A CONTEXTUAL POSTPROCESSING SYSTEM FOR ERROR DETECTION AND CORRECTION IN CHARACTER RECOGNITION

Edward M. Riseman
Allen R. Hanson

Department of Computer and Information Science
University of Massachusetts at Amherst

Technical Report 72C-1
October 1972

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

TABLE OF CONTENTS

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
University of Massachusetts, Amherst, Mass. 01002	UNCLASSIFIED
	2b. GROUP: None

3. REPORT TITLE

A Contextual Postprocessing System for Error Detection and Correction in Character Recognition

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical report

5. AUTHOR(S) (First name, middle initial, last name)

Edward M. Riseman, Allen R. Hanson

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October 1972		31

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-67-A-0230-0007	COINS Technical Report 72C-1
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Office of Naval Research, Code 437 Washington, D.C.

13. ABSTRACT

This paper is an examination of the effectiveness of various forms of contextual information in a postprocessing system for detection and correction of errors in words. 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 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 techniques described in this paper 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% of them. As an example, a set of binary positional trigrams operating on a set of 800 six-letter words can reduce a word error rate of 47% to a reject rate of 8.9% and an error rate of only .38%.

This paper demonstrates that computationally efficient procedures can be used to obtain solutions to difficult classification problems, such as automated handwriting recognition, by dropping final word error rates into acceptable ranges. Finally, a more powerful integrated feedback recognition system is proposed which utilizes information fed back from the contextual postprocessor to both the classifier and feature measurement sections of the system.

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]. In the area of character recognition, the potential aid of contextual tecnhiques appears great. Consequently, most of the investigations involving context have centered there. Humans use context in reading so extensively that quite often they do not even realize they have read a misspelled word correctly. In many cases the word is understood only by the use of contextual information.[1] This information might involve knowledge of the language structure and/or knowledge of the subject topic. Within a word the characters surrounding those in error might be sufficient to suggest the corrections; it has been estimated that the English language is over 50% redundant [28]. However, the syntax and semantics of the surrounding words, sentences, and paragraphs might also contain valuable information.

Contextual techniques have been incorporated in character recognition systems directly into the classification process as well as in a postprocessing system for error detection and possibly error correction. Semantic information has been utilized only a few times and usually in programming languages where the syntax and semantics is well defined [19-21]. In most cases, the contextual analyses were restricted to deciding upon a single word on the basis of the characters within the word; that is, the syntactic relationship of the words in the sentence and the semantics of the sentence were not utilized. The approaches basically involved using either a

[1] In fact the recognition rate on isolated characters is only about 96% [27].

dictionary of words or information, often probabilistic, concerning the structure of the words in the language (or dictionary). Once a dictionary is employed, the system is restricted; even if storage is not a difficulty, dictionary searches and comparisons will increase exponentially with the length of the list. Thus, an increase in computation time of an algorithm with a large dictionary must be carefully considered. 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].

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 improvement in error-reject rates 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 = 456,000 words). The justification for this expense is that one would not have to store a large dictionary (although most usable dictionaries of a language would not be as large as 450K words) but the main point is the removal of the need for lengthy searches and long computation times.

The systems that have been developed generally are directed towards a classifier followed by an independently operating contextual postprocessor for detection and/or correction of errors. Rather than carry out separately the classification of each character on the basis of some set of measurements and then employ letter statistics afterwards, theoretically one procedure could be employed utilizing all this information in an optimal fashion. This would remove any need or use for an error detection system. Raviv [14] considered the problem of designing such an optimal classifier. Quite often, though, assumptions of the independence of measurements (which is almost always a false assumption) are made to reduce enormous computation and storage requirements. This assumption allows one to avoid collecting and storing the joint probability distribution of the set of measurements conditioned upon each pattern class. Assumptions such as these, however, usually destroy the optimality of the procedure and then the system might still benefit from a contextual postprocessing system. In contrast the contextual algorithms that will be presented greatly simplify both the process of collecting and storing contextual statistical information and the computational complexity; many of the errors introduced by the simplifying assumptions just mentioned are rectified through this use of context.

It was first noted by Sitar that almost 50% of the digram probabilities are zero; a much lower density of non-zero 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 zero or non-zero, respectively [18]. 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 that will be discussed.

This paper is an examination of the relative effectiveness of the various forms of contextual information in a postprocessing system for error detection and correction. The analysis will discuss context ranging from a complete dictionary to quantized digram statistics. 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 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, and it is known to be a sample of a dictionary word, then an error has been detected. If the word is to

---

2 The only further non-semantic information that could be available would be the word probabilities.

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. Table II.1 indicates the probability that a word in error contains no more than two errors as a function of word length and the character error rate.[3] It is clear that the bulk of words in error involve a single symbol error if the classifier has a reasonably high recognition rate.

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 word in the dictionary that it is most similar to. We will say it is <u>inherently uncorrectable</u> 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.

---

[3] For this calculation it has been assumed that errors are independent of each other.

Table II.1 - Probability of no more than two errors in a word among all words in error

word length	.01	.05	.10	.20
3	≈1.0	.999	.996	.984
4	≈1.0	.997	.989	.954
5	≈1.0	.995	.979	.914
6	≈1.0	.992	.966	.866
7	≈1.0	.988	.951	.813
8	.999	.983	.933	.756

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 similar words to 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.

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. 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 a large amount of storage or long computation times. 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 dictionary syntax) is in the form of a data base of quantized n-grams [18]. These n-grams, for any n, may be either positional or non-positional, 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 non-positional 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. 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 six-letter words, fifteen ($6 \times 5/2$) of them are required to contain all pairwise positional information contained in the dictionary. As an example, the contextual algorithm using binary digrams will be described. Clearly the algorithm will function similarly with any binary n-gram by extension.

The dictionary may be partitioned by word length. For each of these sub-dictionaries, a pair of letter positions i and j can be used to define a 26 × 26 binary matrix, d_{ij}, called a <u>binary digram</u>. 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 i^{th} position and letter ℓ in the j^{th} 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? It has been demonstrated that in some instances this is an effective approximation of the structure of the dictionary. The selection of a subset of these digrams for error detection was considered in [29].

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 almost 50% 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 more 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 procedures 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 non-zero entries. In essence, the techniques that will be described involve approximating the information in ℓ-grams with binary n-grams, $1 < n < \ell$.

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: $\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_\ell}$ where α_{i_k} is a member of the alphabet. Then we must have

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

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 were available (inherently detectable) but that are undetectable by the dictionary syntax. An example that illustrates this very simply is:

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: $D_{13}(\alpha_{i_1}, \alpha_{i_3})$, $D_{23}(\alpha_{i_2}, \alpha_{i_3})$, $D_{34}(\alpha_{i_3}, \alpha_{i_4})$, $D_{35}(\alpha_{i_3}, \alpha_{i_5})$, $D_{36}(\alpha_{i_3}, \alpha_{i_6})$. If at least two 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 there is only one position, 3, that appears more than once in the detection of the error. If some character is to replace α_{i_3}, then each of the 5 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 5 digrams, we will have utilized all the available information:

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

where D^T_{ij} denotes the transpose of D_{ij}. If the j^{th} 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).

The process that has been described takes place after the position of the error is fixed. The position may be fixed by examining the number of times each position has occurred in the set of digrams that detect an error. If only one error has occurred in a word, only one position can

occur more than once. The violation of this condition implies that more than one error has occurred. On the other hand a word might have multiple errors and it may appear that only a single error has occurred. In this case the word in error might be improperly corrected, thus re-introducing an error.

When binary digrams are utilized by the context algorithm, error correction will be attempted in two cases. In the first case, if it appears that a single error has occurred, one alternative letter exists, and the position of that error appears to be determined, the correction will be made. If only one digram detects the error, there is uncertainty which of the two positions contains the error or even whether there are two errors. However, when there is a choice of more than one position, the correction algorithm can be applied to each position independently. If one position has only one possible letter substitution and the other has none, the correction is 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.

If higher n-grams ($n > 2$) are utilized, an attempt is made to correct 2-error words as well as 1-error words. The positions in error are easier to fix using higher n-grams because a larger number of positional n-grams probably detect the error. Although it will not be discussed here, in many cases, it is not difficult to determine the positions of a pair of errors by counting the number of times each position occurs in the set of n-grams detecting the error. An apparent two-error word is corrected if both positions are correctable (i.e., only one possible letter substitution exists for each error position). If the positions in error are not clearly fixed, the

position(s) may be determined by elimination as in the case of digrams. Again, any other case is considered to be uncorrectable.

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 was able to detect 40 and 60 percent of the errors, respectively. However, only 80 to 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. (This encoding procedure loses all positional information.) 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.

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 results obtained are shown in Table IV.1 for text using a 364-word dictionary. It is obvious that the use of the dictionary is far superior to digram information. A procedure combining the two 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 this case, however, the classifier was used to signal the postprocessor as to which character is being confused; thus the problem of detecting and fixing the position of the error is not considered. 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

Table IV.1 - experiemental results from Vossler & Branston [11]

	DICTIONARY METHOD		DIGRAM METHOD	
INITIAL CHARACTER ERROR RATE	3.2%	19.4%	4.2%	19.2%
CHARACTER ERROR CORRECTION RATE	93%	89%	45%	35%
FINAL CHARACTER ERROR RATE	.22%	2.2%	2.3%	12.5%
INITIAL WORD ERROR RATE	10.9%	55%	14.2%	50.7%
WORD ERROR CORRECTION RATE	93%	92%	45%	40%
FINAL WORD ERROR RATE	.79%	4.6%	7.8%	30.6%

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. 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; however, one can expect that the detectability and certainly the correctability rates will improve directly as a function of word length.

The Data Base

A list of 2755 six-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 six-letter words compiled by the authors. The remaining 2455 words were obtained from Thorndike-Lorge [31] by selecting all six-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 virtually all six-letter words commonly used. From those subsets (comprising the word data base) the syntax was extracted and stored in the binary n-grams.

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

1ND: one non-positional digram requiring 27^2 bits of storage;*

15PD: the entire set of 15 positional digrams requiring 15×26^2 bits of storage;

1NT: one non-positional trigram requiring 27^3 bits of storage;

20PT: the entire set of 20 positional trigrams requiring 20×26^3 bits of storage;

1NQ: one non-positional quadgram requiring 27^4 bits of storage.

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

4PT: a subset of four of the 20 positional trigrams requiring 4×26^3 bits of storage;

6PT: a subset of six of the 20 positional trigrams requiring 6×26^3 bits of storage.

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.

* Note that blanks must be considered in non-positional 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.

A test set of 600 one-, two-, and three-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 shoen in Figures V.1a-c. The detectability rate for all positional trigrams (20PT) on one-error wards varies from 99.8% to 98.6%. As expected, the dictionary algorithm is only slightly higher, varying from 99.8% to 99.6%, while all other n-grams yield lower detectability rates. 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 storege that is employed. Also, there is a surprising payoff from the use of one non-positional digram in one-error detection. Between 30% and 65% of these errors are detected with the use of just 729 bits (less than 50 sixteen-bit-words in a minicomputer).

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 Figure V.2.

Experiment #2. Correctability

The object of this experiment is to determine the conditional correction rate of detectable errors; that is, given that an error has beet detected, what is the probability of correcting it? The correction algorithm has been described elsewhere in che paper. Figures V.3a and V.3b indicate the conditional correction rates obtained on one- and two-error words. No attempt was made to correct any three-error words, 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. In the case of the largest word sets, the dictionary algorithm corrects about 20% more of the one-error words.

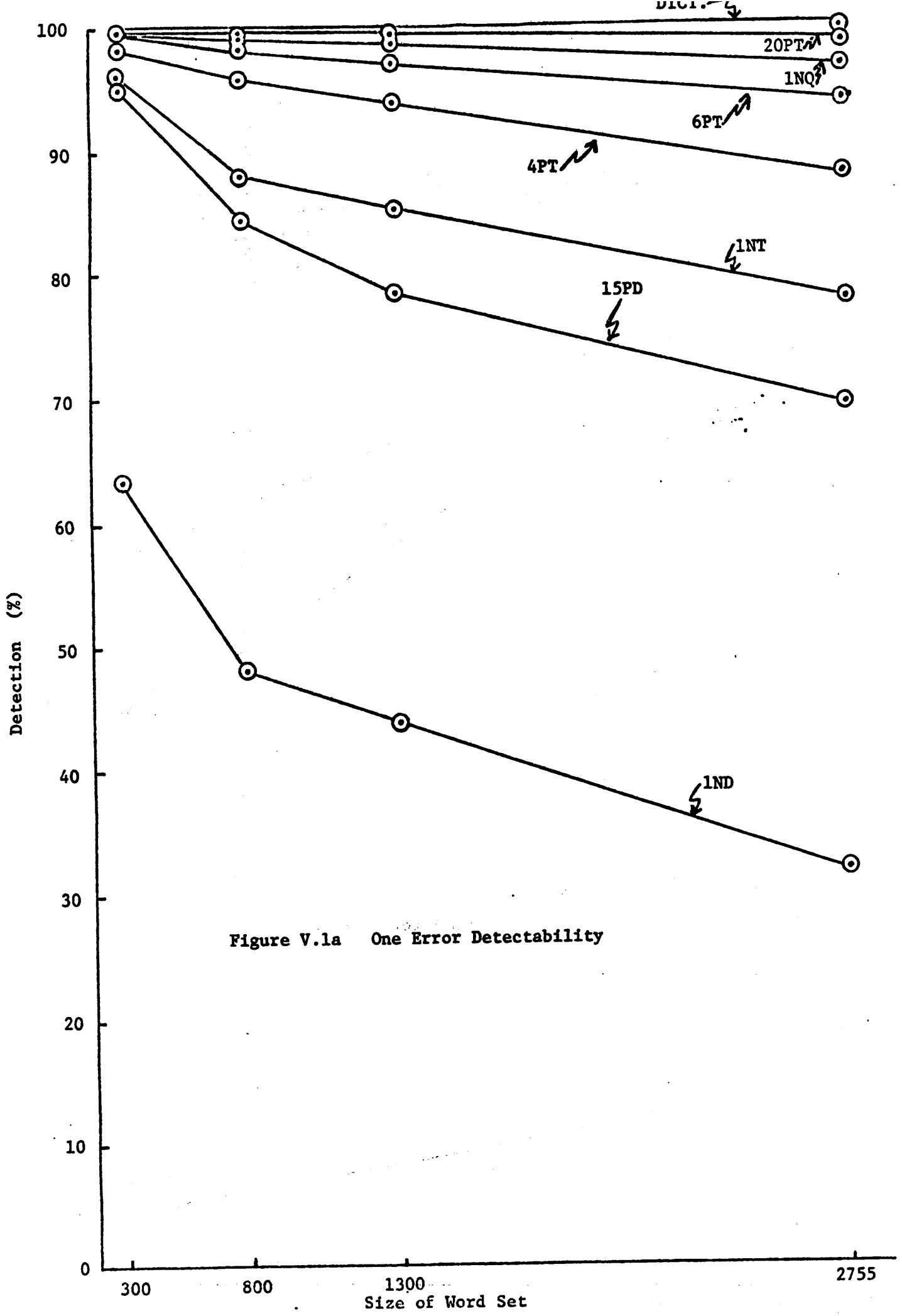


Figure V.1a One Error Detectability

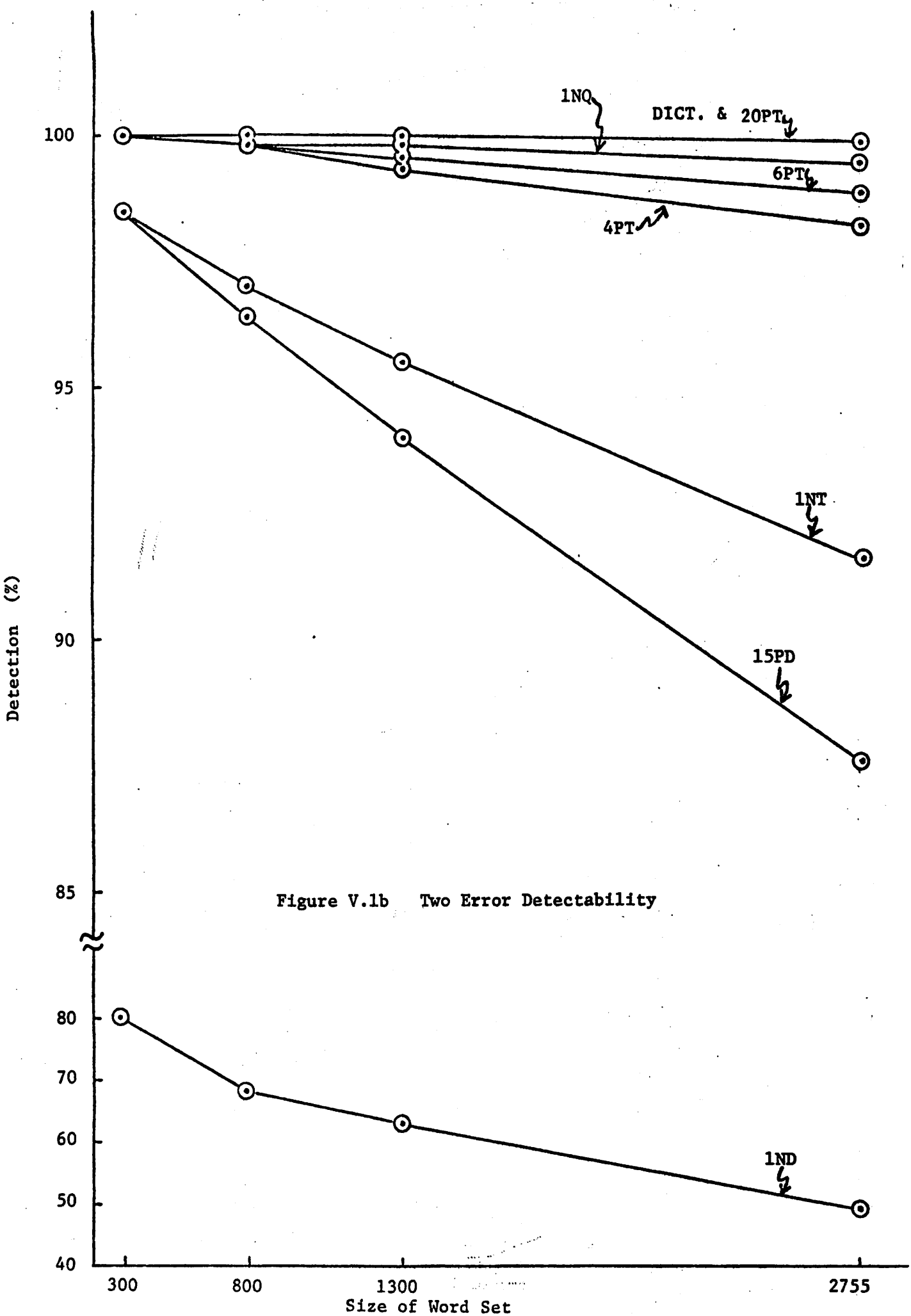


Figure V.1b Two Error Detectability

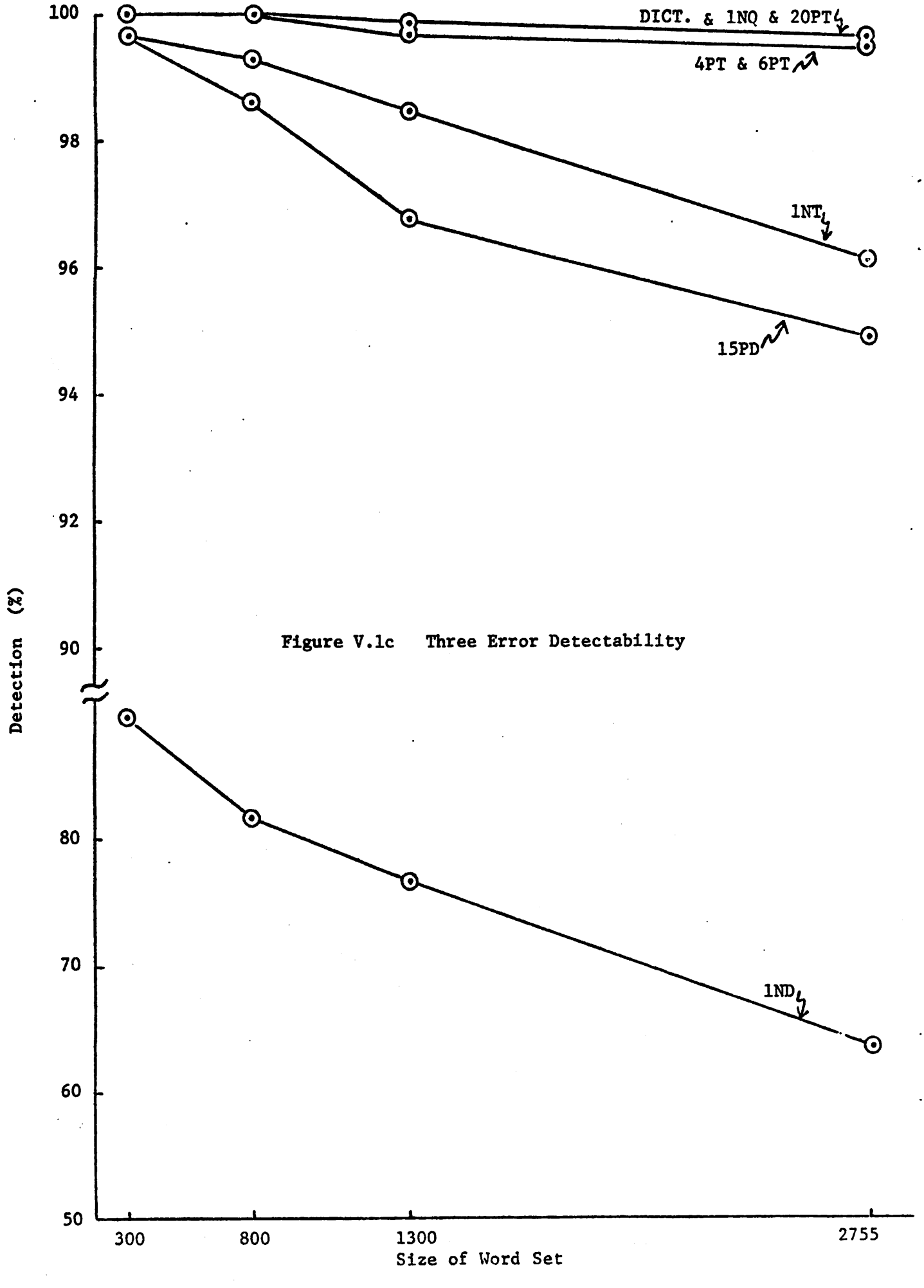
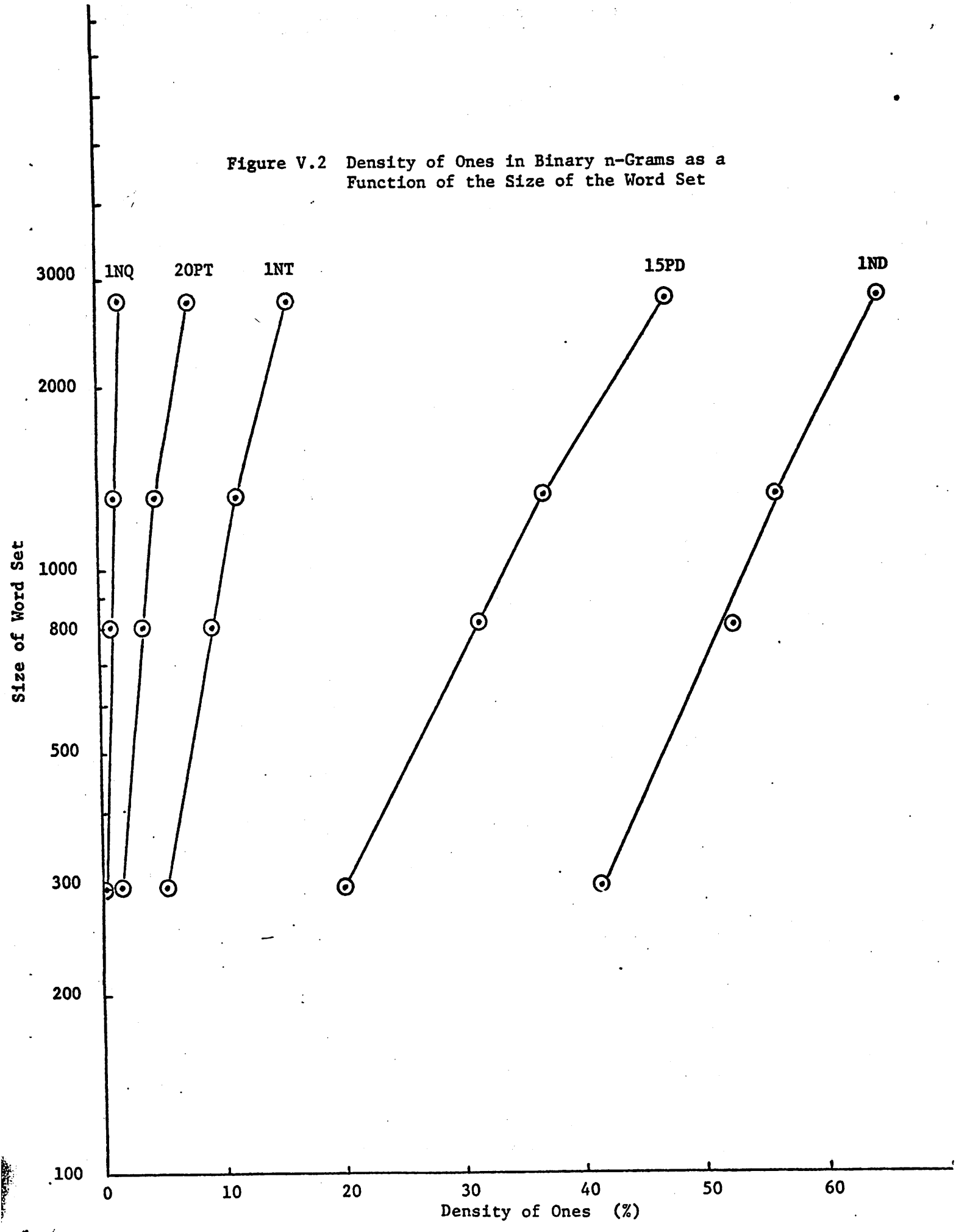


Figure V.1c Three Error Detectability

Figure V.2 Density of Ones in Binary n-Grams as a Function of the Size of the Word Set



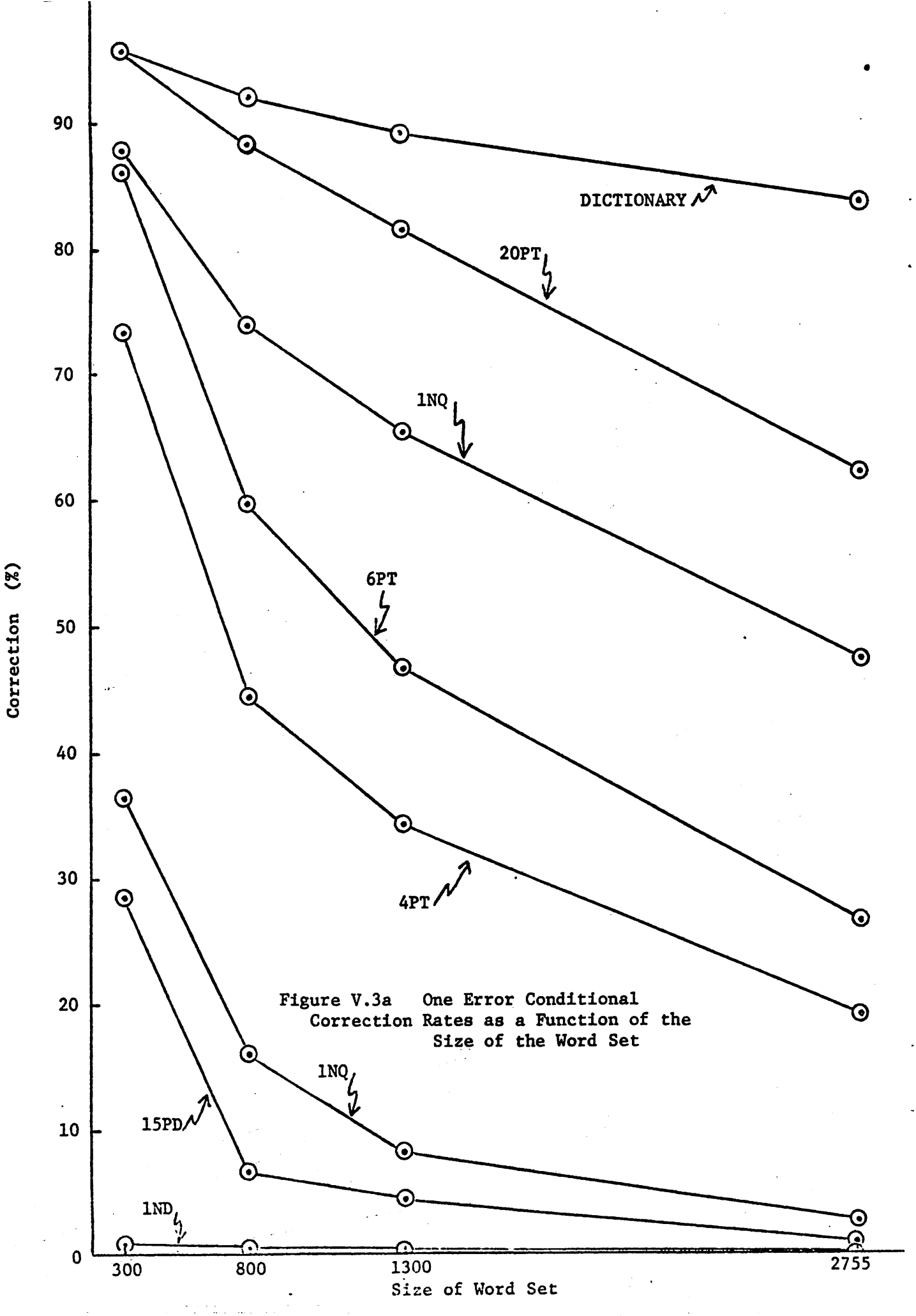
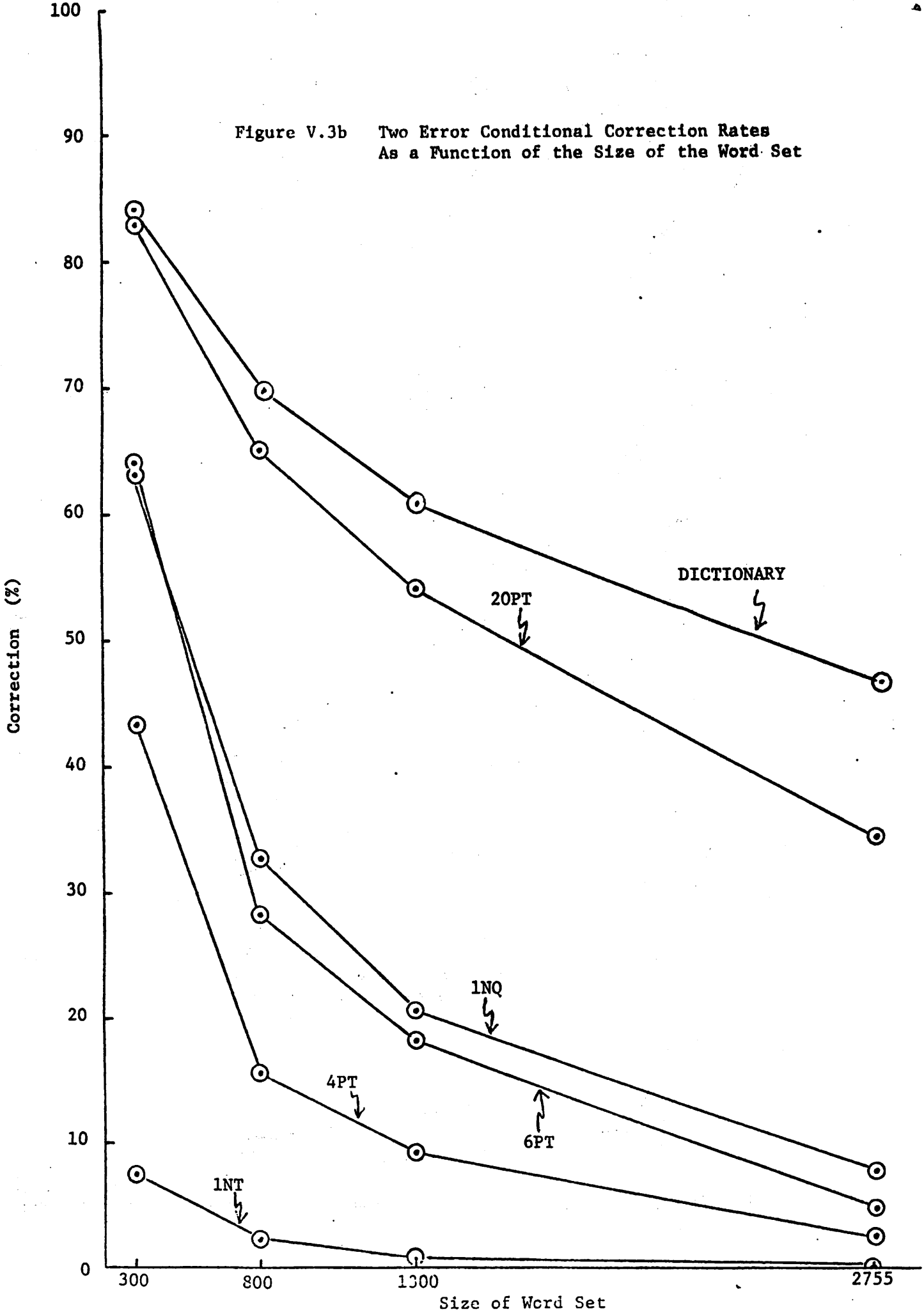


Figure V.3a One Error Conditional Correction Rates as a Function of the Size of the Word Set

Figure V.3b Two Error Conditional Correction Rates As a Function of the Size of the Word Set



The dictionary algorithm showed a sizable improvement over the 20PT only in the case of the larger word sets. A more complete comparison of the algorithms will be carried out in later sections of this paper.

Experiment #3. Analysis of Uncorrectable Errors

A deeper analysis of the detectable but uncorrectable errors was performed. In this experiment correction was attempted on a total of 45,000 one-error words using all positional trigrams. 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 2 or 3 choices, as is summarized in the left-hand portion of Table V.1.

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 some of these cases are eliminated by attempting correction in both positions and 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 V.1. In this portion of the table i/j denotes that there are i alternative characters admissible in one position and j characters for a second position. Once again the cases where there are a total of 2 or 3 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.

Table V.1 - Analysis of uncorrectable errors

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

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	.999
800	4771	3228	226	36	3454	906	240	135	1146	.964
1300	8332	5218	765	214	5983	1158	519	448	1677	.912
2755	16139	7551	2540	1753	10091	1283	951	2061	2234	.764

These results are extremely important since they point out that an attempt at correction is feasible in most of the uncorrectable cases, due to the small number of feasible choices. In Experiment #4, a forced error rate was computed under the assumption that when two choices exist, .8 of the choices could be resolved favorably, and in the case of n choices, $(.8)^{n-1}$ could be resolved favorably. Justification for this high forced guess rate is given in the discussion at the end of the paper.

Experiment #4. Error and Reject Rates

Up to this point we have examined the rates 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,... 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 1-error word, $k > 1$, and then improperly corrected; similarly, a j-error word might be mistaken for a 2-error word, $j > 2$, and improperly corrected. Expected word error and reject rates of the contextual postprocessor can be computed in the following manner:

let ud_i = fraction of undetectable i-error words, $i = 1, 2, 3$;

$d_i = 1 - ud_i$ = fraction of detectable i-error words

uc_i = fraction of uncorrectable i-error words, $i = 1, 2, 3$;

$m_{k\ell}$ = fraction of k-error words corrected improperly as an ℓ-error word, $k > \ell$.

r = assumed classifier character error rate;

f_i = probability of i-error word being output from the classifier.

The probability that the classifier outputs a 6-letter word with i errors can be computed as a function of r by

$$f_i = \binom{6}{i} r^i (1 - r)^{6-i}.$$

Assuming that no more than 3 errors occur (for the character error rates considered this is not unreasonable), the final word error rate E is then

$$E = ud_1 f_1 + (ud_2 + m_{21}) f_2 + (ud_3 + m_{31} + m_{32}) f_3$$

and the final word reject rate R is

$$R = \sum_{i=1}^{3} uc_i f_i.$$

Table V.2 gives the actual percentages of errors that are corrected, rejected, and remain as errors after processing with 20PT and with the dicitonary algorithm. Also computer 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.

Figures V.4a-c show the error, reject, and correction rates for all positional digrams and assumed classifier error rates r = .01, .05, and .10, producing word error rates of .059, .265, and .469 respectively. Shown on the same plots are these initial word error rates W and the forced error rate. The forced error rate is computed assuming a zero reject rate and an 80% forced guess rate, as previously discussed. Plots are shown as a function of the size of the word set. Figures V.5a-c show the same information but for the set 20PT.

Table V.2 - 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

Figure V.4a Error, Reject and Correction Rates
Using all Positional Binary Digrams

r = .01; W = 5.85

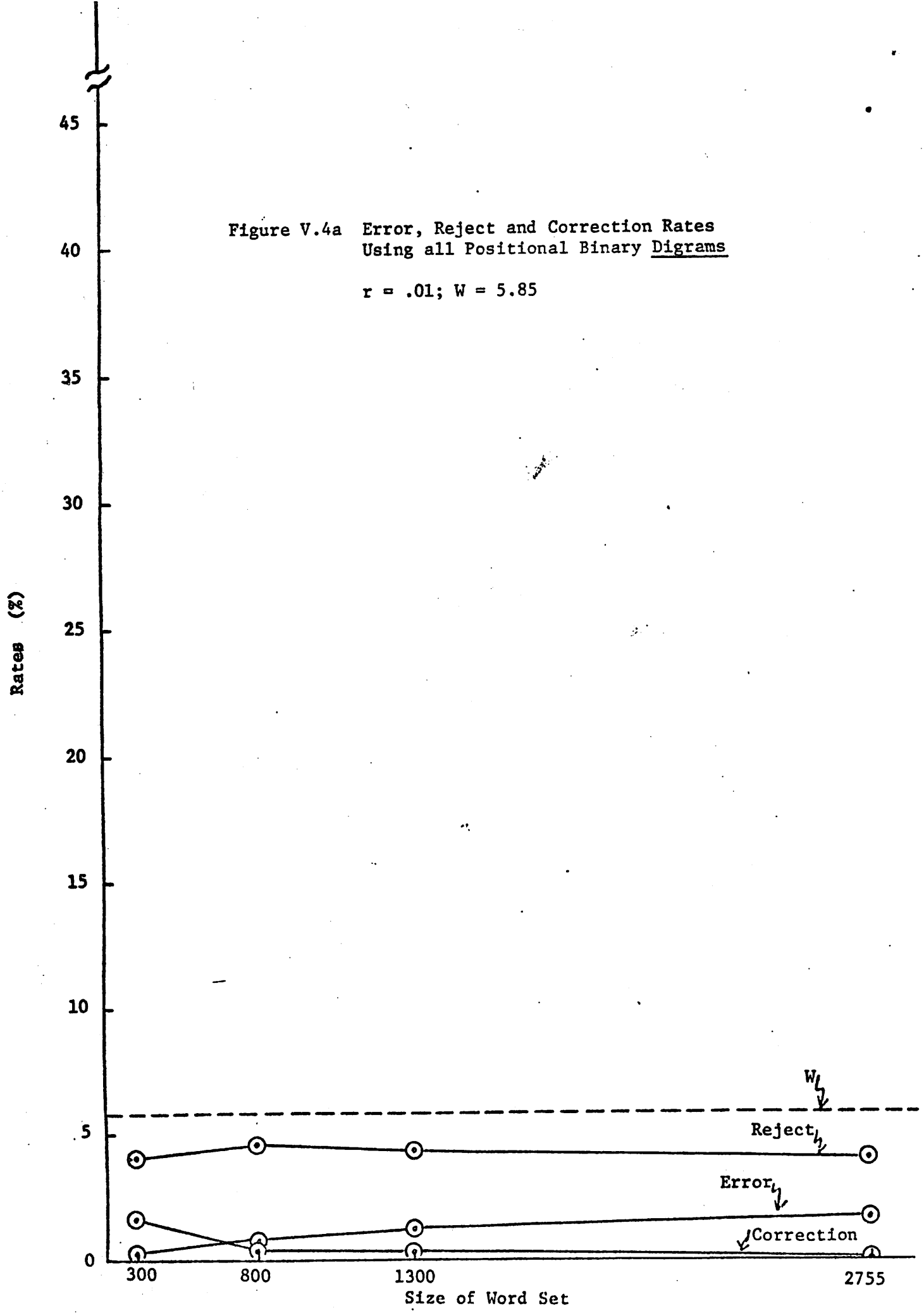
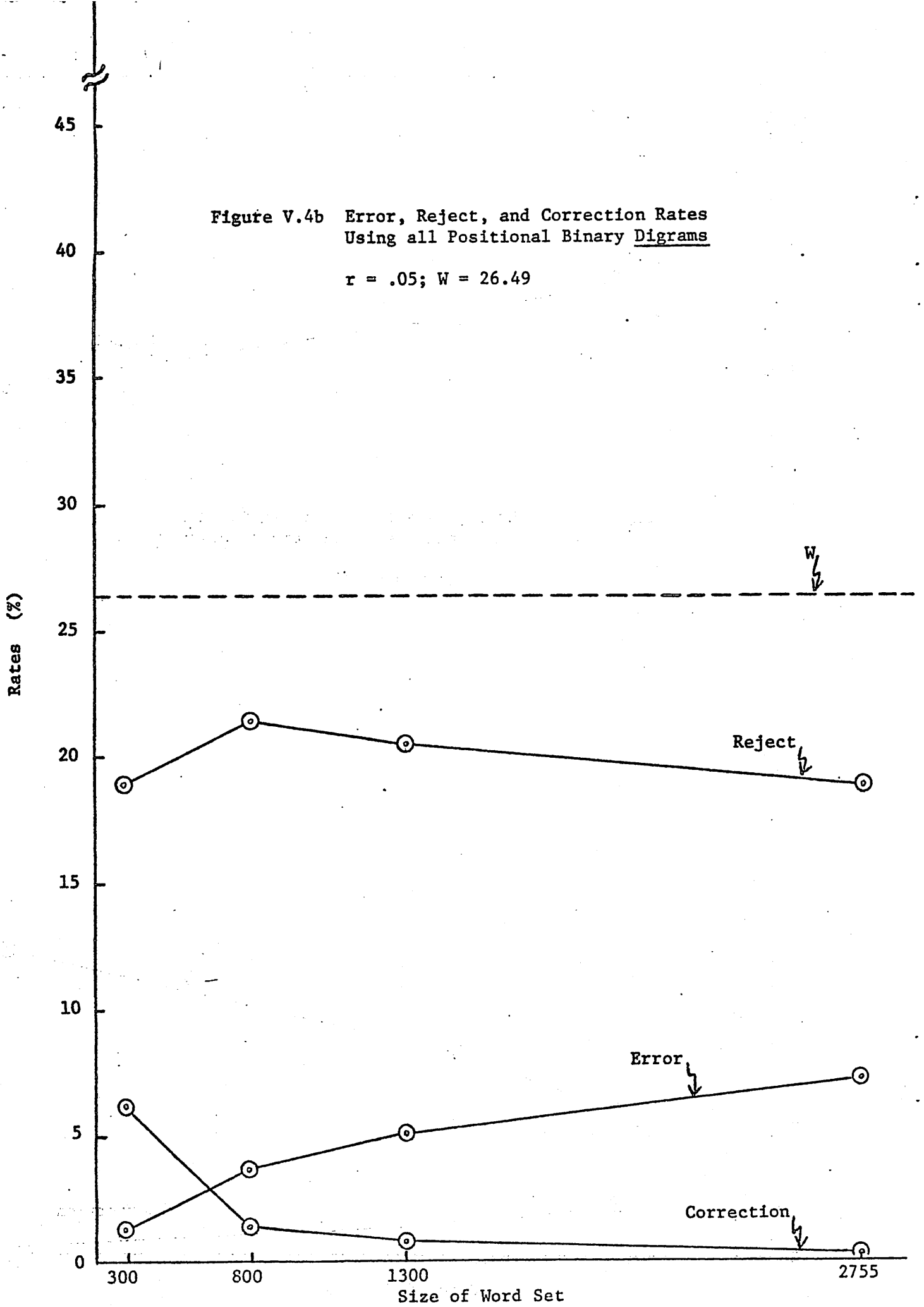


Figure V.4b Error, Reject, and Correction Rates
Using all Positional Binary Digrams

r = .05; W = 26.49



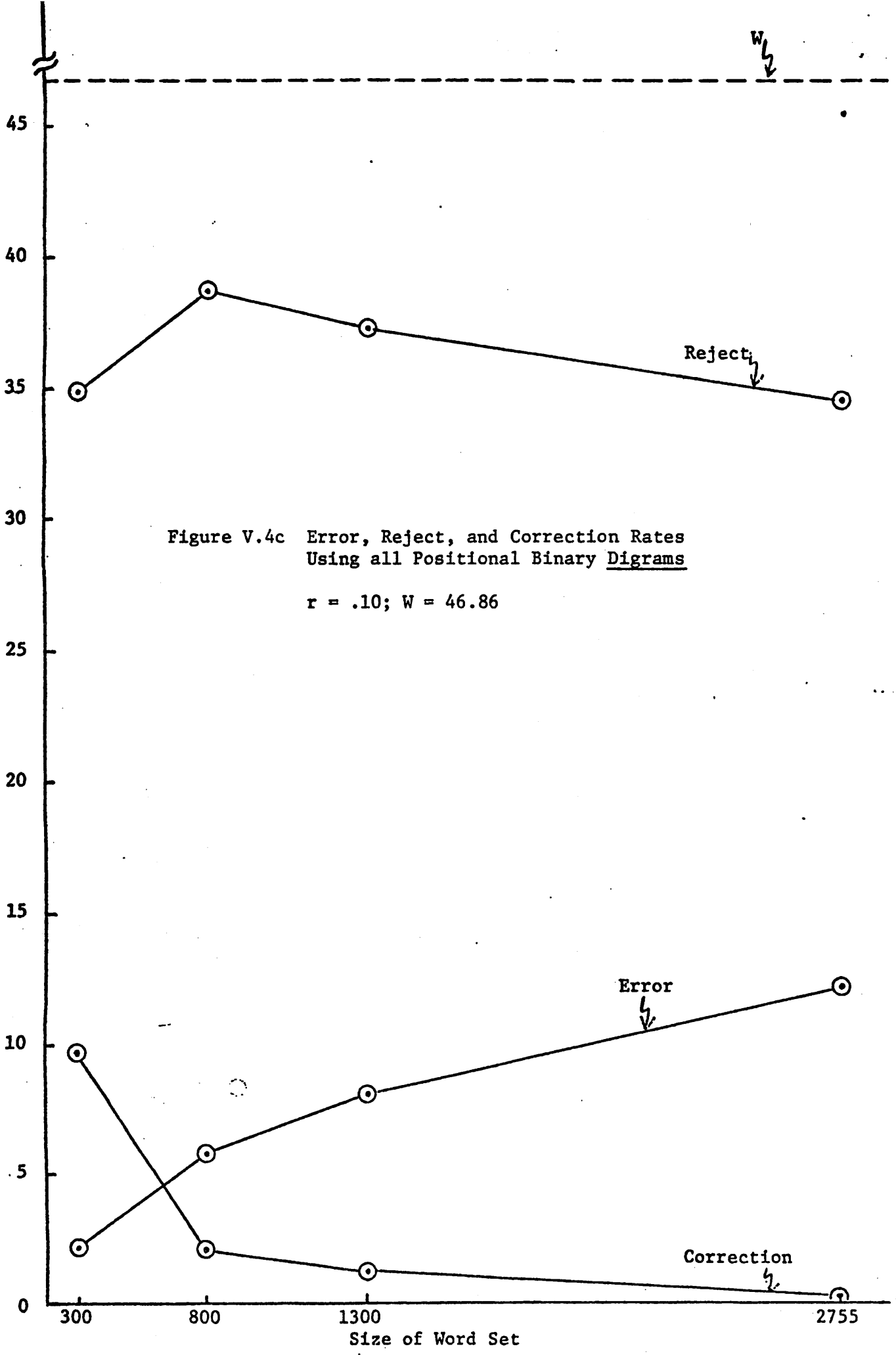


Figure V.4c Error, Reject, and Correction Rates
Using all Positional Binary Digrams

r = .10; W = 46.86

Figure V.5a Error, Reject, and Correction Rates
Using all Positional Binary Trigrams

r = .01; W = 5.85

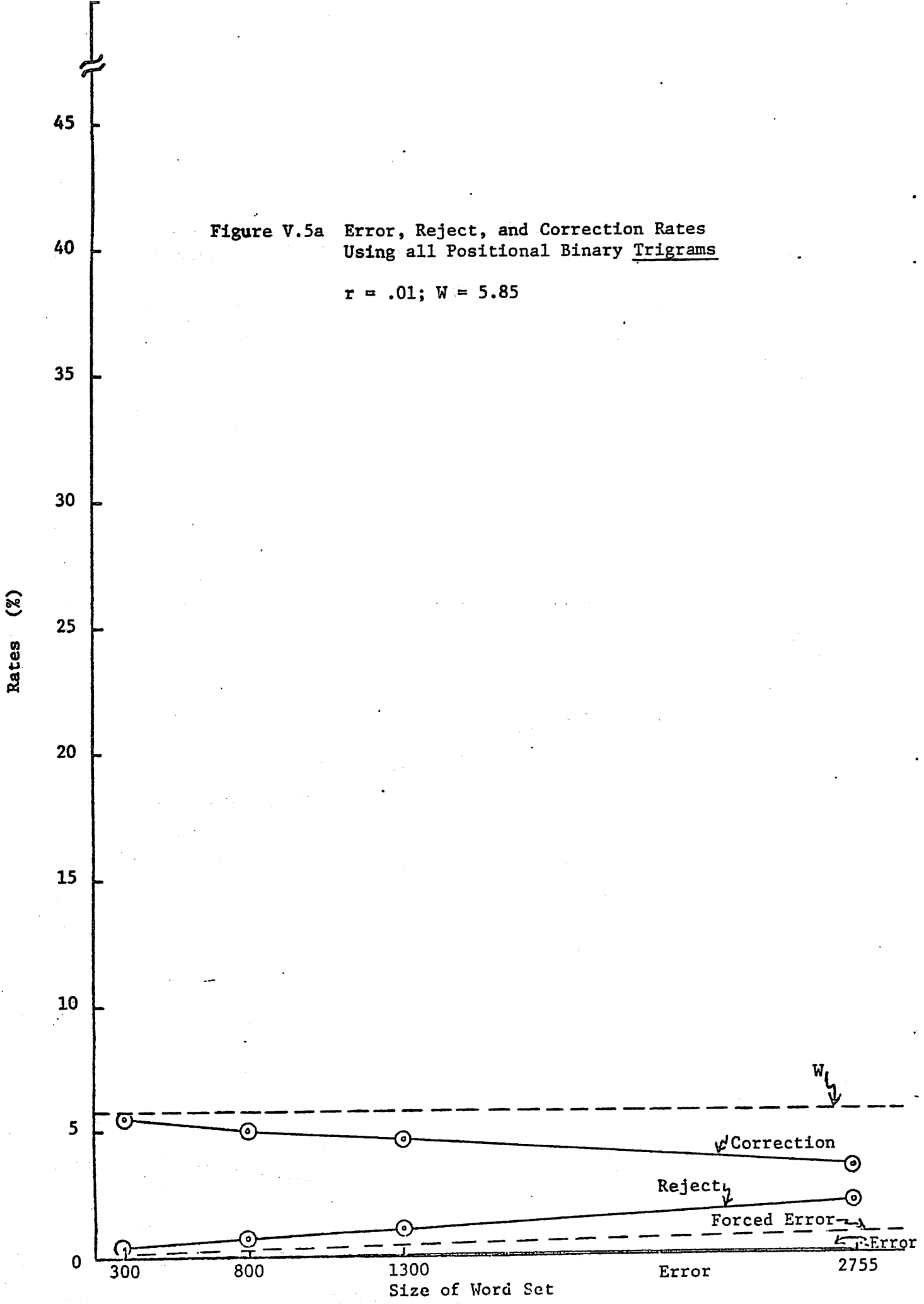
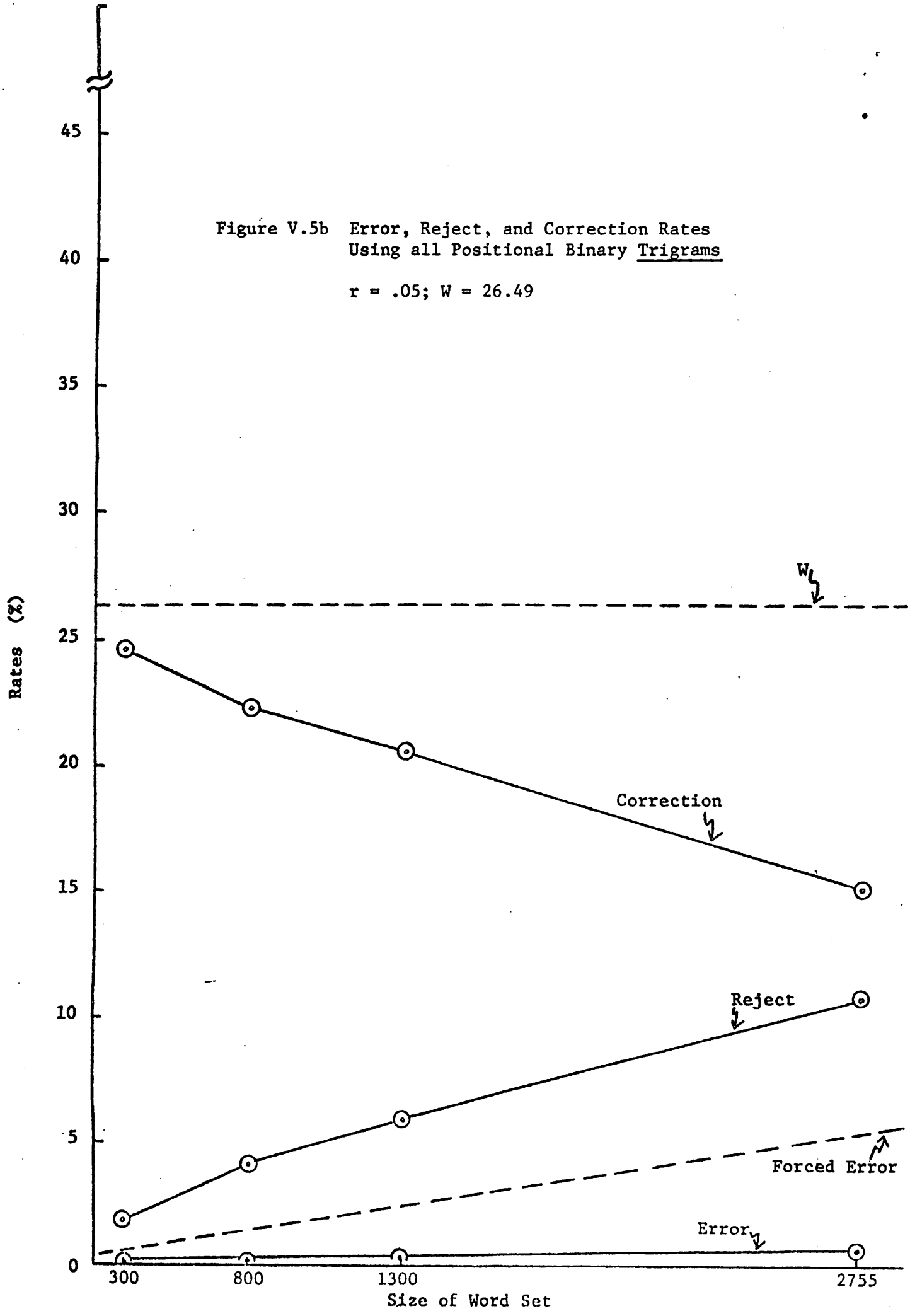


Figure V.5b Error, Reject, and Correction Rates
Using all Positional Binary Trigrams

r = .05; W = 26.49



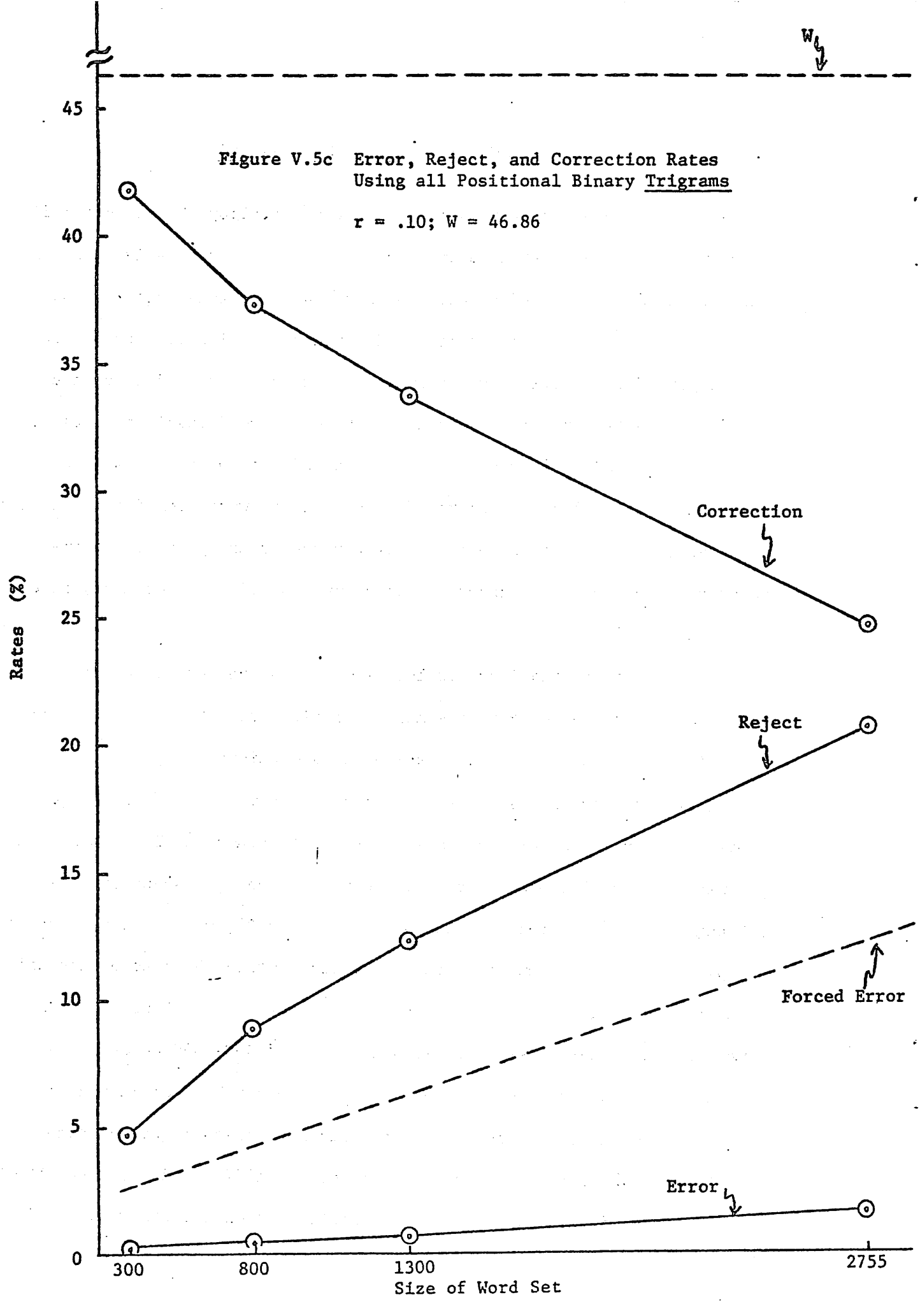


Figure V.5c Error, Reject, and Correction Rates
Using all Positional Binary Trigrams

r = .10; W = 46.86

VI. STORAGE AND COMPUTATIONAL ANALYSIS

Before one can properly compare the contextual algorithms with the dictionary method, one must take into consideration their relative cost, namely the amount of storage and computation necessary for these algorithms.

The space required to store the dictionary is directly proportional to the length of the dictionary. Assuming 5 bits are required to code each of the 26 alphabetic characters, then each 6-letter word requires 30 bits; the required storage is 30m where m is the size of the set of words. Table VI.1 gives the number of bits required for the sets of words considered in this paper as well as the storage necessary for the other types of n-grams. The space required for the n-grams is not a function of the dictionary size and remains constant.

The dictionary requires far less space than the set of 20 positional trigrams; the subset of trigrams must be limited to 4 before the required space is less than the size of the 2755-word dictionary.

There is a strong reason to ignore the disadvantage of the larger storage requirements for the complete set of trigrams, 20PT. Namely, the plummeting cost of storage: at this point in time high speed storage can be bought at 1¢/bit; thus 350,000 bits cost only $3,500, and the price is dropping sharply. We expect that in the near future the cost of the difference in storage will not be significant. Actually, 350,000 bits is not a very large amount of storage, less than 10K of CDC-3600 words which are 48 bits long. One should note that previous research utilizing the actual probabilities of non-positional letter triplets required large amounts of storage (assuming only 16-bit words, 17576 words, or over 280,000 bits, are needed), yet was relatively ineffective in the error correction process

Table VI.1 - Comparison of storage requirements

size of word set	dictionary	n-grams	
m	# bits required		# bits required
300	9,000	1 non-positional digram (1ND)	729
		15 positional digrams (15 PD)	10,240
800	24,000	1 non-positional trigram (1NT)	19,683
		4 positional trigrams (4PT)	70,304
1300	39,000	6 positional trigrams (6PT)	105,456
		20 positional trigrams (20PT)	351,520
2755	82,650	1 non-positional quadgram (1NQ)	531,444

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 $\log_2 n$ ($\log_2 300 - 8.2$, $\log_2 2755 = 11.8$) dictionary accesses (this process must be time sequential) to determine whether or not a word is in the dictionary. On the other hand the full set of trigrams may require up to 20 bit accesses (which may be done in parallel), and proportionally 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 1-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 2-error correction process. Table VI.2 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 10 trigrams which involve a single character position, one must access ten 26-bit rows (or columns), intersect

Table VI.2 - 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

them, and then examine each of the 26 bits 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 out the orders of magnitude difference in speed.

VII. DISCUSSION

The algorithms that have been presented appear to be extremely effective in both error detection and correction. Although the dictionary algorithm is slightly more effective than the set of all positional trigrams (2OPT) for error detection, the detectability rates of the 2OPT are so high already that the difference is negligible. Depending on the size of the word set, detectability rates for 1-error words vary from 98.6% to 99.8% 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 2OPT and 1NQ. One non-positional digram (a single 27 × 27 binary matrix) is not nearly as effective (although one must keep in mind the small number of bits of information required) but all other types detect at least 95% of errors for small word sets. It seems that little of the information necessary for error detection resides in the specific value of the non-zero probability of letter pairs and triplets since very 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 non-zero 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 participate in the detection and correction process.

The dictionary algorithm is somewhat more effective than binary positional trigrams in terms of error and reject rates. For the smaller word data sets, the difference in performance is not very signigicant. For the larger word sets, most of the difference involves a lower correction rate and higher rejection rate on the part of the binary trigrams. However, the remaining error rate is still very small. The positional binary trigrams are certainly far more effective than any previous use of the probabilities of digrams 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 look-up 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.

The contextual algorithm employing the entire set of trigrams yielded 1-error correction rates varying from 61% to 95% of the errors as a function of the size of the word set. The correction rate for 2-error words drops to the 34$ to 83% range. 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 procedures under discussion should be able to transform a poor classification system into a fairly effective one. The 2755 six-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 between 5000 and 10,000 words could easily be sued 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%, then 6-letter words would have a word error rate of 47% or 74% respectively. These systems would be virtually unusable. The high speed contextual system that has been described will reduce the 47% error rate to a .38% error rate and a 9% reject rate; correspondingly the initial 74% error rate results in 1.29% errors and 22% rejects. For lower initial character error rates the system performs better, since a larger proportion of the errors are 1-error words which are wasier to correct than multiple-error words.

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, in 8-letter words the total number of di-grams and trigrams available is 28 and 56, respectively. The implication is that if one invests a greater amount of storage, results superior to those presented in this paper should be obtained.

The assumption of the errors being random substitutions should be discussed. In practice, if a Bayesian classifier is used, there is a bias towards the letters that are a priori more likely. Thus, there would be a tendency for more of the error substitutions to be from the more likely

letters. This would cause some deterioration in detectability rates since the local density of the n-grams is greater about those letters. However, this deterioration can be offset by using statistics on the confusion matrix from the classifier, a process that was ignored in this paper.

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 V.1 that when an error is uncorrectable a very high percentage of these cases involve only 2 or 3 choices. If the information concerning the possible substitutions is available (a 26 × 26 binary matrix might be sufficient), a large proportion of the currently uncorrectable errors might be resolved.

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, as shown in Figure VII.1. 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 approaching detectability rates might be feasible. Further research in this direction is being carried out. 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. This final recognition system is shown in

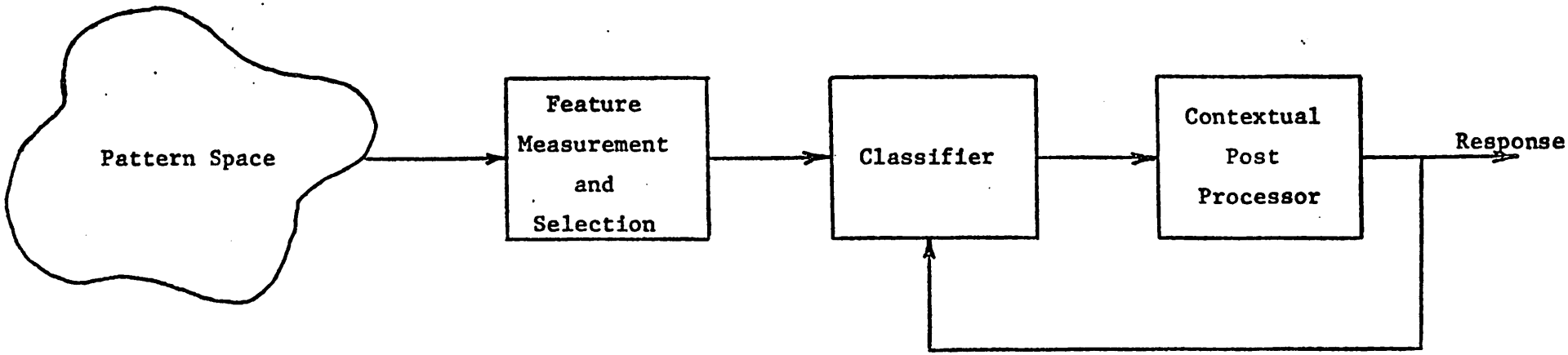


Figure VII.1 Recognition System Utilizing Feedback of Contextual Information: feedback from contextual post processor to classifier indicates the reduced set of alternative characters. Reclassification is performed on the reduced set.

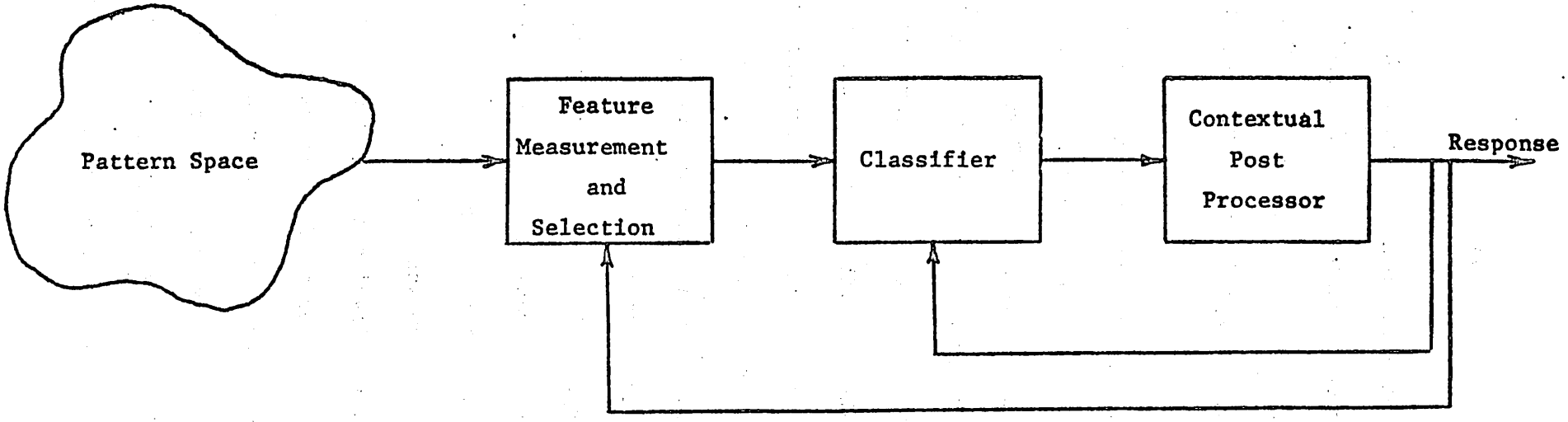


Figure VII.2 Complete Recognition System Utilizing Two Modes of Feedback: feedback from contextual post processor to classifier reduces the number of alternatives when reclassification is indicated; feedback from contextual post processor to feature measurement and selection indicates additional specific measurements to be made.

Figure VII.2, containing both feedback loops. Not shown in this diagram is any of the computation required in the feedback loops. However, the simplicity of the postprocessor we have described might allow this computation to be reasonably straightforward. Research in this direction has been started [30].

A contextual system of the type described might be the key to making handwriting recognition feasible. Relatively poor classifiers (10% to 20% character error rates which yield over 50% word error rates) potentially might have 95% of their errors corrected. The speed of the contextual correction process would enable the system to carry this out on-line during the classification. The handwriting segmentation problem would make alternative attempts of correction for different length words.

Another extension of the techniques that have been discussed is the correction of errors in feature measurements. 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. If the set of features of any general pattern are partitioned into a set of ℓ subsets, then positional binary n-gram statistics could be collected. For example, a positional trigram would involve three of the ℓ subsets and describe the allowable values that features in these subsets can take on. 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.

VIII. CONCLUSION

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, such as handwriting recognition, feasible. Conversely, one might greatly reduce the complexity of pattern classifier if it is followed by a contextual postprocessor with the reliability that has been demonstrated herein.

Acknowledgments

The authors wish to thank Edward Fisher and Stephen 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," Proceedings of the IEEE, Vol. 56, May 1968, pp. 836-862.

[2] M. D. Levine, "Feature Extraction: A Survey," Proceedings of the IEEE, Vol. 57, August 1969, pp. 1391-1407.

[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, March 1959, pp. 17-24.

[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. Control, Vol. 3, 1960, pp. 60-67.

[7] E. J. Sitar, "Machine recognition of cursive script: The use of context for error detection and correction," Bell Telephone Laboratories, Inc., 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. Control, Vol. 5, 1962, pp. 368-384.

[10] R. B. Thomas, M. Kassler, and G. Woolley, "Advanced character recognition techniques study," Tech. Rept. 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. ACM 19th Nat. Conf., Aug. 1964, pp. D2.4-1D2.4-13.

[12] A. W. Edwards and R. L. Chambers, "Can a priori probabilities help in character recognition?" J. ACM, Vol. 2, Oct. 1964, pp. 465-470.

[13] 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.

[14] J. Raviv, "Decision making in Markov chains applied to the problem of pattern recognition," IEEE Trans. Inform. Theory, Vol. IT-13, Oct. 1967, pp. 536-551.

[15] J. T. Chu, "Error Bounds for a Contextual Recognition Procedure," IEEE Transactions on Computers, Vol. C-20, Oct. 1971, pp. 1203-1207.

[16] A. J. Szanser, "Error-Correcting Methods in Natural Language Processing," Information Processing 68, North Holland Publishing Co., Amsterdam. 1968, Vol. II, pp. 1412-1416.

[17] J. R. Welch and K. G. Salter, "A Context Algorithm for Pattern Recognition and Image Interpretation," IEEE Transactions on Systems Science and Cybernetics, Vol. SMC-1, January 1971.

[18] E. M. Riseman and R. W. Ehrich, "Contextual Word Recognition Using Binary Digrams," IEEE Transactions on Computers, Vol. C-20, April 1971, pp. 397-403.

[19] 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, pp. 1139-1149.

[20] H. L. Morgan, "Spelling Correction in System Programming," Comm. ACM, Feb. 1970, pp. 90-94.

[21] R. C. Heinselman, "Computerized Detection and Correction of Spelling Errors in Fortran Programs," MS Thesis, Computer and Information Sciences Dept., Univ. of Minnesota, 1972.

[22] H. T. Glantz, "On the Recognition of Information with a Digital Computer," J. ACM, April 1957, pp. 178-188.

[23] L. Davidson, "Retrieval of Misspelled Names in an Airlines Passenger Record System," Comm. ACM, March 1962, pp. 169-171.

[24] J. J. Giangardella, "Spelling Correction by Vector Representation Using a Digital Computer," IEEE Transaction on Engineering Writing and Speech, Dec. 1967, pp. 57-62.

[25] C. N. Alberga, "String Similarity and Misspellings," Comm. ACM, Vol. 10, No. 5, May 1967, pp. 302-313.

[26] F. Damerau, "A Technique for Computer Detection and Correction of Spelling Errors," Comm. ACM, Vol. 7, No. 3, March 1964, pp. 171-176.

[27] V. Neisser and P. Weene, "A Note on Human Recognition of Hand-Printed Characters," Inf. & Control, Vol. 2, 1960, pp. 191-196.

[28] C. Shannon, "Prediction and Entropy of Printed English," Bell System Technical Journal, Jan. 1951, pp. 51-54.

[29] R. W. Ehrich and E. M. Riseman, "Contextual Error Detection," COINS Tech. Report 70C-4, University of Massachusetts, Amherst, May 1971.

[30] F. W. Stodda and A. R. Hanson, "Pattern Recognition using Contour Analysis," Dept. of Computer and Information Science, Tech. Report 72-5, Univ. of Minnesota, Minneapolis, May 1972.

[31] Edward L. Thorndike and Irving Lorge, The Teacher's Word Book of 30,000 Words, Bureau of Publications, Teachers College, Columbia University, New York 1944 (4th ed. 1963).