Combined Data Compression and Error Correction

S.Sudhan and M.A.Ambroze

Fixed and Mobile Communications, University of Plymouth, Plymouth, UK
e-mail: M.Ambroze@plymouth.ac.uk

Abstract

The primary objective of the project is to familiarize with the concepts of data compression and error correction. The project intends to analyze the impact of data compression on error propagation. As the data is compressed more, error propagation through the transmission channel increases. When this data is decoded, it will bear no resemblance to the original data being transmitted. To reduce error propagation, it is necessary that the data is not compressed to a great extent. Another main aspect which the project intends to investigate is the best combination of compression and error correction that is likely to make data being transmitted to be sufficiently compressed and also less prone to errors.

Keywords

Data compression, Error Correction, LZW, LDPC.

1 Introduction

In the field of information theory, data compression and error correction are two coding subsystems, which though theoretically diverse seem to be codependent on each other. Source coding is the process of removal of redundant information from the input sequence where as error correction introduces redundant information as a means to safeguard data from error introduced due to error correction. Error correction can hence be considered as a necessary evil. It negates the very purpose of compression by increasing the size of compressed data. But if completely ignored can cause serious errors particularly in the case of text data. The level of correction depends on various factors like type of data being compressed, the level of compression required and also on the accuracy of the received data. Even though these two techniques are integral part of information theory, their diverseness requires them to be studied separately for better understanding. Initial part of this literature review will be dealing with data compression.

The project intends to investigate very important aspects which have a direct bearing on how degree of compression and integrity of data being transmitted are affected. The main purpose of such an undertaking is to identify factors that affect compression and error correction in addition to understanding the core concept about its working.

The project will follow a straight forward approach.

- Detailed research on various coding techniques used for data compression and error correction.
- Select one coding algorithm each for data compression and error correction.
- Divide the project into 3 parts
    1. Data compression: This part will deal with data compression alone. Using the selected coding algorithm various text sources will be compressed, transmitted (error introduced randomly) and decompressed. Degree of compression will be observed. By analyzing the results, the impact of data compression on error propagation can be investigated. The actual implementation of the algorithm would require the development of software using C language programming.
    2. Error correction: This part of the project will deal with error correction. Using the selected coding algorithm, the uncompressed text source will be coded (redundancy introduced), transmitted (error introduced) and finally decoded. The process is repeated by varying the number of redundancy bits introduced during coding. The extent of error correction achieved in each case will be observed.
    3. Combined data compression and error correction: The proposed digital communication system will be implemented and simulated by means of software developed for this purpose. The input text source will be compressed to varying degree. In addition to this the degree of error correction will also be varied. Detailed analysis of the simulation results will help in arriving at a proper conclusion regarding the best combination of data compression and error correction.

The report will begin off by delving in to the background of the two coding subsystems which as mentioned earlier are integral part of information theory. Thereafter the report goes on describe the research methodology that was undertaken to achieve the proposed objectives. This section also includes some results that were achieved during the course of trying to gain better understanding about the subject. Results of the simulation obtained by running the program was then analysed and presented towards the end of the report.

2 Overview of Data Compression and Error Correction:

Data Compression: One of the earliest forms of text compression was MOS code, which was invented in 1838. This was one of the simplest form using short codes for most often repeated letters. But the actual birth date of information theory is considered to be 1948. It was then that Claude E. Shannon published his epoch making paper on limitations of trustworthy data transmission over untrustworthy channels and also proposed various methods to achieve these limits. The paper also included the concept of information theory. It was actually the first known work to have established bounds for maximum amount of information that could be transmitted over an untrustworthy channel. The very foundation of data compression

was laid by Shannon and Robert Fano. They together introduced a method of compressing data making use of probabilities of blocks.

This work was later overshadowed by a more efficient coding system – Huffman coding. Huffman coding (1952) is similar to Shannon coding. It can perform compression by reducing redundancy in coding of symbols. It was found to be one of the most efficient fixed length coding methods going around. Or so it was, until the emergence of arithmetic coding which made the idea of replacing input symbol with a specific code obsolete. Instead, it introduced a new and innovative idea of replacing stream of input symbol with a single floating point number. But the main drawback of this method was that a long and complex information required more bits.

More recently dictionary based compression algorithm have become popular. This technique is entirely different from various other compression techniques that were available at the time of its introduction. These coding algorithms encode variable length strings of symbols as single tokens. The token will then form an index to a phrase dictionary. In case the tokens are smaller compared to index phrase, the tokens will replace the phrase. In this way compression is achieved.

In 1979, Abraham Lempel and Jacob Ziv took a different approach to symbol coding. They assigned code words to source words that were repeating or to patterns in a text. This approach was different from the usual approach to assigning code words to symbols in advance. Such was the success of this method of compression that it is still the basis of modern lossless data compression.

In 1984, Terry Welch improved the Lempel-Ziv method, which was known as LZ78 and renamed the method as LZW. This algorithm is now being used by commercial compression software such as PKZIP and by mainframe image formats GIF and some versions of TIFF. (Wolframscience, 2002)

Error Correction: Performance levels that are achieved in case of coded communication system are clearly stated in popular theorems proposed by Claude Shannon, who can well be considered as the father of information theory. He, in 1948, laid the foundations of information theory in a paper entitled “A Mathematical theory Of Communication”. These theorems not only define the limits on efficiency that can be achieved with information theory but also the importance of coding in achieving these limits. The paper published by Shannon in 1948 formulated statistical evaluation of various problems faced in communication, based on earlier works of Hartley, Weiner, Rice, and Kotel’nikov. Shannon’s work contradicted earlier notion that noise places a limit on accuracy that can be achieved in communication. Shannon showed that various communication channel characteristics such as noise level signal power and bandwidth determines a very important parameter called channel capacity represented by C. the channel capacity actually sets an upper limit on the rate at which information transmission is reliably transmitted and received through the channel. Shannon’s work proved beyond doubt that probability of error in the information being transmitted can be made very low by use of long coded transmission signals if the transmission rate of the information is less than the channel capacity C. this shows that noise places limit only on the

information transmission rate and not on the accuracy of the delivered information. (Michelson and Levesque, 1985)

3 Proposed System:

Information source: The information source in this case is a text source. The text will be read manually.

Data Compression (Source coding): Ideally it is preferable to represent source information by the smallest number of bits possible to make the digital communication system efficient. This is done by removing as much redundancy as possible from the source before transmission. This process of compression is also known as source coding.

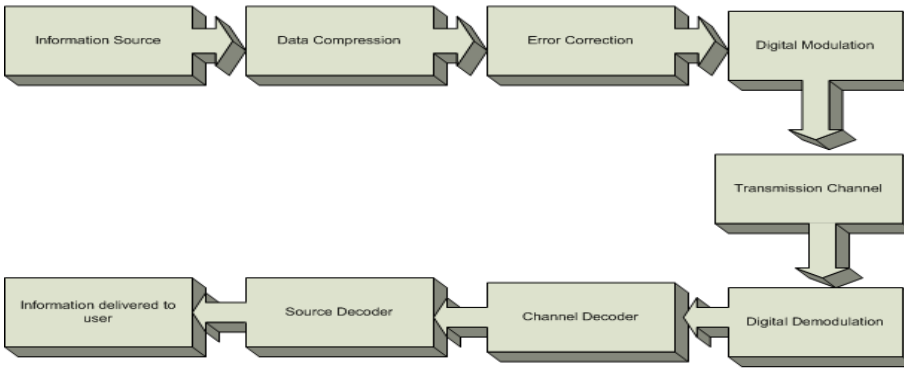

Figure 1: Proposed Digital Communication System

Error Correction (Channel encoder): The encoder receives information bits at the rate R_s and adds redundant bits producing encoded data having a higher rate of R_c. While encoding with a block code, the encoder accepts information in blocks of k bits size and for each k bits it will generate a block of b bits. Typically $n \geq k$. (Michelson and Levesque, 1985)

Digital Modulator: The purpose of the digital modulator is to match the encoder output with that of the transmission channel.

Transmission Channel: Transmission channel is the terminology used to include all the operations that are required to prepare data for transmission in physical channel, the transmission medium and reception operations necessary to bring the received signal upto the point of demodulation. Transmission channel is mainly responsible for introduction of errors in the data being transmitted. Hence during the implementation of the system, introduction of error will occur in this part of the digital communication system.

Digital Demodulation: Demodulation provides an interface between the transmission channel and the function that makes an estimation of the data being

transmitted to the user. The demodulator processes the received data and produces a set of numbers representing an estimate of transmitted symbol. (Michelson and Levesque, 1985)

Channel Decoder: The decoder performs the conversion of demodulator output into symbol decisions that reproduce the actual data being transmitted as accurately as possible. (Michelson and Levesque, 1985)

Source Decoder: The sequence of symbols coming out of channel decoder is received by the source decoder and it then tries to reproduce the original information contained in the original data being transmitted.

4 Research

The project required the entire digital communication system to be simulated by writing appropriate codes in C language. Before the actual programming was undertaken it was necessary to do extensive research on both data compression and error correction. Extensive research was done on both data compression and error correction. From the invention of MOS code in 1838, to actual birth of information theory in 1948 inspired by Claude E. Shanon through to the modern era of commercial compression softwares such as PKZIP and highly efficient error codes such as turbo codes were extensively researched and understood.

The next major objective or rather a dilemma which had to be overcome before starting the project was to decide on the compression technology to be implemented in the project. Of the many compression techniques available, LZW method was chosen. It is not the best encoding algorithm available but for the purpose of the project (which is mainly to study the best combination of compression and error correction), LZW appeared to be lucrative due to its ease of implementation. Having decided on LZW as the means for data compression, an in depth study was done to better understand the LZW technology. The future implementation would require quite an extensive knowledge about the working of LZW technique. After going through coding and decoding algorithm, many examples of LZW coding and decoding were done to familiarize with the process and gain a thorough knowledge.

As the project involves studying the effects of compression and correction in real time transmission it is highly imperative that effect of error propagation not be overlooked. For viewing the effects of error propagation during compression/decompression process, some of the bits of the code were randomly changed and then decompressed. The effects were found to be quite impressive. The change of a single bit produced a rather faulty data after decompression, which bore no resemblance with the actual data being compressed.

The most important part of the project i.e. implementation of the proposed digital communication system involved programming in C language to simulate the entire operation of compression, noise, error correction and decompression. As mentioned earlier of the many compression techniques available LZW was chosen. The LZW algorithm is easy to understand but implementation is not that easy some of the

implementation issues faced while developing the coding for LZW are listed in the commencing section.

4.1 Implementation issues of LZW:

4.1.1 Size of Code Word

Main thing that needs to be settled during the implementation phase of LZW is deciding the size of code word that needs to be used. Some of the things that need to be taken into consideration when deciding on the size of the codeword being used are:

- Code words generally need to be bigger than a length 1 string of the string being encoded.
- Each and every encoded string requires a code word for representing them.
- Code words which are bigger size generally imply that there are more entries in the dictionary. (Dipperstein, 2008).

4.1.2 String representation in the Dictionary

The dictionary in the Lempel-Ziv-Welch algorithm has a unique way of associating strings with code words. Even though code words are of limited length, the LZW algorithm does not enforce a limit on the string length that is encoded.

Strings of arbitrary length are represented by a null terminated array. There may be cases when there may be a large number of strings which may amount to thousands of bytes in length. With the increase in the machines memory size, the memory requirements of the null terminated strings become negligible. (Dipperstein, 2008).

4.1.3 Dictionary Layout

Deciding the dictionary layout affects locating and inserting strings. The strings are stored in the dictionary as a codeword prefix and a byte suffix.

In the beginning, the dictionary has only one entry for a character string. As the process of encoding proceeds, the strings contained in the dictionary expand. There is however an upper limit on the number of strings that can be accommodated in a dictionary. It depends on the size of the code word. If the code words are n bits long, there can be up to 2^n unique code words. (Dipperstein, 2008)

4.1.4 Memory complexities:

The advantage of using 14 or 15 bit long codes is that it gives better compression ratios for large files as they have larger string table to be utilized during it operation. It does however affect the performance drastically for smaller files. On the other hand compression of long files cause the compression ratio to degrade gradually as more of the file is read in. The reason for this is quite obvious. As the string table is of finite size, on adding more strings to the table, it becomes full and no further

strings can be added to it. String table can only function properly for the part of the file that was read in when the table was built. Remainder of the file has different features and hence requires a different string table.

5 Results and Discussion

The input parameters that are required are:

- Input file
- Index size
- Error correction encoding enable or disable
- If encoding enabled: matrix size and noise level

The output parameters are:

- Original file size
- Compressed file size
- Compression ratio
- BER
- PeR

In the commencing section we will be seeing how each of these parameters is related to each other by performing actual compression-error correction process by running the program. The results are plotted in graphs and the results analyzed.

5.1.1 Compression and Decompression Only:

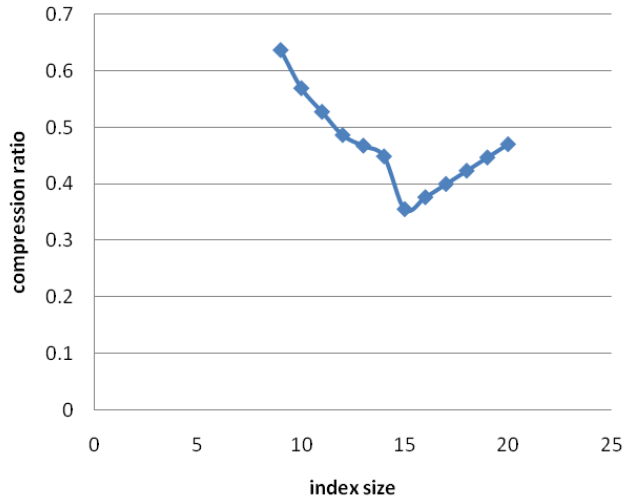

Figure 2: Compression ratio vs index size

- Variation of compression ratio with index size: The level of compression varies with the size of codeword being used. The source data was compressed to varying degrees by altering the index size. This helps us to arrive at the best compression for the source data. The source data used in this case was a file named 1.txt. The file size was about 188.7 KB. The index size was varied from 9-20. The results were plotted with index size on the x-axis and the compression ratio on the y-axis.

  From the graph it becomes clear that the compression ratio decreases with increase in the index size. The compression ratio decreases almost linearly with index size. The best compression occurs at the index size of 15. Beyond that, the compression goes on increasing. Thus, in order to achieve the best compression, the index size selected should be 15. In reality, this shift in behavior from linear decrease to linear increase of compression ratio with index size is due to the wastage of space caused due to use of large code words (above 15) hence the best compression for the file 1.txt occurs when index size of 15 is used.

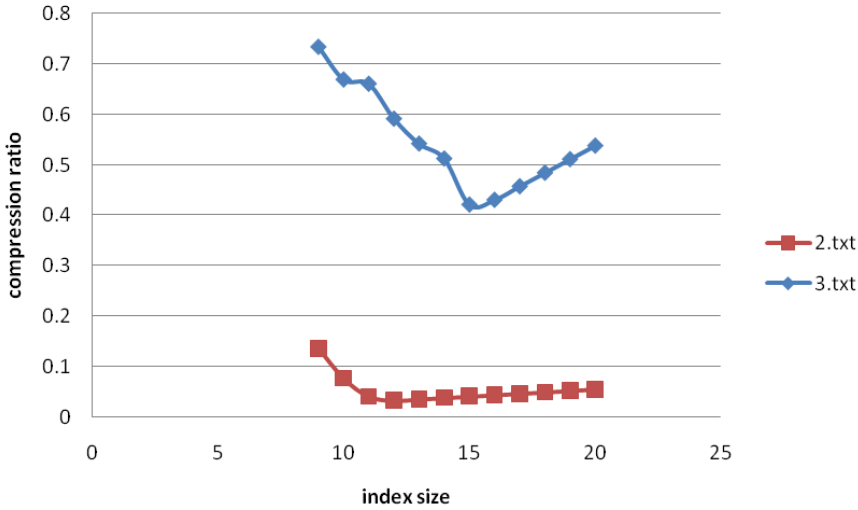

Figure 3: Compression Ratios for different text files

- Compression of different types of text files: Different text files were selected, both of similar size (189 kB) but having different contents. The two files selected were 2.txt and 3.txt. The file 2.txt has more repetition of words than the contents of 3.txt. Hence as a consequence, the file 2.txt is compressed more than the file 3.txt.

  For the text file 2.txt the best compression was achieved at index size of 12 and for 3.txt the best compression was achieved at index size of 15. This is because the dictionary size in the case of 2.txt is much smaller than 3.txt. So if large code words are used there will be wastage of space. This is the

reason why compression occurs at smaller index size for 2.txt as compared with 3.txt which has larger dictionary with more words.

- File Size v.s. Compression ratio:

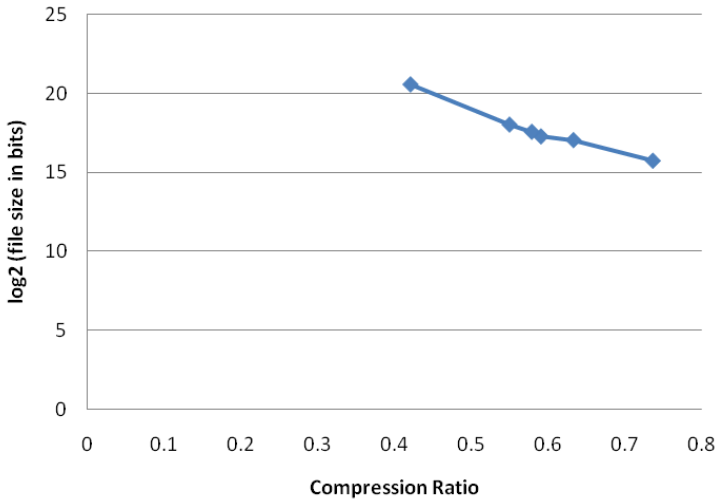

Figure 4: File size vs Compression Ratio

The graph shows the variation of compression ratio when files of different sizes are used. The graph was plotted using files of sizes ranging from 6 kB to 192 kB. The codeword length or the index size was set at 15 which was neither too small nor too big a value. The file sizes were converted to bits and expressed in terms of $\log_2$. On plotting the graph it was found that compression ratio varies inversely with file size i.e. higher the file size, lower was the compression ratio. This means that file of lager sizes were compressed more in comparison with smaller files.

Limitations: Analysis on Error Correction was not possible as LDPC was not properly implemented. Code for LDPC was taken from Ioremap (Ioremap, 2009). Due to compatibility issue with the self developed LZW program, the results obtained for BER and PeR were not consistent.

6 Conclusion

The project was undertaken mainly to familiarise with the basic concepts of compression and error correction, in the process investigating how various parameters has a direct bearing on the efficient transmission of compressed data. Detailed analysis couldn't be carried out due to time constraints as majority of the allotted time was spent on developing the coding. However, most of the objectives were achieved and the results were analysed. The best compression for the file under consideration was found out by trial and it was found to be dependent on the size of the code word. Too large a codeword affects the compression as space is

unnecessarily wasted while too small a codeword doesn't help in realising compression to the full extent. Other results obtained were how different files of same size having different contents get compressed to different levels. Finally the variation of compression ratio with size of the file being compressed was also plotted and conclusion was drawn that for a particular codeword length, better compression can be achieved for larger files.

As already mentioned there are a lot of techniques available for compression as well as error correction. Each of these techniques has undergone a lot of modifications over the years. Still researches are being done on each of these coding techniques as there is still a huge scope for improvement. New, improved and efficient coding techniques are being sought out by researchers in the field of information theory. When compression and error correction are considered separately, the advancements made in each are tremendous. But, as mentioned at the start, these two coding subsystems are inseparable in real world communication system. There is always a trade off between accuracy and the compression. The best combination of compression and error correction hasn't been established yet. Several studies are being conducted to realize this. Eventually, the best combination would depend on the level of accuracy required.

7 References

Dipperstein, M. (2008) "Lempel-Ziv-Welch (LZW) Encoding Discussion and Implementation". Accessed: 10/01/09[Online]. Available at: http://michael.dipperstein.com/lzw/index.html#strings

Gallager, R. G. 1978. "Variations on a Theme by Huffman". *IEEE Trans. Inform. Theory 24*, 6 (Nov.), 668-674.

Michelson, A.M. & Levesque, A.H. (1985) "Error Control Techniques for Digital Communication", John Wiley & Sons, Inc.

University of Birmingham. (2008) "A Brief History of Data Compression" Accessed: 10/06/08[Online]. Available at: http://www.eee.bham.ac.uk/WoolleySI/All7/intro_3.htm

Welch, T.A. (1984) "A Technique for High Performance Data Compression", IEEE Computer, Vol. 17, No. 6, pp. 8-19.

William, R.N. (1990) "Adaptive Data Compression" Kluwer Academic Publishers, London

Wolframscience. (2002) "Some Historical Notes" Accessed: 10/06/08 [Online]. Available at: http://www.wolframscience.com/reference/notes/1069b

Ziv, J. & Lempel, A. (1977). A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 23(3):337–343.

Ziv, J. & Lempel, A. (1978). Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory, 24(5):530–536.