Pattern Recognition Pergamon Press 1976. Vol. 8, pp. 35–45. Printed in Great Britain

CONTEXT IN WORD RECOGNITION

A. R. HANSON*, E. M. RISEMAN† and E. FISHER†

* Division of Language and Communication, Hampshire College, Amherst, Massachusetts 01002, U.S.A.
† Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01002, U.S.A.

(*Received* 25 *October* 1974 *and in revised form* 19 *May* 1975)

Abstract—Relatively low character error rates can often lead to prohibitive levels of word error rates. This paper examines several techniques for integrating an independent contextual postprocessor (CPP) into a full classification system. Using positional binary *n*-grams the CPP can correct many errors directly. In those cases where the correction process leads to ambiguity, the CPP can direct additional processing. Experimental results demonstrate that almost all of the derived improvement results from CPP-directed reclassification. This only requires that the CPP have the classifier likelihood fed forward to it. Therefore, a standardized CPP can be built independently of the rest of the classification system. An initial 45% word error rate is reduced to about a 2% word error rate and a 1% reject rate. Presence of a dictionary allows these figures to be reduced even further.

Context Contextual postprocessor Character recognition Positional binary *n*-grams Error detection Error correction.

1. INTRODUCTION

If pattern recognition machines are to process text in any reliable fashion, the error rates that are often reported in the literature must be reduced by at least an order of magnitude. Typical character error rates of 4–25% result in word error rates (in six-letter words for example) ranging from 21 to 86%, assuming that errors are independently distributed.

Thus, even a 4% character error rate leads to an unacceptably large proportion of words in error. However, 4% is approximately the error rate of humans when they view isolated hand-printed characters.[1] It is obvious that our use of context allows us to make at least an order of magnitude improvement in the character error rate so that very few words are actually mistaken. It is contextual processes such as these that are often lacking in the design of character recognition systems.

There have been a number of contextual processing schemes investigated.[2–12] Many of the effective systems use a dictionary or else estimate the likelihood of letter sequences with letter pair (digram) or letter triplet (trigram) probabilities. Although the use of a dictionary is more effective than *n*-gram probabilities,[5–7] these systems suffer from slow correction speeds in language domains of any practical size. On the other hand, systems which use *n*-gram probabilities usually suffer from one or more of several common difficulties: introduction of undesired complexity in the classification process, exorbitant amounts of storage, or only modest reduction in error-reject rates.[8–12]

This research was supported by the Office of Naval Research under Grant ONR 049-332.

This paper extends the independent contextual postprocessor described in ref. 5 by examining the alternatives available for integrating it into a full classification system. Our overall classification system as shown in Fig. 1 is viewed as a network of 3 subsystems: a feature selection/measurement subsystem, a classifier and a contextual postprocessor. The patterns upon which the system operates consist of individual words input as a sequence of characters. The first two subsystems operate on individual characters while the postprocessor operates on sequences of characters forming words. There have been attempts at integrating the designs of the last two of these subsystems. However, this often leads to an increase in complexity, as in Raviv's[8] integration of context (as trigram probabilities) within a decision-theoretic classifier. Many systems that are interesting from a theoretical perspective turn out to be impractical in application. It is far simpler to build each of these subsystems as an independent module. In this paper, we will examine a character recognition system that is very simple but whose power resides in the key role played by an independently operating contextual postprocessor.

Let us briefly discuss the system which is depicted in Fig. 1 and the environment in which it operates. Conceptually, we think of the feature subsystem as a composite of the following: (a) a large pool of features, (b) a process by which a subset of the pool is selected, usually based upon information from a training set, and (c) a process which measures the value of each feature when a pattern is presented. The features contained in our pool are random n-tuples, line intersections, amount of mass within various windows, and only a very few topological features. Very

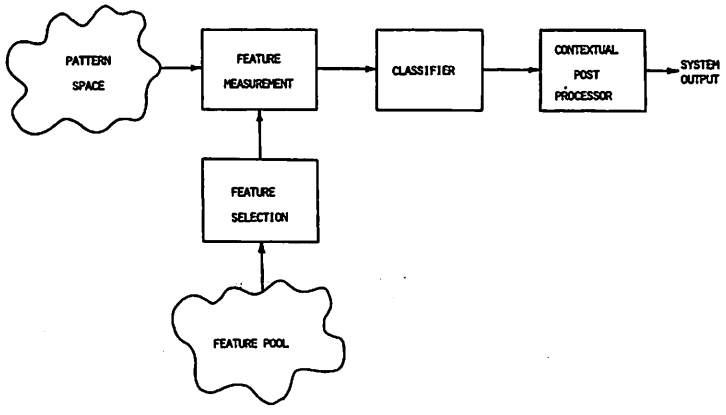


Fig. 1. The three subsystems of a character recognition system: feature subsystem, classifier, contextual postprocessor.

little effort was made to obtain a good pool of features because very good features (such as topologically invariant features) often require large amounts of computation.

The feature selection algorithm is the study of a separate paper;[13] the algorithm employs thresholded measures of non-redundant class pair separation in order to select features. This algorithm was used to select 50 of the 200 features in the pool to act as our primary feature set for classifying characters.

The classifier is quite straightforward. There was virtually no attempt to design a sophisticated decision mechanism. The classifier employed here is a simple Bayesian classifier: select class $j(C = j)$ such that j maximizes the product $P(\overline{X}|C = j)P(C = j)$ where $\overline{X} = (x_1, \ldots, x_n)$ is the vector of measurements on the unknown sample. We made a typical simplifying assumption that the individual measurements $x_i \in \overline{X}$ are independent.* Most assumptions of this type are not valid, but are made to avoid either prohibitively large training sets for estimating the class conditional densities or prohibitive amounts of computation. In any case, we expect errors to occur due to both the violation of such assumptions as well as variations of the character set not accounted for in the training set.

The third subsystem is the contextual postprocessor. It is this subsystem which we expect to be responsible for the overall effectiveness of the character recognizer. This system does not have to be complex in order to compensate for the weaknesses in the first two subsystems. An effective and independent contextual postprocessor was presented in ref. 5. The error rate was significantly reduced by detecting and correcting a large portion of the word errors input. The further improvements to this system are the topic of this paper. In the next two sections, we will review the design, operation, and effectiveness of this CPP and compare it to one of the traditional approaches for employing context in the classification of samples of a language. Then we will examine alternatives for further integrating the CPP into the overall system. Experimental results will be used to assess the value of each of these design variations.

* Results similar to those presented in this paper were obtained under an assumption that the features were multivariate normal with equal covariance matrices.

2. PREVIOUS APPROACHES

2.1 *Context as Markov dependence*

A simple Bayesian classifier was constructed to operate on characters from Munson's handprinted IEEE Data set.[14] This data consisted of three handprinted alphabets from each of 49 authors; two alphabets from each were used for training and one of each was used for testing. The classifier produced a 9.7% character error rate on the 1274 test characters, a value in the range of previous experimenters.[15] Since the whole system is to operate on strings of characters forming words, the test set of characters were used to produce a set of 2755 6-letter words[5] (by random selection among the 49 samples for each letter). The 9.7% character error rate produced a word error rate of 45.8% on six-letter words when characters are classified independently.

As a base against which to measure the contextual processing systems that follow, context was introduced under the assumption that the structure of the input words obeyed a 2nd order Markov dependence; i.e., each character is a probabilistic function of the previous 2 characters (less at the beginning of words). This involves using the set of 17,576 trigram probabilities as a number of previous researchers[8–11] have done. In our case, we use the trigram probabilities to directly weight the likelihoods of the individual characters obtained from the set of features. Thus, for sample character i, we choose j to maximize:

$$P(\overline{X}_i|C_i = j)P(C_i = j|C_{i-1} = k, C_{i-2} = l)$$

where $\overline{X}_m$ and C_m are the vectors of feature measurements and the hypothesized identity of the mth character in the sequence, respectively.

The results are presented in Table 1. The word error rate using the trigram probabilities is reduced,

Table 1. Utility of context as 2nd Order Markov dependence.
(a) Character and word error rates

	CHARACTER ERROR RATE	WORD ERROR RATE
CLASSIFIER WITHOUT CONTEXT	9.7%	45.8%
CLASSIFIER USING TRIGRAM PROBABILITIES	10.2%	29.2%

(b) Distribution of word errors

	NONE	1	2	3	4	5	6	TOTAL
CLASSIFIER WITHOUT CONTEXT	11841	7662	2041	273	23	1	0	10,000
CLASSIFIER USING TRIGRAM PROBABILITIES	15474	2621	1750	1134	531	269	62	6,367

but only to 29.2%. Even more surprising is the result that the character error rate is actually *increased*. This is due to the fact that once an error is made, additional errors are more likely. Table 1 compares the distribution of errors in the 21,841 input words. It is clear that this use of context results in far more words with long sequences of errors. Our conclusion is that this is an ineffective use of contextual information. Although there are more sophisticated ways to integrate the n-gram probabilities into the decision process, these results give a sense of that approach. We do not believe that simple and efficient techniques under assumptions of Markov dependence will result in highly significant reductions in the error rate.

2.2 *Rejection*

One processing option that others have utilized is the rejection of characters whose confidence is below a given threshold, say θ. Our analysis[4] shows that for values of θ up to 0.99, the number of incorrect characters is about the same as the number of correct characters rejected. Thus, the reject rate will increase approximately twice as fast as the error rate will decrease. While this might be quite acceptable, the point is usually reached quickly where a modest decrease in error rate is obtained only at the expense of a very large increase in the reject rate. Thus, this does not allow the order of magnitude improvement we have set as our goal. Reduction of the character error rate to 2–3% leaves a 20–30% character reject rate, causing a very large word reject rate. Although we have not investigated this technique further, it is possible to integrate it with some of the more powerful systems described in later sections.

2.3 *Feedback directed by the classifier*

An additional extension to the classifier involves feedback of low confidence decisions to the feature selection/measurement subsystem. Rather than just choose the class with the maximum likelihood, we can require a minimum likelihood or "confidence" θ before permitting the classifier to output a decision. It is possible to use the confidence threshold not to reject the character, but rather to determine whether to request further measurements selected from the secondary set of specialist features. A subset of the several most likely classes will be formed so that specialist features can be used to reduce the uncertainty between particular class pairs. Reclassification may then be done on the basis of only the new information or on the combined information. The utility of this approach clearly depends upon the quality of the "specialist" features. While results have not been encouraging,[4] the features employed were drawn from the same set as the primary features; results should improve if the specialists are carefully selected from a good set.

3. REVIEW OF THE CONTEXTUAL POSTPROCESSOR (CPP)

The dictionary of words acceptable as input is partitioned by word length into subdictionaries. Although the processor we are describing can operate in general using binary n-grams, for simplicity we will restrict the explanation to binary trigrams and allow the reader to carry out the obvious generalizations. For each subdictionary a triple of letter positions (i,j,k) specify a $26 \times 26 \times 26$ array D_{ijk} called a positional binary trigram. The (l,m,n)th entry of D_{ijk} is defined to be 1 if and only if there exists some word in the subdictionary which contains letters l, m and n in the ith, jth and kth positions, respectively; otherwise, it has value 0. This is equivalent to quantizing trigram probabilities to values of 1 or 0 depending upon whether the probability is non-zero or zero, respectively.

The set of all $\binom{m}{3}$ positional binary trigrams can be used to *detect* errors in m-letter words. For any word output from the classifier (a character at a time), the proper entry in each binary trigram is checked to make sure it is a 1. Any entry of 0 implies that the triple is illegal and there exists one or more errors in the triple of positions.

The set of binary trigrams can also be used to *correct* the errors detected. The position of the error is fixed by analyzing the positions of the subset of trigrams which detect the error under the hypothesis that only one error occurred. If more than one position could account for the trigrams detecting an er-

Table 2. Initial CPP Effectiveness. (a) Analysis of error detection and correction as a function of the number of errors present

	NUMBER OF WORDS					
NUMBER OF ERRORS	INPUT TO CPP	DETECTED	CORRECTED	REJECTED	ERRORS UNDETECTED	ERRORS IMPROPERLY CORRECTED
1	7662	7561	4656	2905	101	---
2	2041	2038	611	1357	3	70
3	273	273	9	248	0	16
4	23	23	----	19	0	4
5	1	1	----	1	0	0
TOTALS	10000	9896	5276	4530	194	

(b) Distribution of words with N errors

NUMBER OF ERRORS N N=	0	1	2	3	4	5	6	TOTAL ERRORS	REJECTIONS
BEFORE CPP	11841	7662	2041	273	23	1	0	10000	0
AFTER CPP	17117	122	42	24	3	3	0	194	4530

ror, then the correction process is attempted for each in the hope that the ambiguity will be resolved by determining that there is no possible correction for all but one of the positions. If there is no single position which could account for the trigrams which detect the error, then there must be multiple errors in the test word. An analogous procedure fixes all possible pairs of positions which could be in error, etc.

For each triple which involves only one position in error, the characters in the two correct positions determine which characters are alternatives for substitution in the error position. This information is a projection of the data in the binary trigram associated with the triple: it resides in the 1's and 0's of the 26-bit vector specified by the characters not in error. We intersect these vectors from all binary trigrams involving the position in error. If there remains only one choice for the character in error, we carry out the substitution. In all other cases, the word is rejected. The case of multiple errors in which the position is not fixed involves still more cases and will not be analyzed here. It should be noted, however, that *if* the number of errors is correctly determined, then the proper correction(s) *must* be in the set of alternatives.

Although there may be several cases to be examined by the CPP, the detection and correction process can be made very fast by accessing information from the n-grams in parallel. Storage requirements are also quite modest since probabilities are quantized.

The CPP is quite effective in the detection and correction of errors as shown in Table 2. About 99% of the word errors are detected and over 50% corrected. It should be noted that about 1/3 of the 104 undetectable errors were undetectable because the resulting incorrect word also appeared in the dictionary and therefore would be undetectable by any process operating only upon independent words. The correction process does introduce new errors since some n-error words are mistaken as m-error words, $m < n$, and are mistakenly corrected. A more careful analysis shows that the ratio of increased corrections to increased errors is 64:1 and 44:1 for 1- and 2-error words respectively. The overall resultant word error rate is less than 1% and the character error rate is 0.23%. However, these impressive improvements have come at the expense of a 21% word reject rate. This rate is in part due to the rejection of words in which 3 or more errors occurred, but is mainly due to ambiguity in the correction process.

The performance of the CPP can be fully appreciated only if one examines some of the word errors

Table 3. Errors corrected directly by CPP. (a) Words properly corrected.

ONE ERROR WORDS			TWO ERROR WORDS		
INPUT	CLASSIFIER OUTPUT	CPP OUTPUT	INPUT	CLASSIFIER OUTPUT	CPP OUTPUT
DISMAL	DYSMAL	DISMAL	LAGOON	IAGOOH	LAGOON
GOSSIP	AOSSIP	GOSSIP	IMPEDE	YHPEDE	IMPEDE
HOODED	HQODED	HOODED	LAVISH	LAVJHH	LAVISH
HUMBLE	HUMBCE	HUMBLE	KNIGHT	XNIGNT	KNIGHT
LEAGUE	LEAGVE	LEAGUE	ENGINE	EHGIXE	ENGINE
LAYOUT	LAYOPT	LAYOUT	DRAGON	DRABOH	DRAGON
TROUPE	TAOUPE	TROUPE	ARMORY	BRMOEY	ARMORY

(b) Words improperly modified

TWO ERROR WORDS			THREE ERROR WORDS		
INPUT	CLASSIFIER OUTPUT	CPP OUTPUT	INPUT	CLASSIFIER OUTPUT	CPP OUTPUT
ASSENT	ABSEMT	ABSENT	BISECT	QSGECT	ASPECT
ODDITY	BODITY	BODILY	HITHER	NYTHEA	NETHER
MORASS	HBRASS	HORASS	SALOON	SAIQOV	SAILOR
SINGLE	JINGIE	JINGLE	SUMMIT	SNMHIT	SUNLIT
FLANGE	FLAACE	FLANCE	ALLIES	ALISZS	ARISES
USEFUL	WBEFUL	WOEFUL	FERVID	FERUSO	PERUSE
COLONY	BOLQNY	BOLINY	RESENT	BEBFNT	BENANT

that are properly corrected as well as mistakenly corrected. Table 3 is a presentation of some of these cases. Most people have some difficulty in correcting two errors in 6-letter words and find 3 errors almost impossible to correct. Although the CPP by itself only corrects 30% of these, some of the more powerful systems presented later handle a much larger percentage. It seems that a large portion of the improper corrections are exactly conversions of a two-error word into a word that differed by only one character from some other in the dictionary. These are precisely the corrections that most people make, but one should note that the CPP is carrying out these corrections without the use of a dictionary of possible words. Hence, it sometimes improperly corrects to words that are not in the language, although, as one might expect, they sound like English words. Some examples are shown in Table 3 (b).

Although the CPP is quite effective in detecting errors, it only carries out correction when there is no ambiguity. It always attempts correction under the assumption that the fewest errors possible have occurred; *i.e.*, it will attempt to correct as a 1 error word if possible, otherwise as a 2-error word, etc. When there is more than one alternative, the CPP we have described will reject the word. This happens in over 45% of the errors detected (21% of all words). In a large majority of these cases, there are only a few choices for letter substitutions, and the remainder of this paper examines ways of correcting these errors. Most of the alternatives for additional processing will be under the direction of the CPP.

4. FURTHER PROCESSING BY THE CPP

The flow of control in the system that has been discussed is quite straightforward. Each subsystem feeds information sequentially to the next subsystem until the CPP outputs a decision or rejects the word. In this section we will introduce additional paths of data flow so that alternatives for processing a character or word at various stages in the recognition process will be available. These paths are shown in Fig. 2 and involve feedback from the CPP to the feature or classifier subsystems. In addition, we will also examine the utility of employing statistics on the types of errors made by the classifier; *i.e.*, the substitution matrix (this dependency is shown as a dotted line).

The CPP has already demonstrated its power by its ability to detect and directly correct word errors

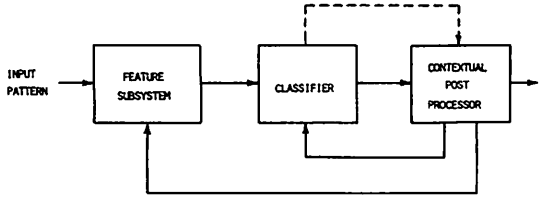


Fig. 2. Additional paths of data flow (solid lines) and design dependency (dotted line).

that are input to it. The results presented in the previous paper[4] were for a CPP that would reject all words where there was any ambiguity in the correction process. In almost all cases where a word is rejected, however, the CPP does supply useful information and can be used to direct a range of actions. Let us outline the options we can make available to the CPP during processing of a character. When an error is detected, the CPP may:

(1) correct the word directly, and output it;
(2) determine the position(s) of the letter(s) in error and then
  (a) refer back to the classifier for reclassification among the remaining choices; or
  (b) feed back to the feature extractor for additional measurements before reclassification is attempted;
(3) be unable to fix the position(s) of the error(s) and either
  (a) reject the word as uncorrectable (possibly with a recommendation list); or
  (b) choose the subset of alternative positions of the error(s) and request additional measurements for the letter(s) in these positions.

In this section, we will discuss each path of information flow in more detail. Although they will be discussed independently, any subset of them can be used concurrently. Unfortunately, there does not seem to be any vehicle by which the different systems can in general be theoretically compared. They are, of course, dependent upon the problem domain and the specific data. The empirical results that follow compare each of the alternative designs and pinpoint which are the most powerful in this problem space and hopefully give insight into other domains.

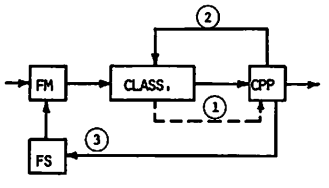
4.1 *Classifier error matrix statistics*

In the case of alternative choices of correction in the CPP (i.e., the 21% of the words rejected), any information which might potentially reduce such uncertainty should be examined. Vossler and Bronston[6] utilized the classifier statistics to determine which type of errors were more likely. In this case, the CPP will use 26^2 probabilities derived from the training set error matrix to determine which of the candidates for the input character was more likely to have produced the error.

In generalizing from a training set to a test data set, one must be aware of the dangers of generalizing from a small sample. Because of the small ratio between the number of characters per class and the number of classes (98:26), we must admit that it is very likely that some mistakes will be made on the test data that were not made on the training set even though the test data reasonably resembles the training set. To impose our subjective evaluation of the problem upon the substitution matrix probabilities, we compute $P(c_i|c_j)$, the probability that a character is

Table 4. Additional Processing by CPP

INFORMATION UTILIZED	ADDITIONAL WORD CORRECTIONS/ERRORS VS. CONFIDENCE LEVEL						
		θ = 0	.5	.9	.990	.999	.9999
①	CORRECTIONS	3476	3349	1918	213	38	1
	ERRORS	1052	752	124	4	0	0
②	CORRECTIONS	4120	4118	3989	3643	3464	3464
	ERRORS	408	406	287	225	170	146
① + ②	CORRECTIONS	4161	4157	4063	3937	3842	3600
	ERRORS	367	365	294	219	174	135
③	CORRECTIONS	3955	3955	3828	3604	3311	3080
	ERRORS	573	571	430	287	193	145
① + ③	CORRECTIONS	4034	4033	3911	3769	3566	3319
	ERRORS	494	491	346	246	178	142
② + ③	CORRECTIONS	4131	4131	4084	3996	3905	3848
	ERRORS	397	393	325	283	237	207
①+②+③	CORRECTIONS	4158	4158	4116	4069	3981	3882
	ERRORS	370	365	317	256	233	211



① To reduce ambiguity in correction, CPP uses probabilities of error matrix estimated from the training set

② When CPP cannot correct, reclassification of the ambiguous choices (on the basis of the original feature measurements) takes place.

③ CPP directs feedback for additional specialist features to reduce ambiguity in correction.

a member of class c_i given that it was classified as c_j:

$$P(c_i|c_j) = \frac{n_{ij}^*}{\sum_{j=1}^{26} n_{ij}^*} \quad \text{where } n_{ij}^* = n_{ij} + h.$$

n_{ij} is the number of training set characters in class i which were classified as class j, and h is the flattening constant discussed by Good.[16]* The net effect of this is that the substitution matrix will not absolutely rule out any of the alternatives presented by the CPP, though it will still provide relative likelihoods of the alternatives.

The results of employing this information is displayed in Table 4. It is labelled as path ① and as a *dotted* line because it is not a path of information flow. Rather, it is design information which can be collected once and permanently incorporated in the CPP. It is interesting that just this simple table of 676 probabilities can be used by the CPP to correct 3476 of the 4530 errors previously uncorrectable (when no confidence level is used). This is due to the small number of alternatives which often result. With a confidence level of 0.5, the increased correction/error ratio is about 4.5:1. Although this is interesting, one would probably not want to re-introduce 1000 of the almost 10,000 errors which were so effectively detected. Here high correction rates are obtained only at the expense of a large number of additional errors. However, in those cases where this final error/reject rate is tolerable, the additional cost of computation and storage is almost nil.

*$h = \frac{1}{100}$ produced the results presented in this paper.

4.2 *CPP-directed reclassification*

Let us consider the case where the position of an error is determined but correction does not take place because there is more than one substitution that is admissible. Almost 80% of all 1-error words and 42% of 2-error words had no more than 3 alternatives for substitution for the error(s). The tail of this distribution, though, is long, and there were as many as 48 alternatives for some 2-error words. Suppose we return the subset(s) of characters to the classifier for reclassification. Even though an error was made initially, it is not unreasonable to expect that the correct character ranked high. Thus, a simple reclassification decision among the few characters remaining might be all that is necessary. Although the process of reclassification can be viewed as feedback, it should be noted that the set of likelihoods of the 26 letters could be *fed forward* for the CPP to use directly for reclassification if necessary.

The implementation of this procedure is not difficult. Because each of the alternatives for substitution in multiple-error words might involve different positions, we can only determine the likelihood of a set of substitutions in context. If a particular character α_i is not in error, then that might imply that some

character α_j must be in error. Thus, the CPP can supply the set of substitutions which thereby form each of the alternative words for correction. The product of the likelihoods of each character in the word is used to compute the likelihood of the whole word and the most likely alternative is selected. This process is computationally feasible only because it is directed by the CPP.

One should fully appreciate the power of this process. Initial erroneous classification has a far greater chance of being corrected by carrying out the identical classification procedure in the restricted pattern subspace. The pattern classes that are confused on the basis of the measurements employed usually are not the same pattern classes that the CPP confuses. For example, although "O's" and "C's" appear visually to be similar and might be mistaken for each other by the classifier, in many cases the contextual statistics would rule out one or the other as impossible. Thus, the expectation is that this procedure might boost the correction rate significantly.

Information path ② in Table 4 presents the results of the CPP-directed reclassification. It is extremely powerful: the increased correction/error rate varies from 10:1 to 24:1 as a function of the confidence level. This process can correct about 4000 of the 4500 words initially rejected. Even in three-error words, the system is relatively effective. Of the 273 3-error words, 177 are properly corrected while 7 are improperly reclassified. Of course, words with more errors can be mistaken as 3-error words and also improperly corrected; this introduces 14 errors. Thus, even attempted 3-error correction has a correction/error ratio of over 8:1.

CPP-directed reclassification seems to be extremely effective even on these 4500 more difficult cases. To get some sense of the problem, Table 5 shows three different words which are originally classified with 3 errors. The CPP using the set of binary trigrams quickly determines substitutions which result in over 40 alternatives for each incorrect word.* Many of these alternatives are in fact actual words. By using the original likelihoods with the reduced set of alternatives, each of the 3-error words is properly corrected. Of course, information from the measurements on the particular character samples (which are not shown here) are being used to carry out the correction. Thus, in the case of the word 'SHOULD', the errors consisted of substituting *B* for *S*, *A* for *H*, and *O* for *D*, errors which were likely to be made and are hard to correct due to the topological similarity of the characters. However, the CPP can rule out many of the difficult substitutions.

The use of *both* the substitution matrix and the reclassification process is also shown in Table 4. Since the relative importance of each type of information was unclear, the overall likelihood was computed as

Table 5. *THREE-ERROR CORRECTIONS*—Three error words which are *properly corrected* during reclassification

ACTUAL WORD	ORIGINAL CLASSIF.	CPP ALTERNATIVES
SHOULD	BAOULD	BABBLE
		BADDLE
		BADILE
		BAFFLE
		BAIDLY
		BANDLE
		BANDLY
		BANGLE
		BANILE
		BANILY
		BANKLE
		BANTLE
		BARBLE
		BARDLE
		BARDLY
		BARELY
		BARILE
		BARILY
		BAROLD
		BAROLE
		BAROLY
		BARTLE
		BASILE
		BASILY
		BASTLE
		BATILE
		BATILY
		BATTLE
		BAUBLE
		BAMBOO
		BLOUSE
		SHOULD
		APOLLO

ACTUAL WORD	ORIGINAL CLASSIF.	CPP ALTERNATIVES
BUSHEL	OUSVEC	OUTLET
		OUTLEY
		OUTREN
		OUTREY
		OUTSET
		OYSVRC
		OLIVER
		BUSHEL
		BUSHER
		BUSKET
		BUSTEN
		BUSTER
		CUSSED
		CUSSEL
		CUSSER
		CUSTER
		DUSTEN
		DUSTER
		FUSTEN
		GUSHER
		HUSSER
		HUSTER
		JUSTER
		LUSTER
		MUSKET
		MUSLEN
		MUSSEL
		MUSTEN
		MUSTER
		PUSSET
		TUSSEL
		TUSSER
		TUSTER
		PUSTIC
		PURVER
		PURVEY
		QUAVER
		SURVER

ACTUAL WORD	ORIGINAL CLASSIF.	CPP ALTERNATIVES
DURING	BVRIHG	BARIAD
		BARIAL
		BARIAN
		BARIED
		BARIEL
		BARILE
		BARILY
		BARINE
		BARINY
		BERIAD
		BERIAL
		BERIAN
		BERIED
		BERIEL
		BERILE
		BERILL
		BERILY
		BERIND
		BERINE
		BERINY
		BIRINY
		BURIAL
		BURIAN
		BURIEL
		BURILE
		BURILY
		BERIAG
		BAKING
		BALING
		BATING
		BELING
		BILING
		BITING
		DARING
		DURING
		PARING
		PIRING
		SIRING

the product of the probabilities obtained from each. This is equivalent to assuming independence of the processes. Although both techniques are better than either alone, it is obvious that the addition of the substitution matrix is only of a little help. One must weigh the slight improvement due to the use of the classifier statistics since it causes the design of the CPP to be dependent on the classifier. This will be discussed further in the conclusion.

4.3 *Feedback to feature system*

Although the process just described seems to be quite effective, it is *possible* that a significant number of errors would be unnecessarily produced. Consider the following line of reasoning. Since an error did occur, the character must in some way be distorted, noisy, or otherwise non-standard. Therefore, the error is caused because the correct character had a low likelihood and might still rank low in any reduced subset. Because of the CPP design, we can be certain that if the position of the error(s) is correctly fixed, then the correct character must be in the reduced set(s). Using this rationale, rather than re-introduce an error, it would be preferable to seek additional measurements since the information in the original measurements is not sufficient. The experiments about to be reported will determine the validity of this argument.

Again we will employ a set of secondary features for the additional measurements. The n best secondary features will be chosen for every class pair involved. If the number of alternative classes is reduced

* The average is 22.6 alternatives for all three-error words which the CPP detects but is unable to correct.

Table 6. CPP System with substitution matrix and reclassification (①+②). (a) Final error-reject rates as a function of confidence level

	CONFIDENCE LEVEL θ = 0	.9	.999
WORD ERROR RATE	2.7	2.2	1.7
WORD REJECTION RATE	0	1.1	2.7
CHARACTER ERROR RATE	.73%	.63%	.51%

(b) Distribution of errors in the 21,841 words

	NUMBER OF ERRORS 0	1	2	3	4	5	6	TOTAL ERRORS	TOTAL REJECTS
NO CONTEXT	11841	7662	2041	273	23	1	0	10000	0
θ = 0	21254	345	152	58	20	9	1	585	2
θ = .999	20885	192	107	50	16	6	1	372	584

to 4, which it usually is, then there would be only 6 class pairs to dichotomize. Taking additional measurements to separate 4 classes should result in the complexity of the problem being exponentially reduced from the 26 category (325 class pair) space.

The results shown in Table 4 suggest that this process, shown as ③, is indeed effective in dealing with the rejected words. However, it is quite surprising that it is not as effective as direct reclassification. This means that our original 50 primary features (which initially produced an error!) were still more effective for reclassification than using a subset of specialist features for each class pair in the reduced set. The results shown are for $n = 3$; however, even if we increase the number of specialist features for each class pair to 12, we gain only slight improvement and it is still worse than ②. This might be a function of the poor quality of our feature pool, although it is unclear whether an improved pool would allow the 50 primary features to be outperformed by the few specialists.

Once the secondary features have been selected and measured, there is the additional possibility of including information from the original measurements in the new classification instead of basing the decision entirely on the few specialist features. This combination of information is denoted as ② + ③ and is slightly better than ② by itself but not better than ② + ① bringing into question the value of the use of the secondary set of features.

4.4 *Summary of whole correction process*

It is worthwhile to summarize the effect of the best version of the whole system. It is not entirely clear which system is best because of the error-correction tradeoff. It is clear that ② is better than ① or ③ alone. A careful examination shows that either ① + ② or ① + ② + ③ yield the best results depending on the confidence level. We will use the simpler version ① + ② as the best system. However, for purposes of independence ② alone may be the most desirable; this choice will be discussed later in the paper.

Table 6 gives a summary of the final correction, error and reject rates for three confidence levels as well as a distribution of the errors in the set of 21,841 words. The 45% word error rate has been reduced to the 2% range with a character error rate reduced from 9.7% to well below 1%.

4.5 *Addition of a dictionary to the CPP*

We have previously argued that dictionary correction systems are inefficient because of the number of words that must be looked up.[5] It takes too long to compare the word in error to the whole dictionary or to look up all words differing by 1 or 2 characters. However, the CPP system that we have described sharply reduces this number of possibilities. If a dictionary were available, then all of the CPP corrections to non-legal words will be avoided. Table 7 presents the same system with a dictionary added so that all potential corrections can be checked for dictionary correctness. The error-rate is cut by more than half and the rejection rate is reduced even more dramatically. The reader can decide for himself whether the additional storage for 2755 6-letter words and the additional computation is worth this improvement in the system.

5. DISCUSSION

The systems we have presented seem to be highly effective. An unacceptably large word error rate of 45% is reduced to the 1–2% range. A classifier making independent errors would have to have a character error rate in the 0.2–0.3% range to obtain these word error rates. A classifier with an initial error rate lower than the 9.7% used in this paper should produce very effective final results. As the initial character error rate is reduced, there are fewer multiple-error words produced. Since it is easier for the CPP to fix the position and correct single errors, we can expect both fewer errors to be produced initially by the classifier, as well as a larger percentage of the errors to be properly corrected. If a classifier with a *character* error rate of 2% is employed (causing a word error rate

Table 7. CPP-Dictionary System. (a) Final error-reject rates as a function of confidence level

	CONFIDENCE LEVEL		
	θ = 0	.9	.999
WORD ERROR RATE	1.2	1.1	.9
WORD REJECT RATE	0	.3	.7
CHARACTER ERROR RATE	.43%	.40%	.36%

(b) Distribution of errors in the 21,841 words

	NUMBER OF ERRORS							TOTAL ERRORS	TOTAL REJECTS
	0	1	2	3	4	5	6		
NO CONTEXT	11841	7662	2041	273	23	1	0	10000	0
θ = 0	21578	94	77	54	27	8	1	261	2
θ = .999	21490	47	66	49	25	7	1	195	156

of over 11%), we expect the final word error-reject rates would each be about 0.4% without a dictionary, 0.2% with a dictionary. One should note that this would be equivalent to about a 0.03–0.06% character error rate, a system that would be nearly impossible to build without the use of context if hand-printed characters comprise the pattern space.

The power of the system we have described resides in the difference between topologically similar characters and contextually similar characters. Of all the alternative systems examined, we feel that the CPP directly carrying out reclassification when it cannot correct a character is all that is necessary. Clearly, this seems to be the most effective process and allows us to avoid going back to the original character sample for additional measurements. This means a standardized CPP can be constructed entirely separately from the rest of the recognition system and allows the same CPP to be operated with many different front ends. It only needs to have the classifier *feed forward* the likelihoods used in making the decision.

The use of the substitution matrix in addition allows slight, but not significant, further improvement. The measurement of additional features directed by the CPP did not outperform direct reclassification. Of course, this is a function of the quality of the secondary features employed and one must be careful in generalizing to other problem domains.

The generalization of these results to text with errors distributed in a non-random manner (run of errors, characters interchanged, etc.) is of practical importance. Although we have just begun to seriously examine these difficulties, we feel that the system described here will have more success dealing with problems of this type than other systems described in the literature. Information concerning specific characteristics of error production such as 2 characters being reversed can be incorporated in the CPP to aid its processing. Errors caused by deletions and insertions of characters can be corrected by inserting a slot between each character to see if any can correct to an admissible word, or by deleting each character to see if it becomes admissible. Of course, this problem requires more examination because of problems such as ambiguity between a 2-error word or a 1-error deletion.

In order to have a complete text recognition system that operates on words of any length, we might need to build a memory to hold a set of binary trigrams for each set of words of a different length. The 2755 6-letter words we used would require about 10K of 32 bit words (although it would be much more efficiently organized as a special 26-bit word, or in general as a function of the number of classes). For efficiency, the memory could be partitioned to access all binary trigrams in parallel. There is reason to believe that for shorter words, the recognition rates reported would still be achieved. There are fewer binary digrams that can be employed for shorter words but there are also fewer words (resulting in sparser n-grams). Error rates should decrease in longer words because of the increase in the number of trigrams that could be employed.

In problem domains where there is contextual knowledge and character recognition systems cannot achieve acceptable recognition rates, a CPP designed in the manner we have described should provide at least an order of magnitude improvement. Further investigations to apply the CPP to postal reading machines are actively underway. Also, there may be many special purpose applications where the size of the input word set is much smaller than 2755. In these cases, much more dramatic results can be expected.

SUMMARY

If pattern recognition machines are to process text in any reliable fashion, the error rates that are often reported in the literature must be reduced by at least an order of magnitude. Typical *character* error rates in the range of 4–25% (in 6-letter words) lead to prohibitive *word* error rates ranging from 21% to 86%. This paper examines ways to extend an independent contextual postprocessor (CPP) developed in[1] and

to integrate it as a major subsystem in a full classification system.

The contextual postprocessor utilizes sets of positional binary n-grams. A binary trigram is a 26^3 binary matrix $Dijk$; the (l, m, n)th entry is 1 if and only if there exists some word in the input set which contains letters l, m and n in the ith, jth, and kth positions, respectively; otherwise, it has value 0. This is equivalent to quantizing trigram probabilities to values of 1 or 0 depending upon whether or not each probability is non-zero. The binary information is quite compact and algorithms for error detection and correction of words are computationally very efficient.

Using the set of all positional binary trigrams, experimental results demonstrate that 99% of errors in a set of 2755 6-letter words are detected. The CPP can also use this information to efficiently correct over 50% of the word errors. These results are compared to integrating context into the classifier by a straightforward Markov process (using trigram probabilities) and shown to be far more effective.

Although these results demonstrate effective processing, about 50% of the detected word errors remain uncorrectable. Many words are not corrected because of ambiguity in the correction process, that is, alternative character substitutions. The remainder of the paper examines ways of correcting these word errors by carrying out additional processing under the supervision of the CPP.

The CPP usually reduces the alternatives for correction down to a few choices. The latter portion of the paper compares various types of additional processing to improve performance. These include:

(a) request for further specification measurements (from a secondary pool of feature specialists) by the classifier whenever confidence of a decision is low; this is an attempt to markedly reduce classifier errors prior to CPP processing;

(b) use of the classifier substitution matrix to resolve ambiguity;

(c) CPP-directed reclassification (without further measurements) of the reduced set of alternatives;

(d) request for further secondary measurements by the CPP prior to reclassification; and

(e) various combinations of the above.

The techniques were compared empirically and show that almost all of the payoff can be obtained by CPP-directed reclassification. In order to accomplish this process, the CPP only needs the set of classifier likelihoods upon which the original classifier decision was based. It can use this data to reclassify in the reduced subspace. This structure allows the CPP to be built independently of the rest of the classifier and standardized. Examples of processing specific 6-letter words with one, two, and three errors give a sense of the quality of the system. In many cases, the system corrects multiple errors where a human finds the task quite difficult.

Experimental results demonstrate the effectiveness of the CPP by analyzing a character recognition system with a weak set of features and the simplest of classifiers. The techniques for additional processing result in more than an order of magnitude improvement. A 45% word error rate is reduced to about 2% word error rate and 1% reject rate. Presence of a dictionary allows these figures to be reduced even further.

REFERENCES

1. U. Neisser and P. Weene, A note on human recognition of hand-printed characters, *Inf. Control.* **2**, 191 (1960).
2. E. M. Riseman and R. Ehrich, Contextual word recognition using binary digrams, *IEEE Trans. Comp.* **C-20**, 397 (1971).
3. R. Ehrich and K. J. Koehler, Experiments in the contextual recognition of cursive script, *IEEE Trans. Comp.* **C-24**, 182 (1975).
4. A. R. Hanson, E. M. Riseman and E. G. Fisher, Context in word recognition, Univ. of Massachusetts, COINS Technical Report 74C-6, August 1974.
5. E. M. Riseman and A. R. Hanson, A contextual postprocessing system for error correction using binary n-grams, *IEEE Trans. Comp.*, **C-23**, 480 (1974).
6. C. M. Vossler and N. M. Bronston, The use of context for correcting garbled English text, *Proc. Ass. Comput. Mach. 19th Nat. Conf.*, pp. D2.4-1–D2.4-13, Aug. 1964.
7. F. Damerau, A technique for Computer detection and correction of spelling errors, *Commun. Assn. Comput. Mach.* **7**, 171 (1964).
8. J. Raviv, Decision making in Markov chains applied to the problem of pattern recognition, *IEEE Trans. Inf. Theory*, **IT-13**, 536 (1967).
9. G. Carlson, Techniques for replacing characters that are garbled on input, in 1966 *Spring Joint Comput. Conf., AFIPS Conf. Proc.*, **28**, 189. Washington, D.C., Spartan, 1966.
10. R. B. Thomas, M. Kassler and G. Woolley, *Advanced character recognition study*, Tech. Rep. 2 DDC-AD 435852, Dec. 1963.
11. 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, 12 Sept., 1961.
12. E. Tanaka, T. Kasai and M. Fujino, A Correcting Method of Garbled Languages Using Languages Informations, *Trans. IECE*, **54-C**, (294–301), (1971).
13. E. G. Fisher, E. M. Riseman and A. R. Hanson, Feature selection using thresholded measures, COINS Tech. Report 74C-9, December 1974—an earlier version of this appeared in Proceedings of 1973 International Conf. on Cybernetics and Society, Nov. 1973, pp. 94–95.
14. J. H. Munson, Experiments in the recognition of hand-printed text, *Proc. 1968-Fall Joint Comput. Conf.*, **33**, 1125 (1968).
15. A. B. S. Hussain, G. T. Toussaint and R. W. Donaldson, Results obtained using a simple character recognition system on Munson's hand-printed data, *IEEE Trans. Comput.*, **C-21**, 201 (1972).
16. I. J. Good, *The Estimation of Probabilities.* Research Monograph No. 30, Cambridge: MIT Press, 1965.

About the Author—EDWARD M. RISEMAN 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 conducted research in switching theory, feature selection, character recognition, and texture analysis. Currently, his principal interests involve the application of contextual information to word recognition, computer vision and mechanized inference systems.

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

About the Author—ALLEN R. HANSON 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. 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, IEEE, and the Pattern Recognition Society.

About the Author—EDWARD G. FISHER was born in Manchester, New Hampshire on April 19, 1946. He received his B.S. in Mathematics from the University of New Hampshire in 1971 and his M.S. in Computer Science from the University of Massachusetts in 1974. He is currently finishing his Ph.D. in Computer Science at the University of Massachusetts. His dissertation topic is "Context in Text Recognition."

He will begin working for Pattern Analysis and Recognition Corporation, Rome, N.Y., in the fall of 1975. His research interests include: pattern recognition, with an emphasis on the use of context; operating systems, measurement and evaluation; and the study of algorithms.

He is a member of ACM, IEEE, the Pattern Recognition Society, and Pi Mu Epsilon.