncorrectability was determined. Some of the errors were uncorrectable because the position of the error was fixed but more than one letter could be used to correct the error. In most of these cases there were very few choices; in fact, a large proportion of this type of uncorrectable error only had two or three choices, as is summarized in the left-hand portion of Table VI.

The other type of uncorrectable error occurs when the position of the error is uncertain. It has been pointed out that if there are two positions in which an error could have occurred, correction is attempted in both positions; some of these cases are eliminated by determining that no character can correct an error in one of the positions. Errors still remain uncorrectable when it cannot be narrowed down to one position via this elimination process. This leads to a distribution of possible substitution characters in multiple positions, as summarized in the right-hand portion of Table VI. In this portion of the table i/j denotes that there are i alternative characters admissible in one position and j characters in a second position. Once again, the cases where there are a total of two or three possible characters distributed across the positions account for a large fraction of the total. The remainder column includes those cases where there are more than two possible positions.

These results are extremely important since they point out that an attempt at correction is feasible in most of the

TABLE V
CONDITIONAL ERROR CORRECTION RATES (IN PERCENT)

	Size of word set	DICT	INQ	20PT	6PT	4PT	INT	15PD	IND
1 error words	300	95.7	88.0	95.5	86.0	73.2	36.4	28.6	.5
	800	92.2	73.9	88.1	59.7	44.5	16.1	6.7	.3
	1300	89.8	65.4	81.4	46.9	34.3	8.0	4.4	0
	2755	84.1	47.8	62.4	26.8	19.4	2.8	.5	0
2 error words	300	84.0	63.1	83.0	64.1	43.2	7.6	---	---
	800	69.5	32.8	65.1	28.4	15.7	2.0	---	---
	1300	60.9	20.8	54.0	18.2	9.2	.7	---	---
	2755	46.8	7.5	34.1	4.6	2.4	0	---	---

TABLE VI
ANALYSIS OF UNCORRECTABLE ERRORS

size of word set	number of uncorrectable errors out of 45,000	number uncorrectable because more than one letter choice				number uncorrectable because position indeterminate				percentage of uncorrectable in which number of choices is 2 or 3
		number of choices			total of 2 and 3	1/1*	1/2*	remainder	total of 1/1 and 1/2	
		2	3	more than 3						
300	1836	1370	0	0	1370	431	31	4	462	99.8
800	4771	3228	226	36	3454	906	240	135	1146	96.4
1300	8332	5218	765	214	5983	1158	519	448	1677	91.2
2755	16139	7551	2540	1753	10091	1283	951	2061	2234	76.4

Note: i/j means that i characters may fit in one position and j characters may fit in the second position.

uncorrectable cases, due to the small number of alternatives. Additional information might be employed to resolve this ambiguity. One possibility is to employ probabilities from the confusion matrix of the classifier. Typically, when a mistake is made, there are only several characters that the correct letter could have been. By choosing the most likely letter to have produced the error from the two or three choices usually available, a large portion of the uncorrectable cases might be satisfactorily resolved. In order to determine the potential impact of this process, in Experiment 4 a forced error rate was computed under the assumption that when 2 choices exist, 0.8 of the choices could be resolved favorably, and in the case of n choices, $(0.8)^{n-1}$ could be resolved favorably.

Experiment 4—Error and Reject Rates: Up to this point, we have examined the ability of the system to detect and correct words with one, two, and three errors. Any classifier with a given character error rate will output some proportion of words with a distribution of 1,2,3,$\cdots$ error words. Since the postprocessor does not know *a priori* the number of errors in a word, a k-error word might mistakenly be detected as a one-error word, $k > 1$, and then improperly corrected; similarly, a j-error word might be mistaken for a two-error word, $j > 2$, and improperly corrected. Expected word error and reject rates of the contextual postprocessor can be computed in a straightforward manner from detectability and correctability data [33].

Since the 20PT algorithm is the only n-gram algorithm that yields performance reasonably close to the dictionary algorithm, we will compare them more closely. Table VII gives the actual percentages of errors that are corrected, rejected, and remain as errors after processing with 20PT and with the dictionary algorithm. Also computed is the percentage of errors that contain a single error. As the character error rate r increases, the probability of words with multiple errors sharply increases. This leads to a smaller percentage of words that are corrected.

The dictionary algorithm is somewhat more effective than binary positional trigrams in terms of error and reject rates. For the 4 increasing sizes of word sets, the 20PT algorithm corrects fewer errors than the dictionary algorithm: approximately 0.5, 4, 8, and 20 percent fewer, respectively. For the smaller word data sets, the difference in performance is not very significant. For the larger word sets, most of the difference involves a lower correction rate and a correspondingly higher rejection rate on the part of the binary trigrams. However, the remaining error rate is still very small.

Fig. 2(a)–(c) shows the error, reject, and correction rates for all positional digrams with word error rates $w = 0.059$, 0.265, and 0.469 (produced by classifier error rates $r = 0.01$, 0.05, and 0.10, respectively). Fig. 3 shows the same information for the set 20PT with a forced error rate added. The forced error rate is computed assuming a 0 reject rate and an 80-percent correct forced guess rate, as previously discussed.

The use of binary positional trigrams should allow a poor classification system to be transformed into a fairly

TABLE VII
COMPARISON OF THE ALGORITHM USING THE ENTIRE SET OF POSITIONAL TRIGRAMS (20PT) AND THE DICTIONARY ALGORITHM—PERCENTAGE OF ERRORS CORRECTED, REJECTED, AND REMAINING

Character ERROR RATE, r	SIZE OF WORD SET	INITIAL WORD ERROR RATE,w	% WORD ERRORS WITH 1 ERROR	DICTIONARY			SET OF POSITIONAL TRIGRAMS (20PT)		
				% WORD ERRORS CORRECTED	% WORD ERRORS REJECTED	% WORD ERRORS REMAINING	% WORD ERRORS CORRECTED	% WORD ERRORS REJECTED	% WORD ERRORS REMAINING
.01	300	5.85	97.5	95.2	4.6	.17	94.9	4.9	.17
	800			91.4	8.4	.19	87.2	12.5	.36
	1300			88.9	10.9	.22	80.3	19.3	.41
	2755			82.9	16.7	.42	60.7	37.9	1.40
.05	300	26.49	87.6	93.3	6.5	.23	92.9	6.8	.23
	800			88.5	11.1	.36	84.4	15.1	.49
	1300			85.6	13.9	.49	76.9	22.3	.74
	2755			78.6	20.5	.82	57.4	40.7	1.86
.10	300	46.86	75.6	89.8	9.9	.38	89.3	10.3	.38
	800			84.0	15.3	.70	79.9	19.3	.80
	1300			80.2	18.8	.95	72.1	26.7	1.21
	2755			72.9	25.7	1.46	52.9	44.6	2.44
.20	300	73.79	53.3	78.7	20.4	.87	78.2	20.9	.85
	800			71.8	26.4	1.72	68.1	30.1	1.74
	1300			67.5	30.3	2.12	60.4	37.3	2.31
	2755			59.6	37.4	3.03	42.9	53.5	3.60

effective one. The 2755 6-letter word set included many words that are not often used. A particular application might not be restricted inordinately if the input set were limited to 800 or 1300 words of any given length. (Note that an overall dictionary of between 5000 and 10 000 words could easily be used by subdividing the words according to their length.) If one assumes an 800 word input set with either character error rates of 10 or 20 percent, then 6-letter words would have a word error rate of 47 or 74 percent, respectively. These systems would be virtually unusable. The high speed contextual system that has been described will reduce the 47-percent error rate to a 0.38-percent error rate and a 9-percent reject rate; correspondingly, the initial 74-percent error rate results in 1.29-percent errors and 22-percent rejects. For lower initial character error rates the system performs better, since a larger proportion of the errors are one-error words which are easier to correct than multiple-error words.

VII. DISCUSSION

The algorithms that have been presented appear to be extremely effective in error detection and correction. These results compare favorably to the previous work that has been discussed, in terms of detection and correction rates, computational efficiency, and extendability to a larger portion of language. The positional binary trigrams are certainly far more effective than any previous use of the probabilities of diagrams and trigrams. There have been methods in the past that have detected a high percentage of errors but these methods have relied upon slow dictionary lookup algorithms or extremely large amounts of data [13], [26]. Herein lies the power of the trigram correction process. Although it is not quite as effective in correcting errors as the use of a complete dictionary, it is orders of magnitude faster than the dictionary correction process. It is this advantage that would allow high speed correction in an on-line process. By enormously reducing the error rate, ineffective classification systems might be made practical.

It seems that little of the information necessary for error detection resides in the specific value of the nonzero probability of letter pairs and triplets since highly effective error detection took place in the algorithms using quantized binary values. The knowledge of whether the probability of some set of letters in certain positions is nonzero is sufficient for error detection. The power of the system seems to reside in the partitioning of information by position. This allows the n-gram matrices to be sparser than position-independent matrices, and also permits several of them to apply nonredundant information in parallel in the detection and correction process. The positional trigrams are not as effective as the dictionary in the correction process, although a large portion of the uncorrectable words are reduced to a few alternatives. Therefore, many of the uncorrectable words might be amenable to other correction procedures, such as collecting additional measurements or employing statistics on the confusion matrix of the classifier.

The use of confusion matrix information of the classifier results in a loss of standardization since the postprocessor is now dependent on the particular classifier employed. On the other hand, the system might be improved a great deal. If one looks at the substitution matrix for a given classifier one sees that when a mistake is made there are usually only several characters that the correct letter could have been. It has been pointed out in Table VI that when an error is uncorrectable, a very high percentage of these cases involve only two or three choices. If the information concerning the possible substitutions is

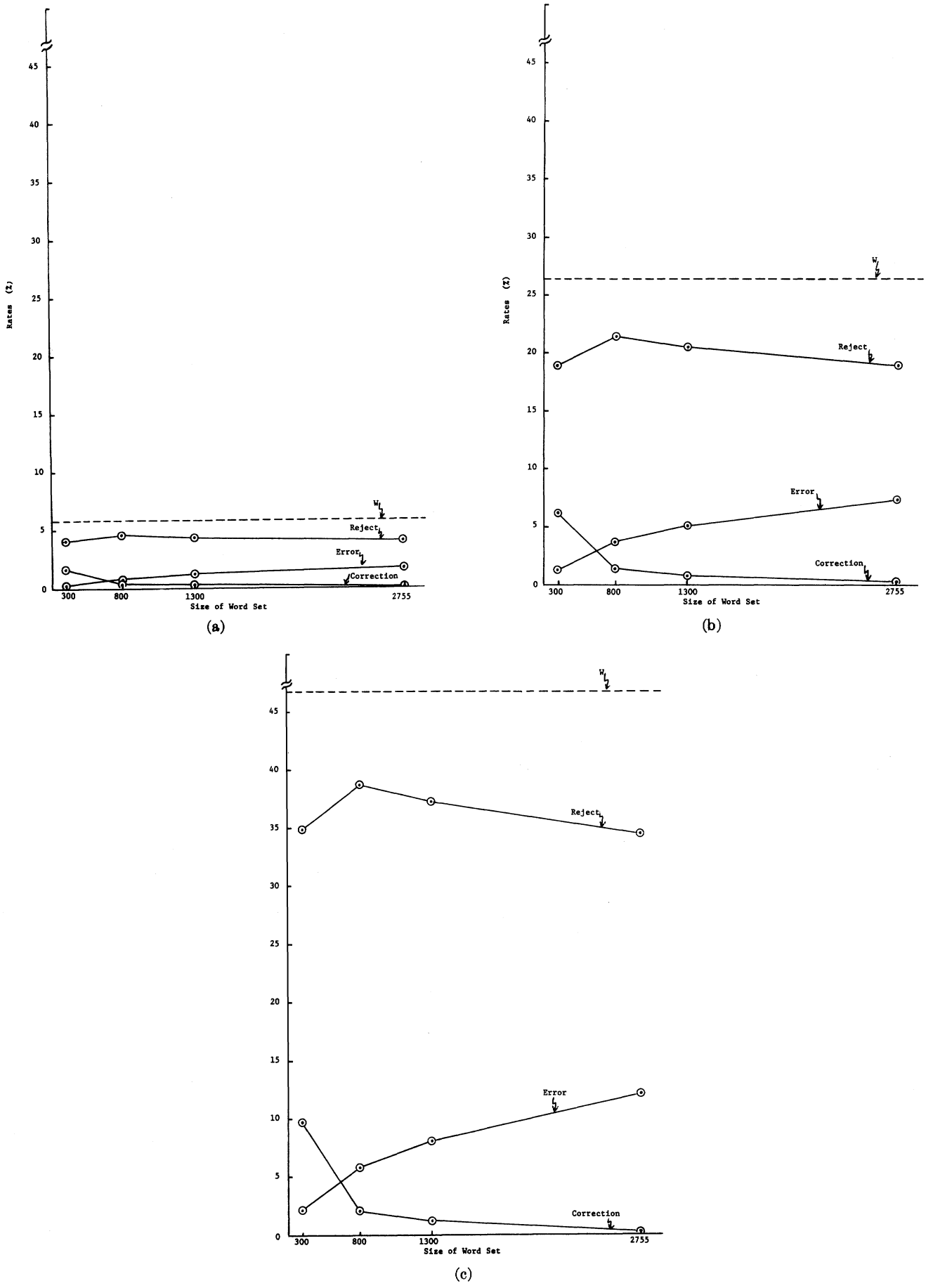


Fig. 2. Error, reject, and correction rates using all positional binary *digrams*. r = classifier character error rate; w = classifier word error rate. (a) $r = 0.01$; $w = 0.059$. (b) $r = 0.05$; $w = 0.265$. (c) $r = 0.10$; $w = 0.469$.

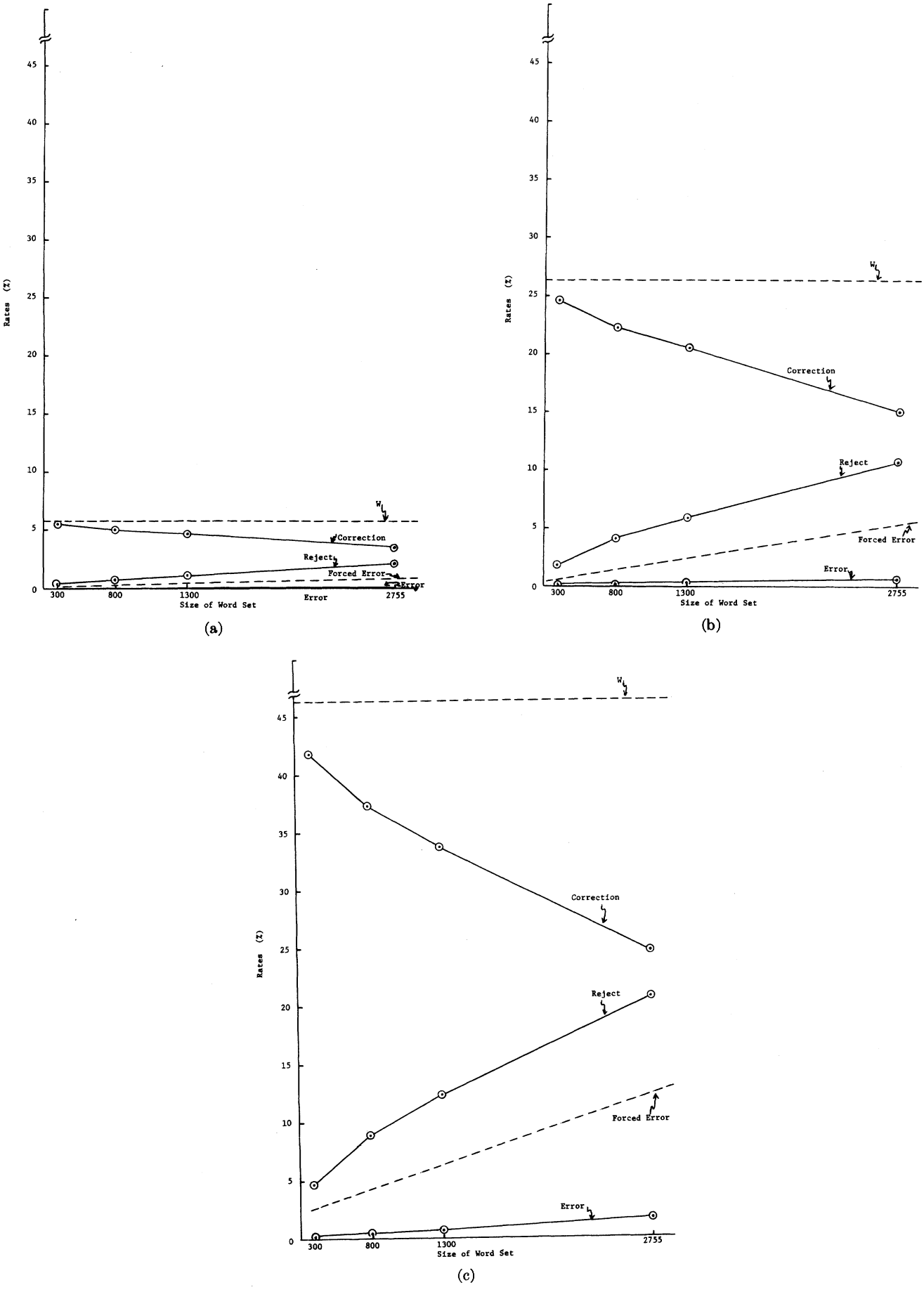


Fig. 3. Error, reject, and correction rates using all positional binary *trigrams*. r = classifier character error rate; w = classifier word error rate. (a) $r = 0.01$; $w = 0.059$. (b) $r = 0.05$; $w = 0.265$. (c) $r = 0.10$; $w = 0.469$.

available (a 26 × 26 binary matrix might be sufficient), many of the currently uncorrectable errors might be resolved.

Only six-letter words were examined in the experiments described. Similar statistics could be employed for sets of words of other lengths. There are fewer positional n-grams for words of shorter length; consequently, the system might detect and correct fewer errors. A possible compensating factor, however, is the existence of fewer short words, which should reduce the density of the binary n-grams and improve the performance of the contextual processor. Since the density is a function of the number of distinct letter pairs that exist across the set of words, we are unsure how much effect this will have. For longer words there are far more n-grams available; for example, 6-letter words involve 15 digrams and 20 trigrams; however, an error in any position only can be in the domain of 5 digrams and 10 trigrams. These last two numbers are indicators of the amount of information being brought to bear in the error correction process. For longer words there are far more n-grams available. In 9-letter words there are 8 digrams and 28 trigrams involving each position. Clearly, one can expect correction rates superior to those presented in this paper if the greater investment in storage and some reduction in computational speed can be tolerated. Otherwise, a judiciously chosen subset of these n-grams, where each position occurs in approximately five digrams and ten trigrams, should produce results comparable to those reported here. The selection of a subset of positional binary digrams for error detection was examined in [29].

These results do have some serious limitations. The detection and correction rates were derived empirically by simulating independent random substitutions. In practice, real data would exhibit somewhat different characteristics. There would be a greater tendency toward sequences of errors due to poor copy. This would produce more difficulty in correction because there would be words with a greater number of errors. This would probably cause only a slight increase in the error rate and a larger increase in the reject rate.

A second difference between these experments and actual data is that the error distribution would be skewed toward common class-pair confusions, say D and O, U and V, etc. Although the letters appear physically the same, the contextual postprocessor might have more or less difficulty with these class pairs, because they may or may not be contextually the same. Thus the error-reject rates might increase or decrease as a function of this distribution. The uniform random distribution only provides "average" results.

The system that has been developed can be extended. A powerful correction system can be achieved by integrating the classifier and the contextual postprocessor. If on-line detection and correction is attempted, one can allow feedback from the postprocessor to the classifier during error correction. Reclassification can be carried out among the few allowable choices, immensely simplifying the original 26-character dichotomization problem. At the loss of standardization of the postprocessor, correctability rates should be greatly improved. An even more integrated recognition system can be utilized by allowing feedback to the measurement system so that further specific measurements can be taken to reduce uncertainty in the correction process when there is more than one alternative. Examination of these modifications has been started [32]–[33].

Another possible extension of the techniques that has been discussed is the correction of errors in discrete feature measurements (i.e., unusual values that lead to erroneous decisions). Each character position of a word can be likened to a feature of the word pattern. From this viewpoint, the contextual system is correcting the value of a feature measurement from one of the possible 26 measurement values to another. In the general pattern recognition problem, "positional" binary n-gram statistics correspond to matrices denoting which of the pairs of values for a feature pair can co-occur. For example, a positional trigram would involve a particular three features and describe the allowable value triplets that these features can simultaneously have in any given pattern. At that point, the detection and correction procedures could be applied without any change. Of course, the success of the system would be highly dependent upon the density of the n-grams; the densities will be a function of the particular problem including the type of patterns and features selected as well as the type of n-grams employed.

It is the authors' contention that contextual recognition techniques in character recognition have not been developed and utilized nearly as much as they should be. The results described in this paper demonstrate that computationally efficient procedures can be used to drastically cut error rates with only modest reject rates. The use of positional binary n-grams might make difficult classification feasible. Conversely, one might be able to greatly reduce the complexity of a pattern classifier if it is followed by a contextual postprocessor with the reliability that has been demonstrated herein.

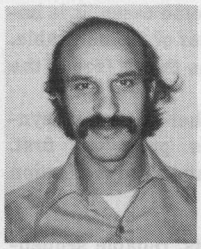
ACKNOWLEDGMENT

The authors wish to thank E. Fisher and S. Fickas for their aid in discussing and programming the algorithms, as well as for gathering the experimental data.

REFERENCES

[1] G. Nagy, "State of the art in pattern recognition," *Proc. IEEE*, vol. 56, pp. 836–862, May 1968.
[2] M. D. Levine, "Feature extraction: A survey," *Proc. IEEE*, vol. 57, pp. 1391–1407, Aug. 1969.
[3] C. J. W. Mason, "Pattern recognition bibliography," IEEE System Science and Cybernetics Group, Oct. 1970.
[4] B. Gold, "Machine recognition of hand-sent Morse code," *IRE Trans. Inform. Theory*, vol. IT-5, pp. 17–24, Mar. 1959.
[5] W. W. Bledsoe and J. Browning, "Pattern recognition and reading by machine," in *1959 Proc. Eastern Joint Computer Conf.* New York: National Joint Computer Committee, Dec. 1959, pp. 225–232.
[6] C. R. Blair, "A program for correcting spelling errors," *Inform. Contr.*, vol. 3, pp. 60–67, 1960.

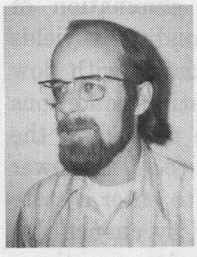
[7] E. J. Sitar, "Machine recognition of cursive script: The use of context for error detection and correction," Bell Lab., Murray Hill, N. J., unpublished memorandum, Sept. 12, 1961.
[8] L. D. Harmon and E. J. Sitar, "Method and apparatus for correcting errors in mutilated text," U. S. Patent 3 188 609, June 8, 1965.
[9] C. K. McElwain and M. B. Evens, "The degarbler—A program for correcting machine read Morse code," *Inform. Contr.*, vol. 5, pp. 368–384, 1962.
[10] R. B. Thomas, M. Kassler, and G. Woolley, "Advanced character recognition study," Tech. Rep. 2 DDC-AD 435852, Dec. 1963.
[11] C. M. Vossler and N. M. Branston, "The use of context for correcting garbled English text," in *Proc. Ass. Comput. Mach. 19th Nat. Conf.*, Aug. 1964, pp. D2.4-1–D2.4-13.
[12] A. W. Edwards and R. L. Chambers, "Can a priori probabilities help in character recognition?" *J. Ass. Comput. Mach.*, pp. 465–470, Oct. 1964.
[13] G. Carlson, "Techniques for replacing characters that are garbled on input," in *1966 Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 28. Washington, D. C.: Spartan, 1966, pp. 189–192.
[14] J. Raviv, "Decision making in Markov chains applied to the problem of pattern recognition," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 536–551, Oct. 1967.
[15] J. T. Chu, "Error bounds for a contextual recognition procedure," *IEEE Trans. Comput.* (Short Notes), vol. C-20, pp. 1203–1207, Oct. 1971
[16] A. J. Szanser, "Error-correcting methods in natural language processing," in *Information Processing 68*, vol. II. Amsterdam, the Netherlands: North Holland, 1968, pp. 1412–1416.
[17] J. R. Welch and K. G. Salter, "A context algorithm for pattern recognition and image interpretation," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-1, pp. 24–30, Jan. 1971.
[18] E. M. Riseman and R. W. Ehrich, "Contextual word recognition using binary diagrams," *IEEE Trans. Comput.*, vol. C-20, Apr. 1971, pp. 397–403.
[19] R. O. Duda and P. E. Hart, "Experiments in the recognition of hand-printed text: Part II context analysis," in *1968 Fall Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 33. Washington, D. C.: Thompson, 1968, pp. 1139–1149.
[20] L. H. Morgan, "Spelling correction in system programming," *Commun. Ass. Comput. Mach.*, pp. 90–94, Feb. 1970
[21] R. C. Heinselman, "Computerized detection and correction of spelling error in Fortran programs," M.S. thesis, Dep. Comput. Inform. Sci., Univ. Minnesota, Minneapolis, 1972.
[22] H. T. Glantz, "On the recognition of information with a digital computer," *J. Ass. Comput. Mach.*, pp. 178–188, Apr. 1957.
[23] L. Davidson, "Retrieval of misspelled names in an airlines passenger record system," *Commun. Ass. Comput. Mach.*, pp. 169–171, Mar. 1962.
[24] J. J. Giangardella, J. F. Hudson, and R. S. Roper, "Spelling correction by vector representation using a digital computer," *IEEE Trans. Eng. Writing Speech*, vol. EWS-10, pp. 57–62, Dec. 1967.
[25] C. N. Alberga, "String similarity and misspellings," *Commun. Ass. Comput. Mach.*, vol. 10, pp. 302–313, May 1967.
[26] F. Damerau, "A technique for computer detection and correction of spelling errors," *Commun. Ass. Comput. Mach.*, vol 7., pp. 171–176, Mar. 1964.
[27] U. Neisser and P. Weene, "A note on human recognition of hand-printed characters," *Inform. Contr.*, vol. 2, pp. 191–196, 1960.
[28] C. Shannon, "Prediction and entropy of printed English," *Bell Syst. Tech. J.*, pp. 51–54, Jan. 1951.
[29] R. W. Ehrich and E. M. Riseman, "Contextual error detection," Univ. Massachusetts, Amherst, COINS Tech. Rep. 70C-4, May 1971.
[30] F. W. Stodola and A. R. Hanson, "Pattern recognition using contour analysis," Dep. Comput. Inform. Sci., Univ. Minnesota, Minneapolis, Tech. Rep. 72-5, May 1972.
[31] E. L. Thorndike and I. Lorge, *The Teachers Word Book of 30,000 Words*, 4th ed. Teachers College, Columbia Univ., New York: Bureau of Publications, 1963.
[32] A. R. Hanson and E. M. Riseman, "System design of an integrated pattern recognition system," in *Proc. 1st Int. Joint Conf. on Pattern Recognition*, Nov. 1973, p. 342.
[33] E. M. Riseman and A. R. Hanson, "A contextual postprocessing system," Univ. Massachusetts, Amherst, COINS Tech. Rep. 72C-1, Oct. 1972.



Edward M. Riseman (S'65–M'68) was born in Washington, D. C., on August 15, 1942. He received the B.S. degree from Clarkson College of Technology, Potsdam, N. Y., in 1964, and the M.S. and Ph.D. degrees from Cornell University, Ithaca, N. Y., in 1966 and 1969, respectively, all in electrical engineering.

In 1969 he joined the Computer and Information Science Department, University of Massachusetts, Amherst, where he is now an Associate Professor. He has taught courses in pattern recognition, artificial intelligence, and the application of computers to education and health care systems. He has conducted research in switching theory, feature selection, and the utilization of context in character recognition. Currently, his principal interests involve the application of contextual information to both word recognition and analysis of outdoor scenes.

Dr. Riseman is a member of Tau Beta Pi, Eta Kappa Nu, and Sigma Xi, and the Pattern Recognition Society.



Allen R. Hanson (S'65–M'68) was born in Jamaica, N. Y. on August 4, 1942. He received the B.S. degree from Clarkson College of Technology, Potsdam, N. Y., in 1964, the M.S. and Ph.D. degrees from Cornell University, Ithaca, N. Y., in 1966 and 1969, respectively, all in electrical engineering.

In 1969, he joined the Department of Computer, Information, and Control Sciences at the University of Minnesota, Minneapolis, as an Assistant Professor where he taught courses in pattern recognition, artificial intelligence, programming languages, and computer aesthetics. In 1973, he joined the faculty of Hampshire College, Amherst, Mass. where he is responsible for computer science curriculum development. He has conducted research in the areas of pattern recognition, artificial intelligence, adaptive systems, and medical data analysis. His current research interests include application of contextual information to pattern recognition problems and the use of semantic structures in vision processing.

Dr. Hanson is a member of Eta Kappa Nu, Tau Beta Pi, the Association for Computing Machinery, and the Pattern Recognition Society. During the summer of 1973, he was cochairman of the First International Conference on Computers in the Humanities.