

in relation to the selected microprocessor and the use of a hardware table look-up for searching is examined.

The use of a multi-level dictionary for text data compression has been suggested in the literature. In this thesis, a two level dictionary has been considered in the context of microprocessors. The performance of a compression scheme is studied by means of the coding time, the compression ratio, and the accesses to the two dictionaries. The effect of the size of the first level dictionary on the performance is examined. A typical implementation of the second level dictionary using CCD (charged coupled device) and its organization are studied.

The final part of this thesis is concerned with the observed frequency distributions of word fragments. Word fragments or n-grams have been studied by several authors in different contexts. This thesis examines the distribution of word fragments for various values of "n" in the light of average word length of the data set, the threshold value used in accepting or rejecting an n-gram, the sample size, and the subject contents of the data sets. It is found that a maximum or a peak occurs in the distributions between $n=4$ and $n=5$ irrespective of the size of the data set, of the threshold frequency, of the average word length and irrespective of the subject content of the data set.

ACKNOWLEDGEMENTS

I wish to express my deepest thanks to my supervisor, Dr. T. Radhakrishnan, for his advice, guidance and helpful comments which have been an invaluable aid throughout both the research and the writing process of this thesis.

My special thanks goes to my friend Mustapha for his moral support and patience.

I am grateful to my friends, Christina and Khalid for their assistance during the final preparation of this thesis.

TABLE OF CONTENTS.

Chapter	page
1 Introduction	1
2 Data Compression: A Literature Survey	7
2.1 Introduction	7
2.2 Terms and Definitions	9
2.3 Text Data Compression	13
2.4 Signal Data Compression	22
2.5 Image Data Compression	25
2.6 Summary	29
3 Three Experimental Data Sets	32
3.1 Data Set1	32
3.2 Data Set2	33
3.3 Data Set3	33
3.4 A Comparison	38
4 A Dictionary Structure for Text Data Compression	42
4.1 Dictionary Based Text Data Compression	42
4.2 A Dictionary Structure for Data Compression	53
4.3 On the Design of the Dictionary Structure	56
5 Data Compression with Microprocessors	73
5.1 Microprocessors in Data Compression	73
5.2 Memory Technology Related to Microprocessor Application	75
5.3 Organization of DICT2 on CCD	80
5.4 Parallel Processors	89
5.5 On-Line Data Compression with Microprocessors	96

6	Use of Word Fragments for Data Compression	104
7	Conclusion	118
	References	121

LIST OF FIGURES

3.1	Word Length Distribution for DATA SET1.....	35
3.2	Word Length Distribution for DATA SET2.....	37
3.3	Word Length Distribution for DATA SET3.....	40
3.4	Word Length versus Percentage of occurrence.....	41
4.1	Data Compression System.....	44
4.2	Partial Coding.....	45
4.3	Graph Representation of MS Algorithm.....	50
4.4	Trie Organization of Dictionaries.....	55
4.5	Storage required versus Accesses Required.....	63
4.6	Dict1 Size versus Hit Ratio.....	70
4.7	Dict1 Size versus Average Processing Time per Word.....	71
4.8	Dict1 Size versus Compression Ratio.....	72
5.1	Memory Technology:Cost/Performance.....	76
5.2	Synchronous Organization.....	79
5.3	SPS Organization.....	81
5.4	LARAM Organization.....	82
5.5	Organization of 1 unit of CCD 460.....	83
5.6	Distribution of Fragments in DICT2.....	85
5.7	Space Time Diagram of Pipeline Architecture.....	91
5.8	A CCDP Processor.....	93
5.9	Multiprocessor Architecture.....	94
5.10	Time Spent in Table Look-up versus Encoding Time.....	101
6.1	Word Length Distribution.....	106
6.2	Effect of Sample Size (DATA SET1).....	108
6.3	Effect of Sample Size (DATA SET2).....	109
6.4	Threshold Effect (DATA SET1).....	112

6.5	Threshold Effect (DATA SET2).....	113
6.6	Word Length Effect (DATA SET1).....	115
6.7	N-gram Distribution without Threshold (DATA SET1).....	116

LIST OF TABLES

2.1	Application of Various Compression Methods.....	29
3.1	Characteristics of DATA SETS1.....	34
3.2	Characteristics of DATA SET2.....	36
3.3	Characteristics of DATA SET3.....	39
4.1	Space Required and Wastage Storage.....	59
4.2	Characteristics of n-grams (DATA SET1).....	60
4.3	Dictionary Accesses Required.....	62
4.4	Summary of Results for DATA SET1 and DATA SET2.....	67
5.1	Table Directory for Dict2.....	87
5.2	Some Types of Commercial Real Time Systems.....	98
5.3	Summary Results of experiment with M6800.....	100

CHAPTER I

INTRODUCTION

Most information systems require direct access to large data bases. Such large data bases impose high storage and maintenance costs. In view of the "information explosion", data compression becomes more important when large amount of data is to be stored or transmitted.

The cost of storing data is a significant part of the total computer system cost. This cost is composed of the direct charges for the storage media, such as disk devices, as well as the costs of transferring the data to and from local and remote storage devices. Data compression results in cost savings by reducing the amount of storage required to store data files. In addition, data compression methods may enable more efficient information retrieval operations as well as more economical transmission of large amounts of data over computer networks. There are several types of data compression techniques and some of them are discussed and summarized in this thesis.

In the past, researchers have studied data compression in different application fields such as:

1) signal data compression

2) text data compression

3) image data compression

In this research, we are concerned mainly about text data compression. Text data compression has been studied by several authors with respect to library and business data files [Heaps,75], [Ruth,72]. Text compression is achieved by reducing the intrinsic redundancy of stored text. Zipf [Zipf,49] has shown that the distribution of words or characters in a large text sample is hyperbolic, and [Booth,67] indicated that 50% of the different terms occur only once, 16% occur only twice and 8% occur only three times. These vocabulary characteristics are useful for building an efficient text compression module.

A decade ago or so, minicomputer applications tended to be primarily in instrumentation, test and control systems. Originally distinguished from general-purpose main-frames by their size, price, function specialization, and absence of a solid software base, minicomputers were often sold as control components of other systems.

Then in the early 1970's, the microcomputer was born. LSI circuit technology has allowed microcomputers to increase performance to where today their capabilities are

almost boundless. Microcomputers gained popularity as components in word processors, data entry terminals, and data communications equipment. They are versatile and have stand-alone capabilities. Microprocessors are proving cost-effective for a number of applications and as more experience is gained their applications will broaden.

The objective of this thesis is to study the feasibility of the use of general-purpose microprocessors for dictionary based text data compression.

Two classes of data rates may be distinguished:

- (1) High data rates in the order of mega bits per second - They occur in the transmission of TV signals and in data transfers with high speed disks.
- (2) Low data rates in the order of tens or thousands of bits per second - They occur in on-line transmission of data from manually operated keyboards.

Due to their low speed, general-purpose microprocessors are not directly suitable for high data rates [Lea,78]. This thesis considers their applications to cases where low data rates are common. For this study, a dictionary based text compression scheme, a coding algorithm, a general-purpose microprocessor, and a set of experimental data have been selected. The encoding time has been analysed empirically and performance curves and tables are presented.

Another part of this thesis considers the implementation of a multi-level dictionary proposed by [Heaps,77]. When microprocessors are employed the size of the first level dictionary versus the performance has been examined as controlled by the following parameters

- hit ratio
- processing time
- compression ratio

Two different experimental data sets were used and the results are compared. The organization of the second level dictionary is described as implemented on a CCD. The cost, organization and performance of such a model is discussed and a multiprocessor architecture is proposed.

The last part of the thesis is the study of word fragments for data compression. Statistics of word fragments or n-grams of textual words have been studied by several authors [Gupta,77], [Suen,79]. Most of these studies have used large experimental data bases and hence were limited to n-grams with "n" varying up to 5 characters. In this thesis a compromise is adopted between the largest "n" examined and the data size. When "n" is increased sufficiently high, the number of different n-grams first increases, reaches a peak, and then decreases. This phenomenon has been studied as a function of the following parameters:

- 1) Average word length of the data base
- 2) Threshold frequency used in accepting or rejecting an n-gram
- 3) Sample size
- 4) Subject matter of the data base (technical, general, etc.)

The thesis is organized in the following way. In Chapter II a literature survey of data compression is presented along with definitions of terms commonly used in data compression.

Chapter III describes three data sets which were used for testing the coding algorithm and building the necessary dictionaries.

Chapter IV contains a discussion on text data compression and the description of three encoding algorithms used for text data compression. This chapter discusses the two level dictionary structure used, the storage requirements of these dictionaries, and the accesses to them. Results for two different data sets are presented and discussed.

Chapter V considers the applications of microprocessors for data compression. Several contemporary technologies are described. An organization of the second level dictionary with CCDs (charge coupled devices) is discussed. A multiprocessor architecture using CCD is proposed for

on-line data compression. Considerations in on-line data compression with microprocessors are discussed. An experiment performed on MOTOROLA 6800 is presented and results are discussed.

Chapter VI contains a study of the application of n-grams for data compression. Several experiments were performed and results are presented.

CHAPTER II

DATA COMPRESSION: A LITERATURE SURVEY

2.1 INTRODUCTION

Most computer-stored information contains redundant information and compression of data is performed by removing, either partly or totally, the redundancy that might be present in the data. As the amount of data in computer systems and in communication between systems such as in computer networks increases, data compression will become more and more useful.

Compression of stored or transmitted data is used because of the following advantages it provides:

- 1) potential increase in transmission of data for a given channel capacity
- 2) reduction of physical storage space of a file
- 3) increase in capacity in mass storage devices, channels and communication lines
- 4) cost savings
- 5) other related benefits such as security

In dictionary based data compression systems, a set of codeable data units and their codes are stored in one or

more dictionaries. When these dictionaries are accessed very often, it becomes necessary to store them in relatively faster access memories and such a necessity may be construed as a "storage overhead". The compression or decompression process takes a finite amount of time and the increased CPU time required for these operations is the price one pays for data compression and is known as "time overhead". The "storage overhead" and "time overhead" are the two disadvantages of data compression.

Different types of data are used in data compression and they may be classified into the following three categories

- 1) signals
- 2) linear strings
- 3) structured data

SIGNALS

Transmission of signals in digital form is more common today than in the past. When analog signals are encountered, they are sampled and quantized into discrete levels to transmit them in digital form. Seismic signals, speech signals, TV signals are some examples of this data type.

LINEAR STRINGS

This category includes numeric and alphanumeric data. They are strings of digits, characters or symbols which appear in general written texts, business files and in specialized data such as computer source programs and object programs. Redundancy and the non-uniform frequency of data units in written texts [Zipf,49] are exploited for compression of such data.

STRUCTURED DATA

This class refers to two or multi-dimensional structured data types such as line drawings, images, graphs, flowcharts, maps, two-tone pictures etc. It may also be combined with written texts as in newspapers, technical papers or "yellow pages" of a phone book.

2.2 TERMS AND DEFINITIONS

In this section, some commonly used terms in data compression are described.

CODEABLE ELEMENT - is a unit or subunit of the given data type selected for coding such as a word, phrase, n-grams or fragments, or a run of identical symbols (e.g. bbb or 000), which is convenient for the process of compression.

CODE - internal code assigned to a codeable element. Codes may be of variable or fixed-length. Variable-length codes,

such as Huffman codes, require complex algorithms for decoding while fixed-length codes are easier to process. Between the two extremes of fixed-length and variable-length codes, there is another method of assigning codes to a codeable element, which is known as "restricted variable length" codes [Ramam,64]. It is a compromise between Huffman codes and fixed-length codes.

DICTIONARY -- list of codeable elements with their corresponding codes as used in a compression scheme. This dictionary is referred to during the coding and decoding process.

ENCODING/DECODING - encoding is the process of generating the compressed data from the original data and decoding refers to the reverse process. The execution time of these procedures constitutes "time overhead" in a compression scheme.

COMPRESSION RATIO - it is defined as the ratio of the number of bits or storage units in the original data to that in the compressed data. This ratio is highly data dependent. Also it is influenced by the dictionary and the encoding process.

REVERSIBLE/IRREVERSIBLE COMPRESSION - reversible compression is a technique in which the original data can be reproduced from the code. Irreversible compression is a technique in which the coded information cannot be reproduced.

INFORMATION DESTROYING/INFORMATION PRESERVING COMPRESSION - in the case of an information-destroying compression scheme, the data not relevant to an application are suppressed from transmission and storage. Obviously, this requires a clear articulation of the user needs and a means of communicating it to the "data compression system". In the information-preserving compression scheme, no data collected is discarded. The information-destroying scheme depends on the property of the user of the information and it is popular with image data compression. An example of this scheme is the class of transform methods.

REAL-TIME COMPRESSION - a real-time compression system is one in which the input data enters the computer directly from the point of origination; it is then compressed "immediately" and the compressed output data are transmitted directly to where they are used. An example of real-time compression is the compression of satellite data, in which large amounts of data have to be transmitted at very high rates from a satellite to the ground station without much storage on-board the satellite.

COMPRESSION SCHEMES - a compression scheme is a method used to compress a given data set. The following are the four general categories of compression schemes used in data compression :

1) Dictionary based substitution :

codeable elements in the dictionary which appear in the file to be compressed are replaced by the corresponding code of the codeable element. Examples of algorithms based on dictionary based encoding are "LONGEST MATCH", "MINIMUM SPACE" and "LONGEST FRAGMENT FIRST" algorithms [Schue,74].

2) Run-length coding :

This scheme replaces a sequence or a run of identical values by a code and it does not require a dictionary. Null Suppression and Pattern Substitution are examples of such coding schemes [Arons,77].

3) Differencing (delta modulation) :

A sequence of data values are considered in this method. The i -th data value is estimated as a function of the past ($i-1$) data values. Then the difference between the actual and the predicted value, which generally requires fewer bits than the actual data value, is transmitted or stored [Gotli,75].

4) Transform methods :

This class of compression methods are based on linear transformations. A well known transform method, for which even special electronics hardware has been devised, is Fourier transform.

2.3 TEXT DATA COMPRESSION

The present section is concerned primarily with compression of textual data appearing in documents or abstracts and in legal, medical and library information systems.

Natural language texts require very high capacity of on-line storage and there is a rapid increase in the number of systems handling such material. This storage can be costly and thus a method which reduces the storage space is desirable.

A natural language, such as English, is highly redundant and it is therefore beneficial if some of this redundancy could be removed and a more compact representation of the text stored. Text compression is achieved by reducing the intrinsic redundancy of stored text. Text strings contain considerable redundancy due to the disparate frequency distribution of characters and words. In a sufficiently large sample of English text, if the different words are ranked in descending order of their frequencies of occurrence, it is found that the word of rank r occurs with a probability p that is given approximately by the equation $p = A/r$ where A is a constant that depends on the length and type of text. This equation is known as Zipf Law [Zipf,49]. The frequency-rank distribution is hyperbolic which implies that 50% of the different words in a data base

occur only once, 16% occur only twice and 8% occur only three time [Booth,67]. Another characteristic of text data bases is their property with regard to growth of vocabulary. If the data base has a total of N terms, then as N increases the number D of different terms increases according to the formula [Heaps,78]

$$D = KN^B$$

where K and B are constants whose values depend on the particular data base. Thus the vocabulary grows logarithmically as a function of the data base size. Zipf law and the logarithmic law are therefore very useful for prediction of data base characteristics and for estimating the efficiencies of various schemes for compression and coding of text data bases.

Language elements and coding

In text compression, the language elements that may constitute a dictionary may be a natural unit like a word or it may be some arbitrary fixed or variable length character string such as phrases, fragments(n -grams) and messages. Each of these language elements has its own advantages and disadvantages as codeable elements.

When constructing the dictionary, a choice exists between fixed-length and variable length codeable elements. In the case of fixed-length elements, the language elements may be single characters, character strings or bit patterns

of fixed-length. Usually, a variable-length code, such as Huffman code, is assigned to the fixed-length elements. Since processing variable-length bit strings requires more processing time than fixed-length elements, variable-length codes are used only when storage space, not processor time is the main concern.

The use of variable-length language elements for compression is common. They may be selected to be words [Heaps,72] or some other convenient set of variable-length character strings. Word-stems and suffixes have been suggested as language elements [Schue,63], [White,67], as well as a combination of words and word phrases [Schwa,67].

After the set of codeable elements are identified, codes must be assigned to construct the dictionary.

Textual words are natural units considered for encoding. In general, the distribution of words in written texts follow the Zipf's Law. In order to obtain optimum compression, variable-length codes have been considered for encoding the textual words. However, it is well known that variable length codes are difficult to manipulate with the conventional computer architecture. Although word coding with variable-length codes provides optimum compression it has some disadvantages. A disadvantage of word encoding is that it requires a large dictionary that is highly data base dependent and grows with each update. Hence encoding and

decoding are slow in operation and expensive in storage requirements.

A compromise between variable-length codes (variable at bit level) and fixed-length codes, is the use of restricted variable-length codes. This coding scheme was proposed by [Heaps,70] and is based on the frequency of occurrence of words in a text data base. The coding scheme may be described as follows:

Let N_1, N_2, \dots, N_k be a sequence of increasing positive integers, and suppose the most frequent $2^{(N_1-1)}$ terms in the dictionary are assigned different codes, each of N_1 bits, of which the first bit is always set to 1. The next most frequent $2^{(N_2-2)}$ terms in the dictionary are assigned different codes of length N_2 bits in which the first bit is always 0 and the (N_1+1) th bit is always 1. The codes will be assigned in this fashion and there may be up to $2^{(N_1-1)} + 2^{(N_2-2)} + \dots + 2^{(N_k-k)}$ such dictionary terms. As mentioned by [Heaps,70], although restricted variable-length codes allow good overall compression of data, they have several disadvantages. These disadvantages are the same as for variable-length codes since these codes are applied to words. One of them is for the same reason as for variable-length codes that is the coding of infrequent terms there is no saving in storage.

Fixed-length codes are easier to manipulate during decoding and one tends to use them. However, Information

Theory requires that in order to obtain optimum compression with fixed-length codes, the associated language elements must occur with equal probabilities [Reza,63]. Therefore, a set of dictionary elements which occur with "approximately" equal frequencies in the data has been considered. Obviously, natural units like words are ruled out by this approach, since their distribution is far from being uniform [Zipf,49]. Even arbitrary fixed-length character strings are eliminated, since it is impossible to find such a set with equal frequencies. Hence, the dictionary will be a set of linguistically meaningless, variable-length character strings. They can be selected to possess the property of equifrequency [Clare,72].

Coding methods with the use of fragments as language elements are attractive because it allows the use of fixed-length codes and reduces considerably the size of the dictionary. This method simplifies the operation of decoding since the code may be used as an index into a dictionary of fragments. Several methods of coding such as Minimum Space, Longest Fragment First and Longest Match algorithms have been studied in this context [Schue,74].

SELECTION OF CODEABLE ELEMENTS

The selection process for codeable elements is a preprocessing phase. This process consists of scanning a representative text sequentially and determining the

language elements which will constitute the dictionary. From the implementation point of view, it is advantageous to restrict the maximum length of a dictionary element to be compatible with the computer's word length. The frequency of occurrence of the codeable element in the data base should exceed a certain threshold value for it to be considered as a dictionary element. It is obvious that a character string or word which occurs only once or twice in the data base should not be included, since the savings in storage space would be insignificant.

In text compression, many types of codeable elements, such as words, fragments (text or word) and phrases or a combination of them, may be chosen to code a given textual data base. There is a problem in determining which type of codeable element would yield the best compression for a particular compression technique. The problem of finding a suitable dictionary for compression coding has been discussed by [Schwa,63], [White,67], [Heaps,72], [Clare,72], [Schue,73], [McCar,73], [Rubin,76] and [Schue,76]. In a text compression scheme the main concern is the creation of the dictionary.

MACHINE ARCHITECTURE FOR COMPRESSION

In the past, text compressor/decompressor modules have been implemented entirely in software. However, Lea [Lea,76], has considered the application of special purpose

hardware with an associative parallel processor, for text compression and decompression tasks. Time spent on compression and decompression routines implemented via hardware is smaller by a factor of 1000 than pure software implementation [Schue,78]. Lea's machine architecture is well suited for fragment encoding and decoding. The implementation of compression tasks via hardware has not been widely studied up to now but is an attractive solution for reducing the computer time taken by the compression and decompression routines.

APPLICATIONS OF TEXT DATA COMPRESSION

Text data compression has many areas of applications such as the following:

1) Business files

Compression of large business files has been discussed in [Ruth,72]. One characteristic of the business files (e.g. customer order record), is the large percentage of zeros and blanks contained in the records. Also, in business records, all characters of all fields are significant, therefore no loss or change of data can be tolerated. The data must be reconstituted exactly. Hence, compression techniques which achieve compression by permanently discarding portions of the data cannot be used for business files. These files are usually quite large and hence compression can reduce the storage and transmission

cost.

2) Compiler messages

Compilers include an extensive body of diagnostic messages. These messages have several common words and phrases, which could form the codeable dictionary. This approach was found to reduce storage space. A number of techniques have been used to minimize the amount of storage required by these compiler messages [Wagne,73], [Hahn,72]. A method of compressing diagnostic messages for compilers through selection of words or phrases is discussed in [Radha,77].

3) Transmission of computer programs

Computer programs, such as COBOL source files, contain some kind of redundancy. The language in which it is written has keywords which repeat quite often all through the program. These programs which are stored on secondary storage devices have to be transmitted either to the central processor or to several distant locations in a network organization. Compression of these programs is desirable to save transmission cost and storage. The compression of a COBOL source file has achieved a compression ratio of 5.91 and a storage saving of 82% [McCar,73].

4) Addresses and proper names

Given names are common items of information in many information storage and retrieval systems. Names and addresses are usually part of customer records in business files. This information if compressed would yield savings in storage space. Walker [Walke,69] has considered compression of names by x-grams.

5) Library documents

Library documents' data bases are large in size and economic storage of such data bases is a problem. Different techniques for compressing such files have been discussed in the past literature [Heaps,74], [Schue,75].

TESTING AND COMPARISON OF DATA COMPRESSION METHODS

Several types of compression methods have been discussed in the past and the performance of each method has been studied and compared in [Arons,77], [Schue,74]. The performance of data compression methods, differs both in terms of percent reduction and computation time. Aronson has compared the performance of several compression techniques only in terms of percent of reduction. He did not consider the computation time. It is to be noted that more complex methods that require more computation time can not be equated to simpler methods, such as the digraph methods of [Snyde,70].

Schuegraf [Schue,74], has studied the performance of three fragment oriented compression techniques such as Minimum Space, Longest Fragment First and Longest Match algorithms. He found that all three algorithms perform equally well as far as compression ratio is concerned. In terms of computing time, the longest match algorithm was found to take the least amount of computer time due to its simplicity.

2.4 SIGNAL DATA COMPRESSION

Electrical signals whose amplitudes vary as a function of time have been widely studied by Communication Engineers. This type of data are encountered in the transmission of speech signals in telephony and video signals in television or facsimile. By means of linear transforms, like Fourier Transform, the signals may be transformed to frequency domain and analysed for transmission or processing. Research in compression of signal data dates back to the early 1950's and an excellent collection of the results are found in the Proceedings of the IEEE, March 1967, a special issue on Redundancy Reduction.

In television and speech applications, the signals obtained from transducers are analog signals. The analog signals are converted to digital form by sampling and quantization for transmission and processing. Basic concepts and problems in digitization and transmission of

digital signals are found in [Kall,75]. Studies in signal data compression have been in two major directions: a) Reduction of band and width, (b) susceptibility of the reduced signal to errors in transmission [Gucci,67]. In data communications, reduction of data or data compression leads to bandwidth reduction and hence the terms data compression and bandwidth compression are used more or less synonymously. Several doctoral theses have been written in the area of bandwidth compression of television signals [Cherr,63].

One of the popular methods used for signal data compression is "run-length coding". A detailed study of the theory of run-length codes, standard run-lengths and their relations to run-length probabilities are found in [Cherr,63]. Run-length coded data is more susceptible to noise than the uncoded data and the "run" information has to be protected against the channel noise. Two-tone or black and white images with only two levels of brightness are common in facsimile transmission of hand-written business documents, engineering drawings, newspaper pages, weather maps and finger print cards. Efficient coding techniques for compression of two-tone images have been studied by [Huang,77].

Another commonly used method for signal data compression is based on the principle of "Differential Pulse Code Modulation or DPCM". Suppose the signal is changing slowly

with time. Then the difference between two successive signal samples will be smaller than the absolute value of the signal. Let $x(1), x(2), \dots$ be the signal values at the sampling times $t = 1, t = 2$ etc.

$$\delta = x(n+1) - x(n).$$

For slowly varying signals

$$|\delta| < |x(n)|$$

$$|\delta| < |x(n+1)|$$

By transmitting δ or some approximations to δ , which will require fewer number of bits than $x(n+1)$, data compression can be achieved. In the "predictive techniques" of DPCM, $x(n+1)$ is predicted as a function F of the past values:

$$\hat{x}(n+1) = F(x(n), x(n-1), x(n-2), \dots)$$

$$\delta = \hat{x}(n+1) - x(n+1)$$

The predictor function F is known a priori both to the transmitter and the receiver [Habib,75].

In almost all methods of signal data compression, the feedback information has been used to obtain a new class of "adaptive methods". For example, the use of adaptive DPCM for speech data compression is discussed in [Rosen,74], and other adaptive methods are surveyed in [Mark,74] and in

[N611,75].

In comparing signal compression systems, the ratio of the power of the signal-to-quantization noise has been used as a main criterion [Rabin,78]. However, the compression ratios, in the order of 3.3 to 1 or 5 to 1, have been reported in some cases [Huang,77]. In all these applications "real-time" needs have been assumed. This implies that signals from transducers are encoded by the transmitter without "much" delay and decoded by the receiver at an acceptable rate. Signal speeds of a few kilo Hertz in speech signals and a few megga Hertz in visual signals are common. The techniques of run-length coding and DPCM have been applied also to text data compression [Smith,76].

Cherry concluded in [Cherr,63] that compression of signals is very largely a problem of technique, of instrumentation, rather than a theoretical problem of principles. In mass communication of visual and audio signals, the receivers are with the common men. Hence, the receivers must be inexpensive, reliable, and simple to operate. Developments in LSI technology offer newer solutions to these problems [Harte,75], [Guthi,78].

2.5 IMAGE DATA COMPRESSION

An image is a more general data type than alphanumeric texts or signals. Different types of images are encountered

in computer data processing satellite images, photographs of objects and scenes, newspaper pages with advertisements, X-rays, engineering drawings, weather maps etc. A class of images, sometimes referred to as "pictures", consist entirely of straight lines or curved lines. Engineering drawings, flowcharts of computer programs, and contour maps are some examples of pictures. They may be stored by describing the boundary lines. In general, an image can be represented by a two or multidimensional matrix of data values where each value is referred to as a "pixel" or picture element. A pixel may be binary valued or integer valued. In the following, a brief account of image data compression is given and detailed surveys can be found in [Rosen,76], [Habib,75], [Pratt,78], [Radha,79].

Techniques in image data compression which is commonly known as "picture coding" can be classified into the following four categories [Radha,79]:

- a) Transform Methods - based on some mathematical transforms for which generally inverse transformations exist.
- b) Encoding Methods - based on run-length codes, PCM techniques, predictive coding and entropy codes like Huffman code.
- c) Compression by Description of Images - in this class of methods, images are described syntactically or otherwise, by means of the constituent parts. They are well suited for pictures.
- d) Specialized Techniques - they depend on the nature of data. For example, the 2-to-1 line interlacing used in conventional television gives "video frames" are to be transmitted successively (about 30 frames of 525 scan lines per second), the transmission of

inter frame difference, instead of the entire frame, leads to data compression [Rose,77].

Transform Methods: This class of methods employs one or more transformation techniques. The transformed data is characterized by a set of parameters. By transmitting a selected subset of these parameters, data compression is achieved.— The following transforms have received a major attention in image data compression and an extensive bibliography on this topic is provided in [Radha,80].

- (1) Fourier Transform
- (2) Karhunen-Loeve Transform
- (3) Hadamard Transform
- (4) Haar Transform
- (5) Cosine Transform
- (6) Slant Transform

Encoding Methods - Run-length codes, Huffman code, and DPCM (Differential Pulse Code Modulation) have been applied to signal data as well as to image data. In fact, for communication purposes an image is scanned by a scanner and converted into signal data. Then what is the difference between signal data and image data? The main difference lies in the two dimensional (multidimensional) nature of images. In a time-varying image signal, there are two coordinates: time and space [Pease,71], [Pratt,78, page 602]; and this gives an additional degree of freedom for data compression.

Compression by Description of Images - Suppose the users' needs in image processing are clearly identified. Then, those details of the images can be extracted and coded for transmission, and that would require far less number of bits than the whole image. However, the discarded details of images can not be recovered and in this sense these methods are called "information destroying" compression schemes.

Extraction of predetermined features and their use to represent a given image is an example of this method. Another example is as follows: In general, the images from earth resources satellites are classified by means of automatic classification methods for further use. Instead of the original raw images, one could classify them for a specified application, code the classified images and transmit such coded images. In coding of classified images for land use study, boundary tracing methods have been found quite useful [Amido,71]. Feature extraction and classification methods hold great promise for data compression of satellite images, but have not been practical up to now. With the advent of LSI (large scale integration) technology and microprocessors, it might be possible in the future to store different methods of feature extraction or automatic classification on-board a satellite and to select one or more of them from a ground control to match the user needs. A summary of the application of the various compression methods are shown in TABLE 2.1.

TABLE 2.1

	DICTIONARY BASED	RUN-LENGTH	DIFFERENCING	TRANSFORM
SIGNALS	NOT APPLIED	COMMONLY APPLIED [PRATT, 67] [CHERR, 63]	COMMONLY APPLIED [PRATT, 67] [ROSEN, 74]	APPLIED [RABIN, 78]
LINEAR STRINGS	COMMONLY APPLIED [HEAPS, 75] [SCHUE, 74]	APPLIED [JEWEL, 76] [SMITH, 76]	APPLIED [DORON, 75] [SMITH, 76]	NOT APPLIED
STRUCTURED DATA	NOT APPLIED	APPLIED [HUANG, 77] [AMIDO, 71]	APPLIED [KRAUT, 7]	COMMONLY APPLIED [ROSEN, 76] [READY, 71]

2.6 Summary

In this chapter, brief accounts on compression of text, signal and image data have been presented. Certain techniques like run-length coding, Huffman codes, and differencing are common to all of them (see TABLE 2.1). At the same time, there are differences in the approaches to compression of text data on the one side and compression of signal or image data on the other side. Studies in the case of the former have been mainly to reduce the storage requirements; whereas that in the case of the latter have been to reduce the bandwidth (or time) requirements.

In the case of signal and image type data, the receiver characteristics and fidelity criteria play a major role in data compression. For example, in some cases the transform methods yield compression ratios in the order of 10 or so [Rosen, 76]. Since the compressed data is transmitted over "noisy channels", study of the susceptibility to channel errors is important. In text data compression these are either irrelevant or not considered in the past studies.

Let the term "composite data" refer to a combination of image and text type data. Composite data are not uncommon. They occur in facsimile transmission of newspapers; "yellow pages"; computer aided instruction with the TV receivers at home and so forth. In the past, while transmitting TV images, one could not afford to use data

compression methods, that would require complex decompression methods; because the TV receivers must be inexpensive. With the advent of microcomputers and their successful use in consumer electronics, such constraints do not exist anymore. On the contrary, when composite data becomes more popular than now, their compression for transmission and storage will also become more important.

CHAPTER III

THREE EXPERIMENTAL DATA BASES

In this chapter three data sets are described. They have been used for generating word fragments and for testing the coding algorithm described in this thesis.

The first two data sets consist of records which are represented by a pair $\langle \text{WORD}, \text{FREQUENCY} \rangle$. Each record in the third data set has one entry which consists of $\langle \text{WORD} \rangle$.

In the experimental data, the term WORD refers to an ordered sequence of alphabetic characters and FREQUENCY is the total occurrence of the word in a chosen textual data base.

3.1 DATA SET1

This data set is a portion of [Kucer,67] data which consists of 2500 most frequently used English words along with their frequencies selected from a text of 1 million words. Kucera's data set has a total of 752,178 word occurrences and consists of 2,991,940 characters.

The average word length of these 2500 "distinct words", that is no word is repeated, is 5.8 characters (denoted as AV_D) and the weighted average word length using

frequency as the weight is 4.6 characters (denoted as AV_N).

The characteristics of this data set are shown in TABLE 3.1 and the word length distribution is shown in FIGURE 3.1.

3.2 DATA SET2

This data set consists of 2500 most frequently used words chosen from the abstracts of the papers in CACM (Communication of the Association for Computing Machinery) published during the years 1968 to 1972.

The set of 2500 most distinct words in the data base gave rise to a total of 33,609 word occurrences. The 33,609 word occurrences accounted for 240,635 characters. The average word length of the distinct words is 6.9 characters and the weighted average word length is 7.2 characters.

The characteristics of this data set are shown in table 3.2 and the word length distribution is shown in FIGURE 3.2.

3.3 DATA SET3

The third experimental data set consists of 1200 distinct words selected from the street names of Montréal.

The average word length of the distinct words in this case is 7.9 characters. No frequency is associated with each word as the frequency information was not readily

TABLE 3.1

L	DISTINCT WORDS	WORDS OCCURRING IN FULL TEXT
2	34	169,362
3	144	207,362
4	430	142,812
5	475	86,320
6	480	55,677
7	427	46,016
9	193	16,672
10	2	167
TOTAL	2500	752,178

$AV_D = 6$ CHARACTERS

$AV_N = 4.6$ CHARACTERS

L = length of the word in characters

Curve A : DICTIONARY FREQUENCY

Curve B : TEXT FREQUENCY

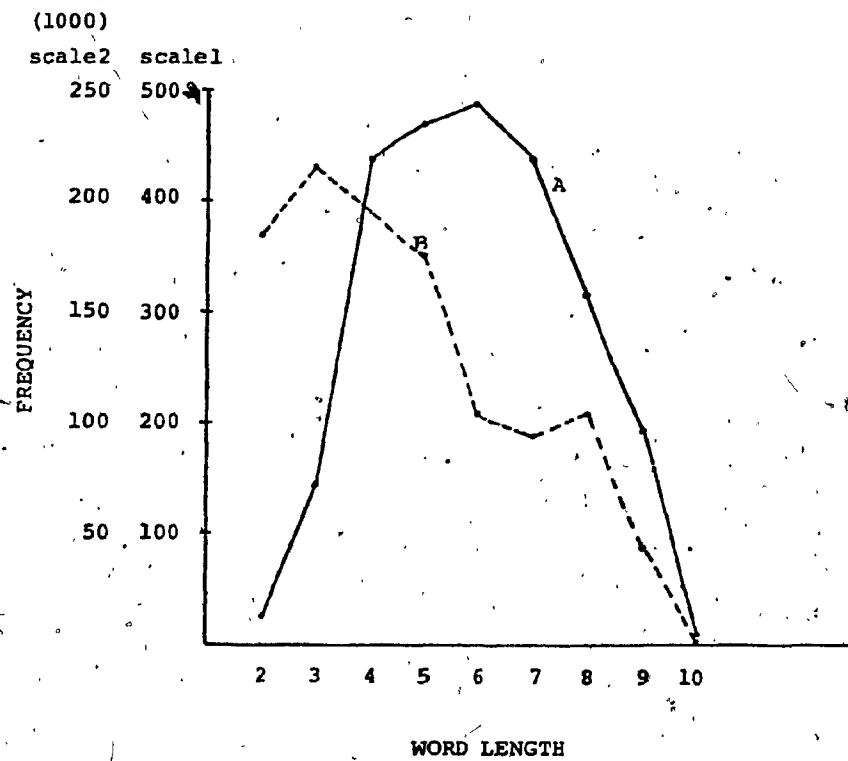


FIGURE 3.1 WORD LENGTH DISTRIBUTION FOR DATA SET1

NOTE: CURVE A CORRESPONDS TO SCALE1
CURVE B CORRESPONDS TO SCALE2

TABLE 3.2

L	DISTINCT WORDS	WORDS OCCURRING IN FULL TEXT
2	157	2035
3	157	1495
4	206	2790
5	269	2857
6	269	3754
7	332	4916
8	343	4985
9	282	3969
10	276	3275
11	108	2086
12	53	760
13	27	387
14	15	269
15	4	27
16	2	6
TOTAL	2500	33,609

$AV_D = 6.9$ CHARACTERS

$AV_N = 7.2$ CHARACTERS

L = length of the word in characters

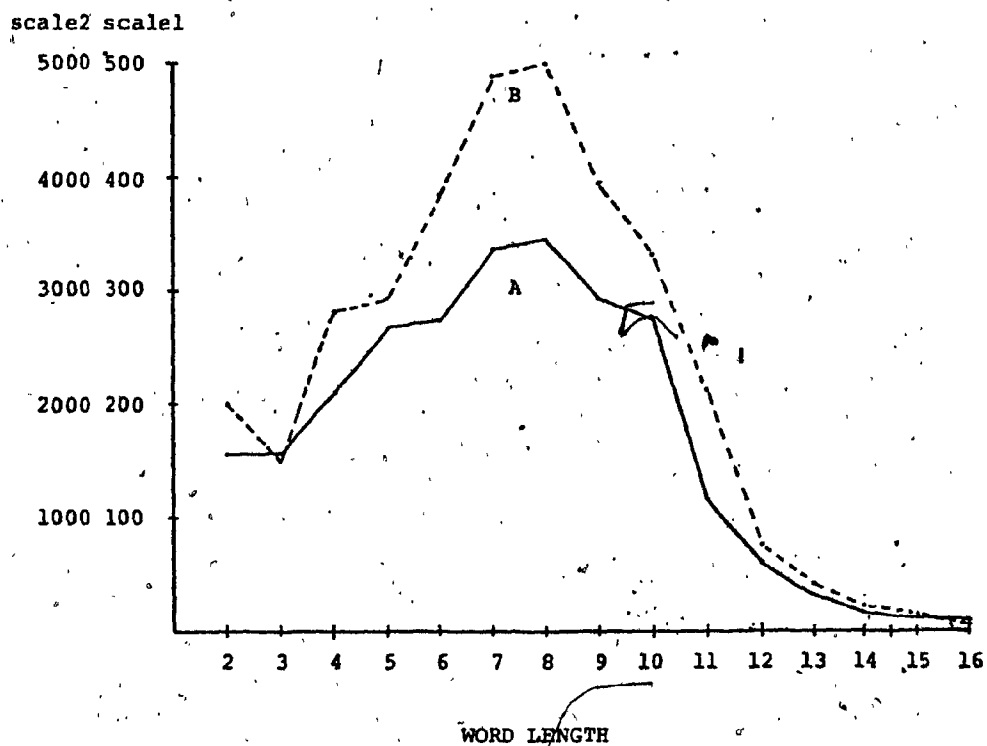


FIGURE 3.2 WORD LENGTH DISTRIBUTION FOR DATA SET2

NOTE: scale1 corresponds to curve A
scale2 corresponds to curve B

available.

The characteristics of the data set are shown in TABLE 3.3 and the word length distribution is shown in FIGURE 3.3.

3.4 A COMPARISON

It can be seen from the graphs that the word length distributions of the three data sets behave differently. Data Set2 has a higher word length ($AV_N = 7.2$) than Data Set1 ($AV_N = 4.6$), because words of length $L=7,8$ occur more frequently in Data Set2, while in Data Set1 words of length $L=2,4$ are more frequent. On the other hand Data Set3 has the highest $AV_D = 7.9$ of the three data sets since words of length $L=6,8$ are more frequent. Yet another fact is that for Data Set1 $AV_D > AV_N$ where as for Data Set2 $AV_D < AV_N$.

This can be explained by referring to the TABLE 3.1 and the TABLE 3.2. In Data Set1, those words of length less than AV_N occur more frequently than others in the text. Thus AV_N is smaller than AV_D . On the other hand in Data Set2, the situation is exactly the opposite. That is, the words of length less than AV_D ($L=2,3,4,5$) occur less frequently than words whose lengths are closer to AV_N ($L=6,7,8,9$). This fact is evident from FIGURE 3.4. Also in Data Set2, words of length greater than 10 characters account for about 10% of the word occurrences whereas in Data Set1 they are insignificant.

TABLE 3.3

L	DISTINCT WORDS
2	7
3	13
4	58
5	154
6	245
7	238
8	230
9	140
10	70
11	29
12	9
13	7
TOTAL	1200

 $AV_D = 7.9$ CHARACTERS

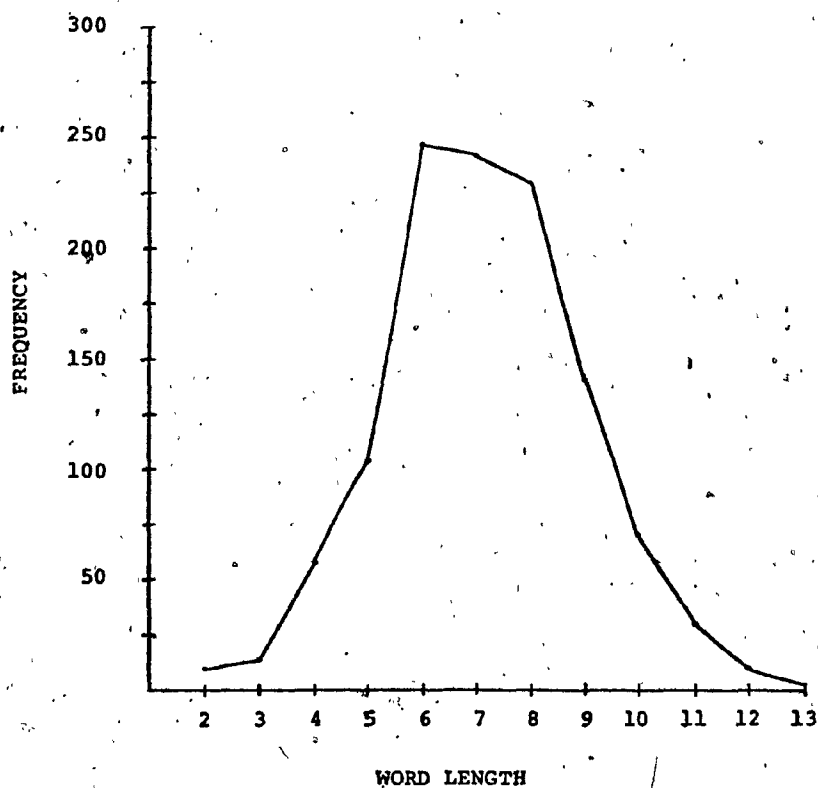


FIGURE 3.3 WORD LENGTH DISTRIBUTION FOR DATA SET3

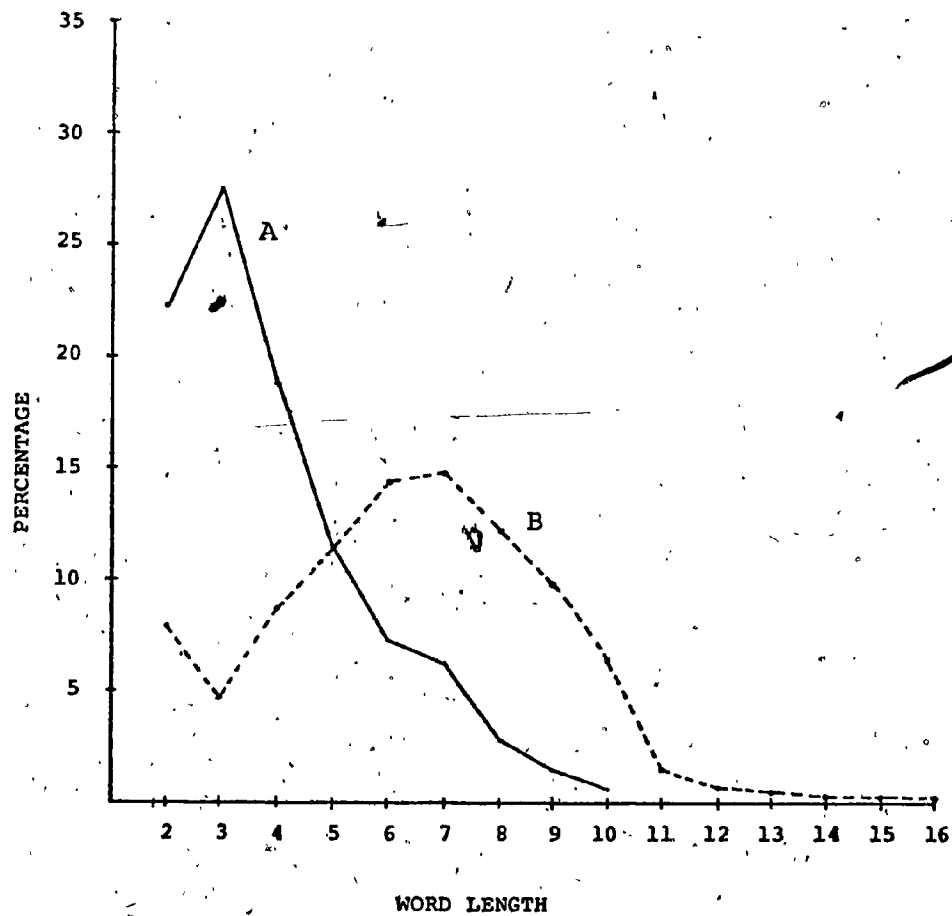


FIGURE 3.4

NOTE: Curve A corresponds to DATA SET1
Curve B corresponds to DATA SET2

CHAPTER IV

A DICTIONARY STRUCTURE FOR TEXT DATA COMPRESSION

In this chapter, three coding algorithms for text compression are described. Their merits as compression methods is discussed along with their disadvantages. Their performances are compared and discussed.

4.1 Dictionary Based Text Data Compression

Alphanumeric texts occur in several data processing applications such as Management Information, Office Automation, Library Information Retrieval and other Information Retrieval Systems. Most of the applications of text compression aim primarily at reducing the computer storage requirements or the transmission time requirements over long distance communication lines.

Data compression methods suitable for textual data are of special interest and a detailed survey of compression methods for non-numeric data bases has been described by [Schue,76]. A widely used method of text compression scheme is based on encoding with the help of a dictionary.

The heart of such a compression scheme is a set of previously chosen language elements, which constitute a

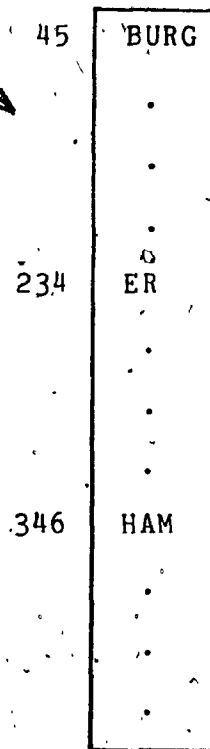
dictionary. In general, the dictionary is ordered so that each element can be uniquely identified by its ordinal position in the dictionary. FIGURE 4.1 shows the basic structure and operation of such a system.

A record to be compressed, be it a sentence or word, is scanned and all elements of the dictionary contained in the record are identified. These matching elements or substrings may constitute the whole record or only part of it. If the substrings represent only part of the record this is then known as "partial coding". The FIGURE 4.2 is an example of such partial coding. For each of the matching dictionary elements, its corresponding code is substituted. The code is shorter in length than the coded element. The code may be chosen to indicate the position of the element in the dictionary for easy decoding. The resulting record is called the "compressed record" and the collection of all these records is the "compressed data base".

Decoding is the reverse of the coding process and will reproduce the original record, provided the same dictionary is used for decoding. For each code, the corresponding dictionary element is substituted to generate the original string.

Selection of the codeable language elements is an important aspect of text compression. Such language elements may be chosen as single characters, words, or

HAM/BURG/ER
input record



346/45/234
output record
(Compressed)

FIGURE 4.1 DATA COMPRESSION SYSTEM

CON/T/RO/V/ERS/IAL

) INPUT RECORD

20

CON

150

ERS

20/T/281/V/150/230

Output RECORD

(Compressed)

230

IAL

281

RO

FIGURE 4.2 PARTIAL CODING

variable length character strings. Several authors have considered the use of common words or phrases as codeable elements for compression.

White, [White,67] has studied compression of printed English text by encoding the most frequently occurring words and phrases, in addition to letter and letter sequences. In his encoder, he obtains the longest possible match between the input text and dictionary entries, and encodes it with fewer bits than the entry itself. Extensive experiments with this scheme indicate that 50% compression can be obtained for general English language texts.

In order to reduce the storage requirements of an exhaustive word coding scheme, [Schwa,63] has designed a split dictionary which contains, not only frequent words, but stems and suffixes from which the less common words can be synthesized.

Columbo, (Colum,69) has discussed the use of word fragments in computer based retrieval systems. To provide the information necessary to make such predictions as to what are the consequences of choosing a particular word fragment as a search term, he developed a dictionary which consists of an alphabetical listing of word fragments, together with the set of words in which each word fragment occurs (parent words). Word fragments were generated by means of a computer program which derives all possible

fragments of four or more characters from the parent word. Once the fragments were generated and sorted, those which were contained in 10 or more parent words were discarded. With a few more selection rules, a reduction of 75% was achieved in the size of the dictionary produced. He noticed that a considerable number of fragments occurred in single words, that is were unique that word. Therefore a second dictionary was created. This dictionary contained words listed alphabetically together with the fragment which uniquely represents them. This pair of dictionaries enabled the selection of appropriate fragments for retrieval of specific words.

Clare, [Clare,72] has proposed the use of variable length character strings as codeable elements. These strings are not limited to be substrings of words, provided they are part of a record or sentence. For this reason, the fragments are called "Text Fragments". Text fragments may contain embedded blanks. Clare et al place an additional restriction on the text fragments that are to be included in a dictionary, namely that the dictionary fragments should occur in the data base with approximately equal frequencies. In their experimental study of a Chemical Titles data base they attempted to identify variable length equiprequent character strings which, in contrast to those treated by [Walke,69], may extend across word boundaries. For a selected threshold value for the frequency, they increase

the fragment lengths until frequencies of all fragments fall below the threshold. Such a procedure was used in an attempt to replace the Zipf distribution of words by a more equiprobable distribution of fragments. They propose that the resulting fragments be used for purposes of indexing, retrieval, and compression of document citations. However, it is not advantageous to replace a fragment of low frequency by a code because of the fairly high cost of including it in the dictionary which must be kept in an easily accessible and hence more expensive storage medium.

A method of selection of equifrequent fragments has been described by [schue,73]. Use of equifrequent character strings has some merits. As has been observed by these authors, equifrequency-property i) allows maximum compression to be achieved by use of fixed length codes, ii) permits the use of a fixed size dictionary and allows more freedom of design.

Three Algorithms for Encoding:

Three algorithms for compression of textual data bases are described in the following.

Minimum Space algorithm (MS)

This algorithm uses the optimum number of fragments possible and thus achieves the highest degree of compression

[Wagne,73].

Assume that the r letters in a record are numbered consecutively from 1 to r and that each letter is represented by a node in a graph. Let node $r+1$ represent a blank space. A k -letter fragment that begins at letter i may be considered as constituting a link from node i to node $k+i$ as shown in FIGURE 4.3. Assign each link a cost of 1 regardless of the value of k . All links are regarded as directed with an arrow that always points from the lower numbered node to the higher numbered one. To locate all the links in the graph of $r+1$ nodes it is necessary to determine all the fragments that are completely contained in the record. Each link represents one possible codeword, and the problem of finding the shortest path, that is the least number of links, between node 1 and node $r+1$ is considered equivalent to the problem of finding the minimum number of codewords to represent the coded record. This is illustrated in FIGURE 4.3 in which the word ILLUSTRATED is subdivided in three different ways to correspond to shortest paths of six links. It may be noted that the inclusion of single letters in the dictionary guarantees at least one path between start and end node, so that a solution always exists.

The disadvantage of this algorithm, however, is its backtracking behaviour and the resulting complexity that results in long execution times.

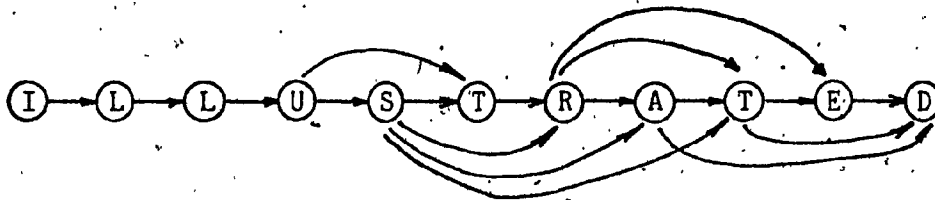
WORD:

ILLUSTRATED

Fragments of length ≥ 2 ($t=10$)

UST, STR, STRA, STRAT, RAT, RATE, ATED, TED

Associated graph



Possible solutions

I/L/L/UST/RATE/D

I/L/L/U/STRA/TED

I/L/L/U/STR/ATED

FIGURE 4.3 GRAPH REPRESENTATION OF MS PROBLEM

Longest Fragment First algorithm (LFF)

This algorithm proceeds in a manner similar to that of the MS algorithm in that all fragments contained in the record must first be located and stored. After all fragments of the record have been stored the LFF algorithm recursively removes the longest fragment and deletes all the subfragments which overlap the one removed. The process is continued until the record is empty. The removed fragments are coded to form the compressed record. For example, using the same fragment dictionary as in FIGURE 4.3, the LFF algorithm subdivides the word ILLUSTRATED into the seven fragments

I/L/L/U/STRAT/E/D

1 2 3 4 5 6 7

whereas only six fragments result from application of the MS algorithm. In order to find the longest codeable fragments, this algorithm locates all the codeable fragments of the input record. Hence it requires long search times and creates large auxiliary tables.

Longest Match algorithm (LM)

The third, and the simplest is, the "longest match" algorithm. Starting with the first character of a record all fragments that consist of this character and subsequent

characters of the record are found. The maximum length of chosen fragments starting at letter one can be p (the maximum length of fragments in the dictionary). Let the longest matching fragment of length less than or equal to p be of length k . This fragment is selected for compression of the record and the selection process is repeated starting at position $k+1$ and is continued until the end of the record is reached. In the instance of the word ILLUSTRATED the LM algorithm selects the six fragments

I/L/L/UST/RATE/D

1 2 3 4 5 6

Since there is no need to look for all the fragments which start at letters $2, 3, \dots, k$ of the record, the coding process is relatively faster. Furthermore, it is not necessary to store all subfragments during coding, hence large auxiliary tables are not required and so less core storage is needed.

As mentioned by [Schue,74], the coding times of the three algorithms exhibit considerable differences. The MS algorithm has the worst time complexity because of its backtracking nature, while the LFF algorithm is somewhat better. Of the three algorithms, the LM algorithm takes the least amount of time for encoding. The differences in compression obtained by the three algorithms given above are found to be hardly significant and hence the coding time may be used as

the criterion for selection of an algorithm. The LM algorithm was found to take approximately one third of the time of the other two algorithms. Also it resulted in about 5% less in saving of the storage space when compared to the other two algorithms. Furthermore, the LM algorithm is easy to implement. Considering all these factors, the LM algorithm has been chosen for the study reported in this research.

4.2 A Dictionary Structure for Data Compression

The following section describes a two-level dictionary organization which is used in this thesis. The selection of the codeable elements for the dictionaries is also explained in this section.

Dictionary Organization:

This dictionary organization was initially proposed by [Heaps,77]. For this research, a two-level hierarchy was used to organize the fragment dictionary. The first level dictionary, DICT1, holds the most frequent and hence the most accessed $2^8 = (256)$ fragments each of which is assigned an 8-bit fixed-length code. The second level dictionary, DICT2, holds the next most frequent $2^{16} = (65,536)$ fragments and their corresponding 16-bit fixed length codes. The dictionaries DICT1 and DICT2 may be arranged in alphabetical order.

The trie organization (Knuth,71) of the two dictionaries has been used in this thesis for fast searching. FIGURE 4.4 depicts such an organization. All fragments that start with the same first letter are grouped as one entity and they are stored consecutively in the memory. There are 26 such entities, each corresponding to the letters A,B,C,D,...,Z. The "table of pointers" in FIGURE 4.4 stores the addresses of the first fragment of each of these entities. There is a one-to-one correspondence between the entries of the table of pointers and the possible initial characters of the fragments, namely A,B,C,D,...,Z. Hence it is no longer necessary to store each initial character with the fragments.

The search is first performed in DICT1 and if it fails, then DICT2 will be searched. If a fragment from DICT1 or DICT2 is matched with the input string, then it is replaced by the corresponding code.

Creation of Dictionaries:

Selection of fragments to set up the dictionaries is normally done after some statistical analysis of a sample data. The problem of selecting a suitable dictionary has been discussed extensively in the literature [Schwa,63] and [White,67].

Codeable fragments can be fixed or variable in length and their codes may also be fixed or variable in length. Each

TABLE OF POINTERS

	DICT1	DICT2
A	1	257
B	3	259
.	.	.
R	150	2003
.	.	.
Z	256	6199

FRAGMENT	CODE
AB	1
ACT	2
BA	3
.	.
.	.
RUN	150
.	.
ZE	256

DICT1

FRAGMENT	CODE
ADD	257
ADDR	258
BAC	259
BET	260
.	.
.	.
RAT	2003
.	.
.	.
ZA	6199
ZO	6200

DICT2

FIGURE 4.4 TRIE ORGANIZATION OF DICTIONARIES

- The underline characters are not stored

of these four combinations has its own advantages and disadvantages. The four schemes are discussed in [Schue,78].

For the purpose of this study, variable length word fragments which are also referred to as "n-grams" were generated. A fixed-length code that relates fragments to their position in the dictionary has been employed. Using the Data Set1, word fragments of length $L=2,3,4,\dots,7$ were generated and their frequencies of occurrence in the data base was accumulated. They were then ordered in decreasing order of the frequency and the 6200 most frequent n-grams were chosen to constitute the dictionaries used in this thesis.

4.3 On the design of the dictionary structure

The storage requirements of DICT1 and DICT2 and accesses to them are discussed in this section, together with the effect of the size of DICT1 on the compression scheme.

Storage analysis of dictionaries:

Each word in Data Set1 was partitioned into word fragments of length $L=2,3,4,\dots,7$. In the case where fixed-length memory cells are used to store fragments, one must decide on the optimum, L' , where fragments of length smaller than L' will be padded with blanks.

The following is an analysis to determine for each cell size how much storage it would take to store all the fragments and how much storage would be wasted due to padding.

Assume a_1, a_2, \dots, a_k denote the letters of a word of length k . The number of n -grams in a k -letter word is given by

$$p = k - n + 1 \text{ where } n \leq k$$

As n becomes larger, the number of n -grams p decreases for a given word.

Let $W_2, W_3, W_4, \dots, W_7$ denote the total number of bigrams, trigrams, tetragrams heptagrams respectively in the selected dictionary. If we were to use a one-byte cell to store the dictionary entries, then the total space required to store the fragments is given by:

$$T_S = 2W_2 + 3W_3 + 4W_4 + 5W_5 + 6W_6 + 7W_7 \text{ cells}$$

There would be no waste of space and the entries are variable in length. Suppose 2-bytes per cell is used, then the total space used is

$$T_S = W_2 + 2W_3 + 2W_4 + 3W_5 + 3W_6 + 4W_7 \text{ cells}$$

and the total amount of wasted space is:

$$W_S = W_3 + W_5 + W_7 \text{ bytes}$$

For a 3-byte cell the above formulae become:

$$T_S = W_2 + W_3 + 2W_4 + 2W_5 + 2W_6 + 3W_7 \text{ cells}$$

$$W_S = W_2 + 2W_4 + W_5 + 2W_7 \text{ bytes}$$

$$= (W_2 + W_5) + 2(W_4 + W_7)$$

The total space used T_S and the wasted space W_S in general would be:

$$T_S = \sum_{i=2}^n \lceil i/c \rceil * W_i$$

where c is the cell size and $\lceil i/c \rceil$ is the smallest integer greater than or equal to i/c . The unused storage due to padding is given by:

$$W_S = \sum_{i=2}^n (c - i \bmod c) * W_i$$

The utilization of space U is defined as follows:

$$U = (T_S - W_S) / T_S$$

From TABLE 4.1 it can be seen that a cell size of 2 or 4 gives the least wastage in storage. Therefore, a logical choice for the size of a dictionary entry would be one of these two. Another reason for this choice could be that most computers have 16-bit or 32-bit words and so the implementation would be easier. The data used to calculate TABLE 4.1 is shown in TABLE 4.2.

For efficient encoding, it is important to minimize the number of accesses to the dictionaries. This minimization

TABLE 4.1

CELL SIZE	SPACE REQUIRED	WASTED STORAGE	UTILIZATION
1	69521	0	100
2	76940	1800	98
3	84675	3600	96
4	90444	1800	98
5	95235	5400	94
6	97326	9000	91
7	100947	12600	88

TABLE 4.2

n=	number of n-grams in dict1 dict2	number of their occurrences in data set 1
2	362	2,239,762
3	2055	1,487,584
4	3814	904,768
5	3564	538,757
6	2826	297,444
7	1800	151,508

might require some compromise in storage.

Let p_2, p_3, \dots, p_7 be the frequency of access of bigrams, trigrams, tetragrams, ..., hepta-grams respectively, then the total number of accesses required for a cell size c is

$$T_A = \sum_{i=2}^n \lceil i/c \rceil * p_i$$

The results are shown in TABLE 4.3. The number of accesses and the storage required for different cell sizes are plotted in FIGURE 4.5.

Study of the effect of the DICT1 SIZE:

The use of two-level dictionaries DICT1 and DICT2 has the same underlying principle as the virtual memory concept, a topic well discussed in the literature [Denni,70], with the exception that elements in memory level M_1 do not appear in memory level M_2 . The gross performance of a virtual memory system that employs a two-level memory hierarchy, M_1 and M_2 , depends on several factors such as:

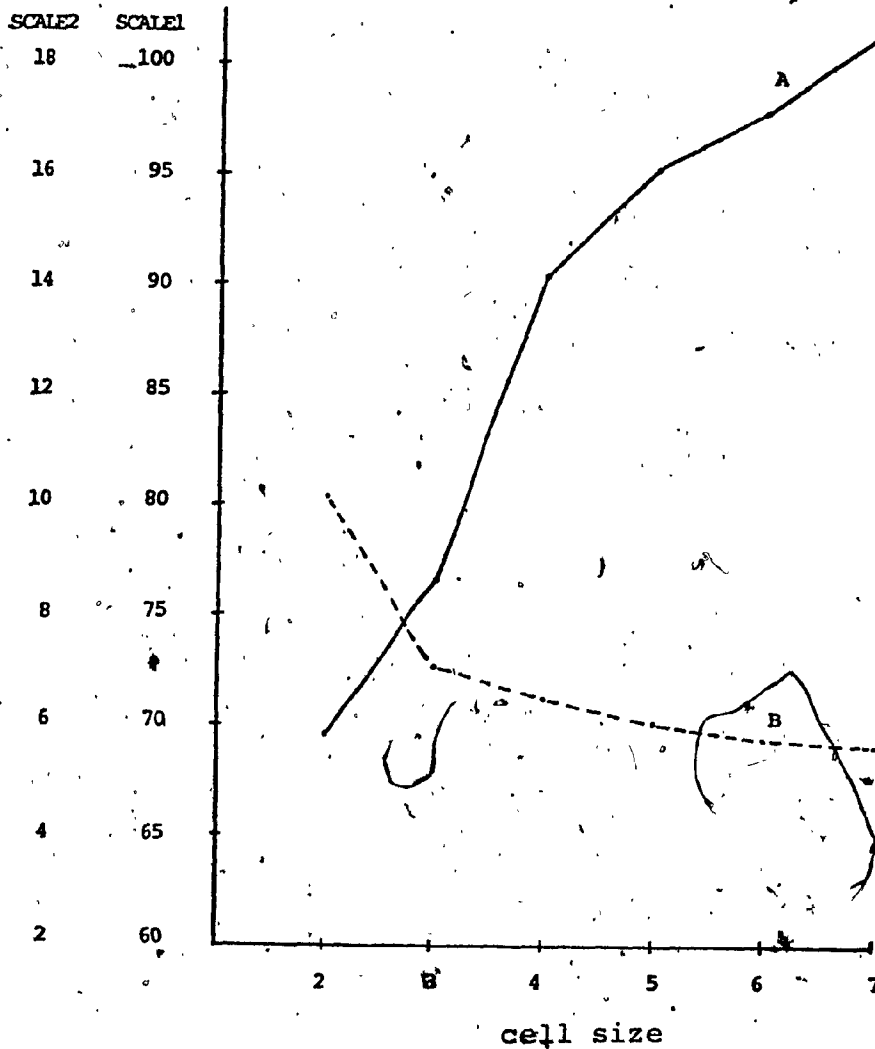
- 1) memory access time of each level
- 2) storage capacity of each level
- 3) hit ratio, H

Let $(N_1 + N_2)$ be the number of data accesses considered, of which N_1 be the number of times the memory level M_2 alone is accessed, then H is given by:

TABLE 4.3

DICTIONARY ACCESSES REQUIRED

CELL SIZE	NUMBER OF DICTIONARY ACCESSES REQUIRED
1	18,100,353
2	10,139,101
3	7,663,808
4	6,607,532
5	6,068,775
6	5,771,331
7	5,619,823



NOTE: CURVE A CORRESPONDS TO SCALE1
 SCALE1 denotes the storage required (1000's)
 CURVE B CORRESPONDS TO SCALE2
 SCALE2 denotes the number of accesses required (1000's)

FIGURE 4.5

$$H = N_1 / (N_1 + N_2)$$

An experiment was performed to determine the size of DICT1 for the best performance. The two fragment dictionaries DICT1 and DICT2 (refer to section 4.2) were used in this experiment and the following factors were studied:

- (a) hit ratio, H
- (b) average processing time per word
- (c) compression ratio

To determine the hit ratio, H, the $(N_1 + N_2)$ accesses were generated using Data Set1. The following illustrates the different values of N_1 and N_2 for different sizes of DICT1:

Size of DICT1	N_1	N_2	H
128	904911	213978	81
256	1063673	94487	92
512	1086293	30279	97
1024	1022194	8376	99
2048	949191	2754	99

It is to be noted that the fragment dictionaries as well as the accesses were generated from Data Set1.

Using Data Set1, the average processing time per word was calculated as explained below.

Let W_T be the total number of words in the Data Set1, to be coded, and T_i be the total processing time to code word i . Then the average processing time per word is defined as follows:

$$T = \frac{\sum_{i=1}^{W_T} T_i}{W_T}$$

The following illustrates the values obtained for different sizes of DICT1, with $W_T = 752178$.

Size of DICT1	T_i (MSEC)	T (MSEC)
128	3470753	4.61
256	2360650	3.14
512	2583802	3.44
1024	3763848	5.00
2048	5887457	7.83

msec = milliseconds

The compression ratio, α , was determined as follows:

$$\alpha = \frac{\text{storage for uncoded data}}{\text{storage for coded data + dictionaries}}$$

The results obtained are shown in TABLE 4.4.

To study the effect of the fragment dictionaries on a different data set, the same factors as mentioned above, were computed for Data Set2. These results are also tabulated in TABLE 4.4. It is to be noted that the poor results in the latter case are due to the fact that the fragment dictionaries were generated from Data Set1.

From TABLE 4.4 we observe the following:

- (1) As the size of DICT1 increases, the hit ratio increases as in cache memories [Hayes,78].
- (2) The compression ratio increases marginally as the size of DICT1 increases. This is due to the coding procedure and can be explained by the following example.

Suppose the word to be coded is the word C O M P U T E R. The two dictionaries DICT1 and DICT2 contain the following fragments:

DICT1	DICT2
CO	COM
TER	COMP
UT	UTER

TABLE 4.4

		SIZE OF DICT1				
		128	256	512	1024	2048
HIT RATIO	DATA SET1	81	92	97	99	99
	DATA SET2	69	83	93	96	98
AVERAGE PROCESSING TIME/WORD (msec)	DATA SET1	4.61	3.14	3.44	5.00	7.83
	DATA SET2	12.62	7.96	6.52	8.60	13.64
COMPRESSION RATIO	DATA SET1	1.96	2.10	2.32	2.57	2.80
	DATA SET2	1.43	1.50	1.62	1.78	1.96

Using the LM algorithm, the coded word would be

CO/M/P/UT/E/R

Now, let's increase the size of DICT1, so some fragments from DICT2 will become part of DICT1.

DICT1	DICT2
CO	COMP
COM	
TER	
UT	
UTER	

Then the coded word becomes COM/P/UTER.

As we can see, more characters are coded than the one previously. The reason is, as the size of DICT1 increases, longer fragments become part of DICT1 and therefore more characters are coded.

(3) The average processing time per word, T_p , increases as DICT1 increases except for $S_D = 128$ and $S_D = 256$. This factor is a complex function of the following:

- (a) fragment coding algorithm;
- (b) the corresponding program;
- (c) dictionary sizes;
- (d) the data structures and the methods used for searching;
- (e) the processor/memory/I-O speed of the computer used.

The distribution of these three factors is shown in FIGURE 4.6, FIGURE 4.7 and FIGURE 4.8.

Summary

From the results of the storage analysis shown in TABLE 4.1, we observe that a cell size of 2 or 4 yields the least amount of wasted storage space in the dictionaries. On the other hand, from TABLE 4.3, we see that a cell size of 4 does not yield the least number of accesses required to the dictionaries. However, the difference between the number of dictionary accesses for a cell size of 4 and 7 is negligible. Also the difference in wastage of storage space for a cell size of 7 is much larger than that of 4. Therefore, a cell size of 4 is an appropriate choice.

The effects of the size of DICT1, from TABLE 4.4 are clear. Obviously, it is desirable to have the hit ratio and the compression ratio as high as possible, but at the same time the average processing time per word should be as low as possible. A trade-off exists between the compression ratio, hit ratio and the size of DICT1 on the one side, and the average processing time per word on the other.

The experiment was performed with Data Set1 and Data Set2, and the same fragment dictionary was used for both. We can see that the results for Data Set2 are not as good as for Data Set1, the reason being that the textual content of Data Set2 is much different from that of Data Set1.

CURVE A CORRESPONDS TO DATA SET1
CURVE B CORRESPONDS TO DATA SET2

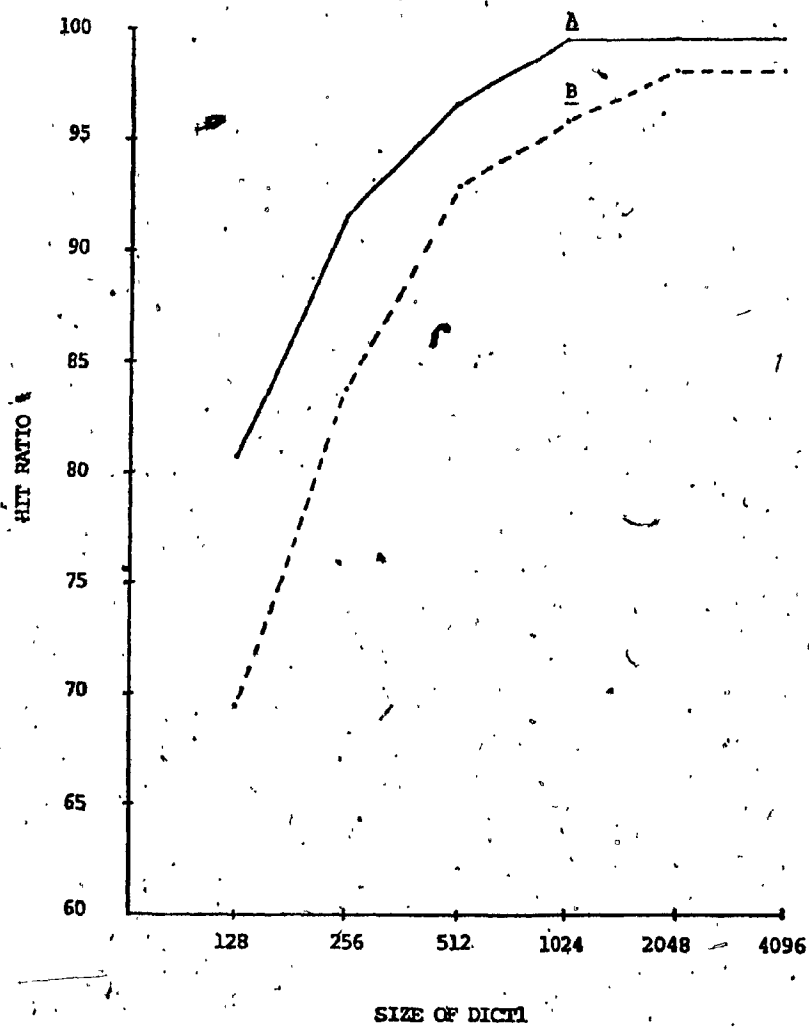


FIGURE 4.6

CURVE A CORRESPONDS TO DATA SET1
CURVE B CORRESPONDS TO DATA SET2

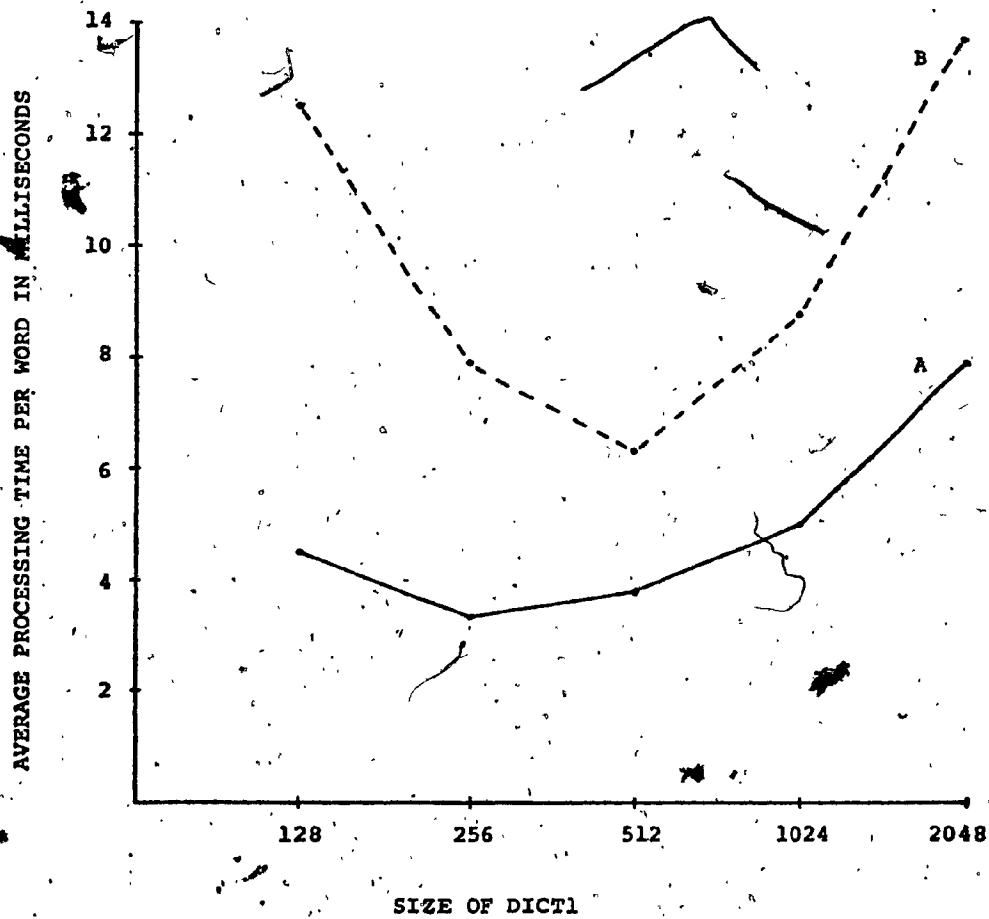


FIGURE 4.7

CURVE A CORRESPONDS TO DATA SET1
CURVE B CORRESPONDS TO DATA SET2

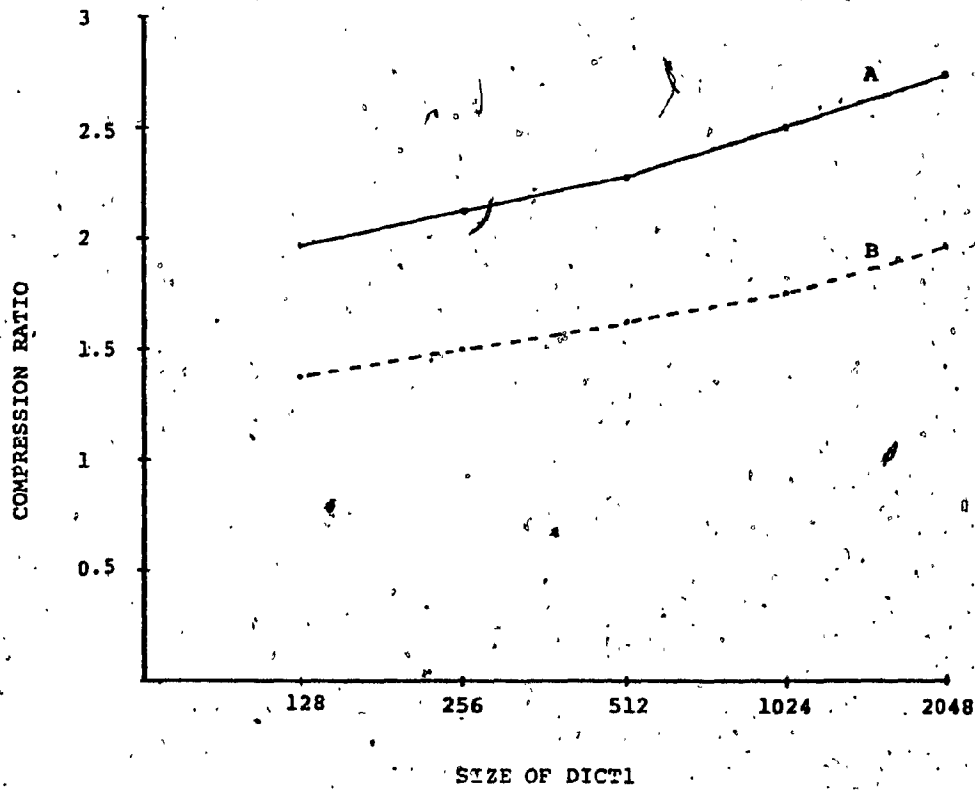


FIGURE 4.8

CHAPTER V

DATA COMPRESSION WITH MICROPROCESSORS

5.1 Micropocessors in Data Compression

Due to developments in electronics technology, particularly in the area of Large Scale Integration (LSI), microprocessors have become widely useful. They are small and light, use less power, programmable for different applications, reliable in performance and they are inexpensive.

Two kinds of microprocessors can be identified

- (i) General purpose microprocessors
- (ii) Bit-slice microprocessors.

General purpose microprocessors are available as a complete computer with processors, memory, control and I/O controllers. They are manufactured with the MOS technology which allows a large number of components and hence a complex function in a single LSI chip. But they are inherently slow and a typical operation such as addition takes 1 or 2 microseconds.

On the other hand, the bit-slice microprocessors employ TTL (transistor Transistor Logic) or ECL (Emitter Coupled Logic) technology which provide higher speed in the order of

a few hundreds of nanoseconds. However, with bit-slice microprocessors one requires extra design efforts and assembly work.

Applications of microprocessors for data compression have been considered by several authors [Murra,77], [Lea,78]; [Radha,79].

Murray, uses 4-bit slice processors to implement a data compression algorithm that is based on Cosine transform and DPCM (Differential Pulse Code Modulation). His data compressor is able to process data at the rate of 200, 400, or 800 kilo bits per second. In the laboratory model, the microprocessor is controlled by a PDP-11 mini computer and the system has been used to compress the television signals. For this application, a single TV frame is considered to consist of 256 lines with 256 pixels per line.

Lea has considered the use of general purpose microprocessors for text data compression and found them to be too slow for disk transmission rates. The processing rate was found to be in the order of 1.1K bytes per second for compression and 4.0K bytes, for decompression. The main objective of his research has been to compress (or decompress) the alphanumeric texts stored on magnetic disks by putting a hardware encoder (decoder) unit in the data path to (from) the disk. In this case, the data rate is in the order of a few megga bytes per second. His compression

algorithm consists of a table of 206 most frequent n-grams along with their codes. If an n-gram of this table occurs in the text to be stored, its occurrence is substituted by the corresponding code which results in data compression. By using a special purpose associative parallel processor for this table look-up, Lea has obtained a processing rate of 4.1 to 6.3 megga bytes per second.

Radhakrishnan, considers on-board data compression of satellite data [Radha,79]. He used "run length coding" and "differential pulse code modulation" as the two techniques to compress the satellite data. He examined the use of bit-slice micros for on-board compression because of its desirable qualities for on-board data compression.

5.2 Memory Technology Related to Microprocessor Applications

The past decade has witnessed dramatic advances in memory, and storage technology, primarily as a result of the tremendous progress in semiconductor technology. Today, semiconductors have almost replaced core as the preferred mainframe memory. The following section introduces two of these memory technologies.

Memory Technologies

The state of memory technologies currently employed in most computer memory hierarchies is shown in FIGURE 5.1. Choice of a technology depends on two principal factors,

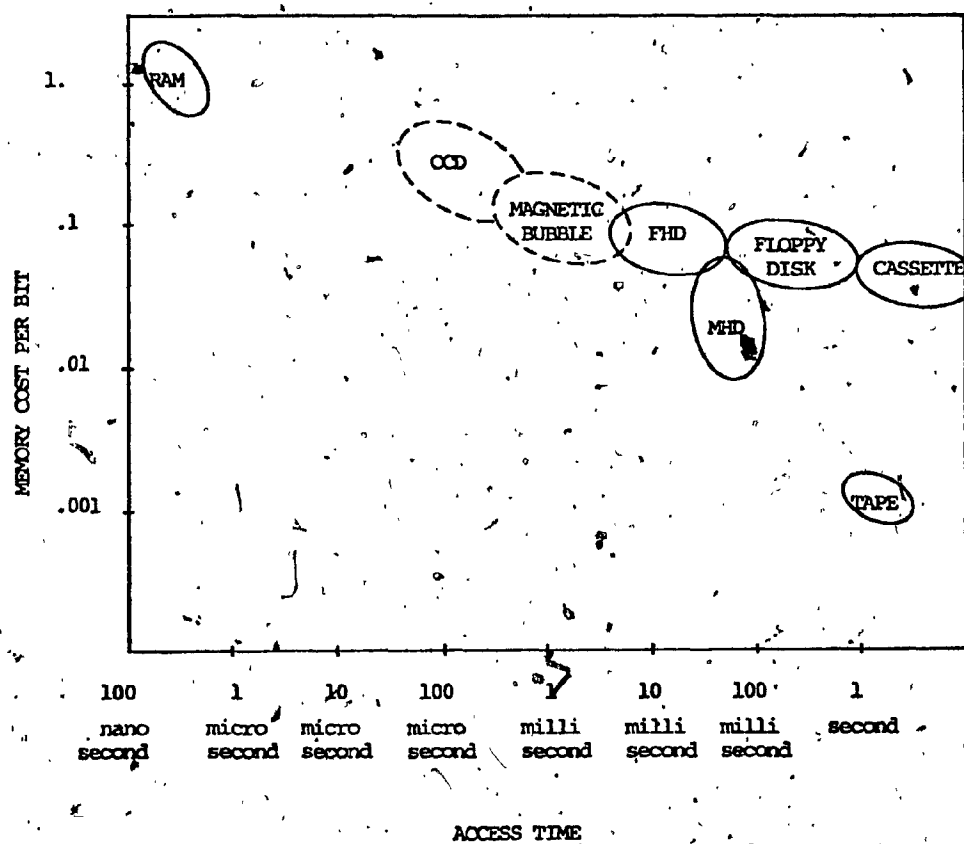


FIGURE 5.1 MEMORY TECHNOLOGY: COST/PERFORMANCE

price per bit and performance evaluated in terms of access time. As shown in FIGURE 5.1, access time increases from the high-performance, high-priced, bipolar and metaloxide semiconductor (MOS) memories on the left to the lower performance, lower-price-per-bit, rotating serial access magnetic memories shown on the right.

Magnetic bubble memories (MBM)

MBM technology has been an active field for almost fifteen years. Today they can be purchased off-the shelf from manufacturers like Texas Instruments. They are available in a wide range of capacities, 16K to 768K bytes, with an average access time of 3 milliseconds [Tex,80]. The major potential advantages of bubbles are their high density, relatively few processing steps in manufacturing, and nonvolatility. In this type of memory, magnetic bubbles are propagated in shift register fashion to transport data from one location to another. Several shift register structures are so organized that the access time to any data is relatively short. Like other magnetic memories, data can be stored indefinitely with no power dissipation. Detailed description of magnetic bubble devices are found in [Bobec,75],[Geusic,76],[Niels,76].

CCD MEMORY ORGANIZATION

The CCD (charge-coupled device), as a serial shift register, might be visualized as a set of flip-flops organized serially, through which information is propagated bit by bit at a rate established by an external clock. The frequency of the clock that shifts data through the CCD shift register can have a wide range, limited at one extreme by the circuit's maximum switching speed, and at the other by maximum allowable refresh time required to maintain the data bits. In contrast to bubble memory, the CCD memory is volatile.

Various organizations have been proposed for CCD memory construction and the choice depends on desired system performance. There are important trade-offs involving clock frequency, number of phases, access times, chip overhead for peripheral circuits, and temperature range, as well as other parameters.

There are three popular CCD memory organizations: synchronous or serpentine, serial-parallel-serial (SPS), and line-addressable random-access memory (LARAM).

The synchronous organization, shown in FIGURE 5.2, is the simplest CCD layout which moves data from cell to cell and from register to register in one continuous stream. It has the longest access time and highest power dissipation of the three schemes. The SPS organization has numerous

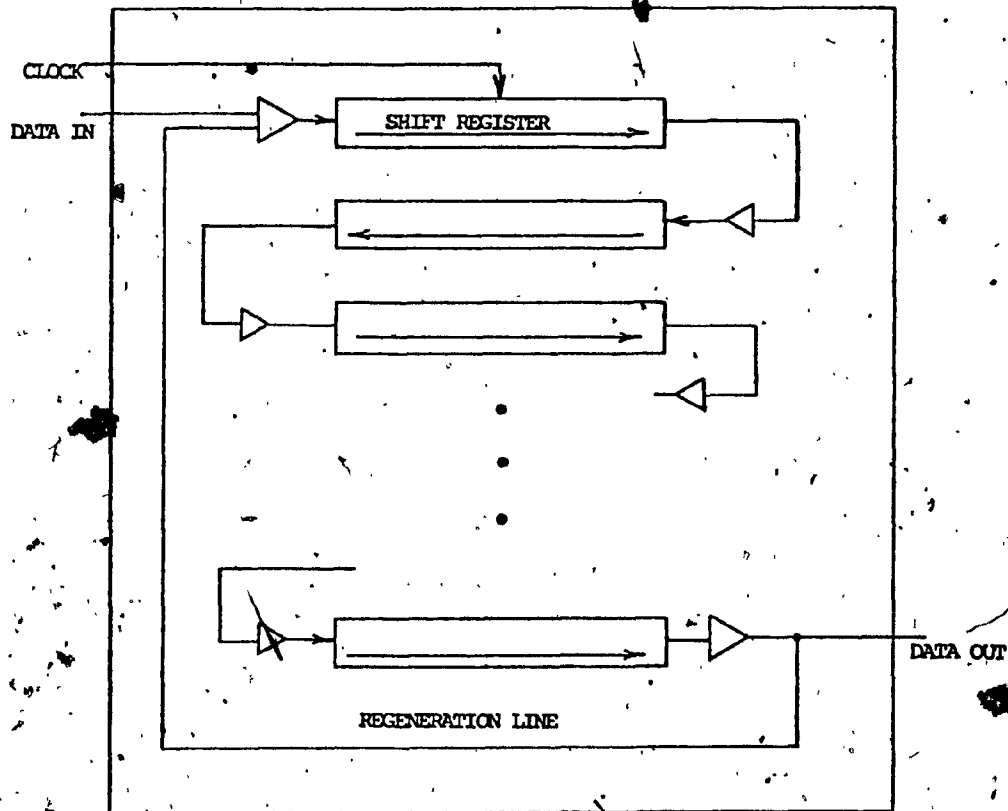


FIGURE 5.2 SYNCHRONOUS ORGANIZATION

parallel storage registers placed between two serial registers. One serial register is meant for input and the other is for output. The FIGURE 5.3 shows such a layout.

The LARAM organization has parallel storage registers that circulate data individually, and the registers are separately addressable at random. Power dissipation is minimal because only one register at a time need operate at higher than standby rate. The FIGURE 5.4 shows a LARAM organization.

A quantitative comparison of these basic organizations, shows that SPS has the lowest cost per bit and lowest power dissipation. The Synchronous scheme offers the best temperature range performance, while the LARAM has shortest access time and lowest clock capacitance drive requirements [Crouc,76].

5.3 Organization of Dict2 ON CCD

It has been mentioned in the earlier section that the LARAM organization provides random access to stored data and has the least access time.

The organization of Dict2 on the CCD will be described using the Fairchild CCD 460. The Fairchild CCD 460 has LARAM organization and can hold 16,384 bits or 2K bytes. FIGURE 5.5 shows the physical organization of CCD 460. Its data rate is 20M bits per second and has a 4-bit parallel

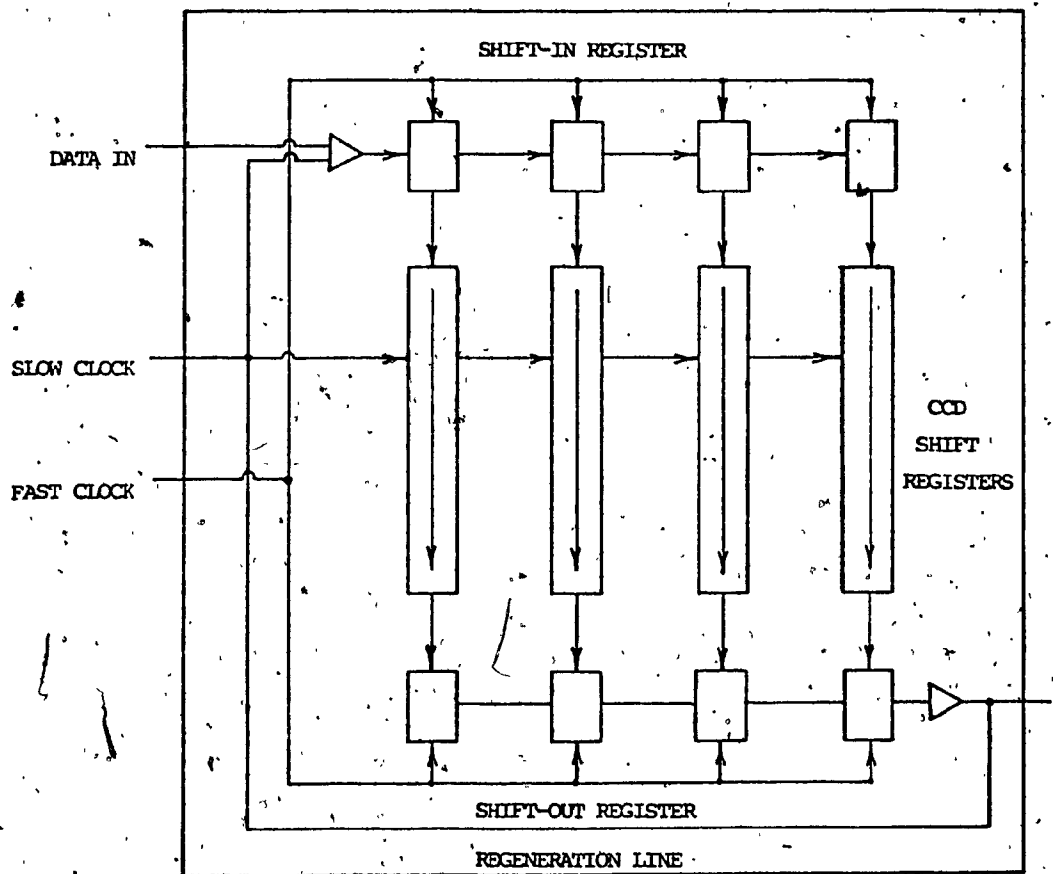


FIGURE 5.3 SPS ORGANIZATION

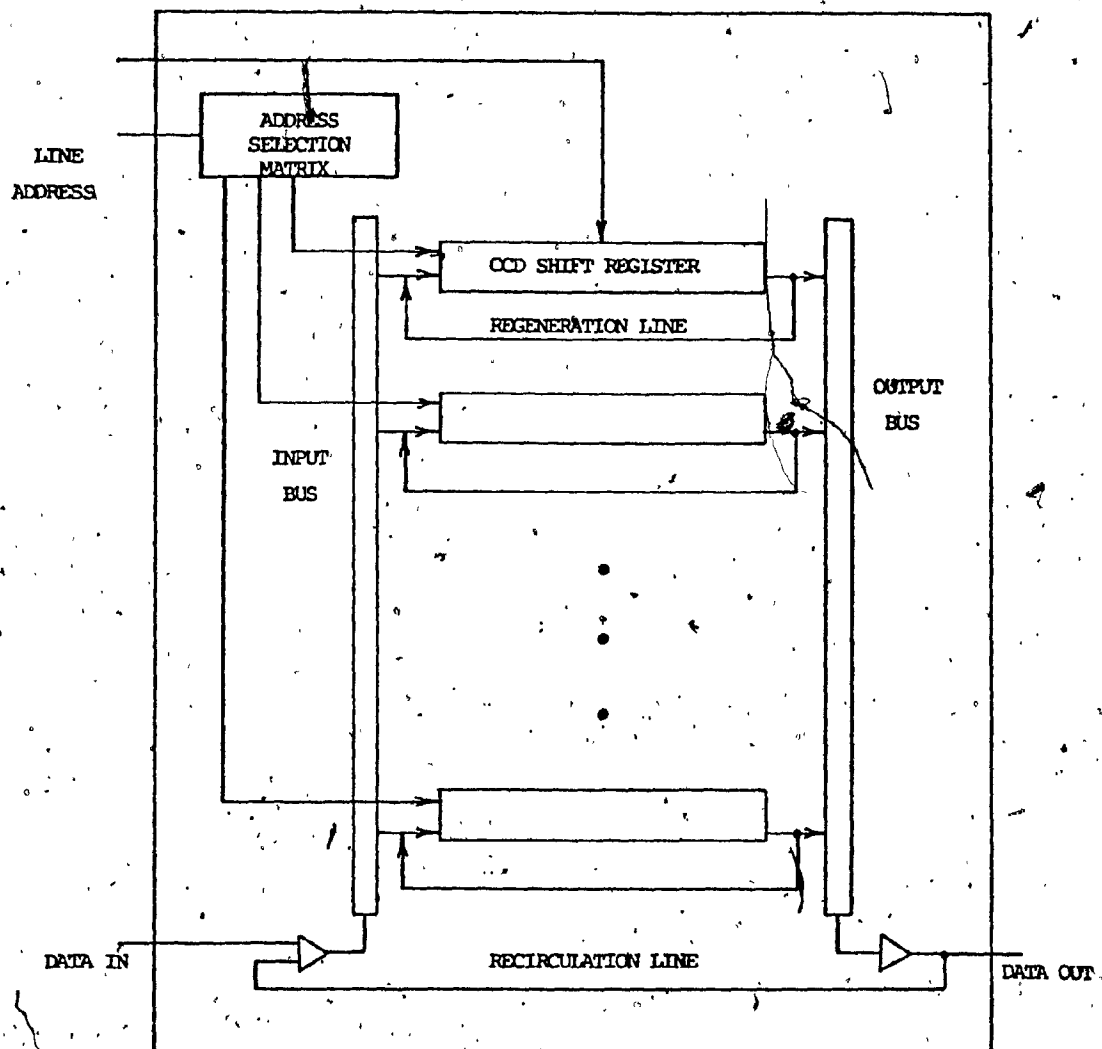


FIGURE 5.4 LARAM ORGANIZATION

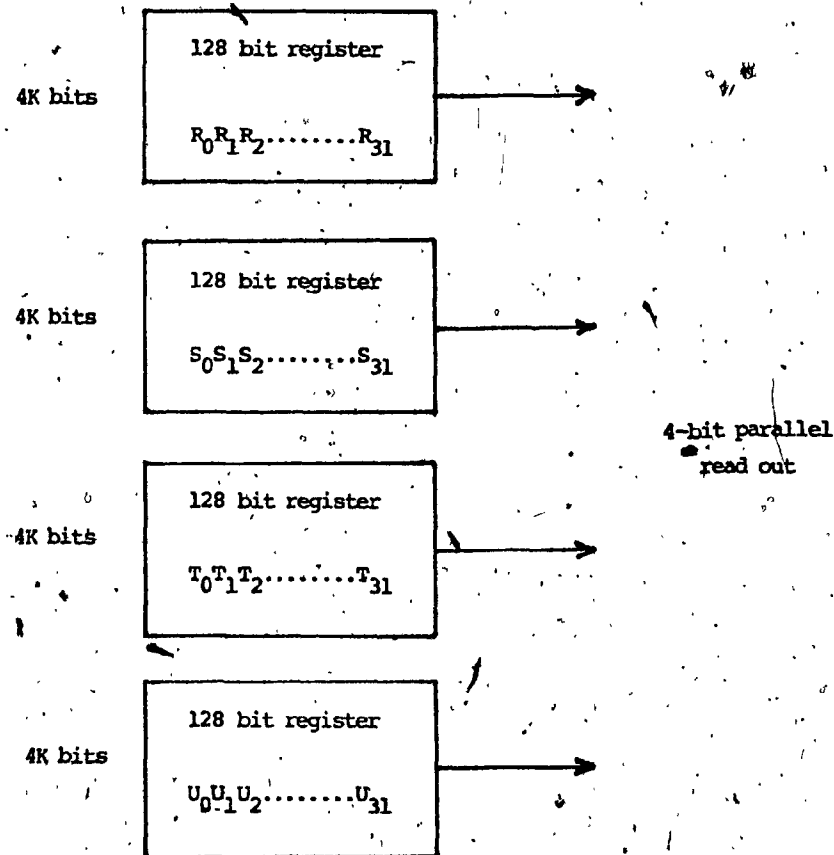


FIGURE 5.5 ORGANIZATION OF 1 UNIT OF CCD 460

Each block (box) contains thirty-two
128-bit long registers

readout. The four bits come from four independent sections each consisting of thirty-two 128-bit registers. Random access at register level is possible; but each register by itself functions as a serial shift register. When R_2 is selected in FIGURE 5.5, for reading or writing, then S_2 , T_2 , and U_2 from other sections are automatically selected for a 4 bit I/O. Average access time is 12.8 microseconds.

The distribution of the fragments to be stored in Dict2 is shown in FIGURE 5.6 and the total number of fragments is 5944. With a fixed size of 5 bytes and 2-byte code per fragment the total storage requirements for Dict2 is 29,720 bytes.

COST OF STORING DICT2 ON CCD 460

The CCD 460 has a unit capacity of 2K bytes and therefore the total number of units required is given by

$$\begin{aligned} N_U &= 41608/2048 \\ &= 21 \text{ units} \end{aligned}$$

At an estimated cost of .1¢/bit the cost, C, of storing Dict2 is

$$\begin{aligned} C &= 21 * 16384 * .1 \\ &= \$344 \end{aligned}$$

into subprocess meets the above demands. Design and fabrication of a pipeline architecture, with a set of three microprocessors, for recognition of typewritten characters are discussed by [Venka,77].

A pipeline architecture with the general purpose microprocessors of today, might require over a hundred processors to meet the high data rates in real time compression of data from the satellites and hence may be impractical. However, it should be remembered that even a two-fold increase in the speed of microprocessors, which does not seem impossible, will halve the number of processors.

Multiprocessor architecture

The main operation of the coding algorithm with LM is comparison or matching of characters. It is possible to implement this operation using the relatively faster bit-slice microprocessors. Such a special purpose processor, may be combined with a CCD unit as shown in FIGURE 5.8 which will be referred to as a CCDP.

If a CCD 460 with a capacity of 64K bits is used in the construction of a CCDP, about 292 five-grams and their codes can be stored in a single unit. Several CCDPs may be connected to a central computer as depicted in FIGURE 5.9 and its operation may proceed as mentioned below:

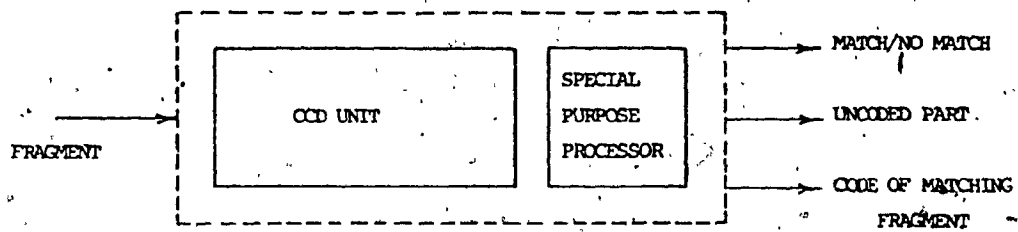


FIGURE 5.8 A OOD PROCESSOR

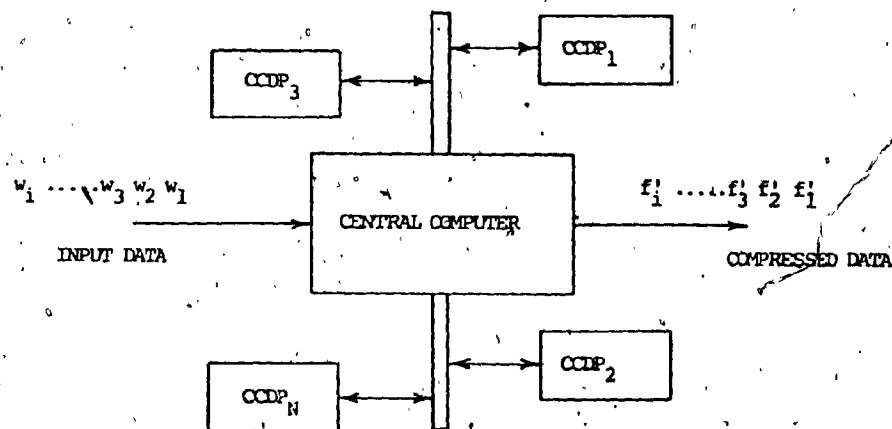


FIGURE 5.9 MULTIPROCESSOR ARCHITECTURE

An input data unit such as a "textual word" is matched by the central computer with a table such as shown in TABLE 5.1, and the appropriate CCDP is selected for communication. For example, the input word COMPUTER would be communicated to different CCDPs as shown below:

CYCLE 1 COMPUTER -----> CCDP3

 < PUTER -----

CYCLE 2 PUTER -----> CCDP15

 PUTER -----

CYCLE 3 UTER -----> CCDP20

 UTER -----

cycle 4 TER -----> CCDP18

 B -----

The coded word would be < PUTB.

Communication between the CCDPs and the central computer may be through a single bus or a multi-bus system. This architecture has a resemblance to the RAP architecture of [Ozkar,75]. Suppose there is a sequence of input data as in a pipeline, and accesses at different cycles are for different CCDPs. Then convenient searching and coding may take place.

Substrings of a single word, such as COMPUTER may go through more than one CCDP for encoding. For instance, from

the previous example COM is coded by the CCDP3 and TER is coded by the CCDP18. Each CCDP communicates to the central computer which in turn may communicate the uncoded substrings to other CCDPs. This necessitates some amount of bookkeeping in the central computer. The effects of such bookkeeping, input data rate, limits of the single or multiple buses, and processor speed may be studied through a simulation model. Results and observations of the experiments discussed in this thesis would be useful for such a simulation study.

5.5 On-line Data Compression with Microprocessors

An on-line system may be defined as one in which the input data enters the computer directly from the point of origination and/or output data are transmitted directly to where they are used, as opposed to an off-line system in which telecommunication data does not go directly into the computer but are written onto magnetic tape or disk, or punched into paper tape or cards for later processing [Marti,72].

A real-time computer system may be defined as one which controls an environment by receiving data, processing them, and taking action or returning results sufficiently quickly to affect the functioning of the environment at that time [Marti,72].

Response time can be defined as that time interval from the operator's pressing the last key of the input to the terminal's typing or displaying the first character of the response [Marti,72]. Where a man-computer dialogue is taking place, the responses must be returned to the man sufficiently quickly so as not to impede his train of thought. Response times between 1 and 5 seconds are typical. In "real-time" systems in which a machine or process is being controlled, response times can vary from a few milliseconds to many minutes. The speed of response differs from one type of system to another according to the needs. In a system for radar scanning a response time of milliseconds is needed. In contrast airline reservation systems accept a response time of about 2 seconds. Some typical real-time systems are listed in TABLE 5.2 [Marti,72].

An Experiment

The coding algorithm LM was written in (MOTOROLA) assembly language and was implemented on the M6800 microprocessor to verify the intended operation of the procedure. An in-core dictionary of size 256 fragments was used. The codeable elements were word fragments and the dictionary was organized as discussed in chapter IV. The average execution time (coding time) needed to compress a word was estimated using two sets of data. One sample was a

TABLE 5.2

SOME OF THE MANY TYPE OF COMMERCIAL
REAL-TIME SYSTEMS:

reponse times between 1-5 seconds

- [1] AIRLINE RESERVATIONS SYSTEMS
- [2] BANKING SYSTEMS
- [3] SALES INQUIRY SYSTEMS
- [4] SALES-ORDER ENTRY SYSTEMS
- [5] POINT-OF-SALES DATA COLLECTION SYSTEMS
- [6] CREDIT-INFORMATION SYSTEMS
- [7] HOSPITAL-INFORMATION SYSTEMS
- [8] TEXT-EDITING SYSTEMS
- [9] DOCUMENT-RETRIEVAL SYSTEMS
- [10] INFORMATION RETRIEVAL SYSTEMS
- [11] LIBRARY CATALOG SYSTEMS
- [12] HOTEL-BOOKING SYSTEMS
- [13] STORES AND INVENTORY CONTROL SYSTEMS
- [14] PRODUCTION DATA COLLECTION SYSTEMS
- [15] MANAGEMENT INQUIRY SYSTEMS
- [16] ENGINEERING DESIGN AIDS
- [17] STOCKBROKER INFORMATION SYSTEMS

set of 500 distinct words from the Time magazine, October 2, 1978 issue. The second was Data Set1 described in chapter III. The fragments were generated from Data Set1. Results are shown in the TABLE 5.3.

The "bulk" of the coding time is found to be in the table look-up. Since 90% of this processing time is spent searching the dictionary, one could improve the searching by means of a hardware table look-up. Suppose n is the table look-up time with the hardware unit and $n \ll u_m$ where u_m is the average coding time. The following table shows the new average coding time u'_m , that takes into account the availability of the hardware table look-up.

Coding Time

n	minimum	maximum	mean (u'_m)
.5 u	.088 msec.	8.00 msec.	1.99 msec.
.1 u	.054 msec.	3.16 msec.	.712 msec.
.05 u	.048 msec.	2.69 msec.	.552 msec.
.01 u	.046 msec.	2.52 msec.	.423 msec.
.005 u	.036 msec.	2.50 msec.	.407 msec.

From FIGURE 5.10 which is a plot of the above results, we can see that a table look-up hardware with $n = 0.1 u_m$ is adequate and no special advantage will accrue with faster devices.

TABLE 5.3

	500 WORDS			2500 WORDS		
	MINIMUM	MAXIMUM	MEAN	MINIMUM	MAXIMUM	MEAN
CODING TIME (msec)	.130	14.6	3.6	.144	10.5	2.7
TIME SPENT IN TABLE. LOOK-UP (msec)	.085	13.4	3.2	.099	10.0	2.5
% OF TIME SPENT IN TABLE LOOK-UP	65%	92%	90%	69%	95%	93%

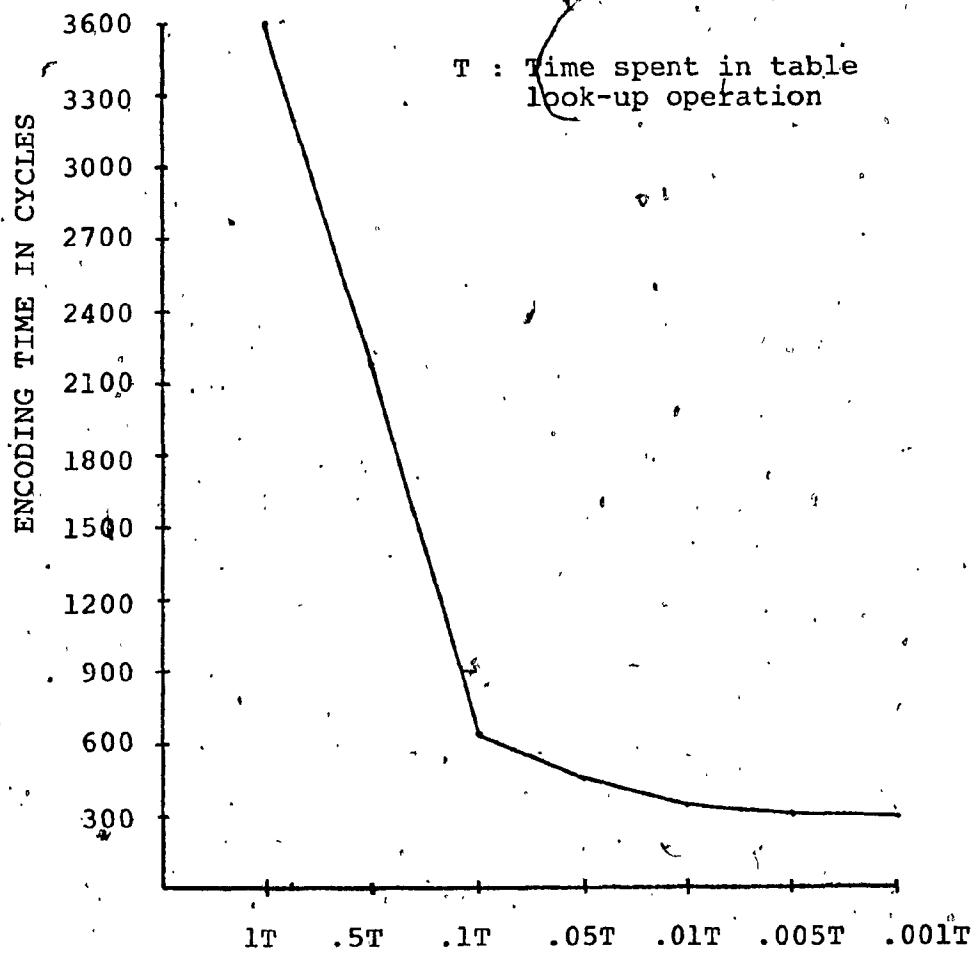


FIGURE 5.10.

From TABLE 5.3 it is seen that the mean coding time per word lies between 2.7 and 3.6 msec. An average word length of 5 characters, would mean a processing rate of 1700 to 2000 bytes per second. By referring to TABLE 5.3 and FIGURE 5.10 it is noted that compression of data in real-time applications like airline reservations or banking, can adequately be supported with conventional microprocessors.

However, for other types of real-time applications such as satellite data or disk to CPU data transmission, where the data rate is in mega bytes per second, this processing rate would not be sufficient. With a hardware table look-up having $n = .01 u_m$, a mean coding time of .423 msec. per word is obtained. Therefore it could process about 2500 words per second. An average word length of 5 characters per word would yield a processing rate of 12,500 bytes per second. Use of hardware table look-up can improve this by a factor of 5. Yet, it is not fast enough for certain applications like satellite data, or disk transmission rates.

In summary, general purpose microprocessors with or without special purpose hardware for table look-up, can be used for data compression of textual data for some real-time applications. Newer memory technologies like CCD and MBM, can be fruitfully used with microprocessors for storing the coding dictionaries.

A multiprocessor architecture with CCDP, such as proposed in this chapter could be simulated and the results obtained in the thesis could be used for such a simulation.

CHAPTER VI

Use of Word Fragments or "n-grams" for Data Compression

Application of n-grams

In the initial stages of this thesis work, n-grams and their occurrences were examined to construct a dictionary for text compression algorithms. This study has led to an interesting observation that in the frequency distribution of n-grams an optimum occurs between $n=4$ and $n=5$. The present chapter is a summary of the above.

Word fragments are substrings of words in written texts and they are also known as "n-grams" or "x-grams". N-grams have been used by investigators for coding, compression, character recognition, spelling error correction and hyphenation of written texts. Statistical distributions of n-grams computed from written texts have been studied in the past by several authors including [Suen,79], [Gupta,77].

Suen has studied the applications of n-grams statistics for natural language understanding and text processing, particularly for character recognition. He computed the n-grams statistics (for $n = 1$ to 5) as they occur in different positions (1 to 11) of the words. He notes that n-gram statistics are strongly influenced by the vocabulary

and characterizes the vocabulary by the number of different words and their frequencies. As the vocabulary increases, longer words and rarely used words appear and the average word length as well as the total number of different n-grams increase.

Suen used Kucera et al data set for his experiment. The data consisted of 922,000 words which had an $AV_N = 4.5$ characters. FIGURE 6.1 shows the word length distribution of this data set.

Gupta has studied the n-gram distributions and used them in building dictionaries for data compression. She generated n-grams (for $n = 2$ to 8) and reported that as n increases the total number of n-grams in the data base decreases whereas the number of distinct n-grams increases. This increase is rapid until n equals 4 .

Gupta used the Marc tapes issued by the U.S. Library of Congress in 1969 as her experimental data. It had 34,056 words of which 10,829 were different. The average word length $AV_N = 5.8$ characters. The word length distribution of this data set is also shown in FIGURE 6.1.

The three data sets described in Chapter III are used for our experiment. N-grams (for $n=2$ to $n=7$) were used for the study reported in this chapter. The word length distribution of Data Set1 and Data Set2 are plotted in FIGURE 6.1 to compare their word length distribution with

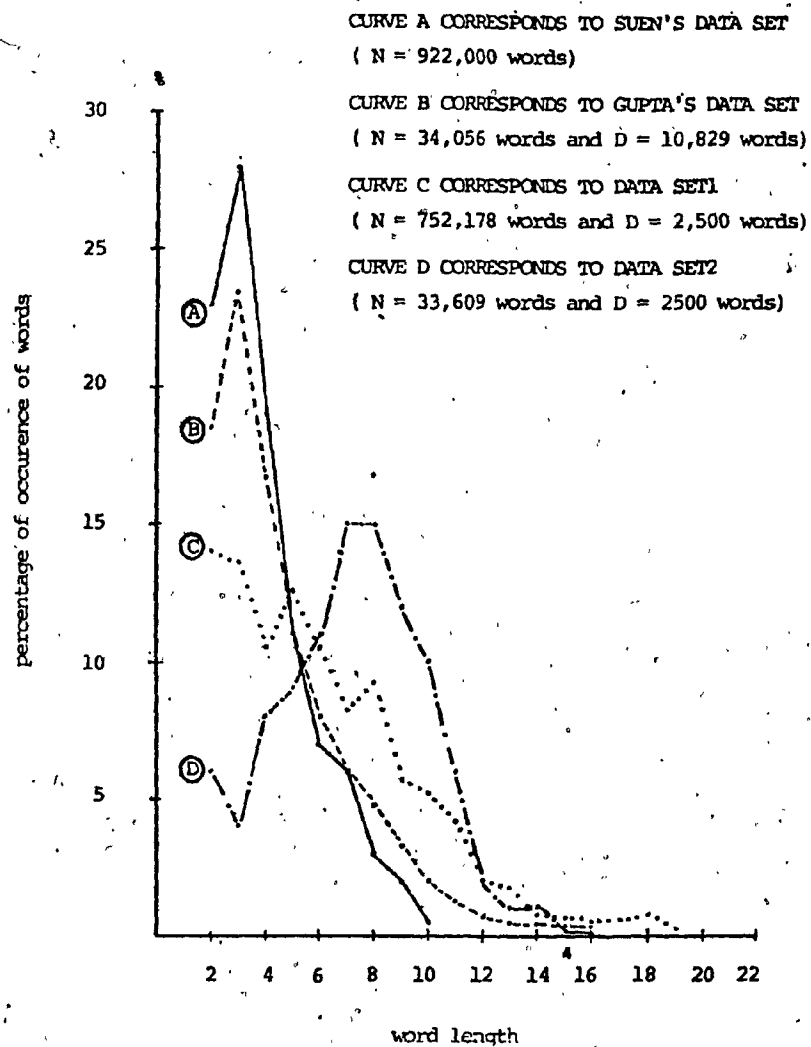


FIGURE 6.1 WORD LENGTH DISTRIBUTION

the other authors. Data Set1 has an $AV_N = 4.5$ characters and Data Set2 has an $AV_N = 7.2$ characters.

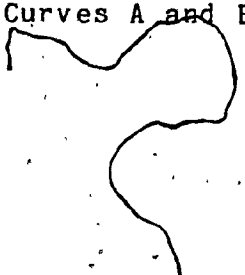
The present study in this thesis examines the distribution of n-grams and their behavior as influenced by the following factors:

- 1) sample size of data
- 2) threshold frequency
a word fragment is not considered if its frequency is less than this specified threshold value.
- 3) the average word length
- 4) subject matter of the data
technical literature versus general english text

EFFECT OF SAMPLE SIZE

For various N , where N represents the number of distinct words in the selected sample, and for a threshold of $T = 50$, n-grams of length $n=2$ to $n=7$ were generated from Data Set1 and Data Set2. The graphs of n versus $\#F$ (number of different n-grams) for various N are plotted in FIGURE 6.2 and FIGURE 6.3. Both data sets contained 2500 most frequent and distinct words which were partitioned into the following sample sizes $N = 500, 1500, 2500$ (most frequent words).

In FIGURE 6.2 the number of different n-grams increases until $n = 3$ for $N = 500$ and then it slowly decreases. Curves A and B peak at different values of n . Also the



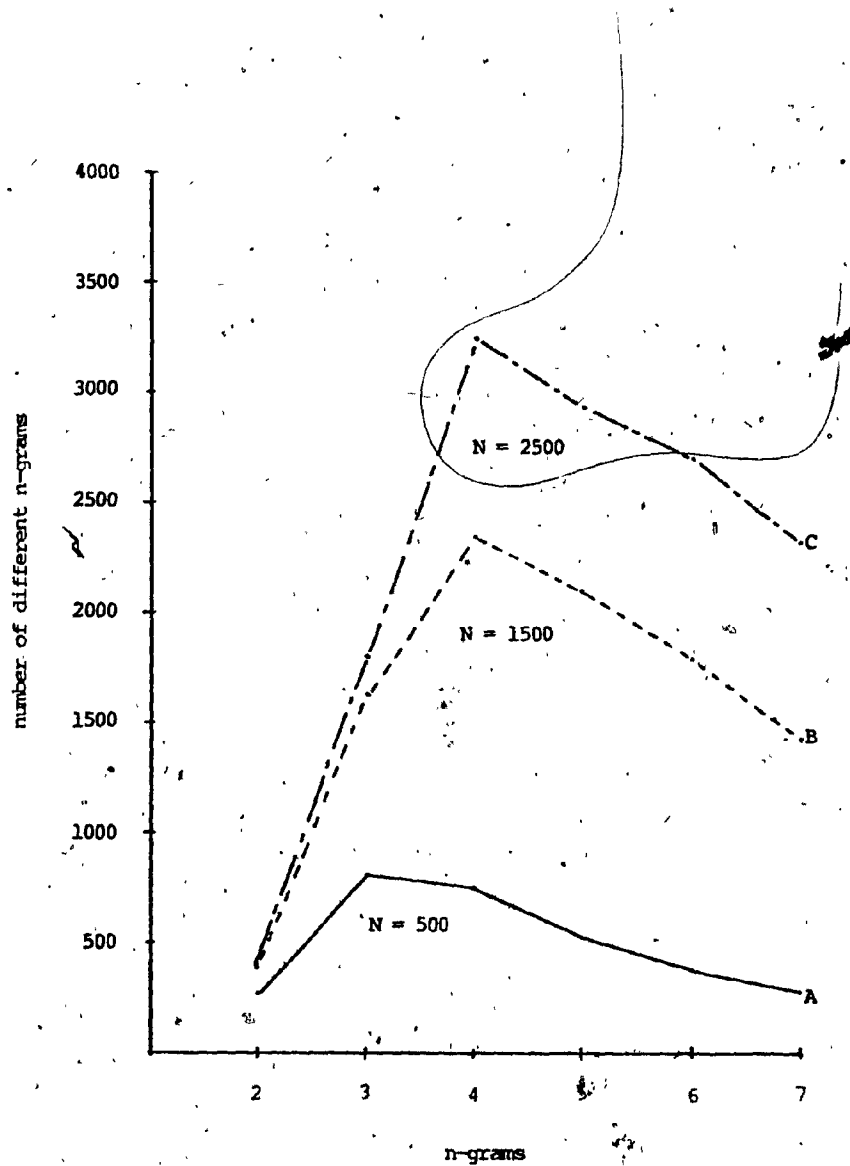


FIGURE 6.2 EFFECT OF SAMPLE SIZE

NOTE: This graph corresponds to DATA SET1

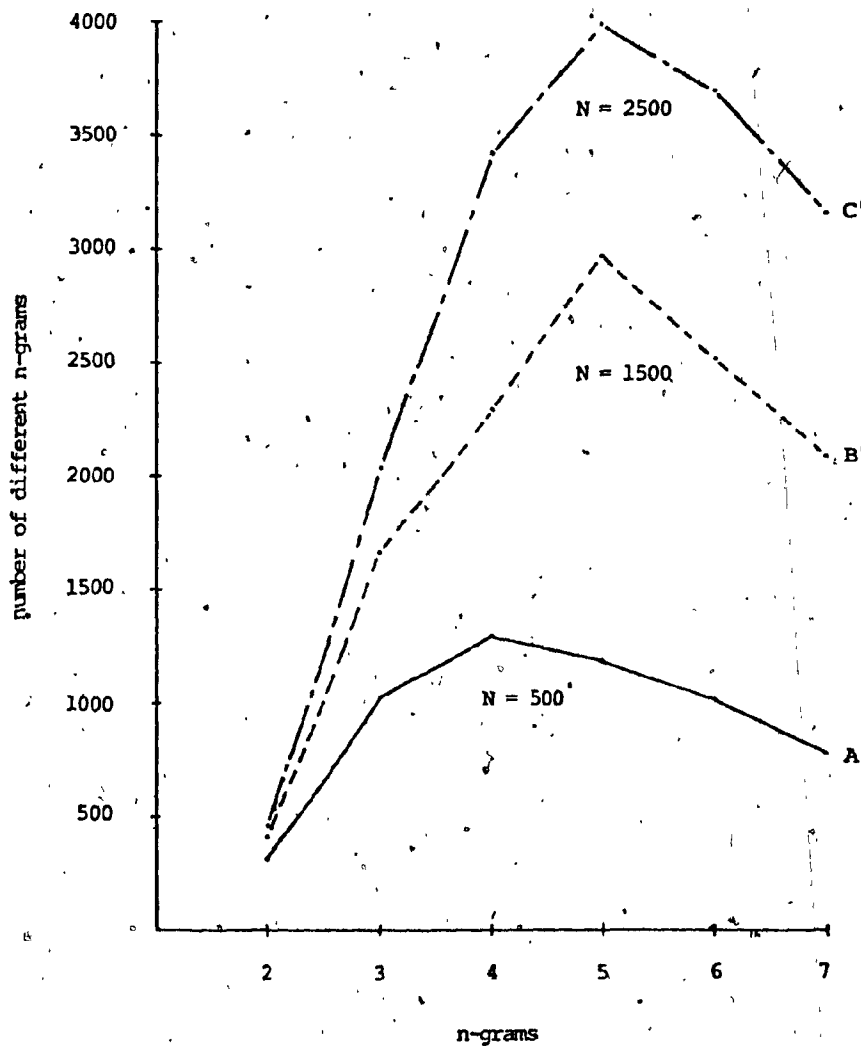


FIGURE 6.3 EFFECT OF SAMPLE SIZE

NOTE: This graph corresponds to DATA SET2

curves A and B have associated AV_D 's 4.8 and 5.6 respectively. One is tempted to correlate the AV_D with peaking at n . But this is not confirmed by the experiment with Data Set2, the results of which are shown in FIGURE 6.3. In the case of Data Set2 (technical words from CACM), high frequency words are longer than those in Data Set1 for $N = 500$. When N is increased to 1500, the AV_D 's change in opposite directions in these two data sets. This change is much more pronounced in Data Set1 than Data Set2.

From FIGURE 6.2 we can observe that as the vocabulary increases the number of different n -grams as well as the word length increase which is in conformity with the investigations by other authors. It may be noted that as the vocabulary size is increased from 500 words to 2500 words, the number of distinct bigrams has increased from 300 to 420, whereas the number of distinct tetragrams has increased from 600 to 3200.

From FIGURE 6.3 we observe that for $N = 500$ the n -grams increase up to $n = 4$ and then decrease. Curves B' and C' peak at $n = 5$. It is to be mentioned that from this data set, more n -grams were generated (for the same number of words) than in Data set1. This is explained from the fact that Data set1 has an $AV_D = 5.8$ characters while Data Set2 has an $AV_D = 6.9$ characters. Also Data set1 has words of maximum length of 10 characters while this data set has more

technical words which extend up to 16 characters.

The above results indicate that curves A to C' peak between $n = 4$ and $n = 5$. The curves of FIGURE 6.2 and FIGURE 6.3 have the same distribution. Another observation is that as N increases, the AV_D for Data Set2 decreases while that of Data Set1 increases.

EFFECT OF THRESHOLD FREQUENCY

For various T , where T is a threshold frequency chosen arbitrarily, n -grams were generated for both data sets. The n -gram size versus $\#F$ for various T is shown in FIGURE 6.4 and FIGURE 6.5.

In FIGURE 6.4 curves D, E and F peak at $n = 5$ while curves A, B and C peak at $n = 4$. For technical words in Computer Science as shown in FIGURE 6.5, the curve A' peaks at $n = 3$ while curves B' and C' peak at $n = 4$.

The two data sets behave similarly with a maximum occurring between $n = 4$ and $n = 5$. As the threshold value increases the total number of n -grams decreases and the peak shifts towards lower values of n .

EFFECT OF AVERAGE WORD LENGTH

In order to study the effect of AV_D on n -grams, the Data Set1 has been used. The n -grams versus $\#F$ for $T = 30$ and

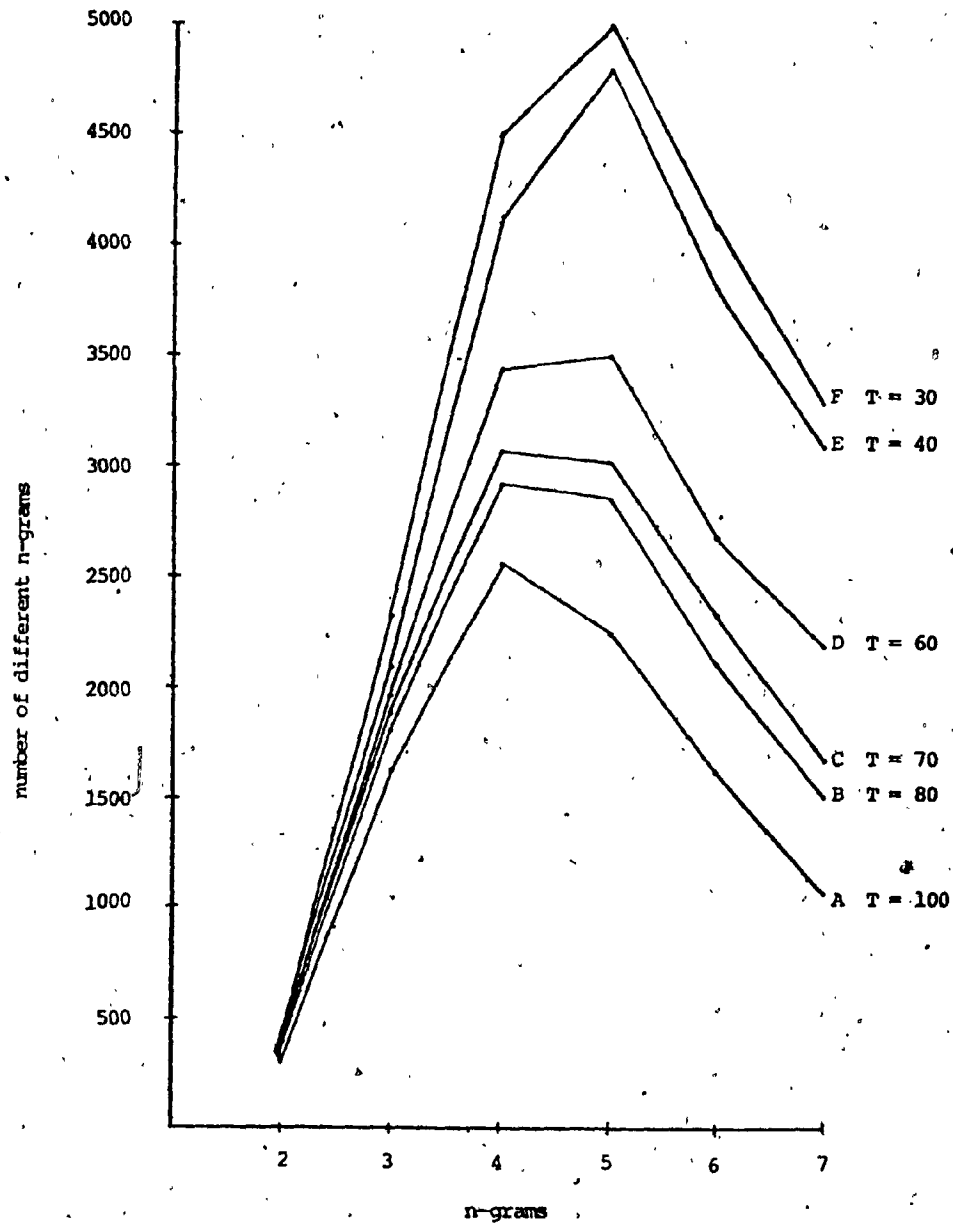


FIGURE 6.4 THRESHOLD EFFECT

NOTE: This graph corresponds to DATA SET1

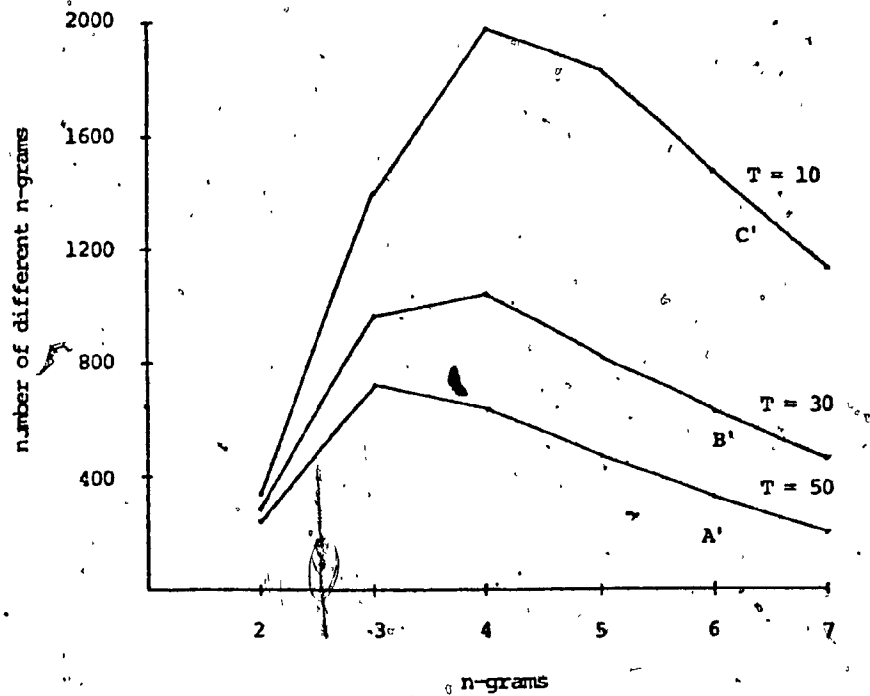


FIGURE 6.5 THRESHOLD EFFECT

NOTE: This graph corresponds to DATA SET2

$T = 100$ are shown in FIGURE 6.6, for three different values of AV_D . The average word length of the 2500 dof Data Set1, distinct words (AV_D) is 6 characters which corresponds to curve C in the figure. By considering the words of sufficient length, a subset of these words was taken so that $AV_D = 7$ characters. Similarly another subset was formed to have $AV_D = 8.4$ characters. The curves B and A in FIGURE 6.6 correspond to these two subsets respectively.

For $AV_D = 6$ characters (curve C and C'), the number of n-grams increases until $n = 5$ for $T = 30$ and $n = 4$ for $T = 100$. For $AV_D = 7$ characters (curves B and B'), the peaks occur at $n = 5$ for $T = 30$ and $n = 4$ for $T = 100$. For $AV_D = 8.5$ characters (curves A and A'), the peaks occur at $n = 5$ for $T = 30$ and at $n = 4$ for $T = 100$. As the AV_D increases, in all cases, the total number of fragments generated decreases. For all three average word lengths when $T = 30$, the peak occurs at $n = 5$; and when $T = 100$ the peak occurs at $n = 4$.

EFFECT OF SUBJECT MATTER

For this study, n-grams (for $n=2$ to $n=7$) were generated from Data Set1, Data Set2 and Data Set3. No threshold value was used. All three data sets have different characteristics as described in Chapter III. The n-grams versus #F for all three data sets is shown in FIGURE 6.7.

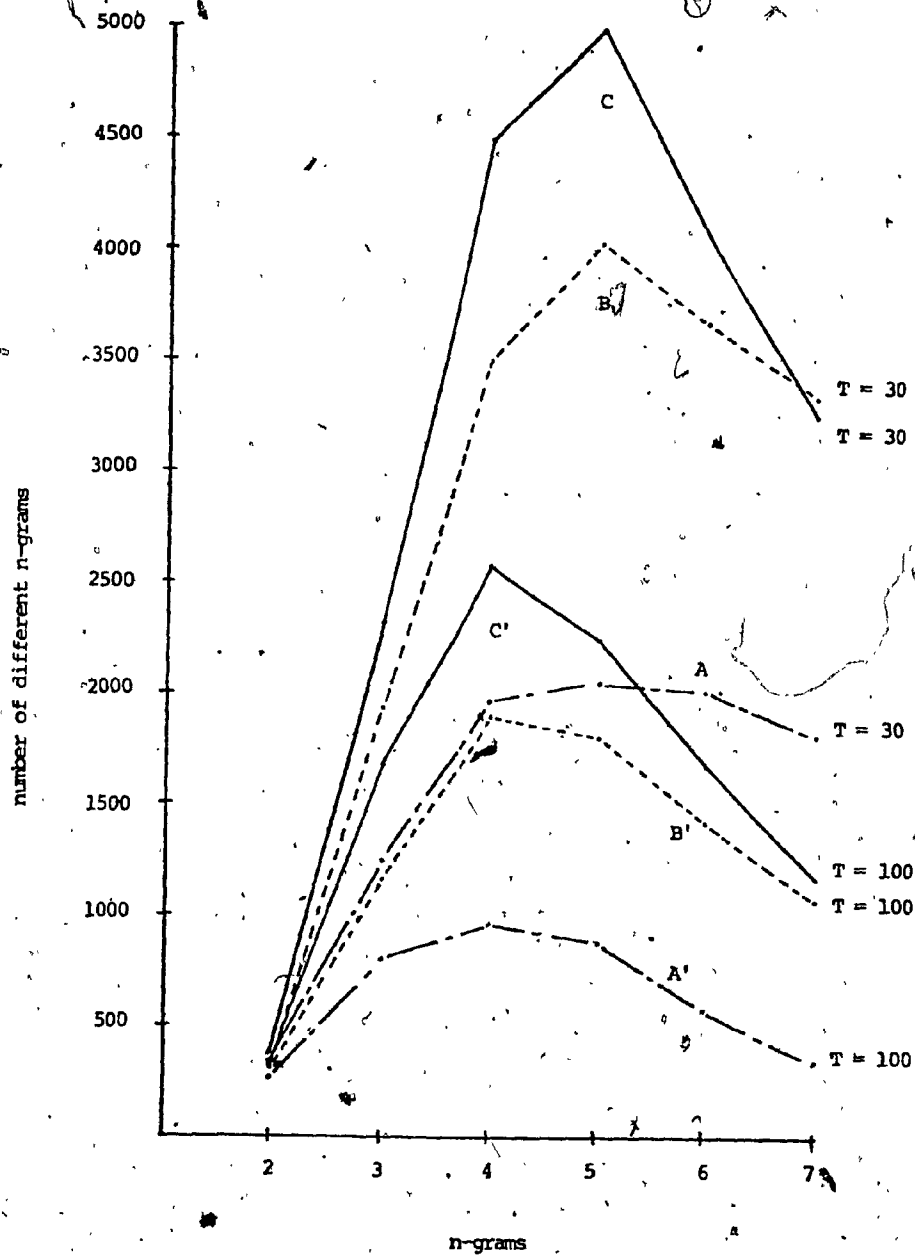


FIGURE 6.6 WORD LENGTH EFFECT

NOTE: This graph corresponds to DATA SET 1

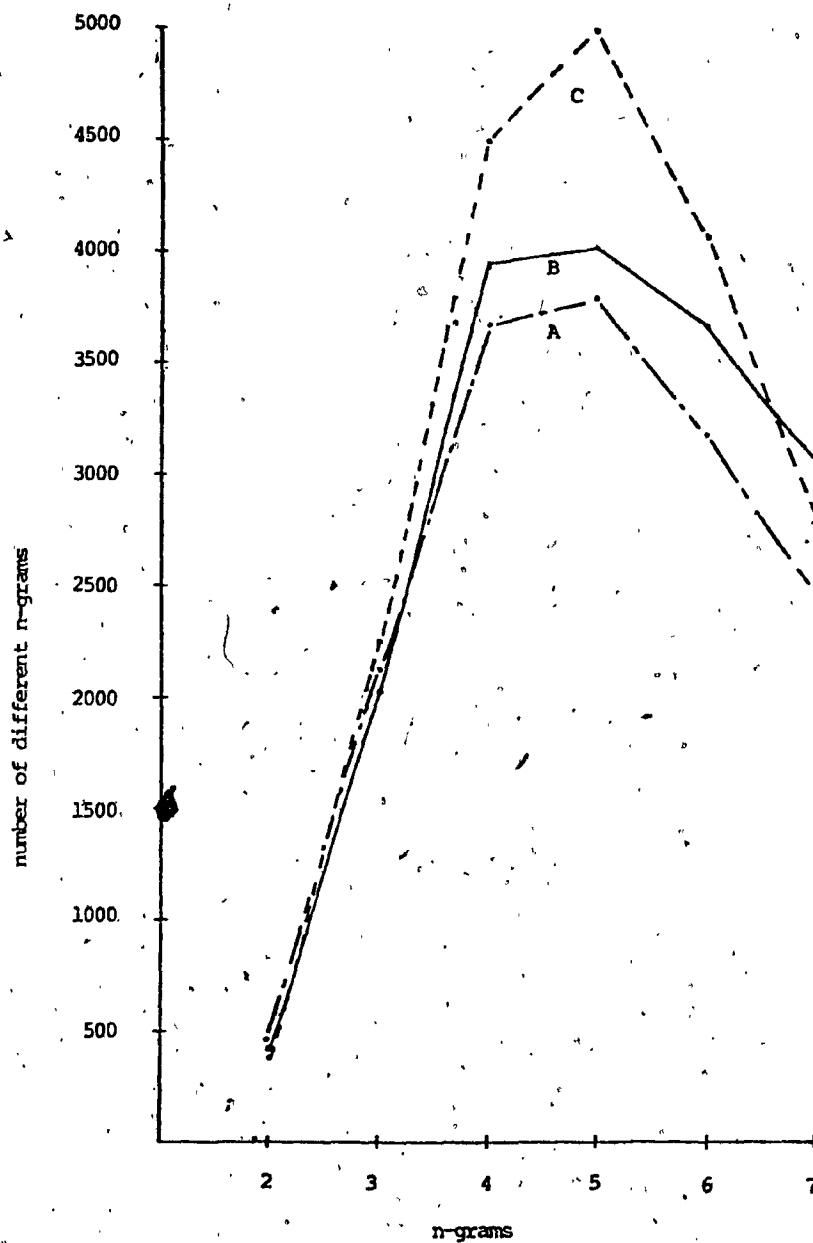


FIGURE 6.7 N-GRAM DISTRIBUTION WITHOUT THRESHOLD

NOTE: Curve A corresponds to DATA SET3
Curve B corresponds to DATA SET2
Curve C corresponds to DATA SET1

In FIGURE 6.7, curve A, B and C all peak at $n = 5$. In the curve C, corresponding to Kucera et al's data base as general English text for $n > 5$, the drop is rapid. This may be explained as the consequence of smaller AV_D of Data Set1.

All three curves have the same distribution even though their subject matter, AV_D , and word length distributions are different.

Considering all the results shown in FIGURE 6.2 through FIGURE 6.7 it is found that a maximum occurs between $n=4$ and $n = 5$ irrespective of the size of the data base, of the threshold frequency, of the average word length (AV_D) and irrespective of the subject matter of the data base.

CHAPTER VII

CONCLUSION

The results presented in this thesis are mainly experimental. The results obtained in the first part of the thesis indicate the suitability of a microprocessor for text data compression where low data rates are involved. With the use of M6800 microprocessor, the average encoding time per textual word was found to be between 2.7 and 3.6 milliseconds for the longest match algorithm. This could accomodate on-line applications such as airline reservations or banking where the response time required is in the order of one second. In these cases data originates from manually operated keyboards. The number of words processed per unit time can be improved by a factor of five with the use of a hardware table look-up.

It was found that a cell size of 4 bytes would yield the least amount of fragmentation in storage. On the other hand, in a two level dictionary scheme a cell size of 4 does not result in the least number of accesses to the second level dictionary. The ratio of the space required for both dictionaries with the cell size 7 to that with the cell size 4 (TABLE 4.1) is 7. This contrasts with the corresponding ratio of dictionary accesses (TABLE 4.3) that is .8. Therefore, when fixed-length dictionary entries are desired a cell size of 4 is appropriate. In a two

level dictionary scheme, it is found that as the size of the first level dictionary increases, the hit ratio increases, the compression ratio increases but marginally and the average encoding time also increases. The results differ for the two experimental data sets, because those word fragments which were generated from the first data set are applied to both.

From the study of the frequency distributions of n-grams it was found that a maximum or a peak occurs between $n=4$ and $n=5$. This phenomena is consistent when the parameters such as the size for the data set, the threshold value, the average word length of the data set, and the subject content are varied.

FUTURE EXTENSIONS

The proposed multiprocessor architecture with CCD, necessitates some amount of bookkeeping in the central computer. The effects of such bookkeeping, input data rate, limits of the single or multiple buses, and processor speed may be studied from a simulation model.

In several application areas, composite data that is images combined with text, are common. There are many compression techniques applied to either images or text but not both. The application of compression techniques to composite data is worth examining.

The occurrence of a maximum in the frequency distribution of n-grams may further be examined using large data samples.

However, this would require a large amount of computer time.

A microprocessor implementation of text compression methods can be incorporated into on-line data terminals. Banking, airline reservations, and word processing are some possible application areas. This is essentially an instrumentation problem.

In dictionary based data compression, construction of a suitable dictionary of data elements for coding is a major problem. As seen from FIGURE 4.8, the compression obtained by using a given dictionary is very much data dependent. One method used for constructing a dictionary is to sample the data base to be compressed and to examine the frequencies of the data elements of interest in the sample and their suitability for inclusion in the dictionary. Clearly this approach assumes the a priori availability of the data base or a representative sample of the data base. Starting from an arbitrary dictionary, can methods be devised to adapt that dictionary for a data base? This is a problem for future research. Probes and counters can be introduced in the dictionary to judge the contribution of an existing dictionary element to compression. But how to know about the data element not in the dictionary which might be relatively better? Yet another problem in this context is as follows: When a dictionary changes should one recode the already coded parts of the data base? Should one restrict changes in the dictionary so that no recoding is required; if so what is the price paid for this restriction?

REFERENCES

- [Amido,71] Amidon, E.L. and Akin, G.S "algorithmic Selection of the Best Method for Compressing Map Data Strings". Communications of the ACM, December 1971, Vol. 14, No. 12.

- [Arons,77] Aronson, J "Data Compression - A Comparison of Methods". Institute for Computer Sciences and Technology, N.B.S Special Publication 500-12, June 1977.

- [Bobec,75] Bobeck, A. H. et al., "Magnetic Bubbles - An Emerging New Memory Technology". IEEE Proceedings, vol.63, no.8, pp 1176-1195, 1975.

- [Booth,67] Booth, A.D. "A "Law" of Occurences for Words of Low Frequency". Information Control, 1967, 70, pp 386-393.

- [Cherr,51] Cherry, C., Kubba, M.H., Pearson, D.E. and Barton, M.P. "an Experimental Study of the Possible Bandwidth Compression of Visual Image Signals". Proceedings of the IEEE, 1963, Vol. 51.

- [Clare,72] Clare, A.G., Cook, G.M., and Lynch, M.F. "The Identification of Variable-length, Equi-frequency Character Strings In A Natural Language Data Base". Computer Journal 15 (1972).

- [Colum,69] Columbo, D.S., and Rush, J.E. "Use of Word Fragments in Computer Based Retrieval Systems". Journal of Chemical Documentation , 9(1969).

- [Crouc,76] Crouch, H.R., Cornett, J.B., and Edward, R.S. "CCD in Memory Systems move into sight"., Computer Design, 1976, pp 75-80.

- [Denni,70] Denning, P.J. "Virtual Memory". Computing Surveys. 2, Vol. 3, pp 153-189, 1970.

- [Econo,71] Economoto, H., and Shibata, K. "Orthogonal Transform Coding System for Television Signal", IEEE Trans. Electromagnetic Compatibility, Special Issue on Walsh Functions, EMC-13,3; August 1971, pp 11-17.
- [Geusi,76] Geusic, J.E. "Magnetic Bubble Devices: Moving From Lab To Factory". Bell Labs. Record, pp 263-267, November 1976.
- [Gottl,75] Gottlieb, D., Hagerth, S.A., Lehot, P.G.H. and Rabinowitz, H.S. "A Classification of Compression Methods and their Usefulness for a Large Data Processing Center". National Computer Conference, 1975, pp 453-458.
- [Gucci,67] Guccione, S.A. and Haddad, A.H. "A Comparison of Some Data Compression Systems". 1967
- [Gupta,77] Gupta, S. "The Use of Fragments for Data Compression". Master of Computer Science Thesis, CONCORIDA UNIVERSITY, Montreal, CANADA (1977).
- [Habib,75] Habibi, A. "Study of On-board Compression of Earth Resources Data". Final Report, TRW No. 26566, September 1975.
- [Hahn,72] Hahn, K.W. and Athey, J.G. "Diagnostic Messages". Software - Practice and Experience, 2, pp 347-352, 1972.
- [Hayes,78] Hayes, J.P. "Computer Architecture and Organization". McGraw-Hill, 1978.
- [Heaps,70] Heaps, H.S and Thiel, L.H. "Optimum Procedures for Economic Information Retrieval". Information Storage and Retrieval, 1970, 6, pp 137-153.
- [Heaps,72] Heaps, H.S. "Storage Analysis for a Compression Coding for Document Data Bases". INFOR 1972, 10, pp 47-61.

- [Heaps,74] Heaps, H.S. "Data Compression of Large Document Data Bases".:Journal of Chemical Information and Computer Sciences, Vol. 15, No. 1, pp 32-39, 1974.
- [Heaps,78] Heaps, H.S. "Information Retrieval, Computational and Theoretical Aspects". New-york Academic Press, 1978.
- [Huang,77] Huang, T.S. "Coding of Two-Tone Images". IEEE Transactions on Communications, Vol. COM-25, No. 11, November 1977, pp 1406-1424.
- [Huffm,52] Huffman, D.A. "A Method for the Construction of Minimum Redundancy Codes". Proc. Inst. Radio Engrs. 1952, 40, pp 1098-1101.
- [Kalle,75] Kallenbach, P.A. "Introduction to Data Transmission for Information Retrieval". Information Processing and Management, Vol. 11, pp 137-145, 1975.
- [Knuth,73] Knuth "Sorting and Searching". The Art of Computer Programming, Vol. 3, Addison Wesley, 1973.
- [Kucer,67] Kucera, H., and Francis, W.N. "Computational analysis of Present Day American English". Brown University Press, Providence, RI(1967).
- [Lea,78] Lea, R.M. "Text Compression with an Associative Parallel Processor". The Computer Journal, 1978, Volume 21, Number1, pp 45-56.
- [Mark,74] Mark, J.W. "An Innovations Approach to Adaptive Data Compression in Data Transmission". IEEE Transactions on Communications, Vol. COM-22, No. 10, October 1974, pp 1618-1629.
- [Marti,72] Martin, J. "Systems Analysis for Data Transmission". Prentice-Hall Inc., 1972.
- [McCar,73] McCarthy, J.P. "Automatic File Compression". National Computing Symposium, 1973, pp 511-516.

- [Murra,77] Murray, G.G. "Microprocessor System for TV Imagery Compression". Proceedings of the International Optical Computing Conference, 1977, SPIE Volume 119, Applications of Digital Image Processing (IOCC 1977).
- [Niels,76] Nielsen, J.W. "Bubble Domain Memory Materials". IEEE Transactions on Magnetics, Volume MAG-12, Number 4, pp 327-345, 1976.
- [Noll,75] Noll, p. "A Comparative Study of Various Schemes for Speech Encoding". Bell System Technical Journal, Vol. 54, No. 9..no. 9 9, pp 1597-1614, November 1975.
- [Ozkar,76] Ozkarahan, E. A., Schuster, S. A. and Sevcik, K. C. "Performance Evaluation of a Relational Associative Processor". Technical Report CSRG-65, February, 1976.
- [Pease,71] Pease, R.F.W and Limb, J.O. "Exchange of Spatial and Temporal resolution in Television Coding". Bell System Technical Journal, Vol. 50, No. 1, 1971, pp 191-200.
- [Pratt,78] Pratt, W.K. "Digital Image Processing". John Wiley and Sons, 1978 (part 6).
- [Rabin,78] Rabiner, L.R. and Schaefer, R.W. "Digital Processing of Speech Signals". Prentice Hall, p 232, 1978.
- [Radha,80] Radhakrishnan, T. and Laplante, P. "A Selected and Classified bibliography on Compression of Signal, Text, and Image data". Technical Report 80-005, CONCORDIA UNIVERSITY, Dept. Of Computer Science 1980. (available from authors)
- [Radha,79] Radhakrishnan, T. and Seco, R. "Microprocessor Controlled Compression of Remotely Sensed Data". Presented at the Mini and Microcomputers Conference held at Anaheim, California, 1979.

- [Radha,79] Radhakrishnan, T., Laplante, P. and Seco, R. "Compression of Imagery Data". Presented at the 2nd International Conference on Remote Sensing, 1979.
- [Radha,77] Radhakrishnan, T. and Heaps, H.S. "Compaction of Diagnostic Messages for Compilers". Software - Practice and Experience, Vol. 7, pp 139-144, 1977.
- [Ramam,64] Ramamoorthy, C.V. "Document Compaction by Variable Length Encoding". Proc. Am. Soc. Infor. Sci., 1964, 1, pp 507-513.
- [Reza,63] Reza, F. "An Introduction to Information Theory". (New-York: McGrawhill, 1963).
- [Robin,78] Robiner, L.R. and Schafer, R.W. "Digital Processing of Speech Signals". Prentice Hall, 1978, p 232.
- [Rose,77] Rose, J.A, Pratt, W.K. and Robinson, G.S. "Interframe Cosine Transform Image Coding". IEEE Trans. Commun., Vol. COM-25, No. 11, 1977.
- [Rosen,74] Rosenthal, L.H., Robiner, L.R., Schafer, R.W., Cumiskey, P., Flanagan, J.L. "A Multiline Computer Voice Response System Utilizing ADPCM Coded Speech". IEEE Transactions on Acoustics, Speech, and Signals. Proc. Vol. ASSP-22, No. 5, pp. 339-352, October 1974.
- [Rosen,76] Rosenfeld, A., and Kak, A.C. "Digital Picture Processing". Academic Press, 1976.
- [Rubin,76] Rubin F. "Experiments in Text File Compression". CACM, 1976, Vol. 19, pp 617-622.
- [Ruth,72] Ruth, S.S. and Kreutzer, P.J. "Data Compression for Large Business Files". Datamation 1972, pp 62-66.

- [Schue,73] Schuegraf, E.F., and Heaps, H.S. "Selection of Equipfrequent:Word Fragments for Information Retrieval". Information Storage and Retrieval, Volume 9 (1973), pp 697-711.

- [Schue,74] Schuegraf, E.F., and Heaps, H.S. "A Comparison of Algorithms For Data Base Compression By The Use of Fragments As Language Elements". Information Storage and Retrieval, 10 (1974), pp 309-319.

- [Schue,76] Schuegraf, E.J. and Heaps, H.S. "Query Processing in a Retrospective Document Retrieval System that uses Word Fragments as Language Elements". Information Processing and Management, Vol. 12, pp 283-292, 1976.

- [Schue,76] Schuegraf, E.F. "A Survey of Data Compression Methods for Non-Numeric Records". Canadian Journal of Information Science, 2(1976).

- [Schwa,63] Schwartz, E.S. "A Dictionary for Minimum Redundancy Encoding". Journal of the Association of Computing Machinery, 10(1963), pp 413-439.

- [Schwa,67] Schwartz, E.S. and Kleiboemer, A.J. "A Language Element for Compression Coding". Information Control, Volume 10, pp 315-333, 1967.

- [Snyde,70] Snyderman, M., and Hunt, B. "The Myriad Virtues of Text Compaction". Datamation 16, Number 16, pp 36-40 (1970).

- [Smith,76] Smith, A.J. "A Queuing Network Model for the Effect of Data Compression on System Efficiency". NCC, 1976, Vol. 45, pp 457-465.

- [Suen,79] Suen, C.Y. "N-Gram Statistics for Natural Language Understanding and Text Processing". IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-1, Number 2, April 1979.

- [Venka,77] Venkatesh, K. "A Microprocessor Based Character Recognition System". Master of Computer Science Thesis, Concordia University, Montreal, Canada, June 1977.