CONTEXTUAL ERROR DETECTION

Roger W. Ehrich
Edward M. Riseman
Department of Computer and Information Science
Technical Report 70C-4
University of Massachusetts
Amherst, Mass. 01002

CONTEXTUAL ERROR DETECTION

by R. W. Ehrich and E. M. Riseman

Abstract

Much of the contextual information which characterizes words of a finite dictionary can be represented by a set of binary matrices called binary digrams. It is shown that such a representation is particularly useful for designing contextual post-processors for improving the performance of word recognition systems. Since binary digrams usually contain redundant information about the dictionary, an important problem in the design of a contextual post-processor is the selection of a set of binary digrams which gives a maximal amount of information about the dictionary.

In this paper a heuristic procedure is presented for selecting a "good" set of binary digrams, and it is shown how the solution to this problem can be used to achieve contextual error detection in words by using a very simple algorithm. Simulation results are given which verify the selection procedure. This same selection procedure can be used in designing contextual post-processors for word recognition systems, since in such a system only the letter distributions are different.

INTRODUCTION

In word dictionaries of finite size, it is usually true that the statistics of certain letter pairs characterize the words in the dictionary more accurately than other pairs. In this paper, we discuss a heuristic procedure for selecting good subsets of letter pairs for characterizing a dictionary by the contextual constraints it imposes on letter sequences. In this section, we discuss the motivation for this problem.

In a recent paper [1], we have demonstrated that a character recognizer can achieve a low error rate if a dictionary and post-processor is added, even if the classifier outputs as many as 5 or 6 alternatives, without confidences, for each letter it must recognize. Our system is designed to be as simple as possible and to require as little storage as possible. Briefly, it functions in the following manner.

Suppose that either by accident or by design a character recognizer cannot determine uniquely the character being scanned. It outputs instead a set of alternatives, called a <u>substitution set</u>, to a contextual post-processor. At the end of the word, the contextual post-processor selects those sequences which can be formed from the substitution sets and which may have been the input word to the recognizer. Suppose that the character A_k occurs in the i^{th} position of some dictionary word and that A_ℓ occurs in the j^{th} position of the same word. This information is recorded by placing a 1 in the $(k,\ell)^{th}$ position of a 26 x 26 binary matrix, D_{ij}, called a <u>binary digram</u>.

Thus, $D_{ij}(k,\ell) = 1$ implies that some dictionary word has A_k in position i and A_ℓ in position j. Obviously, for length n words there are n(n-1)/2 possible digrams with $1 \leq i \leq j \leq n$.

The post-processor uses some subset of the n(n-1)/2 possible digrams to form from the substitution sets only those letter sequences for which the appropriate digram entries are all 1's. Any remaining sequences are looked up in the dictionary. We have shown [1] that such a procedure requires considerably fewer computations than, say, checking the dictionary words one by one to see if they can be formed from the substitution sets. In our simulations we have verified, for example, our prediction of a 1.8% error rate with no rejection on a 300 7-letter word dictionary with 5 letter substitution sets or .08% with 2-letter substitution sets. It is not our intention here to discuss our computational algorithm; let it suffice to say that the examples cited required, on the average, about 350 and 30 digram references per word, respectively this computation comparing favorably with other methods.

It is not hard to see that the n(n-1)/2 digrams contain much redundant information. For this reason and for reasons of storage economy, one would normally select some subset of the available digrams instead of using all of them, and the selection will drastically affect the computational speed of the algorithm. Since this is a very difficult problem, we will content ourselves with a procedure for making "good" choices.

Essentially, we are viewing words as Markov sequences with position dependent transition matrices in which probabilities are quantized

to be either zero or non-zero. From yet another point of view, we are approximating n-gram statistics of dictionary words with selected binary digrams at very substantial savings in storage costs.

THE SELECTION PROBLEM

Since the main use of binary digrams is in discriminating between letter sequences which are in the dictionary and those which are not, it is convenient to formulate a related problem where we are interested only in error detection. Suppose the letters in dictionary words are modifed randomly according to some fixed error rate. Given that we wish to select k of the $n(n-1)/2$ digrams to represent the dictionary, choose those k digrams which maximize the probability of deciding correctly whether or not a given word with random errors is in the dictionary. Given a word and a set of k digrams, we will decide that the word is in the dictionary if the appropriate entry of <u>each</u> digram contains a one.

The choice of digrams will depend upon the probability distribution of letter pairs in the words with errors. Let $0 \le E \le 1$ be the error rate and let $p^d_{ij}(k,\ell)$ be the probability of observing A_k in position i and A_ℓ in position j of a dictionary word. Then the distribution for words with errors is given by

$$p_{ij}(k,\ell) = (1-E)^2\, p^d_{ij}(k,\ell) + E^2 \sum_{t,s \neq k,\ell} \frac{p^d_{ij}(t,s)}{25^2} + E(1-E)\left[\sum_{t\neq k} \frac{p^d_{ij}(t,\ell)}{25} + \sum_{s\neq \ell} \frac{p^d_{ij}(k,s)}{25}\right]$$

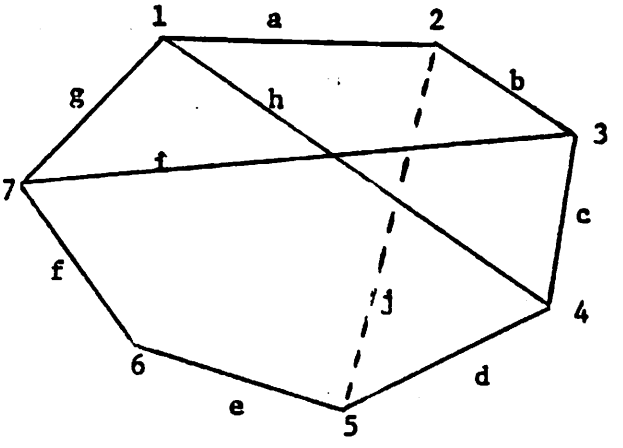
$$= (1-E)^2\, p^d_{ij}(k,\ell) + \frac{E^2}{25^2}\left(1-p^d_{ij}(k,\ell)\right) + \frac{E(1-E)}{25}\left[\sum_{t\neq k} p^d_{ij}(t,\ell) + \sum_{s\neq \ell} p^d_{ij}(k,s)\right].$$

Here we assume that the errors in the words to be recognized are uniformly distributed, but obviously a similar analysis exists for other distributions. We also assume all dictionary words to be equally likely to appear.

Our procedure will be to select digrams sequentially in such a way that each new digram will have a maximal probability of detecting an error which was not detected by a previous digram. In order to be able to choose a new digram precisely, we would have to recompute after every step, the pairwise distribution of letters based on those sequences which have been eliminated by previous digrams. This becomes an extremely complex combinatoric problem which we wish to circumvent by some rough approximations.

As a preliminary procedure, we will always choose those digrams D_{ij} whose zeros coincide with the highest possible probabilities of the distribution p_{ij}, i.e. that digram D_{ij} for which $P_{ij} = \sum_{k,\ell} \overline{D_{ij}}(k,\ell) p_{ij}(k,\ell)$ is maximum. Here $\overline{D_{ij}}$ denotes the Boolean complement of D_{ij} and P_{ij} is the probability that D_{ij} alone will cause rejection of an input word. As more digrams are selected, the problem becomes more complicated because the digrams contain redundant information. Consider positions i,j and h and suppose D_{ij} and D_{jh} have been selected. All that can be gained from selecting D_{ih} in addition corresponds to those positions where D_{ih} has a zero and the Boolean matrix product $D_{ij}D_{jh}$ is one. Notice that $D_{ih}(k,\ell)$ is zero wherever $D_{ij}D_{jh}(k,\ell)$ is zero. Thus, in considering D_{ih}, the corresponding value of P_{ih} is now $P_{ih} = \sum_{k,\ell} (D_{ih}(k,\ell) \oplus D_{ij}D_{jh}(k,\ell))\ P_{ih}(k,\ell)$, where $\oplus$ is the exclusive OR, and the selection procedure is still to select that digram whose corresponding P_{ih} is maximum. More complicated cases which arise are treated similarly. Consider the case illustrated in the digram below,

where the solid lines indicate digrams already chosen and the dashed line is a candidate for the next choice. The vertices correspond to letter positions.



For this example, one begins by enumerating all paths from 2 to 5, which can be obtained by enumerating all the circuits containing branch j. From graph theory, the upper bound can be easily seen to be $2^{N(G)}$ paths, where the nullity of the graph, N(G), is the number of branches minus the number of vertices plus one. In this case, we obtain the 8 paths from 2 to 5:

P_1 = agfe $\qquad$ P_5 = ahd

P_2 = bcd $\qquad$ P_6 = bchgfe

P_3 = bighd $\qquad$ P_7 = ahcife

P_4 = agicd $\qquad$ P_8 = bife

For each path, compute the Boolean matrix product of the corresponding digrams. For example, for P_1 we obtain $D_{21}D_{17}D_{76}D_{65}$, where $D_{ij} = D_{ji}^t$. Let $\hat{D}_{25}$ be the logical AND of these matrix products. Then

$$P_{25} = \sum_{k,\ell} (D_{25}(k,\ell) \oplus \hat{D}_{25}(k,\ell))\ P_{25}(k,\ell).$$

In this case, P_5 causes P_3 to be redundant because D_{12} contains all the zeros of $D_{17}D_{73}D_{32}$. Similarly, P_4, P_6 and P_7 are redundant due to P_2, P_1 and P_8, respectively, so that

$$\hat{D}_{25} = D_{21}D_{17}D_{76}D_{65} \wedge D_{23}D_{34}D_{45} \wedge D_{21}D_{14}D_{45} \wedge D_{23}D_{37}D_{76}D_{65}.$$

The appendix is a description of the actual digrams selected and the paths considered in the problem examined in the section on experimental results.

The problem with this procedure is that as more and more digrams are selected the distributions $p_{ij}(k,\ell)$ change. Suppose D_{ij} has been selected but that neither D_{jh} nor D_{ih} have been chosen. Some of the information in D_{jh} is contained in D_{ij}. For example, suppose the r^{th} column of D_{ij} is all zero which implies that no dictionary word has A_r in position j. This information is also in D_{jh} since its r^{th} row will be zero, and choice of D_{jh} will not tell us much we didn't already know. The following example for a 2-letter alphabet [A,B] will motivate an approximate procedure for updating the distributions p_{ij} wherever a new digram is selected. Suppose D_{12} has been selected and we wish to evaluate D_{23}, where

$$D_{12} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad D_{23} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$P_{12} = \begin{bmatrix} .3 & .4 \\ .2 & .1 \end{bmatrix} \qquad P_{23} = \begin{bmatrix} .3 & .2 \\ .3 & .2 \end{bmatrix}$$

Since D_{12} has been selected there is a probability of .2 that A in position 2 will be rejected as an error. Therefore, given that A occurs in position 2, the probability is .2/(.2 + .3) = .4 that it will be rejected by D_{12}. Since .4 of the A's in position 2 are rejected by D_{12}, D_{23} is no longer so effective in rejecting A's in position 2.

If we assume that $p_{23}(j,k)$ is independent of position 1 then

$$p_{123}(i,j,k) = p_1(i)\ p_{2|1}(j|i)\ p_{3|1,2}(k|i,j) \approx p_1(i)\ p_2(j)\ p_{3|2}(k|j)$$

and $$p_{23}(j,k) \approx p_2(j)\ p_{3|2}(k|j).$$

Thus $p_{23}(j,k)$ can be updated by reducing the first row of p_{23} by .4. This gives

$$p'_{23} = \begin{bmatrix} .3(.6) & .2(.6) \\ .3 & .2 \end{bmatrix} = \begin{bmatrix} .18 & .12 \\ .3 & .2 \end{bmatrix}$$

Now, the probability of D_{23} rejecting a word is .38 instead of its value independent of D_{12} of .50. Therefore, once a digram D_{ij} is selected all distributions P_{jh} must be updated according to

$$p'_{jh}(\ell,m) = \left[1 - \frac{\sum_k p_{ij}(k,\ell)\ \overline{D_{ij}}(k,\ell)}{\sum_k p_{ij}(k,\ell)}\right] p_{jh}(\ell,m)$$

and similarly all distributions p_{gi} are modifed. These new probability distributions are now used in selecting the next digram.

In the preceding equation, there is some question as to whether the distributions $p_{ij}(k,\ell)$ should be the same as at the beginning of the procedure or the updated distribution from the previous step. Since our approximation is very rough, we feel that the original distribution should be used, since otherwise there could be considerable cumulative error.

EXPERIMENTAL RESULTS

The methods described have been tested to determine the improvement in error detection on a sample of 300 7-letter words. A letter

error rate of 1% was introduced into the words on several trials yielding a 7.0% word error rate over 3000 samples. Table 1 is a comparison of three different digram selection techniques. The first is a random choice of digrams; the second is an analysis of the digrams on the basis of initial letter pair distributions. Neither of these two selection methods utilized the techniques for reducing redundancy that have been described. The third column shows the results when redundancy is taken into account. Note that in the very large majority of cases, there is significant improvement varying from about 20% to over 50% reduction in the number of errors. In no case is there an increase in undetected errors. Note that with only 8 digrams 86% of the errors are detected and that the original 7% error rate has been reduced to 1%; if all 21 digrams are used, only 6 undetected errors of the original 210 remain, a final error rate of .2%. This compares favorably with the use of a single non-positional binary trigram which requires more storage and results in 18 undetected errors. (Non-positional refers to the more usual case of the digram or trigram taking on values irrespective of the position in the word; consequently, a non-positional binary digram or trigram is just the logical OR of all the positional binary digrams and trigrams, respectively.)

No. of digrams	1	2	3
1	177	171	171
2	150	149	121
3	132	99	82
4	101	94	59
5	78	69	56
6	64	48	48
7	57	40	38
8	56	40	29
9	50	40	23
10	45	35	19
11	24	27	18
All 21 digrams - 6 errors			
One non-positional trigram - 18 errors			

TABLE 1

Table 1 - Undetected errors out of 210 vs. number of digrams used. Letter error rate is 1%, yielding an initial word error rate of 7% of the 3000 samples. Table gives remaining undetected errors.

Column 1 - Random choice of digrams.

Column 2 - Digram choices made independently and based only upon initial letter pair distributions $p_{ij}(k,\ell)$; choose k largest

$$P_{ij} = \sum_{k,\ell} \overline{D_{ij}}(k,\ell) p_{ij}(k,\ell).$$

Column 3 - Digram choice made sequentially and based upon both minimum redundancy and updated pair distributions $p_{ij}(k,\ell)$; full procedure described in paper.

CONCLUSION

We have presented an analysis of redundancy in a most useful representation of the contextual information contained in finite dictionaries. In our model, the probabilities in positional binary digrams have been quantized to 1 or 0. Such techniques can be applied in other instances of the general problem of deciding which of the vast amount of contextual information should be used when finite storage resources are available. Because of quantization binary trigrams are also feasible, and these will be useful for very large dictionaries or where better error detection is desired.

Our analysis can certainly be improved by exact updating of all joint probabilities whenever a digram is selected, but we feel that our method enables us to make good enough choices. In our procedure, the updated probabilities become less and less accurate because of our independence assumption and because we do not update all probability distributions.

Experimental data verifies the improvement obtained by considering redundancy in information. It further shows the ability to obtain a high error detection rate with an extremely simple use of context. Certainly error detection is reduced as the dictionary is enlarged, since context becomes less and less. In such cases we need to store more information about the dictionary.

APPENDIX

Table 2 is a list of the digrams in the order they were chosen and the set of paths that were needed to evaluate the non-redundant

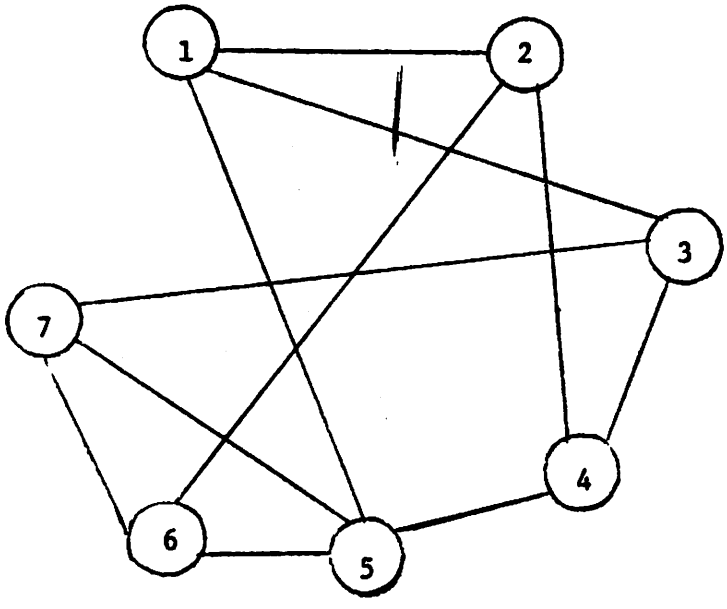


TABLE 2

DIGRAM SELECTED	PREVIOUS PATHS TO BE CONSIDERED
D_{67}	
D_{12}	
D_{34}	
D_{57}	
D_{26}	
D_{37}	
D_{15}	$D_{12}D_{26}D_{67}D_{75}$
D_{24}	$D_{21}D_{15}D_{57}D_{73}D_{34} \wedge D_{26}D_{67}D_{73}D_{34}$
D_{45}	$D_{43}D_{37}D_{75} \wedge D_{42}D_{21}D_{15} \wedge D_{42}D_{26}D_{67}D_{75}$
D_{13}	$D_{12}D_{24}D_{43} \wedge D_{15}D_{57}D_{73} \wedge D_{12}D_{26}D_{67}D_{73} \wedge D_{15}D_{54}D_{43}$
D_{56}	$D_{57}D_{76} \wedge D_{51}D_{12}D_{26} \wedge D_{54}D_{42}D_{26}$

information in the selection of each digram. Note that this technique of determining redundancy is applied only where a path between the two positions of the digram to be evaluated already exists using digrams previously selected. This set of digrams corresponds to the data obtained in Column 3 of Table 1. The digram, D_{ij}, that was selected at each step was that digram that resulted in the maximum P_{ij} using the methods described in the paper.

REFERENCES

[1] E. Riseman and R. Ehrich, "Contextual Word Recognition Using Binary Digrams," *IEEE Transactions on Computers*, C-20, April, 1971.

[2] B. Gold, "Machine Recognition of Hand-Sent Morse Code," *IRE Trans. Inform. Theory*, Vol. IT-5, March, 1959, pp. 17-24.

[3] W. W. Bledsoe and J. Browning, "Pattern Recognition and Reading by Machine," *1959 Proc. Eastern Joint Computer Conf.*, New York: National Joint Computer Committee, December 1959, pp. 225-232.

[4] C. R. Blair, "A Program for Correcting Spelling Errors," *Inform. Control*, Vol. 3, 1960, pp. 60-67.

[5] E. J. Sitar, "Machine Recognition of Cursive Script: The Use of Context for Error Detection and Correcting," Bell Telephone Laboratories, Inc., Murray Hill, N. J., unpublished memorandum, Sept. 12, 1961.

[6] L. D. Harmon and E. J. Sitar, "Method and Apparatus for Correcting Errors in Mutilated Test," U.S. Patent 3 188 609, June 8, 1965.

[7] C. K. McElwain and M. B. Evens, "The Degarbler - a Program for Correcting Machine Read Morse Code," *Inform. Control*, Vol. 5, 1962, pp. 368-384.

[8] R. B. Thomas, M. Kassler and G. Woolley, "Advanced Character Recognition Techniques Study," Tech. Report,4, DDC-AD 435852, December, 1963.

[9] C. M. Vossler and N. M. Bronston, "The Use of Context for Correcting Garbled English Text," in *Proc. ACM 19th Nat. Conf.*, August, 1964, pp. D 2.4-1 - D 2.4-13.

[10] A. W. Edwards and R. L. Chambers, "Can A Priori Probabilities Help in Character Recognition?" *J. ACM*, Vol. 2, October, 1964, pp. 465-470.

[11] G. Carlson, "Techniques for Replacing Characters that are Garbled on Input," in *1966 Spring Joint Computer Conf., AFIPS Conf. Proc.*, Vol. 28, Washington, D.C.: Spartan, 1966, pp. 189-192.

[12] J. Raviv, "Decision Making in Markov Chains Applied to the Problem of Pattern Recognition," *IEEE Trans. Inform. Theory*, Vol. IT-13, October, 1967, pp. 536-551.

[13] R. O. Duda and P. E. Hart, "Experiments in the Recognition of Hand-printed Text: Pt. II Context Analysis," in *1968 Fall Joint Computer Conf., AFIPS Proc.*, Vol. 33, Washington, D.C.: Thompson, 1968.

[14] C. S. Christensen, "An Investigation of the Use of Context in Character Recognition Using Graph Searching," Ph.D. dissertation, Center for Applied Mathematics, Cornell University, Ithaca, New York., Tech. Report., AFOSR 68-2470, November, 1968.