Entropy Reduction of English Text Using Variable Length Grouping

Vincent Norman Ast Jr.
Portland State University

AN ABSTRACT OF THE THESIS OF Vincent Norman Ast Jr. for the Master of Science in Applied Science presented July 14, 1972.

Title: Entropy Reduction of English Text Using Variable Length Grouping

APPROVED BY MEMBERS OF THE THESIS COMMITTEE:

[redacted]

Robert Rempfer, Chairman

[redacted]

Pah I. Chen

[redacted]

Jack C. Riley

[redacted]

Selmo Tauber

It is known that the entropy of English text can be reduced by arranging the text into groups of two or more letters each. The higher the order of the grouping the greater is the entropy reduction. Using this principle in a computer text compressing system brings about difficulties, however, because the number of entries required in the translation table increases exponentially with group size. This experiment examined the possibility of using a translation table containing only

selected entries of all group sizes with the expectation of obtaining a substantial entropy reduction with a relatively small table.

An expression was derived that showed that the groups which should be included in the table are not necessarily those that occur frequently but rather occur more frequently than would be expected due to random occurrence. This was complicated by the fact that any grouping affects the frequency of occurrence of many other related groups. An algorithm was developed in which the table originally starts with the regular 26 letters of the alphabet and the space. Entries, which consist of letter groups, complete words, and word groups, are then added one by one based on the selection criterion. After each entry is added adjustments are made to account for the interaction of the groups. This algorithm was programmed on a computer and was run using a text sample of about 7000 words.

The results showed that the entropy could easily be reduced down to 3 bits per letter with a table of less than 200 entries. With about 500 entries the entropy could be reduced to about 2.5 bits per letter.

About 60% of the table was composed of letter groups, 42% of single words and 8% of word groups and indicated that the extra complications involved in handling word groups may not be worthwhile.

A visual examination of the table showed that many entries were very much oriented to the particular sample. This may or may not be desirable depending on the intended use of the translating system.

ENTROPY REDUCTION OF ENGLISH TEXT USING VARIABLE LENGTH GROUPING

by

VINCENT NORMAN AST JR

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
APPLIED SCIENCE

Portland State University
1972

TO THE OFFICE OF GRADUATE STUDIES AND RESEARCH:

The members of the Committee approve the thesis of Vincent Norman Ast Jr. presented July 14, 1972.

[redacted]

Robert Rempfer, Chairman

[redacted]

Pah I. Chen

[redacted]

Jack C. Riley

[redacted]

Selmo Tauber

APPROVED:

[redacted]

Nan-Teh Hsu, Head, Department of Applied Science

[redacted]

David ~~F~~. Clark, Dean of Graduate Studies and Research

July 28, 1972

ACKNOWLEDGMENTS

I would like to thank the following people who helped make this thesis possible.

First, Dr. Robert Rempfer, who was my thesis advisor. Some of the important ideas of this project originated as a result of our many interesting conversations.

Dr. Bill Wilson and Mike Arts of the University of Portland and Bob Coffin of Portland Public Schools were very generous in allowing me the many hours of time on their Honeywell computers.

LeRoy Younglove wrote some of the subroutines used in the computer programs.

Finally, my wife, Nancy, typed the thesis and offered many suggestions as to its wording and layout. Most of all, her patience and encouragement during these past years were very much responsible for my successful venture through graduate school.

TABLE OF CONTENTS

PAGE

ACKNOWLEDGMENTS iii

LIST OF TABLES. vii

LIST OF FIGURES viii

CHAPTER

I INTRODUCTION. 1

Conventional Storage Techniques 1

Storage and Entropy 2

Reduction of Storage
Entropy Considerations

Variable Length Grouping. 6

II DEVELOPMENT OF THE ALGORITHM. 8

Calculation of Change of Entropy. 8

Total Entropy
Forming a Group
The Change in Entropy
Generalization

The Grouping Criterion. 11

Selecting an Entry
Interaction of Entries

The Basic Algorithm 13

Tables
The Process

CHAPTER		PAGE
	Further Considerations.	14
	Simplification of ΔH Calculation	
	Method of Grouping	
	Modification of the Algorithm	
	When to Stop the Process	
III	IMPLEMENTATION CONSIDERATIONS	23
	The Machine and the Program	23
	Text Sample	24
	Initial Preparation	
	Input Editing	
	Table Structure	
	The Alphabet Table.	26
	Structure	
	Binary Search	
	Table Maintenance	
	Group Table	29
	Preparation of the Group Tape	29
	Requirements	
	Method of Preparation	
	Adjustment of Tables.	32
	Adding an Entry to the Alphabet	
	Removing an Entry from the Alphabet	
IV	RESULTS AND DISCUSSION.	35
	The Tables.	35
	Text Sample Table	
	The Group Table	
	Entropy Reduction	36
	Run History	
	Alphabet Summary	
	Effect of the Double Blanks	
	Interpretation of the Curve	
	Equation of the Curve	

CHAPTER PAGE

Distribution Curves 41

Group and Word Distribution
Frequency Distribution

V CONCLUSIONS AND RECOMMENDATIONS 45

Conclusions 45

Entropy Reduction
Table Content

Recommendations 46

Changes in Procedure
The Group Table and Sample Sensitivity

REFERENCES CITED. 48

APPENDIX. 49

LIST OF TABLES

TABLE PAGE

I Summary of Entropy Data Taken at Checkpoints During the Computer Run. 37

II Letter and Group Distribution Taken at Checkpoints During the Computer Run 42

III Run History 50

LIST OF FIGURES

FIGURE PAGE

1 Flow Chart of the Basic Algorithm Used for Building the Alphabet Table. 15

2 Flow Chart of the Modified Algorithm Used in This Experiment. 21

3 Computer Memory Utilization Used in the Experiment. 23

4 Example of Text Storage Showing the Use of Item Marks Before and After the Group, TION, Was Formed. 26

5 Sample Taken from the Alphabet Table. 27

6 Example from the Text Table of the Group, PRESENT, Being Formed Showing All the Related Groups Which Are Affected. 33

7 Graph of Entropy per Letter Versus Alphabet Size. 39

8 Lower Curve of Figure 7 Drawn on Logarithmic Scale. 40

9 Graph of Letter and Word Group Distribution as a Function of Alphabet Size 42

10 Graph of Group Frequency Distribution at Selected Checkpoints During the Computer Run 44

CHAPTER I

INTRODUCTION

In recent years there has been considerable use of the computer in applications which require the storage of large amounts of alphabetic data. For example, with computer assisted instruction complete courses are programmed and stored on a computer system which then feeds the information in bite-size pieces to students sitting before a typewriter keyboard or a cathode ray tube (1). In another example, abstracts of scientific publications can be stored in a computer system along with keywords which allow a user to research hundreds of journals by merely sitting at a console (2).

One of the considerations in the design of such systems is economically providing enough machine storage space to hold the large amount of data required. Clearly, any method which would reduce the amount of data to be stored without degrading the performance of the system would be advantageous. One such technique of reduction is the topic of this paper.

I. CONVENTIONAL STORAGE TECHNIQUES

Present day business computers, such as the IBM 360 series or the Honeywell 200 series, have a memory structure in which magnetic cores are arranged in small stacks, called bytes, of nine cores each, the whole memory commonly consisting of thousands to hundreds of thousands of such bytes. The reader is referred to the individual manuals for specific details (3,4).

Each core is always magnetized to saturation in one of two possible states. Because of this binary property each core is able to store one binary digit, or one "bit" of information.

In storing alphabetic data the Honeywell 200 uses only 6 of the bits in each byte. Since each bit has 2 possible states, the whole byte has 2^6 or 64 possible states. Each state corresponds to one of 64 characters making up the character set or alphabet for the machine. (IBM has provisions for using 8 bits of each byte giving a 256 character set, since 2^8 equals 256.) The character set usually consists of the letters A through Z, the numerals 0 through 9, the blank, and a series of special symbols such as the comma, period, ampersand, etc.

Because of the tremendous cost of core storage it is usually used only for storing programmed instructions for the machine and for storing only the data which is currently being processed. The bulk of the data is stored on mass storage devices such as disks or drums which typically have a capacity of millions to billions of bytes (5).

II. STORAGE AND ENTROPY

Reduction of Storage

One way to reduce the number of bits required to store data on a machine would be to simply adopt a smaller character set. For example, a set with only 32 characters would require only 5 bits per character. It is doubtful whether 32 characters would be sufficient, however, since it allows only 5 additional characters over the 26 letters and the space.

The older Teletype system uses such an arrangement by making the numerals and punctuation "upper case" and the letters "lower case" (6). The actual data transmission becomes somewhat greater ~~that~~ than 5 bits per character, however, since additional characters must be transmitted to control the shifting of cases.

Another example of storage reduction is employed on the IBM 1130 system (7). Rather than the byte the standard unit of storage is the "word" consisting of 16 bits. Using the IBM 8-bit code or the Honeywell 6-bit code only 2 characters can be stored per word. However, by using a 40 character alphabet and a modulo 40 arithmetic, 1130 users are able to store 3 characters per word which averages 5 1/3 bits per character.

Entropy Considerations

Equally Probable Events. If there are n possible outcomes of an event, all equally probable, the entropy, H, is defined by:

$$H = \log_2 n$$

or: $$H = -\log_2 p$$

where p equals 1/n and is the probability of the occurrence of any given outcome. The reader is referred to any standard text on information theory (8,9,10,11,12,13).

The unit of entropy is the "bit", the derivation of the term being similar to that for the magnetic cores. The entropy, or the information content, of an event is the number of YES/NO questions that must be answered in order to determine the outcome of the event. For example, the entropy of one member of a 64 character alphabet is:

$H = \log_2 64$

which is 6, meaning that it requires 6 YES/NO questions to be answered or 6 magnetic cores to be properly magnetized in order to determine the particular character.

Entropy figures with fractional bits can be considered as averages over several events. For example, the entropy of a 40 character alphabet is:

$H = \log_2 40$

which is approximately 5.322 bits per character. Three such characters would have an entropy of 15.97 bits or almost 16 bits. It can be seen that the IBM 1130 scheme mentioned earlier of packing 3 characters per 16-bit word is quite efficient.

The entropy figure of a group of alphabetic characters, therefore, provides a theoretical figure for the number of bits required for storage. Since any storage system need not, and usually can not, be 100 per-cent efficient, the actual storage required will usually be greater than that given by the entropy figure. It cannot be less, however; this is one of the principle theorems of communications theory.

If the alphabet would be restricted to just the letters A through Z and the space, the entropy would be:

$\log_2 27 = 4.75$ bits per character.

Unequally Probable Events. In the preceding discussion no mention was made that in a typical sample of characters such as from a passage of English text, the frequency or probability of occurrence is not the same for all members of the alphabet. The true entropy is defined by computing the entropy for each member and taking a weighted average:

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where p_i is the probability of occurrence of the ith member of an alphabet of n characters.

Determining the probability of occurrence of the members must be done by actual count of a typical sample. Results by Shannon (14) show that the entropy of 27-letter English text is about 4.03 bits per character. Using the Huffman code (13), 27-letter English has actually been coded with an average of 4.12 bits per character. A user of such a scheme would need only a 27 entry coding table and a simple translating program.

In addition to the unequal distribution of English letters there is also a Markov effect in that the probability of occurrence of any character depends on what character it follows. Shannon again shows that considering this property the entropy of 27-letter English is about 3.42 bits per character. A translating scheme would require the task of building a 27^2 or 729 entry coding table.

Further estimates by Shannon show that if the Markov analysis is expanded to include the effects of the previous 2 characters, the entropy is about 3.0. A translating scheme, however, would require a table with 27^3 or almost 20,000 entries. While such a table could be built it is now leaving the realm of simplicity in that it would require a large amount of computer core storage for operation.

Final estimates by Shannon show that ultimately, the entropy of 27-letter English may be reduced to less that 1.3 bits per character taking into account large strings of letters preceding each given letter.

The conclusion seems to be, then, that although the entropy of English text can be reduced from 4.75 bits down to 1.3 bits, a practical translating device using a Markov table could not reduce the storage requirements much below 3.4 bits per character otherwise the size of the table would be prohibitive.

III. VARIABLE LENGTH GROUPING

Preliminary studies by the author showed that a frequency table of two-letter groups had many entries which did not occur at all even with a moderate sample. For example: GX, ZN, QA, QB, etc. The effect was even more noticeable with three- and four-letter groups. In addition to this there were also many which occurred only a few times throughout the whole sample.

Because of this property I thought that perhaps it would be possible to build a two-letter translating table containing only the frequently used groups. If a group was encountered during the translation which was not in the table it could be handled on an individual character basis. The result might be a large reduction in the table size at the expense of only a slight increase in entropy.

Generalizing on this principle I thought it possible to include in this table the frequently used three-, four-, and even higher order letter groups and enjoy the advantages of even greater entropy reduction and yet have a translating table of reasonable size. Finally, there would be no need to limit it only to letter groups, but complete words and even word groups could be included. The entries in this table plus the original 27 letters could all be considered as members of a huge alphabet.

This thesis describes the process by which such a table or alphabet was built. The biggest problem was determining which groups should make up the table. What is meant by a "frequently used group"? Will the presence of a certain group in the table influence the effectiveness of another group?

Another objective is to determine the relation between entropy reduction and table size. How low will the entropy go? Is there an optimum table size?

Finally, as an additional point of interest, what proportion of the table will consist of letter groups, what proportion of words, and what proportion of word groups?

CHAPTER II

DEVELOPMENT OF THE ALGORITHM

I. CALCULATION OF CHANGE OF ENTROPY

Total Entropy

Starting with the definition of entropy given earlier, namely:

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

the entropy per character of an actual sample of text can be written:

$$H = -\sum_{i=1}^{N} \frac{x_i}{x_T} \log_2 \frac{x_i}{x_T}$$

where:

N = number of entries in the alphabet

x_i = number of occurrences of the ith letter

x_T = total number of characters in the sample.

The total entropy of the entire sample, H_T, is then H times x_T:

$$H_T = -\sum_{i=1}^{N} x_i \log_2 \frac{x_i}{x_T}$$

Expanding the logarithmic quantity:

$$H_T = -\sum_{i=1}^{N} x_i (\log_2 x_i - \log_2 x_T)$$

$$= \sum_{i=1}^{N} x_i \log_2 x_T - \sum_{i=1}^{N} x_i \log_2 x_i$$

But the $\log_2 x_T$ is a constant and can be pulled out of the summation. Also:

$$\sum_{i=1}^{N} x_i \text{ is simply } x_T.$$

Therefore:

$$H_T = x_T \log_2 x_T - \sum_{i=1}^{N} x_i \log_2 x_i$$

Or in expanded notation:

$$H_T = x_T \log_2 x_T - x_1 \log_2 x_1 - x_2 \log_2 x_2 - \ldots - x_N \log_2 x_N$$

<u>Forming a Group</u>

Suppose now that the first two letters of the alphabet are considered as a group, this group now being considered a new "letter" in the alphabet. The number of occurrences of this group in the sample will be called x_A. The following conditions now would be true:

1. Individual occurrences of the first letter of the alphabet now occur only $x_1 - x_A$ times.
2. Similarly the second letter occurs $x_2 - x_A$ times.
3. There are now only $x_T - x_A$ "letters" in the entire sample.

The new entropy, H'_T, is:

$$H'_T = (x_T - x_A) \log_2 (x_T - x_A) - (x_1 - x_A) \log_2 (x_1 - x_A) - (x_2 - x_A) \log_2 (x_2 - x_A) - \ldots - (x_N - x_A) \log_2 (x_N - x_A) - x_A \log_2 x_A$$

<u>The Change in Entropy</u>

The change in entropy is: $\Delta H = H'_T - H_T$.

$$\Delta H = x_T \log_2 (x_T - x_A) - x_A \log_2 (x_T - x_A) - x_T \log_2 x_T -$$
$$x_1 \log_2 (x_1 - x_A) + x_A \log_2 (x_1 - x_A) + x_1 \log_2 x_1 -$$
$$x_2 \log_2 (x_2 - x_A) + x_A \log_2 (x_2 - x_A) + x_2 \log_2 x_2 -$$
$$x_A \log_2 x_A$$

Notice that all terms beyond x_2 have cancelled. Combining terms:

$$\Delta H = x_T \left[\log_2 (x_T - x_A) - \log_2 x_T\right] -$$
$$x_1 \left[\log_2 (x_1 - x_A) - \log_2 x_1\right] -$$
$$x_2 \left[\log_2 (x_2 - x_A) - \log_2 x_2\right] +$$
$$x_A \left[\log_2 (x_1 - x_A) + \log_2 (x_2 - x_A) - \log_2 (x_T - x_A) - \log_2 . x_A\right]$$

$$\Delta H = x_T \log_2 \frac{x_T - x_A}{x_T} - x_1 \log_2 \frac{x_1 - x_A}{x_1} - x_2 \log_2 \frac{x_2 - x_A}{x_2} +$$
$$x_A \log_2 \frac{(x_1 - x_A)(x_2 - x_A)}{(x_T - x_A)\, x_A}$$

Because of the original choice in the order of subtraction a positive ΔH means that the total entropy will increase and a negative ΔH indicates a decrease.

Generalization

This process can be expanded to a grouping of the first n letters of the alphabet. (Where n is not to be confused with N, the total number of letters in the alphabet.) The following will be true:

1. The frequency of the new group is x_A.
2. The individual occurrences are $x_1 - x_A, \ldots, x_n - x_A$.
3. There will be $x_T - (n - 1)\, x_A$ "letters" in the sample.

Using the same development as before, it is found that:

$$\Delta H = x_T \log_2 \frac{x_T - (n-1)\,x_A}{x_T} - x_1 \log_2 \frac{x_1 - x_A}{x_1} -$$

$$x_2 \log_2 \frac{x_2 - x_A}{x_2} - \ldots - x_n \log_2 \frac{x_n - x_A}{x_A} +$$

$$x_A \log_2 \frac{(x_1 - x_A)\,(x_2 - x_A)\,\ldots\,(x_n - x_A)}{(x_T - x_A)^{n-1}\,x_A} \qquad (1)$$

Although this derivation assumed the grouping of the first n letters of the alphabet, the actual sequencing of the alphabet is arbitrary and so the derivation is valid for the grouping of any n letters. The derivation fails, however, when any letter in a group is repeated; this will be discussed later.

II. THE GROUPING CRITERION

Selecting an Entry

The purpose of the preceding derivation has been to develop a criterion to determine whether or not there would be an advantage in including a given letter group in a machine translation table. If the ΔH of a particular grouping is negative this means that if this group would be included in the translation table the number of bits required to code this group would be less than the number of bits required to code the individual members making up the group; the number of bits saved in the entire sample being ΔH. On the other hand if the ΔH is positive it would require more bits to code the group than to code the members; hence, such a grouping should not be included.

The importance of equation 1 is that with a given sample of text it is possible to predict the effect of any possible grouping before actually performing the grouping. The ideal procedure for building a table, then, would be to compute the ΔH of all possible groupings immediately discarding any with a positive ΔH. The choice for the first new entry into the alphabet might be the one with the most negative ΔH.

Interaction of Entries

The next choice of entry would seemingly be the group with the next lowest ΔH. Proceeding in this fashion one would simply select entries in increasing order of ΔH until the total entropy of the sample was sufficiently low or until the table reached a pre-determined size. Unfortunately this is not possible because the grouping process changes the frequency of occurrence of many of the other characters in the alphabet and therefore changes the ΔH of many other potential groupings. Four particular cases can be illustrated by examples:

1. If the new group is, say, TH and occurs b times then the frequency of occurrence of individual T's and H's is each reduced by b thereby affecting the ΔH of all other groups containing T or H.
2. If the new group is TH then the possible group, THE, which originally was a three letter word is now only a two letter word since the group, TH, is now itself a "letter". The ΔH of THE is thereby affected.
3. The original THE has as subgroups TH and HE. Binding TH together now eliminates any further consideration of possible HE binding. The only HE's remaining in the sample are those

which were not associated with a leading T (such as the SHE), so the ΔH of HE is affected.

4. In addition, all grouping causes a slight decrease in x_T, the total number of characters in the sample, which will have some effect on every ΔH.

Because of this interaction it is necessary to recompute all the ΔH's before selecting the next entry. Since the computation of all the ΔH's is a lengthy process, the time required to build the alphabet is greatly increased. Moreover, the only way the effects of case 3, above, can be determined is by actually scanning the text sample, again a lengthy process.

III. THE BASIC ALGORITHM

Tables

The process to be described requires a computer with a large memory in which can be stored three tables: the alphabet table, the text sample, and the group table; in addition to the program.

Alphabet Table. This table initially contains only the basic alphabet but will grow as new groups are added. The space allocation must, therefore, be large enough to contain the anticipated number of entries of the final alphabet. Each entry consists of the particular letter (or letter group) along with a count of the number of times it occurs in the text sample.

Text Sample. This table contains the complete sample of text being analyzed together with a mechanism for combining letters into groups.

Group Table. This table ideally contains all two-, three-, four-, and higher order letter groups, including complete words and word groups. Obviously, such a table would require an impossible amount of memory so some assumptions are made which limit the entries as will be described later. Associated with each entry is a frequency count as in the alphabet table.

The Process

The first step is to scan the group table computing the ΔH for each group and selecting the entry that is most negative. See flow chart in Figure 1.

The next step is to transfer the selected group from the group table to the alphabet table.

Finally the text sample is scanned and, at every occurrence of the selected group components, grouping marks are set. In addition, every letter and letter group associated with the selected group is examined and its count in the alphabet table or group table is decreased by one.

With the grouping accomplished and all tables adjusted the process can then be repeated.

IV. FURTHER CONSIDERATIONS

Simplification of ΔH Calculation

Need for Simplification. The previous derivation of ΔH, while correct, requires the time consuming calculation of several logarithms for each entry. In addition, the expression was lengthy and difficult to analyze. Furthermore, it was not valid for repeated occurrences of

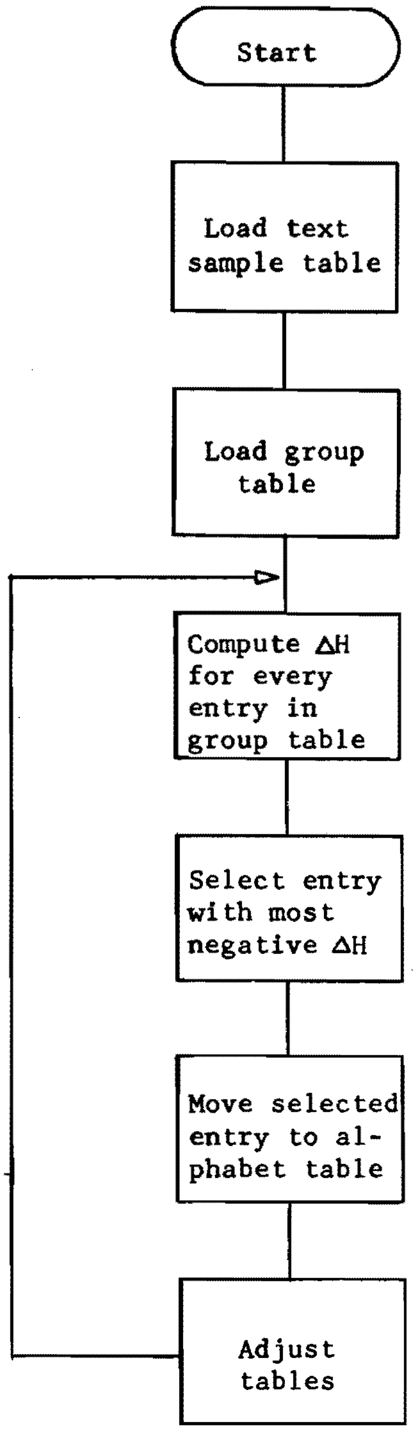

Start
Load text sample table
Load group table
Compute ΔH for every entry in group table
Select entry with most negative ΔH
Move selected entry to alphabet table
Adjust tables

Figure 1. Flow chart of the basic algorithm used for building the alphabet table.

the same letter. These problems can be lessened somewhat by making the assumption that the frequency of occurrence of the new group is much smaller than that of the components making up the group. The reasonableness of this assumption will be discussed later.

Derivation. Equation 1 can be rewritten:

$$\Delta H = x_T \log_2 \left[1 - \frac{(n-1)\, x_A}{x_T}\right] - x_1 \log_2 \left[1 - \frac{x_A}{x_1}\right] - x_2 \log_2 \left[1 - \frac{x_A}{x_2}\right] - \ . \ . \ . - x_n \log_2 \left[1 - \frac{x_A}{x_n}\right] + x_A \log_2 \frac{(x_1 - x_A)(x_2 - x_A) \ . \ . \ . \ (x_n - x_A)}{(x_T - x_A)^{n-1}\, x_A} \ .$$

Assume: $x_A \ll x_1$

Then: $\frac{x_A}{x_1} \ll 1$

Using the first term of the logarithmic expansion:

$$\log_e \left[1 - \frac{x_A}{x_1}\right] \doteq -\frac{x_A}{x_1}$$

Note the natural base. Changing to base 2:

$$\log_2 \left[1 - \frac{x_A}{x_1}\right] \doteq -\frac{x_A}{x_1} \log_2 e$$

Note also that $x_1 - x_A \doteq x_1$. Making a similar assumption regarding $x_2, \ . \ . \ . \ , x_n$ (and therefore x_T):

$$\Delta H \doteq (\log_2 e)\left[- x_T \frac{(n-1)\,x_A}{x_T} + x_1 \frac{x_A}{x_1} + x_2 \frac{x_A}{x_2} + \ldots + x_n \frac{x_A}{x_n}\right] +$$

$$x_A \log_2 \frac{x_1\, x_2 \ldots x_n}{x_T^{n-1}\; x_A} \tag{2}$$

The entire expression inside the brackets reduces to, simply, x_A.

$$\Delta H \doteq x_A \log_2 e + x_A \log_2 \frac{x_1\, x_2 \ldots x_n}{x_T^{n-1}\; x_A}$$

Dividing each x term by x_T:

$$\Delta H \doteq x_A \log_2 e \frac{(x_1/x_T)\,(x_2/x_T) \ldots (x_n/x_T)}{(x_A/x_T)} \tag{3}$$

But x_1/x_T is simply the probability of occurrence of x_1. Similarly for $x_2 \ldots x_n$, and x_A. Therefore:

$$\Delta H \doteq x_A \log_2 e \frac{p_1\, p_2 \ldots p_n}{p_{12 \ldots n}} \tag{4}$$

Discussion. Equation 4 is interesting in that the numerator of the fraction is simply the probability that the group would occur at random considering the probability of occurrence of its components. The denominator is the actual probability of occurrence.

Notice that there is only one logarithm. With over 6000 items in the group table and nearly 600 expected cycles of operation there will be nearly four million ΔH calculations. By using the simplified formula at least three logarithms per calculation or over 12 million logarithm calculations would be eliminated. At several milliseconds per logarithm the result would be the saving on many hours of precious computer time.

The reader will also recall that the original derivation of equation 1 was not valid for multiple occurrences of the same letter in a group. It can easily be verified that the derivation of this special case is quite similar to the original case and when simplified becomes the same as equation 4. Therefore, the programming would be further simplified using equation 4 since there are no special cases to consider.

Justification. The validity of equation 4 depends on the assumption that a particular group occurs much less often than any of its components. Preliminary experiments by the author showed that for three- and four-letter groups this was generally true and the error introduced averaged less than 5% for the former and less than 2% for the latter. However, the error was much greater for two-letter groups, averaging about 35%. In all cases the estimated value was higher (less negative) than the true value. Since the purpose of the ΔH figure is to determine the order of entry into the alphabet, the effect of the error would be simply to delay the entry of a particular group.

Since the policy of selecting entries according to ΔH ranking was chosen only because it seemed like the reasonable thing to do, and since, as will be shown shortly, the actual ΔH contribution of all entries in the alphabet table are constantly fluctuating, the effect of delaying an entry due to ΔH error is almost impossible to analyze. Errors in ΔH of 5% or less are almost certain to be masked by other effects. The only area that may cause some concern is the 35% error in the two-letter groups. However, if the final results show that the overall contribution of two-letter groups is small, then even this concern may be minimal.

Taking all these factors into consideration, I decided to use the simpler formula, actually equation 3, for calculation of ΔH.

Method of Grouping

In determining the ΔH of a group, the components initially are single letters; however, as the alphabet grows the components of a group may themselves be other groups. Problems may then arise in that it may be possible to group a potential entry in several different ways. For example, suppose the alphabet contains the entries: HIS, IS, T and TH (among all the others). To determine the ΔH of THIS the grouping could either be T HIS or TH IS, each grouping possibly giving a different ΔH.

Ideally, all possible groups should be tried but this would be extremely time consuming and difficult to program. The procedure used here is that which would probably be used by a typical translator which is always to take as large a bite as possible going from left to right. (The division would then be, in this example, TH IS.) Any sacrifice in entropy reduction must, therefore, be considered as a property of the translator and not as error in the experiment.

Modification of the Algorithm

Discussion was made earlier concerning the effect of a new entry into the alphabet on the ΔH of the remaining potential entries. The effect of such an entry on the other entries already in the alphabet should also be examined. Four cases will be shown with examples:

1. Groups which are contained in the new group. For example, THE enters the alphabet; what happens to the ΔH of TH which is already in the alphabet? The frequency of free TH's, x_{TH}, is

reduced. The frequency of free T's and H's is not affected so p_Tp_H/p_{TH} increases. (p_T is the probability of occurrence of the letter T, etc.) The magnitude of ΔH for the TH decreases and the sign could even become positive.

2. Groups which contain the new group. TH enters, what happens to THE? x_{THE} remains constant but the fraction inside the log is now $p_{TH}p_E/p_{THE}$ rather than $p_Tp_Hp_E/p_{THE}$. $p_{TH} > p_Tp_H$ (a requirement for TH to enter). The log term, and therefore ΔH, becomes less negative.
3. Groups which overlap the new group. HE enters, what happens to TH? The fraction p_Tp_H/p_{TH} decreases since some free H's were used up. Therefore the ΔH becomes more negative.
4. Groups which are not related to the new groups. The fraction $x_1x_2 \ldots x_n/x_T^{n-1}x_A$ in equation 2, affected only by a small decrease in x_T will increase and the log will gradually become less negative.

This examination shows that in cases 1,2, and 4, especially 1 and 2, the contribution of an alphabet entry to the total entropy reduction is diminished and could even increase the entropy. For this reason the basic algorithm is expanded so that after a new entry is placed into the alphabet, the ΔH of all other entries in the alphabet (except single letters) is re-computed. Any entry who's ΔH exceeds that of the most recent entry is removed and returned to the group table; the tables are readjusted and the text is regrouped. Later on, of course, this group may be returned to the alphabet. A flow chart of this modified algorithm is shown in Figure 2.

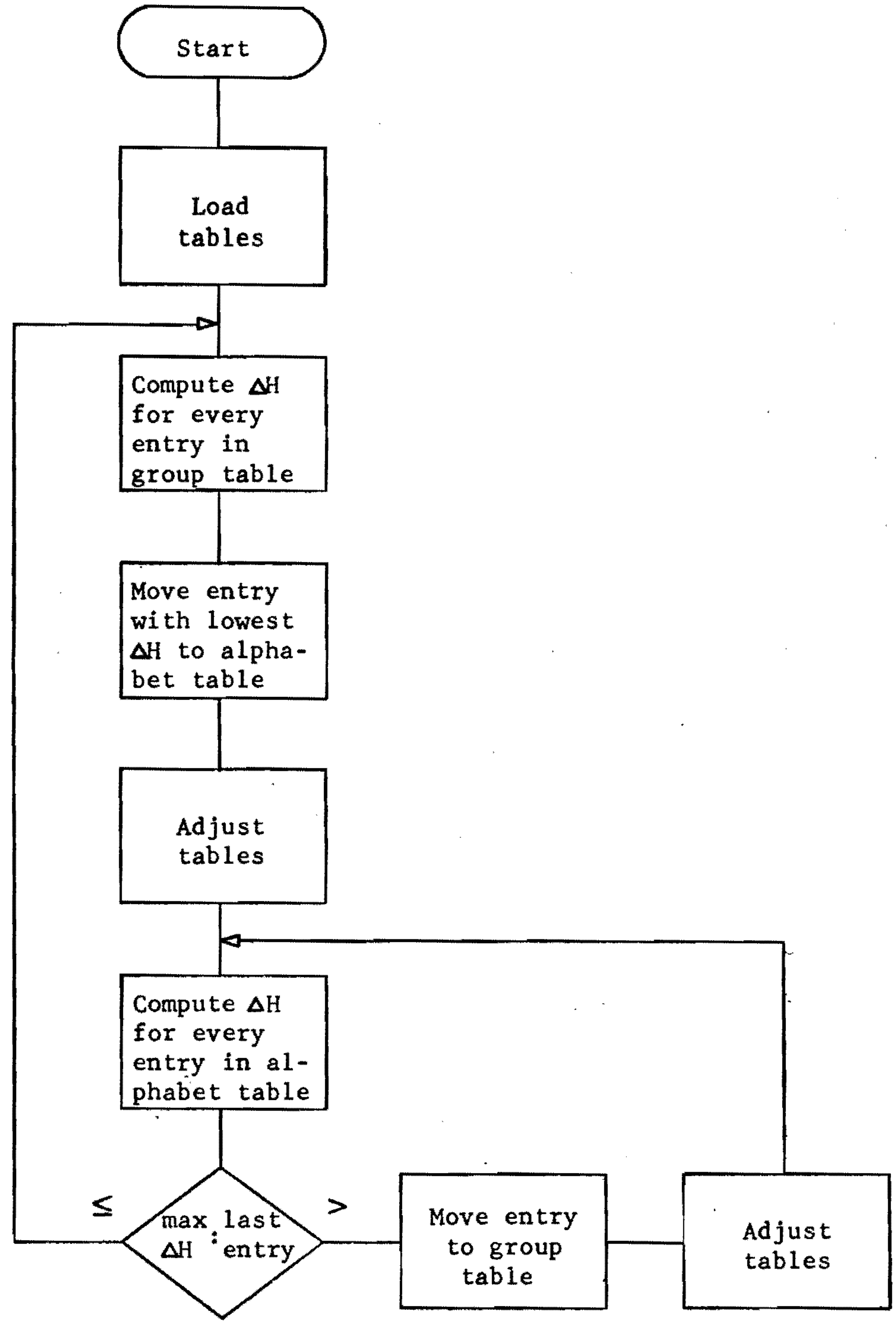

Start
Load tables
Compute ΔH for every entry in group table
Move entry with lowest ΔH to alphabet table
Adjust tables
Compute ΔH for every entry in alphabet table
max ΔH : last entry
≤
>
Move entry to group table
Adjust tables

Figure 2. Flow chart of the modified algorithm used in this experiment.

When to Stop the Process

Theoretically, the most efficient (and totally useless) alphabet would be one which had only one entry, the whole sample; the total entropy would be zero! The translator could handle only one message--the sample.

This cannot happen here because of some limitations placed on the entries in the group table as will be described in the next chapter. What will happen is that the program will run until all entries having a negative ΔH have entered the alphabet. On test runs using very small samples of text, this occurred in a rather short time.

Using a large sample, unfortunately, it may not be economical, in terms of computer time, to allow the program to run to completion. It was hopefully assumed that the graph of the relation between entropy and table size would decay rather quickly and then gradually level off. Periodic printouts from the computer during the run would allow the operator to observe the progress and terminate the run after the curve appeared to level off.

CHAPTER III

IMPLEMENTATION CONSIDERATIONS

I. THE MACHINE AND THE PROGRAM

The computer used for this project was a Honeywell 1250 with 131,072 bytes of memory. The memory cycle time of this machine is 1.5 microseconds per byte. As described earlier, each byte consists of nine bits, of which, six are used for actual data storage. The seventh and eighth bits, called word marks and item marks respectively, are used for identifying field boundaries. The ninth bit is used by the machine for error checking purposes.

The memory was divided into four areas as shown in Figure 3. The program was stored in the first 8000 positions. The alphabet table started at 25000 and expanded downward. The text sample began at 25000 and loaded upward and the group table began at the top end of memory and loaded downward, the two tables almost meeting at about 78000. To

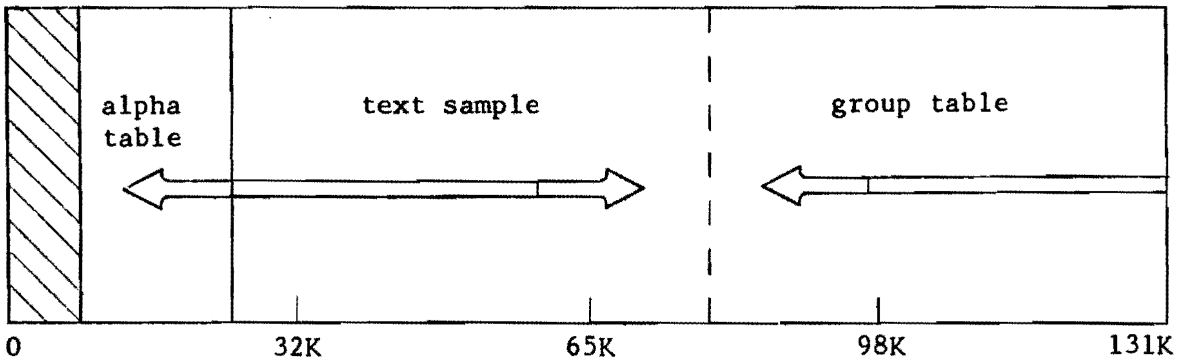


Figure 3. Computer memory utilization used in the experiment. (Cross-hatched area was used for the program.)

assure the greatest possible efficiency in working with individual characters the program was written in assembly language.

The program was designed so that whenever an entry was added to or removed from the alphabet table it was printed out together with the computed ΔH figure. Also, at the beginning of the run and after every 50 entries into the alphabet table the complete alphabet table was printed out together with the frequency and entropy of each entry. As each entry was printed it was also punched out on a card so the cards could be used for later machine analysis.

Since this experiment was to require over 15 hours to run, a restart capability was included. At regular intervals during the run the operator could cause the entire memory to be written out and saved on magnetic tape. If the main memory were accidentally destroyed due to mechanical failure or power failure, it could be rebuilt and the program restarted from the point at which the most recent save was performed.

II. TEXT SAMPLE

Initial Preparation

The text sample used in this experiment was a selection of about 7000 words from a United States history text, The Democratic Experience, by Niebuhr and Sigmund.

The text was punched onto standard 80 column cards using columns 6 through 80. The first five columns were reserved for a card number so the cards could be re-sequenced in case the deck should accidentally be dropped. In order to preserve the original word spacing a continuous format was used; that is, punching continued from the end of one card onto the beginning of the next with no regard to word boundaries. The only exception is that each paragraph began on a new card, indented five spaces. All numerals and punctuation were included even though they were not used in this particular experiment.

<u>Input Editing</u>

At the beginning of the run all the cards containing the text were read into the text table during which the following editing took place:

1. All characters other than the letters A through Z and the blank were suppressed.
2. To simplify the delimiting of words all single blanks were changed to double blanks. Each blank could then be thought of as a "half blank". Between each pair of words there would be two "half blanks", one belonging to each word.
3. With the period suppressed, all sentences would terminate with a double blank. Therefore, all occurrences of double blanks were replaced with blank--period--blank. Each blank ("half blank") delimited its associated word and the period delimited the two sentences without actually belonging to either sentence. Although these periods appeared as part of the text sample table, they were not included in any of the character counts.
4. All strings of blanks, such as those occurring between paragraphs, were reduced to double blanks and handled as such. Therefore, there was no delimiting of paragraphs in the text sample table.

Table Structure

As each character of the edited text was loaded into the text table, the item mark bit of that character was set. This indicated that the text was composed of individual characters. When the run began and grouping occurred, the groups were indicated by removing all the item marks of a group except for the one over the left hand character of the group. The right hand end of a group need not be marked since the next character beyond would be the beginning of a new group and would, therefore, have an item mark. Figure 4 shows an example of text first in its original form and then after the group, TION, was formed. Characters having the item mark bits set are indicated with an "i".

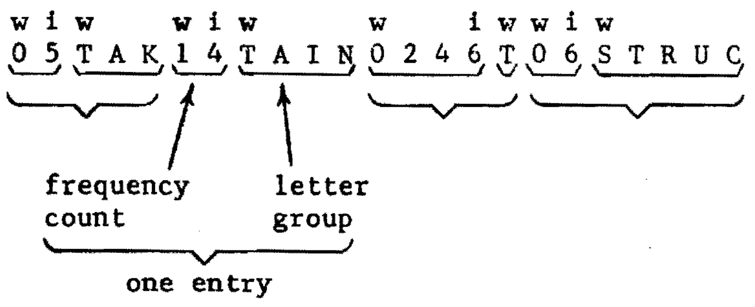
```
i i i i i i i i i i i i i i i i i i i i i i i i i i i i
A     N A T I O N A L     R E G U L A T I O N     W A S

i i i i i i       i i i i i i i i i i i       i i i i i
A     N A T I O N A L     R E G U L A T I O N     W A S
```

Figure 4. Example of text storage showing the use of item marks before and after the group, TION, was formed.

III. THE ALPHABET TABLE

Structure

The alphabet table can best be described by looking at Figure 5 which shows a small sample taken at the termination of the run. Each



<u>Figure 5</u>. Sample taken from the alphabet table.

entry consists of a letter (in this case, T), or a letter group (TAK, TAIN, STRUC) with a word mark over the leftmost character. To the left of each such field is a numeric count of the number of occurrences of that group. The count field has a word mark on the left end and an item mark on the right. It can be noticed that the entries are in alphabetical sequence, the table going from right to left.

At the beginning of the run, of course, the only entries were A through Z and the blank. Each of these initial entries had a four digit count field initially set to zero. The proper counters were incremented as the text sample table, described previously, was loaded.

Binary Search

Since the running of this program would require millions of references to the alphabet table, it is necessary to be able to find an item in the shortest time possible. For this reason I decided to use the method known as the binary search.

In the normal binary search, the item to be located is compared with the item in the middle of the table. The result of the comparison determines whether the desired item is above or below this mid-point; that is, it tells in which half of the table the item is located. Next the mid-point of the particular "half" is selected and the process repeated. Each cycle of the process halves the area of search and the process continues until the area is reduced down to a single item. It can readily be seen that the number of cycles required to find an item in a table of n items is simply $\log_2 n$. A 1000 item table, for example, requires only 10 search cycles to retrieve an item.

Because of the variable item length structure of the alphabet table it was necessary in this program to modify the standard binary search. Each time a mid-point was selected it was necessary to scan the immediate area until an item mark was found then compared with the entry to the right.

Table Maintenance

The use of the binary search requires that all items in the table be in alphabetical sequence. Therefore, when a new item was added to the table this sequence had to be preserved. This was accomplished by first performing a binary search. Even though the item was not yet in the table, the search found the point at which the item was to be inserted. Next, every character in the table from the low end up to the insertion point was moved downward to make room for the new item which was then inserted. Fortunately, on this machine, the entire "moving down" could be accomplished with a single instruction.

To remove or delete an item from the table it was necessary only to use a binary search to locate the item then "move up" the lower part of the table enough positions to overlay the old entry.

IV. GROUP TABLE

The structure of the group table was identical to that of the alphabet table except that it was much larger. Look-up was also accomplished by an identical binary search.

Because of the difficulty in determining the frequencies of thousands of groups, the table was initially loaded from a tape which had been prepared earlier and will be described in the next section.

Since no items were ever to be added to the group table, the add/delete capability was not included. When an item was "removed" from the group table (to be placed in the alphabet table) it was not really removed, but its count field was simply set to zero. Similarly an item was "restored" to the group table by merely restoring the count field.

V. PREPARATION OF THE GROUP TAPE

Requirements

Although the original concept of the group table was that it would contain all groups of all sizes it is obvious that such a table would be prohibitively large. Therefore, in order to have a table which would fit into about 50,000 memory positions, some editing was required.

The first requirement was that no letter groups were allowed which extended over word boundaries. The assumption is that the letters within a given word were not influenced by the letters within a neighboring word and therefore would not form useful groups. (Work done by Newman and Gerstman (15) indicates that this might not be entirely true, however.) Groups larger than a word were allowed only if they were groups of complete words. Similarly, word groups which extended across sentence boundaries were not allowed.

Initial attempts at building a group tape indicated that the table was still much too large and further reduction was necessary. I noticed that a large percentage of the table consisted of entries which occurred only a few times. By eliminating all entries which occurred less than five times the table was finally reduced to a size which would fit into the allowable memory. Whether or not this cut-off would have an effect on the eventual alphabet table depended on the relation between this cut-off and the frequency distribution of the groups. This will be discussed in later chapters.

Method of Preparation

The first step in the tape preparation was to read in the text sample from cards and write out all possible 40 character groups onto a work tape. The cards were the ones described earlier in reference to the loading of the text sample table. The previously described editing of the card data was also employed. The groups written out were such that each group consisted of a single new character from the card plus the 39 previous characters. The first group, therefore, consisted of characters 1 through 40; then the next group included characters 2 through 41; the next, 3 through 43; and so forth. The figure, 40, was chosen because of programming considerations but proved to be of no limitation in itself as will be seen.

As the first step in fulfilling the editing requirements, given earlier, any 40 character group which began with a period or a double blank was suppressed from the work tape.

After the entire work tape was built it was sorted into alphabetical sequence using a standard utility routine supplied by Honeywell. The effect of the sorting was to collect all like groups together.

The next step was to read in the sorted work tape and count the occurrences of each set of like groups or parts of groups and write out onto another work tape. The process was as follows: Forty counters were used, one corresponding to each character position in a group. Whenever a group was read that was identical to the previous group all counters were incremented by one. If only the first n characters were identical, a record consisting of the first n+1 characters of the old group was written out together with the contents of the (n+1)st counter. Additional records were then written for the n+2, n+3 positions on up to 40. All counters from one to n were incremented by one; all counters above n were reset to one.

In addition, another suppression routine prevented the following groups from being written out:

1. Items ending with a double blank since the second blank would belong to the next word and groups were not to cross word boundaries.
2. Items ending with a period since groups were not to cross sentences.
3. Items containing a double blank that did not begin and end with a blank. If a group did begin and end with a blank it would have been a complete word or possibly group of words,

which was allowed.

4. Items whose counter was less than five.

Since parts of groups were held in memory as others were written out, the output tape was again out of sequence. The final step, then, was to re-sort the tape, the result being the complete group tape.

Since the memory requirements were not large the entire process was performed on a smaller machine, a Honeywell 1200 with a 32K memory. The programs were written in Fortran.

VI. ADJUSTMENT OF TABLES

As was described in the last chapter, whenever an entry was added to or removed from the alphabet table, the remaining entries in all the tables may have required adjustment. This process is now examined in detail.

Adding an Entry to the Alphabet

Once a new group had been added to the alphabet the program scanned down the text sample until the first occurrence of this group was found. Figure 6 shows an example of the group, PRESENT, being processed. The components of this group (PRES, EN and T) are each located in the alphabet table and their corresponding counters each decremented by one. All the other items listed in Figure 6 are then located in the group table and similarly decremented. (The symbol, Δ, indicates a blank.) It will be noted that each of these items are composed of two or more groups, at least one of which is contained in PRESENT and at least one of which is not. The reason for the decrementing, of course, is that after PRESENT is formed into one group then all the other items

```
i i         i i   i       i   i i i     i       i     i
▵ ▵ T H E ▵ ▵ R E P R E S E N T A T I V E S ▵ ▵ O F ▵ ▵


Affected groups:                  E N
                                  E N T
  ▵ R E P R E S                   E N T A
  ▵ R E P R E S E N               E N T A T I V
                                  E N T A T I V E S ▵
  R E P R E S
  R E P R E S E N                 T
                                  T A
  P R E S                         T A T I V
  P R E S E N                     T A T I V E S ▵
```

Figure 6. Example from the text table of the group, PRESENT, being formed showing all the related groups which are affected.

listed will no longer be candidates for possible grouping. Groupings which completely contain the new group, such as REPRESENT, are still possible, however, and so these groups are not decremented.

It will also be noted that parts of groups, such as EPR, are not decremented, even though they probably exist in the group table, because EPR would have been previously decremented when the group RE or PRES was formed.

Since letter groups are not allowed to extend over word boundaries all the groups shown are all included within the word, ▵REPRESENTATIVES▵. Had the new group been a complete word the above procedure would have been the same except that the groups processed would have been groups of complete words.

Since any group originally occurring less than five times was not included in the group table it is possible that a given item, such as ENTA in Figure 6, might not be found in the group table. The program took advantage of the fact that, in this case, ENTATIV and ENTATIVES▵ would also have been omitted, and so no lookup of these items would have been attempted.

If a counter in the group table were ever decremented to zero, the item was considered as deleted, as was explained earlier. (The routine which computed ΔH on the group table passed over such items.) If a counter in the alphabet table went to zero, however, the delete routine was automatically entered and the item was removed.

After all the counters had been decremented, the new group was officially formed by removing all the item marks except the one on the left hand character. In this example, the item marks over the E and the T in PRESENT would have been removed leaving only the one over the P.

The scan of the text sample then resumed and the process was repeated. The actual groups processed may have been entirely different, however, since the next occurrence of PRESENT may have been PRESENTATION.

Removing an Entry from the Alphabet

If an item had to be removed from the alphabet due to a change in its ΔH, the process described was reversed.

When the desired group was found during the scan of the text table the first step was to re-group the old group into the largest subgroups possible. Item marks were set at the left end of each such subgroup.

The same alphabet table and group table adjustments were made as before except that the counter for each entry was incremented by one rather than decremented.

CHAPTER IV

RESULTS AND DISCUSSION

I. THE TABLES

Text Sample Table

The exact size of the text sample was not known until the text table was loaded. The initial printout showed that the table contained 51143 characters. Inspection of the memory listing showed that the table required 51475 bytes. The difference of 332 is accounted for by the periods, which were not counted. Since the first and last characters of the sample were periods there were, therefore, 331 sentences in the sample.

The total number of blanks was 14106. Since each word had a blank at each end there were, therefore, 7056 words. Of these 7056 pairs of blanks 331 would have occurred in the original sample at the ends of sentences. The remaining 6722 were, therefore, added by the program. The total number of effective characters in the sample, therefore, was 51143 minus 6722 or 44421.

The Group Table

The group table contained 6442 entries and required 47417 bytes of memory. Single-word groups accounted for 224 entries or about 3.5% of the total entries. A summation of the word frequencies totaled 4889, meaning that 70% of all words in the sample were contained in the group table. The remaining 30% were lost because of the lower limit cutoff that was imposed.

There were 93 two-letter groups or 1.4%, accounting for 915 double words of text. This would cover 26% of the sample assuming no overlap of groups. Since there was probably some overlap, the percentage would be less.

There were only 12 three-word groups and only 1 four-word group, ∆THE∆∆ARTICLES∆∆OF∆∆CONFEDERATION∆, and none higher. This four-word group contained 34 letters meaning that the 40-letter restriction imposed during the building of the group tape had no limiting effect.

II. THE ENTROPY REDUCTION

Run History

The printout of the complete history of the run is shown in the appendix in Table III. Each entry is terminated by a comma so it can be determined which entries end with a blank. Those beginning with a blank appear indented. The DELTA H column shows the amount of entropy reduction of the entire sample resulting from each entry. Notice that some entropy figures are positive quantities, and they are indented for clarity. These represent entries leaving the alphabet rather than entering.

It can be seen that the entropy changes are quite large at first and then become smaller. Toward the end of the run they are quite small and change rather slowly. Because of this, and since the program had already run for 15 hours, I decided to terminate the run after 600 cycles of operation.

Alphabet Summary

After every 50 entries had been added to the table a listing of the complete alphabet was automatically performed. These places are indicated in the run history as "checkpoints". The actual listings were too lengthy to be included in the appendix, but a summary of each listing is shown in Table I.

TABLE I

SUMMARY OF ENTROPY DATA TAKEN AT CHECKPOINTS DURING THE COMPUTER RUN

Check-point	Total Entries	Total Groups	Total Entropy	Entropy per Char	Correctd Entropy
0	27	51143	195954	4.41	4.08
1	76	34477	155659	3.50	3.24
2	124	29837	143632	3.23	2.99
3	171	26635	135816	3.06	2.83
4	216	23679	129628	2.92	2.70
5	263	21918	125059	2.82	2.61
6	313	20268	121156	2.73	2.53
7	355	19292	118313	2.66	2.46
8	399	18217	115671	2.60	2.41
9	448	17219	133466	2.55	2.36
10	493	16484	111521	2.51	2.32
11	539	15843	109975	2.48	2.29
12	584	15222	108787	2.45	2.26

There were 13 listings; listing 0 was printed after the initial load before any grouping occurred. The table starts with the original 27 characters and stops with 584.

The "Total Groups" column is shown mainly for interest and it is not analyzed further. From the first entry, 51143, the total character count was determined since at this point each "group" consisted of but a single character. At the end of the run each group consisted of an average of about 3.4 characters.

The figures in the "total entropy" column were obtained by actual summation of every entry in the alphabet and not simply an accumulation of ΔH's since (as was discussed earlier) the ΔH calculations used an approximation formula and the ΔH's were subject to degeneration. The entropy per character was computed by dividing the total entropy by the number of effective characters in the text, seen earlier to be 44421. A graph of this figure plotted against table size is shown as the solid curve in Figure 7.

Effect of the Double Blanks

Since the text sample table contains extra blanks for ease in word delimiting, the effect has been that the computed entropy figures are somewhat high. No attempt was made in the program to correct these figures; however, it was possible to make some corrections and estimates at the end of the run.

First, as was shown earlier, at the beginning of the run 6722 blanks were added to the sample. By re-computing the initial total entropy, considering 6722 fewer blanks, a figure of 181324 was obtained giving an entropy per character of 4.08 bits. This, incidentally, is very close to Shannon's estimate for single characters of 4.03 (14).

At the end of the run a hand count of the remaining blanks in the memory listing was made. All single blanks except those next to a period were eliminated since a single blank implies that its partner was absorbed into a group which is, therefore, properly delimited. The only blanks counted, then, were one each of 273 pairs plus 168 next to

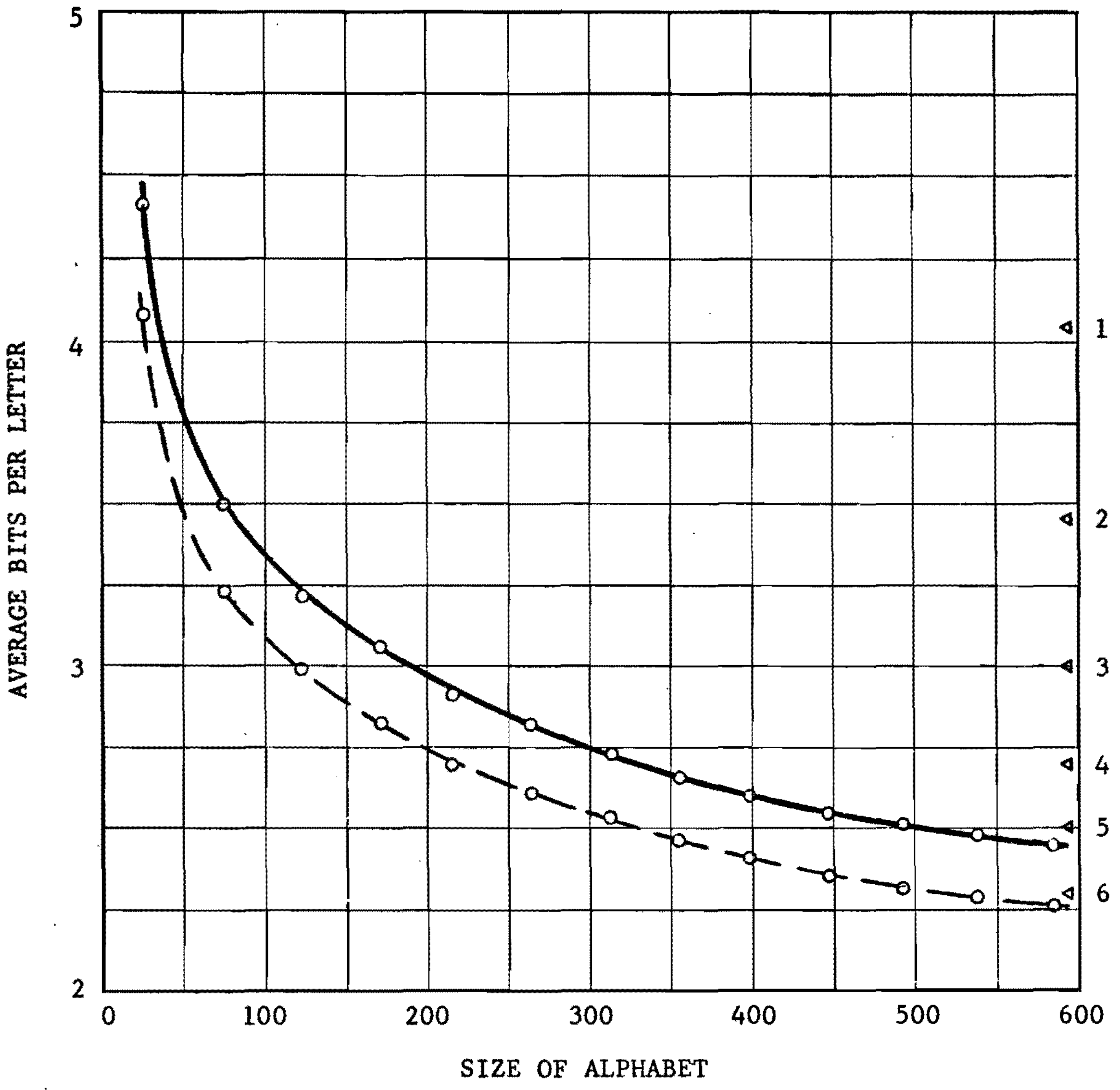


Figure 7. Graph of entropy per letter versus alphabet size.

periods or 441 rather than an actual count of 2992. Recomputing on this basis yielded a total entropy of 100579 or an entropy per character of 2.26 bits.

It was interesting that the ratio of entropy correction in both cases was 0.92. Assuming that the correction at the other 11 points might have been roughly the same a second set of points was computed and shown as a dashed line in Figure 7.

Interpretation of the Curve

The points along the right side of Figure 7 show the entropy estimates of Shannon for single-letter groups up to six-letter groups. It can be seen that with less than 100 items in the alphabet the entropy has already been reduced to that obtainable with a two-letter table. At 200 and 300 entries it surpasses a three- and four-letter table respectively. At 500 to 600 entries the entropy is further reduced although other factors may limit its usefulness as will be seen shortly.

Equation of the Curve

It was originally hoped that the curve would somewhere show a definite leveling off point. One seemingly good point would be where the slope changes most rapidly, which is the point at which the third derivitive goes to zero. However, the log log plot in Figure 8 produced

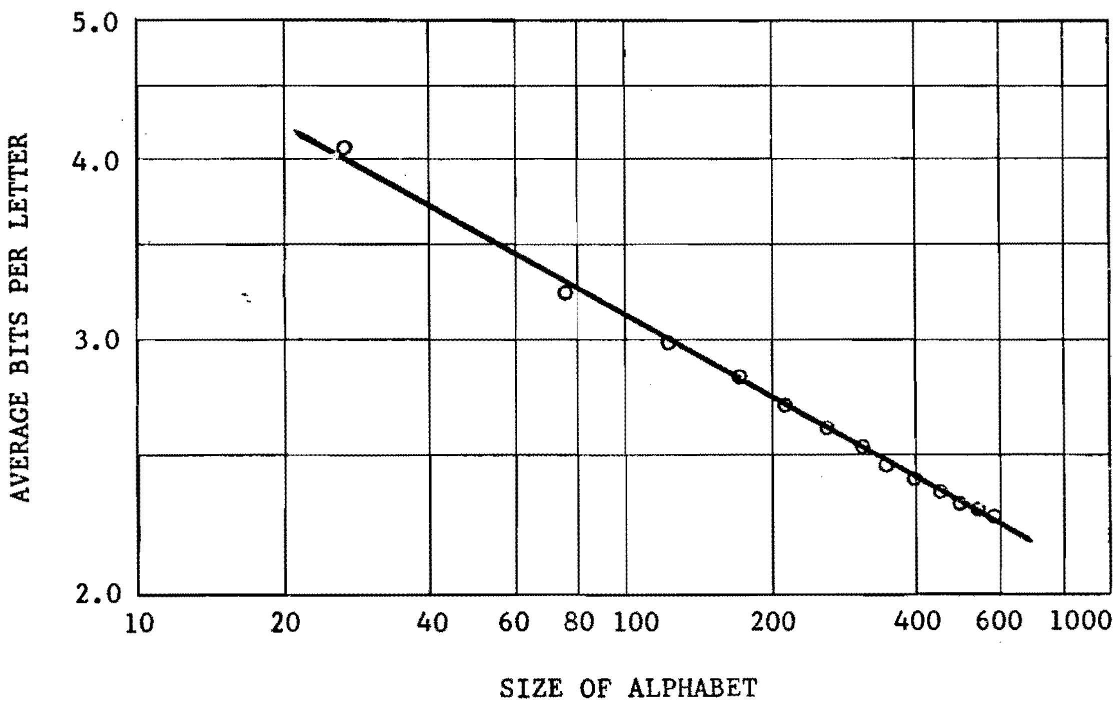


Figure 8. Lower curve of Figure 7 drawn on logarithmic scale.

almost a straight line, indicating that the data was basically exponential. Since none of the derivatives of an exponential go to zero, there seems to be no nice way to define where the curve levels off.

By determining the slope and the intercept of the line in Figure 8 the equation is found to be:

$$\log_{10} H = -.19 \log_{10} N + .875$$

where N is the number of entries in the alphabet. The original curve in Figure 7, then, has an equation:

$$H = 10^{.875} N^{-.19}$$

or roughly:

$$H = 7.5 / \sqrt[5]{N}$$

III. DISTRIBUTION CURVES

Group and Word Distribution

An analysis of the run history in the appendix was made to determine what proportion of the entries consisted of letter groups, single words, double words, etc. The results are shown in Table II and on the graph in Figure 9.

These results show that once the curve stabilizes the proportions remain fairly constant except for a slight rise in letter groups toward the end. Of significance is the fact that letter groups and single words make up 90% to 95% of the table. The small remainder is composed essentially of two-word groups, and the higher order groups contribute virtually nothing.

Also shown is the percentage of two-letter groups. This was included since there was some discussion in an earlier chapter on the use

TABLE II

LETTER AND GROUP DISTRIBUTION TAKEN AT CHECKPOINTS DURING THE COMPUTER RUN

Check-point	Total Entries	% Letter Groups	% 1-word Groups	% 2-word Groups	% Higher Groups	% 2-1tr Groups
0	27	100.0	-	-	-	-
1	76	71.1	25.0	2.6	1.3	0.0
2	124	62.1	31.5	4.8	1.6	1.6
3	171	59.1	35.1	3.5	2.4	1.8
4	216	59.7	34.7	3.7	1.9	4.2
5	263	56.7	36.9	4.9	1.5	3.4
6	313	58.8	35.5	4.5	1.3	3.5
7	355	59.2	34.1	5.6	1.1	3.4
8	399	60.2	32.8	6.0	1.1	3.8
9	448	62.7	30.1	6.2	.9	4.0
10	493	64.3	29.0	5.9	.8	5.1
11	539	66.0	27.3	5.9	.8	5.6
12	584	66.4	26.7	6.2	.7	6.0

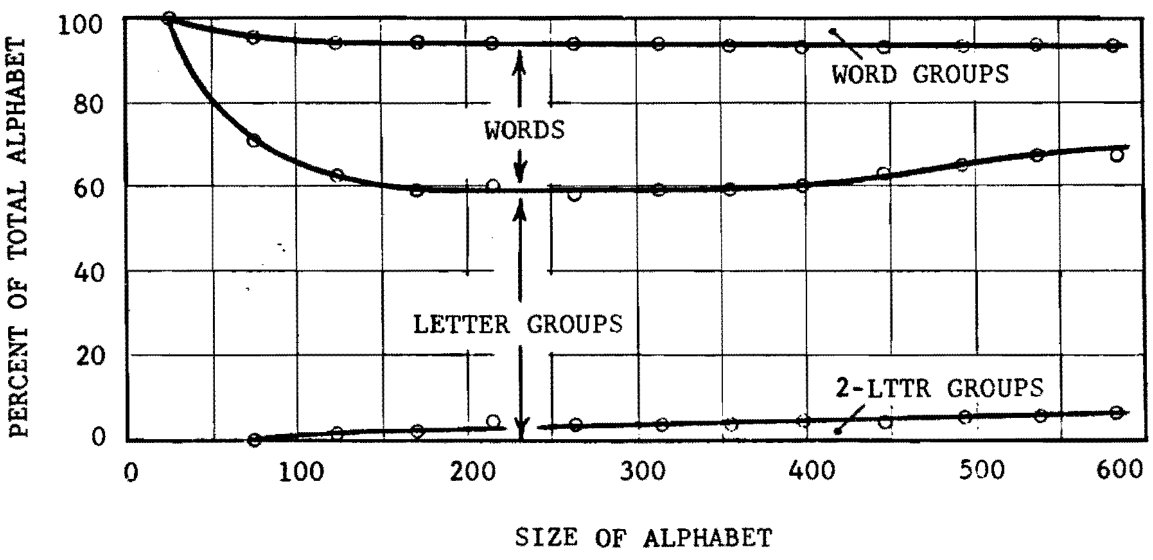


Figure 9. Graph of letter and word group distribution as a function of alphabet size.

of the simplified ΔH approximation, in that there could be considerable error when applied to two-letter groups. It is seen that the percentage of such groups is quite small.

Frequency Distribution

To observe the effect of the low limit cut-off of five during the building of the group tape, a count was made of how many items in the alphabet occurred at given frequencies. These were computed from the 13 checkpoint listings. The results of selected tabulations are shown in Figure 10 as a frequency polygon.

The actual frequencies extend from 5 up to several thousand; however, everything of interest occurs only below about 20, so that is all that is shown.

By observing the behavior of the curve in the known area one can make some guesses as to what might have happened in the area below five. It can be seen that with 76 items in the alphabet none occurred less than 9 times, the peak being around 12 or so. The cut-off apparently had no effect. At 171 items the peak has shifted to the left and it appears that a few items with a frequency less than 5 would have occurred if there had not been the cut-off. At 216 it's hard to tell what the curve is trying to do, but at 313 and beyond the curve is definitely peaking in the region below five.

Had the cut-off not occurred, many low frequency items with a good ΔH probably would have gone into the alphabet and the entropy would, therefore, have been lower than that obtained here.

On the other hand, it seems undesirable to include a large number of low frequency items in the alphabet because many of these items might be found only in this particular sample. This thought opens up the whole area of sample sensitivity and is discussed briefly in the last chapter.

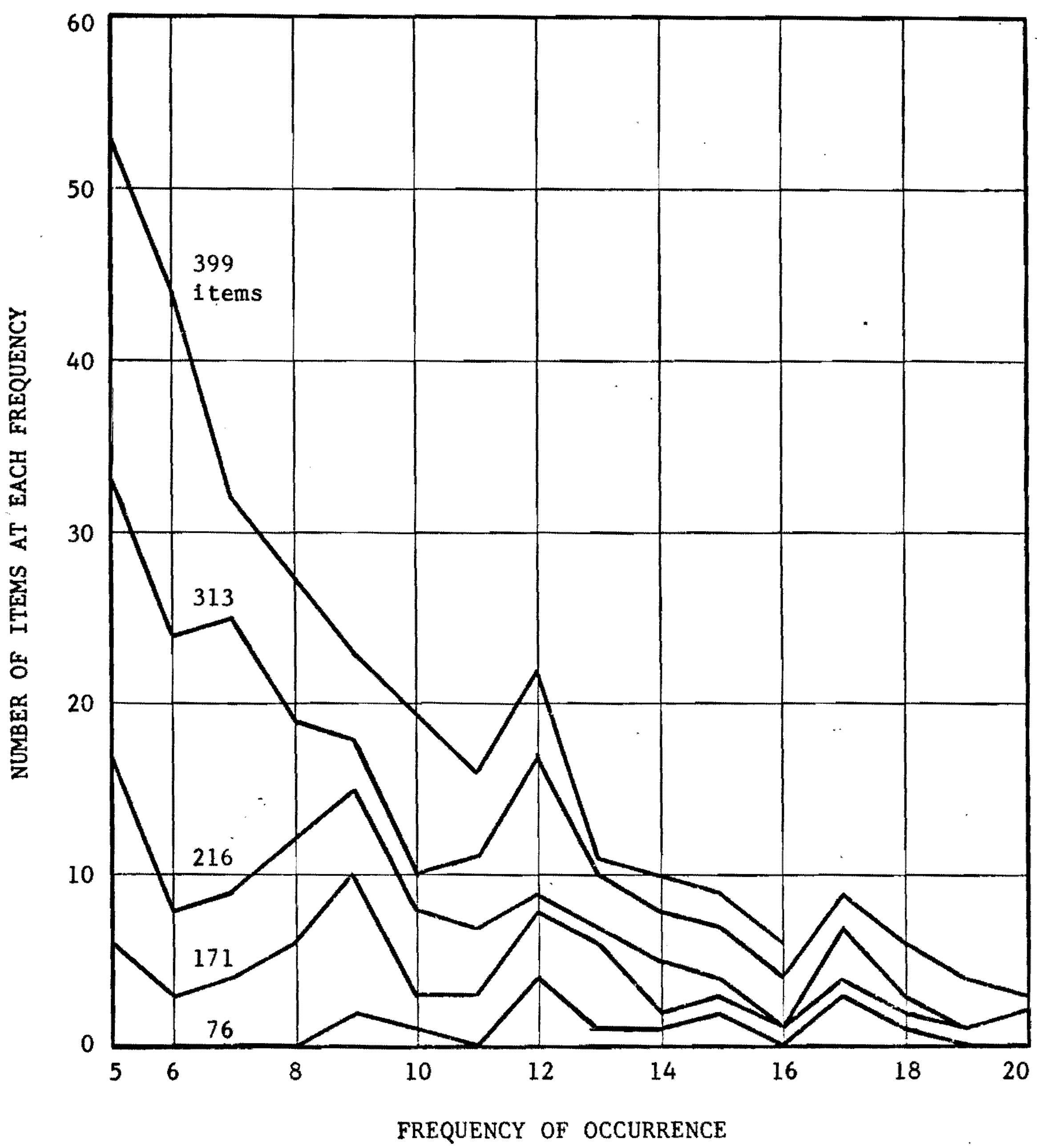


Figure 10. Graph of group frequency distribution at selected checkpoints during the computer run.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

I. CONCLUSIONS

Entropy Reduction

Considerable reduction in the entropy of English text can be accomplished with a relatively small table by grouping the text into variable length groups. Reducing the entropy to below 3 bits per character can be accomplished with an alphabet of less than 200 entries. Using a larger alphabet of around 500 words it may be possible to reduce the entropy to below 2.5 bits.

Because of the simplifications made these figures can only be taken as an estimate. Had the ΔH equation not been simplified and had the low frequency cut-off not been necessary, the entropy results would probably be even lower. However, if numerals and punctuation had been included the entropy would not be as low. Also, there will be an entropy increase due to losses in any coding scheme used in a translator.

Table Content

It is definitely worthwhile to include both letter groups and single words in the alphabet. I would doubt the use of word groups, however, since their contribution is small and involves extra programming complexity. In any event, groups larger than two words are definitely not worth handling.

II. RECOMMENDATIONS

Changes in Procedure

Accurate ΔH Prediction. Since the percentage of two-letter groups in the alphabet was small, it suggests that the effect of the simplified ΔH equation on the total entropy reduction is minimal. It is possible, however, that the ΔH error is somewhat the cause of the low percentage, since the error of approximation has the greatest effect on two-letter groups. I would suggest that the next run incorporate the original ΔH equation even though the computer time would be increased considerably. Comparison of the results, however, would indicate the degree of error and perhaps allow a decision to be made as to a preferred method.

Elimination of Word Groups. Since I found that the contribution of word groups was small it would be worthwhile eliminating them from the processing thereby simplifying the programming in this area. The building of the group tape would also be speeded up since it would not be necessary to allow for the large maximum group size of 40 characters.

Use of Complete Alphabet. Since the ultimate goal of this experment is to develop a workable translating system the run should be made with the complete alphabet rather than just 27 letters. Since there is little redundancy in the numerals and punctuation their entropy is high; however, in normal text their usage is usually low. Their overall effect on the reduced entropy remains to be seen.

The Group Table and Sample Sensitivity

The one area that still requires considerable work is that of preparing the group tape. Some editing is definitely required in order to finish with a group table that will fit into memory. Yet, the arbitrary cut-off at five items left much to be desired.

Closely related to this problem is the one of sample sensitivity which was not at all studied in this experiment. As more and more groups are taken from a particular sample and placed in the alphabet the more the alphabet becomes tailored to that one particular sample. This tailoring can be reduced only by insuring that the selected groups are truly representative samples of English itself and not just the particular sample.

But where is the line to be drawn between "English itself" and a particular sample? And, perhaps, some sample dependence is desirable. If a translating system is to process only history text then the inclusion of "pure history" words is desirable even though the system may now perform poorly with chemistry. A good translating system could have several alphabets in disk storage, and at the beginning of each message it could load into main memory the one tailored for a particular text.

One method of building the group tape might be to build many tapes each from a different sample. Then, according to some statistically determined criterion, select those groups which appear on most of the tapes and reject those which appear on only a few. How much diversity in the selection of the samples depends again on the intended diversity of the translating system.

REFERENCES CITED

1 Zinn, Karl L., "Instructional Uses of Interactive Computer Systems," Datamation, v. 14, pp. 22-27, Sept., 1968.

2 Lipetz, Ben-Ami, "Information Storage and Retrieval," Sci. Am., v. 215, pp. 224-242, Sept., 1966.

3 IBM Corp., IBM System/360 Principles of Operation, Poughkeepsie, IBM Corp., 1967.

4 Honeywell Inc., Honeywell Series 200 Programmers' Reference Manual, Wellesley Hills, Honeywell Inc., 1965.

5 Honeywell Inc., Direct-Access Devices and Controls, Wellesley Hills, Honeywell Inc., 1968.

6 Henny, Keith, Radio Engineering Handbook, New York, McGraw-Hill, 1959, p. 24-4.

7 IBM Corp., 1130 Commercial Subroutine Package, White Plains, IBM Corp., 1968, pp. 15-17, 21-23.

8 Abramson, Norman, Information Theory and Coding, New York, McGraw-Hill, 1963.

9 Edwards, Elwin, Information Transmission, London, Chapman and Hall, 1964.

10 Hyvarinen, L. P., Information Theory for Systems Engineers, Berlin, Springer-Verlag, 1968.

11 Pierce, J. R., Symbols, Signals and Noise, New York, Harper and Brothers, 1961.

12 Raisbeck, Gordon, Information Theory, An Introduction for Scientists and Engineers, Cambridge, M.I.T. Press, 1963.

13 Reza, Fazlollah, An Introduction to Information Theory, New York, McGraw-Hill, 1961.

14 Shannon, C. E., "Prediction and Entropy of Printed English," Bell System Tech. J., v. 29, pp. 147-160, 1951.

15 Newman, F. B. and Gerstman, L. J., "A New Method for Analyzing Printed English," J. Exptl. Psychol., v. 44, pp. 114-125, 1952.

APPENDIX

The following pages are a computer printout showing all groups as they were going into or coming out of the alphabet table. The effect of each entry on the total entropy is also shown.

Groups which appear to be indented one space are those which begin with a blank. All entries are followed by a comma. This comma is not part of the entry but was added by the program, so that it would be possible to tell which entries end with a blank.

Entries leaving the alphabet are shown with a positive ΔH, which is also indented on the listing.

Although entries are selected in order of increasing ΔH, an occasional item may appear out of sequence due to a particular grouping of the previous entry.

It can be noted that a few entries do not have a ΔH figure. These are groups which were "used up" forming other groups, and so were purged from the alphabet with no effect on the entropy. Single letters were not purged but were listed with an asterisk. (Q was the only case.)

After every 50 cycles of operation a checkpoint listing was produced consisting of a complete alphabet printout. These printouts are not included here but the checkpoints are indicated on the listing.

TABLE III

COMPUTER RUN HISTORY

ALPHABET ENTRY	DELTA H	
THE ,	5187-	
GOVERNMENT,	2633-	
TION,	2026-	
AND ,	1664-	
OF ,	1638-	
STATES ,	1535-	
CON,	1217-	
LEGISLATURE,	937-	
ING ,	936-	
THE ARTICLES OF CONFEDERATION ,	819-	
CONGRESS ,	822-	
WAS ,	816-	
ED ,	814-	
THAT ,	810-	
CONSTITUTION,	809-	
CON,		655
OULD ,	799-	
WHICH ,	742-	
TO ,	734-	
THE,	738-	
FEDERALIS,	714-	
WITH,	678-	
CON,	642-	
IN ,	652-	
FOR,	644-	
REPRESENT,	629-	
POWER,	599-	
WERE ,	553-	
DELEGATES ,	537-	
LY ,	505-	
MASSACHUSETTS ,	452-	
THE ARTICLES ,	451-	
ENT,	449-	
STATE ,	435-	
CONVENTION,	429-	
PRO,	419-	
POLITICAL,	415-	
HOWEVER ,	394-	
IGHT,	372-	
TIVE ,	368-	
VIRGINIA ,	366-	
THEIR ,	367-	
FROM ,	365-	
AMERICA,	354-	
COMMERC,	353-	
POPULA,	347-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H	
BY ,	346-	
CENTRAL GOVERNMENT ,	344-	
COMPROMISE,	333-	
HAMILTON,	332-	
INDEPEND,	324-	
CHECKPOINT 1		
RATIFICATION ,	320-	
AUTHORITY ,	317-	
REVOLUTION,	317-	
CONTINENTAL ,	309-	
NEW YORK ,	304-	
NATIONAL,	302-	
THER,	299-	
BE,	297-	
HAD ,	290-	
VER,	280-	
INTEREST,	273-	
SYSTEM ,	267-	
CONFEDERATION ,	262-	
THE ARTICLES OF CONFEDERATION ,		168
PEOPLE ,	261-	
BUT ,	246-	
LEGISLA,	242-	
OF THE ,	237-	
ON ,	235-	
COLONI,	237-	
NORTHWEST ,	235-	
UNDER ,	234-	
DOCUMENT ,	231-	
WH,	229-	
OUGH,	227-	
BETWEEN,	226-	
ORDINANCE ,	225-	
EACH ,	223-	
AFFAIRS ,	222-	
STRONG,	219-	
ANTIFEDERALISTS ,	219-	
LARGE,	218-	
AL ,	215-	
AND,	216-	
THE DECLARATION OF ,	217-	
TERRITOR,	213-	
BRANCH,	210-	
BELIEVED THAT ,	210-	
BILL OF RIGHTS ,	209-	
MAJORITY ,	202-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H	
EXECUTIVE ,	202-	
PER,	201-	
MOST ,	200-	
COUNTRY,	200-	
IN,	200-	
NOT ,	204-	
MADISON ,	190-	
UNITED STATES ,	188-	
QU,	186-	
G,	*	
REGULA,	188-	
ALLY ,	187-	
CHECKPOINT 2		
GOVERNOR,	187-	
DIFFERENCE,	185-	
IT ,	183-	
SUCH ,	181-	
GREAT,	181-	
PRESIDENT,	176-	
NEW ,	176-	
JAMES ,	176-	
AFTER ,	175-	
THEY ,	172-	
TH,	170-	
THE,		120
THERE,	170-	
OPPOSITION ,	170-	
IN REGARD TO ,	169-	
HOUSE,	169-	
APPROV,	168-	
OUT,	167-	
ECONOMIC,	167-	
THEMSELVES ,	167-	
NCE ,	166-	
ESTABLISH,	162-	
ITS ,	161-	
WOULD ,	160-	
COULD ,	164-	
OULD ,		133
SHOULD ,	284-	
VOTE,	159-	
FOREIGN ,	158-	
JUDICIAL ,	157-	
APPOINT,	154-	
SOME ,	153-	
PRINCIPLE,	153-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H	
ATION ,	152-	
WEST,	151-	
JOHN ,	151-	
AGAINST ,	151-	
THE ARTICLES OF CONFEDERATION ,	149-	
PLAN ,	148-	
PUBLIC,	147-	
DOMESTIC ,	146-	
FFIC,	144-	
EFFEC,	147-	
GENER,	144-	
FAVOR,	144-	
COM,	143-	
FARMERS ,	142-	
ABLE ,	140-	
TRADE ,	140-	
OR,	140-	
FAVOR,		126
RE,	150-	
REVOLUTION,	289-	
CHECKPOINT 3		
REVOLUTION,		49
THEIR OWN ,	142-	
ELEC,	140-	
CONTROL,	140-	
WAR,	139-	
THE LOWER ,	137-	
BRITISH ,	136-	
REPRESENTATIVES ,	135-	
AGREED ,	134-	
TER,	133-	
FEDERAL ,	134-	
DI,	133-	
DE,	138-	
FREEDOM ,	132-	
MUCH ,	131-	
THEM ,	130-	
EX,	129-	
ALREADY ,	129-	
ACCEPT,	129-	
LEADERS,	126-	
CREAT,	126-	
ALL,	125-	
FAVOR,	124-	
PROPOS,	123-	
TIONS ,	123-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H	
IVE ,	122-	
TIVE ,		114
PROPOS,	123-	
TIONS ,	123-	
IVE ,	122-	
TIVE ,		114
EN ,	123-	
AN,	137-	
AR,	141-	
WAR,		93
ON,	126-	
END,	132-	
TOWARD ,	125-	
NUMBER,	123-	
SPECI,	116-	
UN,	115-	
PRE,	115-	
UR,	115-	
ER,	144-	
TER,		66
NUMBER,		114
ATE,	119-	
FIRST ,	119-	
ISSUE ,	117-	
GEORGE ,	116-	
THESE ,	114-	
RECOGNI,	114-	
NUMBER ,	114-	
MORE ,	114-	
THIRTEEN ,	113-	
BALANCE ,	112-	
AVE ,	112-	
TAX,	113-	
ECT,	111-	
CHECKPOINT 4		
TRIBU,	110-	
THE FEDERALISTS ,	110-	
NORTH ,	110-	
TIES ,	109-	
OUS,	109-	
VISION ,	108-	
STRU,	107-	
ESS,	107-	
PROBLEM,	106-	
SIMPLY ,	104-	
HENRY ,	104-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H	
SHAYS ,	103-	
INTO ,	103-	
IMPORT,	103-	
THE STATES ,	100-	
AS ,	99-	
AMONG ,	98-	
MUST ,	98-	
CHECK,	97-	
CAME ,	97-	
TWO ,	97-	
ITS OWN ,	97-	
TO BE ,	96-	
POLIC,	95-	
MONEY ,	95-	
LIKE ,	94-	
DEBT,	94-	
ES ,	93-	
TIES ,		78
SECTION,	94-	
ENC,	94-	
INDEPENDENCE ,	102-	
RATIF,	93-	
ATED ,	91-	
THE CONSTITUTION ,	91-	
MARKET,	91-	
LOCAL ,	91-	
WITHOUT ,	90-	
BOTH ,	89-	
THREE ,	88-	
SECOND,	87-	
PROPERTY ,	87-	
MIGHT ,	84-	
RIGHT,	87-	
IGHT,		77
SUPP,	83-	
THIS ,	82-	
JAY ,	82-	
YEARS ,	82-	
CALLED ,	82-	
SMALL,	84-	
ALL,		58
ALL ,	93-	

CHECKPOINT 5

SHIP,	81-	
CH,	83-	
ACK,	83-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H
PROVID,	81-
CLOSE ,	81-
WHO ,	81-
SOUTH,	80-
MEET,	80-
CENTRAL ,	80-
MENT,	81-
IGHT,	79-
PERIOD ,	79-
EVERY ,	79-
LAND ,	78-
ALSO ,	78-
OW,	77-
WORK,	79-
WAR,	80-
OLVED ,	78-
COLL,	78-
ECTIONS ,	77-
OVER ,	77-
FUL,	76-
WHILE ,	76-
WHEN ,	77-
MATTER,	76-
FOUND,	76-
EVEN ,	76-
WIL,	75-
TIES ,	75-
STATE LEGISLATURES ,	75-
TORS ,	74-
TIME ,	75-
RESULT,	75-
BRI,	74-
ARGU,	74-
CAUSE ,	73-
A ,	74-
ATIVE ,	74-
OTHER,	74-
MAKE ,	74-
POINT,	73-
MEN ,	73-
REMAIN,	72-
EQUAL ,	72-
ORIGIN,	71-
ITY ,	70-
ARY ,	70-
NE,	70-
IZE ,	69-

TABLE III--Continued

CHECKPOINT 6

ALPHABET ENTRY	DELTA H	
STATE GOVERNMENTS ,	69-	
AT ,	69-	
EMBER,	68-	
ACT,	68-	
UP,	67-	
CONSIST,	67-	
ALMOST ,	67-	
TAIN,	66-	
TEMPT,	65-	
IGN,	65-	
THERE WAS ,	65-	
TERMS ,	65-	
POPULATION ,	65-	
IT WAS ,	65-	
HIS,	65-	
DEPEND,	65-	
APP,	65-	
APPOINT,	109-	
APPOINT,		19
AMERICAN,	65-	
EN,	65-	
EN ,		18
ENC,		43
END,		46
ENCE ,	95-	
NO ,	66-	
VOTED ,	64-	
OLUTION,	64-	
POS,	64-	
DANGER,	64-	
DIRECT,	64-	
ONE ,	63-	
ONLY ,	68-	
WHERE ,	63-	
WH,		61
WHO,	76-	
WHO ,		36
OF GOVERNMENT ,	63-	
INTER,	63-	
INTEREST,	143-	
INTEREST,		36
EST,	64-	
DECIS,	63-	
FORE ,	62-	
FFER,	62-	
FRAM,	63-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H	
THAN ,	62-	
REPRESENTATION IN ,	61-	
FEW ,	61-	
RESTR,	60-	
RATHER ,	60-	
AGR,	61-	
PHIL,	60-	
THROUGH,	59-	
THERE WERE ,	59-	
THERE,		47
COMMERCIAL ,	59-	
ING,	58-	
CHECKPOINT 7		
SPAIN,	59-	
HELD ,	58-	
ANCE ,	58-	
NCE ,		5
ANCE ,		57
CE ,	59-	
ENCE ,		54
AD,	58-	
EAS,	59-	
PASS,	59-	
TRANS,	58-	
TOO ,	57-	
LEGISLATIVE ,	57-	
METH,	56-	
HAND,	57-	
ALTH,	56-	
WOULD BE ,	56-	
POPULAR ,	56-	
POPULA,		23
CURR,	57-	
MITT,	55-	
IOUSLY ,	55-	
THUS ,	55-	
BASI,	55-	
THOUGH,	54-	
IONS ,	53-	
ECTIONS ,		38
TREA,	53-	
RANK,	53-	
THE CONVENTION ,	53-	
ON THE ,	54-	
SO,	53-	
EITHER ,	53-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H	
ESTION,	52-	
ATIONS ,	52-	
TOWN,	52-	
HAVE ,	52-	
ISH,	52-	
LONG ,	51-	
ISSI,	51-	
IC,	52-	
ION ,	55-	
ID,	59-	
DID ,	54-	
DURING ,	53-	
WITH THE ,	51-	
MEAN,	51-	
COMMERCE ,	51-	
COMMERC,		
SIX,	50-	
SET ,	50-	
MIL,	50-	
CESS,	49-	
THERE,	49-	
ANGE ,	48-	
AMP,	48-	

CHECKPOINT 8

ALPHABET ENTRY	DELTA H	
REGULATE ,	48-	
PROTECT,	48-	
PART ,	48-	
INDI,	48-	
MEND,	48-	
MEDI,	48-	
EXPERI,	43-	
SERV,	47-	
LE ,	47-	
LAW,	47-	
ANY ,	47-	
ENCY ,	47-	
ENCE ,	47-	
LESS ,	46-	
GRESS,	49-	
ESS,		44
DGE ,	46-	
NEW GOVERNMENT ,	46-	
IN THE ,	46-	
FOR THE ,	47-	
IMPER,	46-	
PORTION,	45-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H
OUNT,	45-
DUC,	45-
SEVER,	45-
COLONIES ,	45-
OLD ,	44-
RO,	44-
FIN,	44-
DIS,	44-
IST,	44-
ERS ,	44-
ESS,	45-
STA,	45-
POWERS ,	45-
SU,	45-
PUT,	44-
LPH,	43-
JU,	43-
GOVERNMENT SHOULD ,	43-
GIV,	43-
OFFIC,	42-
IZED ,	41-
MIS,	41-
RI,	41-
SH,	41-
MA,	41-
AT,	41-
DRA,	42-
IRE ,	41-
MAR,	42-

CHECKPOINT 9

AL,	41-
CLA,	42-
ACY ,	42-
AB,	42-
YEAR ,	41-
COMPL,	40-
GENERALLY ,	40-
MENTS ,	39-
ELECTION ,	40-
ITION ,	41-
NEVER,	39-
COULD BE ,	39-
BEEN ,	40-
OSE ,	38-
ORDER,	38-
MER,	39-

TABLE III--Continued

ALPHABET ENTRY	DELTA H	
OCK,	38-	
ACC,	39-	
CRE,	39-	
IF,	38-	
ITI,	38-	
IM,	40-	
IMPER,		36
FEDERALIST,	38-	
FEAR,	38-	
OOD,	37-	
OP,	38-	
OPER,	52-	
PORT,	37-	
EFFECT,	38-	
EFFEC,		
ISE ,	37-	
ARE ,	37-	
REVOLUTIONARY ,	37-	
ANT,	36-	
TERRITORY ,	36-	
UND,	35-	
TH,	35-	
TH,		9
THERE,		26
ALTH,		27
THIR,	41-	
PRESS,	35-	
EDED ,	35-	
AIL,	35-	
LVE ,	35-	
SLA,	36-	
UPON ,	35-	
UNION ,	35-	
IMPER,	35-	
DO,	35-	
TAK,	34-	
AM,	34-	
GAVE ,	34-	
END,	33-	

CHECKPOINT 10

MEND,		30
ECTIONS ,	33-	
ANCE ,	33-	
AG,	33-	
ASS,	33-	
RIGHTS ,	33-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H	
URE ,	32-	
WE,	33-	
EFF,	32-	
FUS,	32-	
FAC,	32-	
AP,	32-	
NATION ,	32-	
ELECT,	32-	
TIL,	31-	
IT,	32-	
ITI,		26
ITIES ,	48-	
IA ,	35-	
IAL ,	34-	
COLONIAL ,	32-	
COLONI,		
PLA,	31-	
ULT,	31-	
EL,	31-	
ELEC,		26
ELY ,	58-	
DELE,	32-	
ARLI,	32-	
ATELY ,	31-	
ENTLY ,	31-	
ACH,	31-	
WA,	31-	
SPE,	31-	
SPECI,	69-	
MON,	31-	
FORM,	31-	
FAR,	31-	
ARD,	31-	
CONTINENTAL CONGRESS ,	31-	
STATE,	30-	
TER,	30-	
EPT,	30-	
FR,	30-	
FO,	31-	
URY ,	29-	
COMM,	29-	
THE RIGHT ,	29-	
SEE,	29-	
INTERESTS ,	29-	
FROM THE ,	29-	
WH,	28-	
VERY ,	28-	
RY ,	28-	

TABLE III--Continued

CHECKPOINT 11

ALPHABET ENTRY	DELTA H	
CLO,	28-	
OD,	28-	
COD,		25
CHO,	29-	
TO THE ,	28-	
THAT THE ,	28-	
DEBTS ,	28-	
ROW,	27-	
LOW,	30-	
MAN,	27-	
INGS ,	27-	
FULLY ,	27-	
FACT,	27-	
EV,	27-	
WON ,	27-	
LACK,	27-	
IS ,	27-	
IOUS ,	27-	
COULD NOT ,	27-	
CENT,	27-	
BRANCHES ,	27-	
IES ,	27-	
TAXES ,	27-	
THEN ,	26-	
SON ,	26-	
NOW,	26-	
ALTH,	26-	
UD,	25-	
IZ,	25-	
IZED ,		20
ANISH,	25-	
THERE,	25-	
PROPOSED ,	25-	
PROPOS,		22
OR ,	25-	
WOR,	25-	
WORK,	35-	
POWER TO ,	25-	
BR,	25-	
BRI,	37-	
BRI,		18
ATT,	25-	
ADD,	25-	
FOR ,	25-	
FORCE ,	33-	

TABLE III--Continued

ALPHABET ENTRY	DELTA H
ADV,	25-
TY ,	24-
ST,	25-
STATE,	22
SENT ,	24-
ENC,	24-
TEN ,	24-
STRUC,	24-
AY ,	24-
ALS ,	24-

CHECKPOINT 12