Intelligent Communication for Low Bit Rate Channels

David K. Asano
Communications Research Laboratory,
Ministry of Posts and Telecommunications,
Koganei, Tokyo 184, Japan,
E-mail: *david@kohnolab.dnj.ynu.ac.jp*.

Ryuji Kohno
Division of Electrical and Computer Engineering,
Yokohama National University,
Yokohama 240, Japan,
E-mail: *kohno@kohnolab.dnj.ynu.ac.jp*.

Abstract

In this paper, intelligent compression techniques to allow transmission of text and image data over low bit rate channels are proposed. For text, a compression technique for English language text is presented. Compression is carried out by deleting characters that are not necessary for comprehension by a human reader. Decompression is carried out on a sentence by sentence basis using a database consisting of words and English grammar information. The possible sentences are constructed and then the incorrect sentences are eliminated until only one possibility remains. When this technique is used in combination with a lossless compression scheme such as Lempel–Ziv, the text can be compressed to a higher degree than is possible with only traditional compression techniques. For images, a technique to convert sequences of images into sequences of commands is described. Instead of transmitting the actual images, motion descriptions of the image relative to the previous image are sent. This results in high compression.

I Introduction

Traditional data compression techniques use the statistics of the data to reduce the number of bits required to represent the data. In these methods, the data are thought of as numbers. This approach works well for many kinds of data, such as images and text. However, the amount of compression that can be obtained is limited by the entropy of the data. In order to improve the compression, information other than just the statistical properties of the data must be used.

In contrast, intelligent communication techniques use the meaning of the information in order to decrease the amount of data used to represent the information. These two approaches are shown in figure 1.

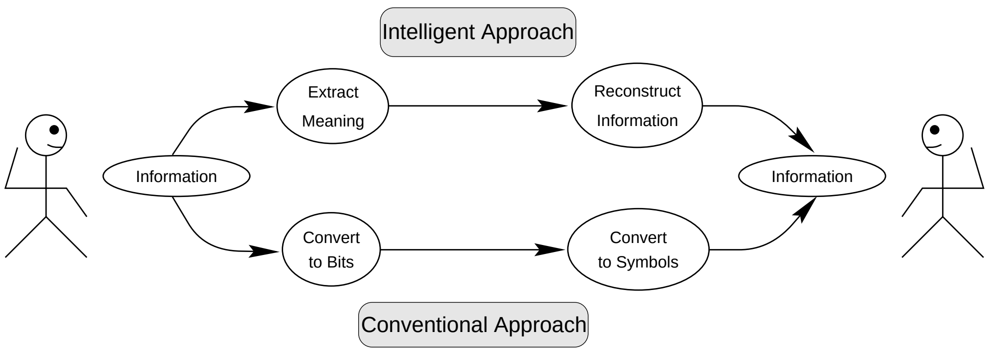

Figure 1: Conventional vs. intelligent communication.

In this paper, intelligent communication techniques for transmission of text and image data over low bit rate channels are proposed. For text, we propose a compression algorithm for natural English language text that takes into account the grammatical properties and meanings of the text as well as the statistical properties. By restricting the type of data to English text, we can increase the amount of compression possible.

For images, a technique to convert sequences of images into sequences of commands is described. Instead of transmitting the actual images, motion descriptions of the image relative to the previous image are sent. This results in high compression.

II Text Compression

A System Description

The communication system that is considered in this paper is shown in Fig. 2. We assume that two English speaking people are communicating over a noise–less communication channel using written text. This situation can arise, for example, when two people are communicating through a computer network by typing at a computer terminal.

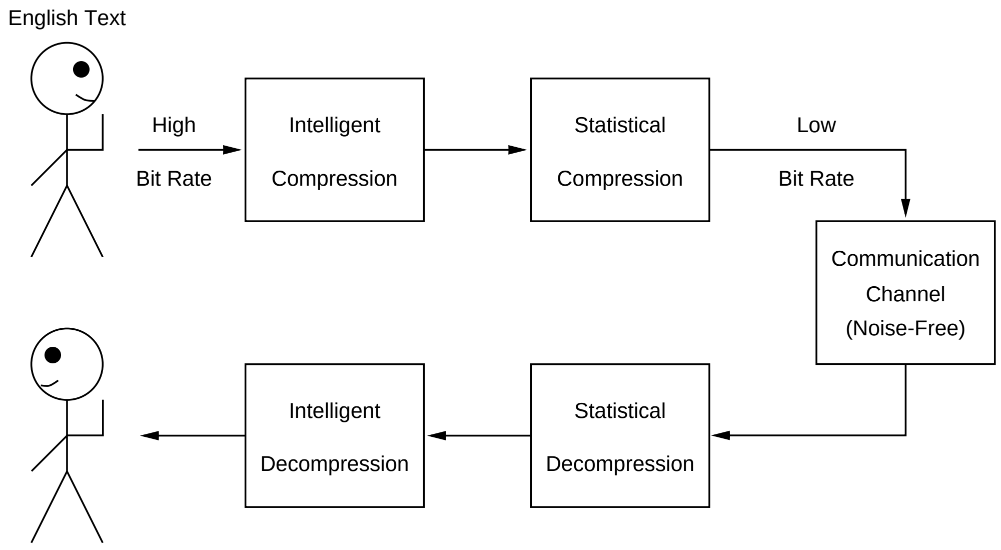

Figure 2: Communication system model.

This scenario can also be interpreted as storage of English text files on a computer disk. In this case, the transmitter section of Fig. 2 corresponds to compression and writing of the text file, while the receiver section corresponds to decompression and reading of the text file.

When we refer to a “noise–less” communication channel, we also include those noisy channels that use channel coding to reduce the probability of bit error to a negligible level. In this case, the noise–less channel would refer to the combination of the noisy channel and the channel encoder / decoder pair.

We wish to consider only noiseless channels so that we can focus on the compression problem. Of course, it is possible to consider the joint problem of compression and channel coding, but that is beyond the scope of this paper.

The compression part of the system is divided into “intelligent” compression and “statistical” compression. The intelligent compression part of the system removes characters from the text until no more characters can be removed without changing the meaning of the text. In this type of system, the intelligent compression part is lossless in the sense that the meaning of the text remains the same, but is lossy in the sense that the original text may not be exactly reconstructed by the decompression algorithm.

The statistical compression part of the system is a lossless, one–pass universal compression scheme. In this paper, we use the Lempel–Ziv (LZ) algorithm [1] [2]. LZ schemes work by creating a dictionary of character sequences. Therefore, roughly speaking, text that has fewer different types of character sequences can be compressed to a higher degree. Generally, text that has fewer types of characters has fewer different character sequences. As an example, consider text consisting of only the characters “a” and “b” versus text using “a”, “b” and “c”. For the former case, the possible one and two character sequences are the following six: “a”, “b”, “aa”, “ab”, “ba” and “bb”. For the latter case, there are 12 possible one and two character sequences.

From the above description, we can see that there are three compression mechanisms at work. The first is the obvious reduction in text that can be achieved by removing characters. The second is the compression of the text by the statistical compression algorithm. The third is the improvement in the performance of the statistical compression algorithm by reduction of the types of character sequences. To maximize the combined compression ratio, the intelligent compression algorithm should be designed to produce text that is matched to the statistical compression algorithm. In this paper, as a first step, we consider the two compression algorithms independently.

At the receiver side, first, the universal compression operation is reversed with a decompression algorithm. Second, the original text is estimated with an intelligent decompression algorithm. The goal of the intelligent decompression algorithm is to reproduce the text so that its meaning is the same as the original text.

B Intelligent Text Compression

In general, in order to compress the entire text, words that are deemed to be unimportant to the overall meaning of the text can be deleted. This requires comprehension of the text. A simpler, but less efficient, approach is to treat the text on a word by word basis. Here, the unimportant letters in each word are deleted. If the meaning of the text is considered on a sentence basis, the letters in a word that are unimportant are those letters that can be deleted without making it impossible to regenerate the original sentence.

In general, this can be thought of as a mapping from “English word” space to a “character string” space as shown in Fig. 3. As shown in the figure, the mapping is in general a many-to-one mapping. We assume that the mapping can be reversed so that the possible words corresponding to a particular character string are known.

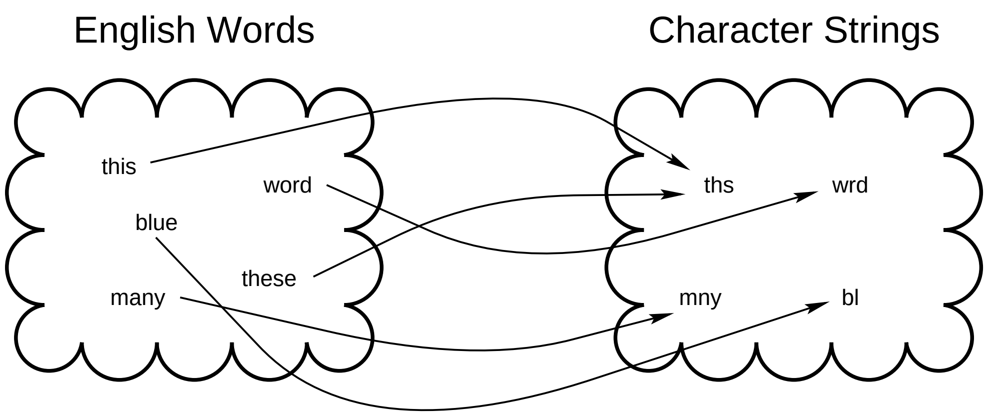

Figure 3: Mapping from “English word” space to “character string” space.

A well known shorthand technique for English is to omit vowels, i.e., ’a’, ’e’, ’i’, ’o’ and ’u’, when writing. This is due to the fact that vowels are not as important as consonants for comprehension of a sentence. Therefore, as a first step, we use this as a basis for compression. Specifically, we delete the vowels in a word to obtain its compressed version. The mapping for English words to character strings is done by storing the words known by the compression algorithm in a database along with their compressed versions.

The intelligent compression algorithm is shown in Fig. 4. First, a word is read from the input stream. Second, the word is checked against a database to see if the word is known. If the word is known, then the compressed version of the word is read from the database. If the word is not known, then the word is not compressed. Third, the compressed word, or the original word if it is not known, is sent to the output stream.

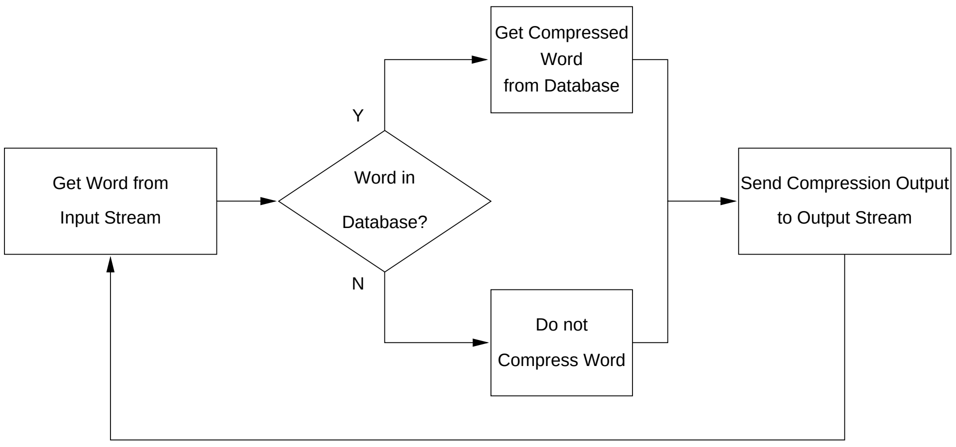
Get Word from Input Stream
Word in Database?
Y
N
Get Compressed Word from Database
Do not Compress Word
Send Compression Output to Output Stream

Figure 4: Intelligent compression algorithm.

C Compression Results

To estimate the compression improvement obtained by the intelligent compression algorithm, we compared the size of several English text files to the size of the files after intelligent and statistical compression and to the size of the file when only statistical compression is done. The results are shown in Table 1. The improvement shown in the table is the difference in compression ratios between combined IC & SC and SC only. Here, we define the compression ratio as the size of the compressed file expressed as a percentage of the original file size, i.e.,

$$\text{Compression Ratio} = \frac{\text{compressed file size}}{\text{original file size}} \times 100\%. \tag{1}$$

From the table, we can see that the combined intelligent and statistical compression system improves the compression ratio by an average of roughly 7.0%.

File 2 and file 4 are truncated versions of files 1 and 3 respectively. Notice that the size of file 2 is the same as the size of file 1 after intelligent compression. Likewise, file 4 is the same size as file 3 after intelligent compression. Comparing the results for files 1 and 2, we can see that statistical compression of file 1 after intelligent compression is less efficient than statistical compression of file 2. In other words, the efficiency of the statistical compression algorithm has been worsened by deletion of vowels. This indicates that vowel deletion does not produce suitable character strings for LZ–type compression. However, the overall system does result in an improvement in compression performance.

D Intelligent Text Decompression

The intelligent compression algorithm produces character strings that may correspond to more than one English word. In order to decompress the text, the grammatical properties and meaning of the words must be used in order to determine which word is the correct one. This process is outlined in figure 5.

In order to use the grammatical properties of the text, a complete sentence is required. Therefore, the decompression algorithm reads character strings until the end of the sentence is detected before any decompression is done. Once the character strings that form a complete sentence have been read, the possible sentences are generated.

Since there may be more than one word that corresponds to each character string, there is in general more than one possible sentence corresponding to the received character strings. The number of possible sentences is given by the product of the number of possible words for each character string, i.e., for a sentence made up of K character strings, the number of possible sentences is given by

$$N_s = \prod_{i=1}^{K} N_w(i), \tag{2}$$

where $N_w(i)$ is the number of possible words for the ith character string.

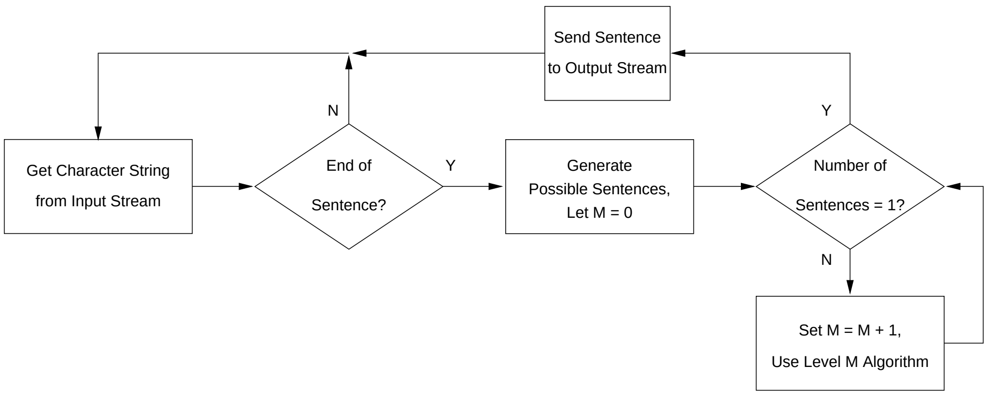
Send Sentence to Output Stream
N
Y
Get Character String from Input Stream
End of Sentence?
Y
Generate Possible Sentences, Let M = 0
Number of Sentences = 1?
N
Set M = M + 1, Use Level M Algorithm

Figure 5: Intelligent decompression algorithm.

The goal of the decompression algorithm is to reduce the number of possible sentences to one. Here, we use a multi–level algorithm, consisting of several sub–algorithms, to reduce

Table 1: Compression Comparisons

Size	File 1	File 2	File 3	File 4
Original	53533	40730	65746	47022
After IC Only	40730	30823	47022	33621
After SC Only	17395	13396	26685	19693
After IC & SC	14496	11131	21327	15648
Improvement	5.4%	5.6%	8.1%	8.6%

IC: Intelligent Compression
SC: Statistical Compression (LZ77)

the number of possibilities step by step. This scheme has the advantage that the time required to decompress the character strings can be shortened for simple sentences.

The level zero sub–algorithm simply generates the possible sentences. The higher level sub–algorithms gradually increase in complexity by using grammar and word meanings. Finally, as a last resort, the highest level sub–algorithm arbitrarily chooses one of the remaining sentences. This level is necessary in case none of the lower level sub–algorithms can reduce the number of possible sentences to one.

E Test Program

In order to evaluate the above decompression algorithm, we created a C program. A number of simplifications are made so that the basic ideas can be tested. Later, these simplifications will be removed so that the program can deal with a wider range of English text.

The program uses a database of words that also includes the words' grammatical type and compressed versions. The meanings of the words are not included in the present database in order to simplify the program. The meanings of the words will be included in a later implementation so that a higher level sub–algorithm can be added to the decompression system.

We assume that the text is grammatically correct so that grammatical techniques can be used. To aid the grammatical sub–algorithm, the grammatical types of the words are divided into categories such as noun and adjective. A complete list is given in Table 2. The verb categories may be further divided by using the tense of the verb, e.g., past tense, but this is not considered in the present program.

Table 2: Grammatical Word Types

verb (transitive)	adjective
verb (auxiliary)	adverb
verb (intransitive)	article
noun (countable)	pronoun
noun (uncountable)	preposition
noun (proper)	conjunction

On the compression side, the program only compresses words that are contained in the database. All other words are considered to be proper nouns. A database updating function is planned so that unknown words can be added to the database.

To simplify the program further, we assume that only certain sentence types are possible. These are listed in Table 3. In these sentence types, the subject and object may consist of a noun or an adjective and a noun. In either case, an article may also be present at the beginning of the subject or object.

Using the test program, we were able to confirm that the decompression algorithm worked successfully for a limited vocabulary. The program could not uniquely determine the original sentence in cases when the two words of the same type were converted to the same character string by the compression algorithm, e.g., the words "beer" and "bear" are both compressed to "br" and are both nouns. This problem can be corrected by using separate character strings for the two words, e.g., "beer" could be converted to "ber" and "bear" can be converted to "bar". This results in only a slight performance loss.

III Image Compression

A System Description

The image transmission system is shown in figure 6. The images that we consider here are facial images. A CCD camera connected to a workstation is used to load the image. The workstation then processes the image and converts it to a series of low rate commands that describe the movement of the image. This information is transmitted over a low rate channel and then the original image is reconstructed using a workstation at the receiver.

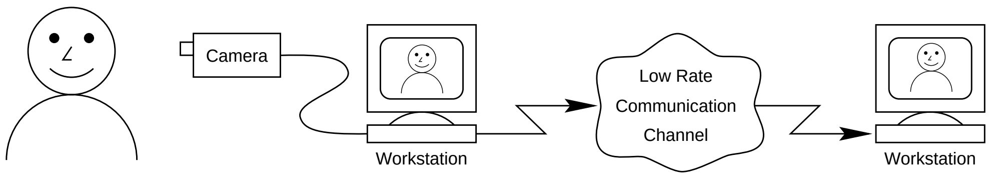

Figure 6: Intelligent image transmission system.

B Intelligent Image Compression

A block diagram of the compression method is shown in figure 7. First, a reference image is transmitted. This provides the receiver with an image which can be used to reconstruct the subsequent images. Next, image frames are scanned by the camera and compared to the previous image. The workstation extracts movement information such as “head tilted sideways” and “eye closed”. This information is converted into a set of commands and transmitted.

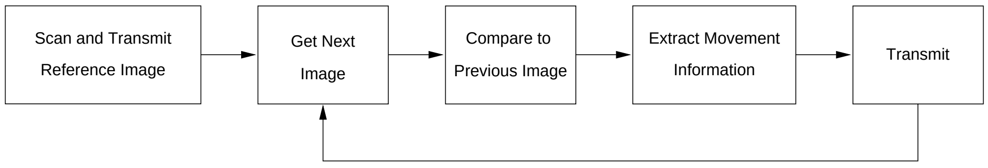

Figure 7: Intelligent image compression algorithm.

An 8–bit colour image of 200 by 200 pixels transmitted at 30 frames per second requires a data rate of 9.6 megabits per second. Using the proposed method with 2–byte commands, the images can be sent at a rate of 480 bits per second.

Table 3: Sentence Types

Subject - Verb	
Subject - Verb - Object	
Subject, Object:	(article) noun
	(article) adjective - noun

C Intelligent Image Decompression

A block diagram of the decompression technique is shown in figure 8. The reference image is first received and stored. Next, the movement information for each frame is received. The receiver uses the movement information and the previous image to reconstruct the current frame from the reference image.

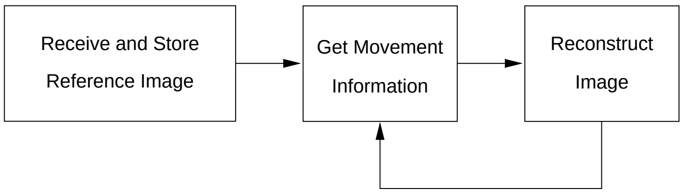

Figure 8: Intelligent image decompression algorithm.

IV Conclusions

In this paper, we have proposed compression techniques for text and image data to facilitate transmission over a low bit rate channel.

For text, we have proposed a compression algorithm specifically designed for English. The algorithm uses a combination of intelligent and statistical compression to take advantage of the redundancy in English text. Using this algorithm, we can compress English text 5% to 8% more than conventional universal compression algorthms such as Lempel–Ziv.

Decompression of the compressed text is accomplished by using a database of words and their properties. The text is processed in units of sentences so that the grammatical properties of the text can be used. The actual algorithm is a multi–level one. This approach can reduce the decompression time for simple sentences and also makes it easy to add new decompression techniques.

A test program was written to evaluate the compression algorithm. Results showed that the algorithm operated successfully for a limited number of words and sentence types.

As for images, a technique to convert sequences of images into sequences of commands was proposed. Instead of transmitting the actual images, motion descriptions of the image relative to the previous image are sent.

References

[1] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” *IEEE Trans. on Information Theory*, vol. 23, pp. 337–343, Mar. 1977.

[2] J. Ziv and A. Lempel, “Compression of individual sequences via variable–rate coding,” *IEEE Trans. on Information Theory*, vol. 24, pp. 530–536, May 1978.