Abstract:
In data compression systems, most existing codes have integer code length because of the nature of block codes. One of the ways to improve compression is to use codes with noninteger length. We developed an arithmetic universal coding scheme to achieve the entropy of an information source with the given probabilities from the modeler. We show how the universal coder can be used for both the probabilistic and nonprobabilistic approaches in data compression. to avoid a model file being too large, nonadaptive models are transformed into adaptive models simply by keeping the appropriate counts of symbols or strings. The idea of the universal coder is from the Elias code so it is quite close to the arithmetic code suggested by Rissanen and Langdon. The major difference is the way that we handle the precision problem and the carry-over problem. Ten to twenty percent extra compression can be expected as a result of using the universal coder. The process of adaptive modeling, however, may be forty times slower unless parallel algorithms are used to speed up the probability calculating process.

UNIVERSAL CODING WITH DIFFERENT MODELERS IN DATA COMPRESSION

by

Reggie ChingPing Kwan

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

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

July 1987

APPROVAL

of a thesis submitted by

Reggie ChingPing Kwan

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citation, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

August 2nd 1987	[illegible]
Date	Chairperson, Graduate Committee

Approved for the Major Department

August 2nd 1987	[illegible]
Date	Head, Major Department

Approved for the College of Graduate Studies

12 August 1987	[illegible]
Date	Graduate Dean

STATEMENT OF PERMISSION TO USE

Signature [illegible]

Date July 30, 87

TABLE OF CONTENTS

TABLE OF CONTENTS--Continued

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In data compression systems, most existing codes have integer code length because of the nature of block codes. One of the ways to improve compression is to use codes with noninteger length. We developed an arithmetic universal coding scheme to achieve the entropy of an information source with the given probabilities from the modeler. We show how the universal coder can be used for both the probabilistic and nonprobabilistic approaches in data compression. To avoid a model file being too large, nonadaptive models are transformed into adaptive models simply by keeping the appropriate counts of symbols or strings. The idea of the universal coder is from the Elias code so it is quite close to the arithmetic code suggested by Rissanen and Langdon. The major difference is the way that we handle the precision problem and the carry-over problem. Ten to twenty percent extra compression can be expected as a result of using the universal coder. The process of adaptive modeling, however, may be forty times slower unless parallel algorithms are used to speed up the probability calculating process.

Chapter 1

INTRODUCTION TO DATA COMPRESSION

A Basic Communication System

Communications are used to provide terminal users access to computer systems that may be remotely located, as well as to provide a means for computers to communicate with other computers for the purpose of load sharing and accessing the data and/or programs stored at a distant computer. The conventional communication system is modeled by:

1. An information source
2. An encoding of this source
3. A channel over, or through, which the information is sent
4. A noise (error) source that is added to the signal in the channel
5. A decoding and hopefully a recovery of the original information from the contaminated received signal
6. A sink for the information

This model, shown in figure 1, is a simplified version of any communication systems. In (3), we are particularly interested with sending information either from now to then (storage), or from here to there (transmission).

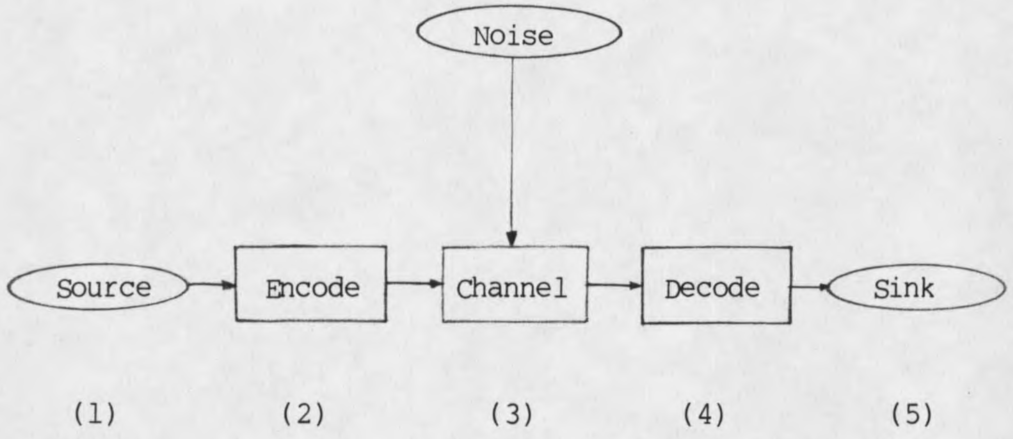

Figure 1. A basic communication system.

The Need For Data Compression

There are three major reasons that we bother to code:

1. Error detection and correction
2. Encryption
3. Data compression

To recover from noise, parity check and Hamming (4,7) codes [1] are introduced to detect and correct errors respectively. To avoid unauthorized receivers, encryption schemes like public key encryption system [2] are used. Finally, to reduce the data storage requirements and/or the data communication costs, there is a need to reduce the size of voluminous amounts of data. Thus, codes like Shannon-Fanno [3], Huffman [3], LZW [4], and Arithmetic [5] and [6] etc. are employed.

Data compression techniques are most useful for large archival files, I/O bound systems with spare CPU cycles, and transfer of large files over long distance communication links especially when low band width is available.

This paper discusses several data compression systems and develops a universal coder for diffferent modelers.

Data Compression and Compaction - A definition

Data compression/compaction can be defined as the process of creating a condensed version of the source data by reducing redundancies in the data. In general, data compression techniques can be divided into two categories -- nonreversible and reversible. In nonreversible techniques (usually called data compaction), the size of the physical representation of data is reduced while a subset of "relevant" information is preserved, e.g. CCC algorithm [7], and Rear-compaction [8]. In reversible techniques, all information is considered relevant, and compression/decompression recovers the original data, e.g. LZW algorithm [4].

Theoretical Basis

The encoding of an information source can be illustrated by figure 2.

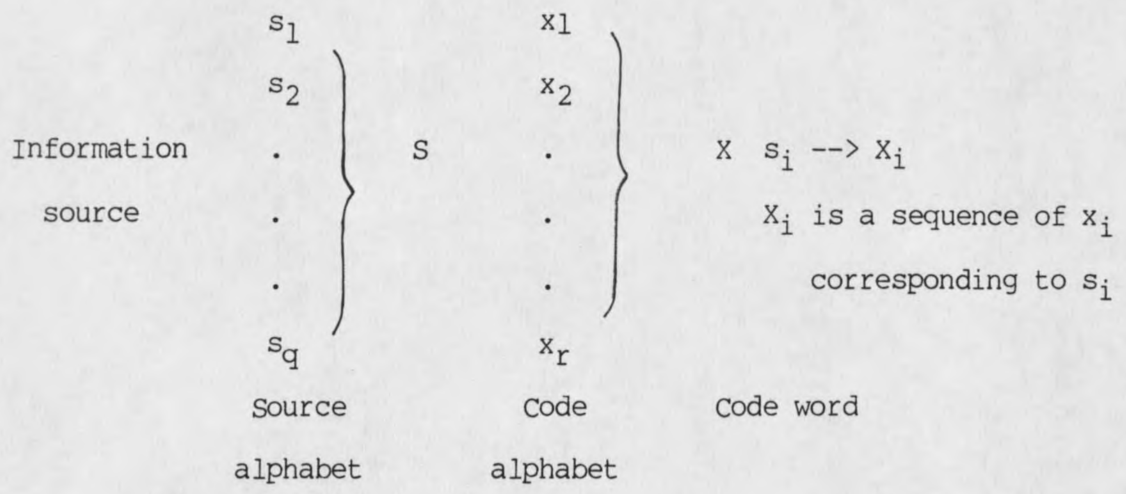

Figure 2. Coding of an information source.

In fact, figure 2 is the component labeled (2) in figure 1. The simplest and most important coding philosophy in data compression is to assign short code words to those events with high probabilities and long code words to those events with low probabilities [3]. An event, however, can be a value, source symbol or a group of source symbols to be encoded.

Tables 1 and 2 are used to illustrate the idea to achieve data compression when the probabilities are known. Consider the language Kikian of a very primitive tribe Kiki. The Kikian can only communicate in four words # = kill, @ = eat, z = sleep, and & = think. The source alphabet S = { #, @, z, & }. Assume an anthropologist studies Kikian and he encodes their daily activities with a binary code. Then, the code alphabet X = { 0, 1 }. Without knowing the probabilities of the tribe´s activities, or source symbols, we constructed the code for the source symbols as table 1.

With the probabilities of each activity, or symbol, being known, we can employ our coding philosophy to construct a new code, as shown in table 2.

Table 1. Kikian code.

Symbol	Code A
#	00
@	01
z	10
&	11

Table 2. Compressed Kikian code.

Symbol	Probability	Code B
#	0.125	110
@	0.25	10
z	0.5	0
&	0.125	111

The average length L of a code word using any code is defined as

$$L = \sum_{i=1}^{q} |C(S_i)| * p(S_i) \qquad (1.1)$$

where q = number of source symbols

S_i = source symbols

$C(S_i)$ = code for S_i

$|C(S_i)|$ = length of the code for symbol S_i

$p(S_i)$ = probability of symbol S_i

The average length L_A of a code word using code A is obviously 2 bits because $|C(S_i)| = 2$ for all i and $p(S_i)$ is always one. On the other hand, the average length L_B of a code word using code B is 1.75 bits, calculate as

$$L_B = 0.125 * 3 + 0.25 * 2 + 0.5 * 1 + 0.125 * 3.$$

That is, for our encoding system for the Kikian we have found a method using an average of only 1.75 bits per symbol, rather than 2 bits per symbol. Both codes, however, are instantaneous (see Appendix B), and therefore uniquely decodable. It is because all code words from both codes are formed by the leaves of their code tree as in figure 3.

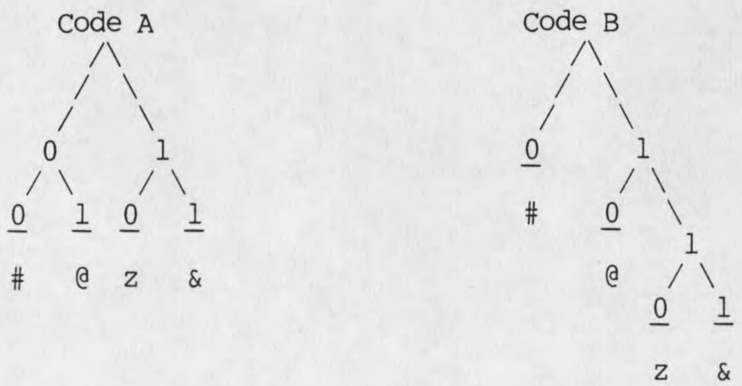

Figure 3. Code trees of Code A and Code B.

A typical day of the Kikian could be sleep, eat, sleep, kill, sleep, think, eat, and sleep. If a Kikian has a diary, z@z#z&@z will be the symbols appear on a typical day. On the anthropologist's note book, it would be 1001100010110110 without considering the probabilities of different activities. With the probabilities being known, however, the daily activities of the Kikian can be represented by 01001100111100. In other words, to represent the same information, the daily activities of the Kikian, we can either use 16 bits or 14 bits.

Code B, in this case, is clearly better than code A even though they are both uniquely decodable which is the major criteria in data compression. Nevertheless, at this point we cannot conclude if code B gives us the best compression for the given probabilities, though it turns out to be true.

To determine what is the best compression we can get for the given probabilities, let us look at the information content of each event first. Let $p(S_i)$ be the probability of the symbol S_i. We can say that the information content of S_i is:

$$I(S_i) = - \log p(S_i) \qquad (1.2)$$

bits of information if log base two is used [3].

Code B uses only one bit to represent z and three bits to represent @ because the information content of z is less than @.

$$I(z) = -\log_2 p(z) = -\log_2 0.5 = 1 \text{ bit}$$

$$I(@) = -\log_2 p(@) = -\log_2 0.125 = 3 \text{ bits}$$

Intuitively, if an event is highly unlikely to happen and happens, it deserves more attention or it contains more information.

To determine if a code has the shortest average code length L, we have to compare L with the entropy H(S). If symbol s_i occurs, we obtain an amount of information equal to

$$I(S_i) = - \log_2 p(S_i) \text{ bits}$$

The probabilities that S_i is $p(S_i)$, so the average amount of information obtain per symbol from the source is

$$\sum_S p(S_i)\ I(S_i) \text{ bits}$$

The quantity, the average amount of information per source symbol, is called entropy H(S) of the source

$$H(S) = - \sum^q_{i=1} p(S_i) \log_2 p(S_i) \text{ bits} \qquad (1.3)$$

In code B, $L_B = H(S) = 1.75$ bits, and therefore code B provides the best compression with the given probabilities according to Shannon First Theorem

$$H_r(S) =< L < H_r(S) + 1. \qquad (1.4)$$

The proof of the above theorm can be found in any information or coding book. It is obvious that the lower bound of L is the entropy. The upper bound of L, on the other hand, can be explained by the nature of block coded. The length of any block code is always an integer, even though the information content is not necessarily an integer because $|C(S_i i)|$ form (1.1) is calculated as $|C(S_i)| = |\log p(S_i)|$.

Components of a Data Compression System

The fundamental components of a data compression system are:

1. Information source
2. Modeler
3. Encoder
4. Compressed file
5. Decoder.

In a digital image compression system, a predictor may be used with the modeler to capture hopefully not so obvious redundancies. For the purpose of our discussion, we would concentrate on modelers, encoders and decoders. Components of a data compression system are shown in figure 4.

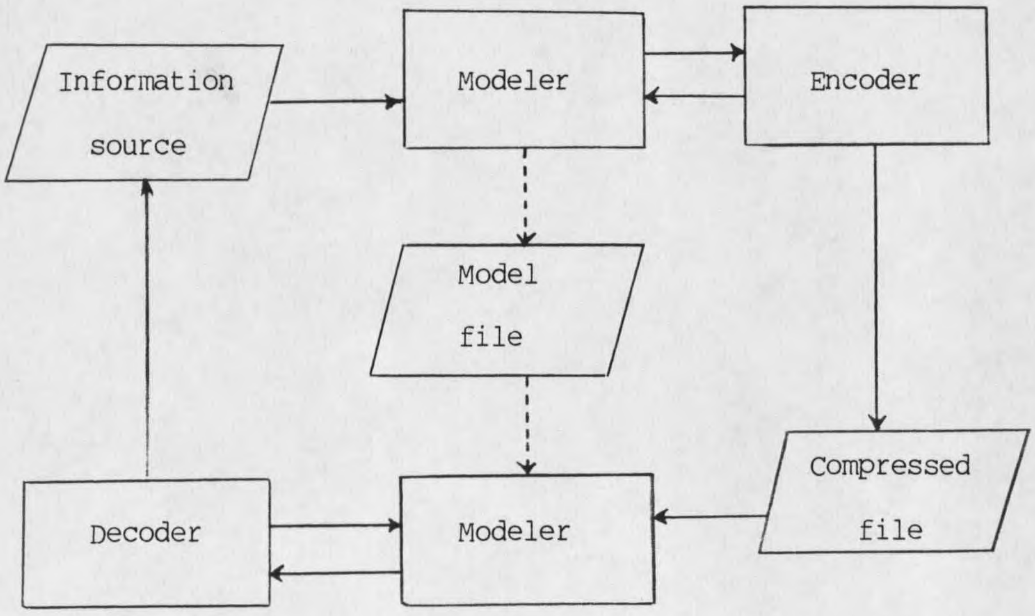

Figure 4. Components of a data compression system.

Information Source

For communication purposes, an information source can be thought of a random process {S(i)} that emits outcomes of random variables S(i) in a never ending stream. The time index i then runs through all the integers. In addition to the random variables S(i), the process also defined all the finite joint variables (S(i), S(j), ...). The most frequently studied information sources are either independent, or Markovian.

A source is said to be independent if $p(S_i) = p(S_i|S_j)$, where $j = 1,\ldots,q$. Therefore, it is often called zero-memory source.

A more general type of information source with q symbols is one in which the occurrence of a source symbol S_i may depend on a finite number m of preceding symbols. Such a source (called Markovian, or an m^{th}-order Markov source) is specified by giving the source alphabet S and the set of conditional probabilities

$$p(S_i|S_{j1},S_{j2},\ldots,S_{jm}) \quad \text{for } i=1,2,\ldots,q;\ j_p=1,2,\ldots,q \qquad (1.5)$$

For an m^{th}-order Markov source, the probability of a symbol is known if we know the m preceding symbols.

Modeler

The modeler is responsible for capturing the "designated" redundancies of the information source. Designated redundancies can be viewed as the frequency measures used in the coding process.

In a straight forward data compression system like Huffman code [3], the modeler simply counts the occurrence of each symbol S_i, for

$i=1,2,\ldots,q$. In other more sophisticated systems, e.g. LZW algorithm [4], the modeler is responsible for putting new strings in the string table. Though the algorithm does not attempt to subdivide the process into modeling and coding, modeling and coding are actually done separately and will be shown in chapter 3.

Depending on how the probabilities are calculated, we can distinguish between two cases: the adaptive and the nonadaptive models discussed in detail in chapter 3. In a nonadaptive model, a model file is generated as in figure 4. The model file contains the necessary probabilities for both the encoder and the decoder.

Encoder

The encoder encodes the symbols or strings with their probabilities given by the modeler. The basic data compression philosophy, symbols with higher probabilities are assigned with the shorter code words, applies here. The encoder also ensures the codes to be uniquely decodable for reversible compression.

In our data compression system, all the encoder needs from the modeler are the probabilities of all symbols and the previous symbols. The whole encoding process will be **adding** and **shifting.**

Compressed File

The compressed file can be thought of a file with a sequence of code words or code alphabets. With X the set of the code alphabets, the compressed file x should be simply a subset of X^*.

Decoder

The decoding process also involves the modeler regardless of the nature of the modeler (adaptive or nonadaptive). If nonadaptive modeler is used, a model file is required with the compressed file to provide all the necessary statistics.

In our data compression system, the decoder goes through a **subtract** and **shift** process to recover the original source.