Lossless text compression using GPT-2 language model and Huffman coding

Md. Atiqur Rahman[1,*] and *Mohamed* Hamada[1,†]

[1]School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu City, Fukushima, Japan

Abstract. Modern daily life activities produced lots of information for the advancement of telecommunication. It is a challenging issue to store them on a digital device or transmit it over the Internet, leading to the necessity for data compression. Thus, research on data compression to solve the issue has become a topic of great interest to researchers. Moreover, the size of compressed data is generally smaller than its original. As a result, data compression saves storage and increases transmission speed. In this article, we propose a text compression technique using GPT-2 language model and Huffman coding. In this proposed method, Burrows-Wheeler transform and a list of keys are used to reduce the original text file's length. Finally, we apply GPT-2 language mode and then Huffman coding for encoding. This proposed method is compared with the state-of-the-art techniques used for text compression. Finally, we show that the proposed method demonstrates a gain in compression ratio compared to the other state-of-the-art methods.

1 Introduction

It is not easy to manage the increasing amount of data produced every day, especially in medical centers and on social media. To save the data in a digital device requires too much storage space. According to the research of the 6th edition of DOMO's report [**1**], more than 2.5 quintillion bytes of data are produced daily, and it is growing. It further estimates that approximately 90% of the world's data has been produced between 2018 and 2019. It also calculates that each person will create 1.7MB of data per second on the earth by 2020. There are three possible solutions to this problem, using better hardware or software or a combination of them. Nowadays, so much information is created that it is impossible to design new hardware competing with data creation because of the limitations of hardware construction reported in [2]. Therefore, developing better software is the only solution to the problem.

A solution from the software viewpoint is compression. A way of representing data using fewer bits than an original is called compression. Compression reduces the consumption of storage and bandwidth; and increases transmission speed over the cyber world [3-4]. Compression is applied in many areas such as audio, video, text, images,

[*]e-mail: atick.rasel@gmail.com
[†]e-mail: mhamada2000@gmail.com

Etc. Two types of compression are lossless and lossy. The less important and irrelevant data are removed permanently for lossy compression, whereas lossless preserves every detail. Lossless compression eliminates only statistical redundancy. In short, lossy allows a little bit of degradation in data and lossless reconstructs perfectly from its compressed form [5-6]. There are many applications of lossless compressions, such as electronic document compression, medical imagery, zip file format, and facsimile transmissions of bitonal images [7-9].

Our primary focus in this article is to compress a text file. Usually, Burrows-Wheeler Transform (BWT), Huffman coding, LZW (Lempel-Ziv-Welch), LZMA, Gzip, Bzip2, and Deflate are the most popular text compression algorithms [10-11]. Some statistical compression techniques such as Huffman coding, arithmetic coding put the shorter codes to the characters repeated more frequently. On the other hand, LZ77, LZW compress a text file based on a dictionary where a set of sub-strings is created and assigns them a pointer reported in [12-13]. Storer et al. in [14] show that though LZW is a good compression technique, its searching complexity is very high. Salomon in [15] shows that Deflate is Huffman and LZSS based text compression algorithm and provides better compression, but it is slow due to searching the longer and duplicate substrings. Gzip is an LZ77 and Huffman coding-based text compression algorithm and provides a speedy compression than Deflate reported in [16-17]. Rahman et al. show a Burrows-Wheeler transform (BWT), pattern matching, and Huffman coding-based text compression technique in [18] and Claims a better compression than Deflate, Bzip2, Gzip, LZMA, and LZW. It also shows that the method is a bit slow.

Burrows-Wheeler transform (BWT), a reversible technique that converts a set of characters into runs of similar characters [19], and the technique is used by Bzip2, a lossless text compression algorithm [20]. Though many text compression techniques have already been developed, current technology needs a more effective text compression strategy.

From this point of view, we propose a straightforward but efficient lossless text compression procedure using GPT-2 language model and Huffman coding in this paper. Our primary focus is on compression, not the speed of compression. There are three steps in our proposal. First of all, we split a large text file into a set of small files and then apply Burrows-Wheeler transform to each file individually to speed up transformation. Secondly, we use a list of uniquely defined keys to reducing the length of each file. Finally, the GTP-2 language model and then Huffman coding is applied for compression. We compare the proposed approach against some standard and advanced popular alternatives. In this article, background studies are discussed in segment 2. The proposed technique is explained in segment 3. In segment 4, we show the experimental outcomes and analysis. We finally conclude the paper in segment 5.

2 Background studies

Huffman is a lossless entropy coding technique that constantly generates a most effective tree [21] and works on the contrary path of Shannon-Fano coding. It sorts the probabilities in descending order and makes a tree connecting the two lowest probabilities each time, and the tree is finally used for encoding. There are two barriers to Huffman coding. First of all, it is extremely sensitive to error and may smash nearly the entire message for only modifying one or two bits in transmission. Secondly, it provides a relatively lower compression rate as it assigns a code-word for each pixel

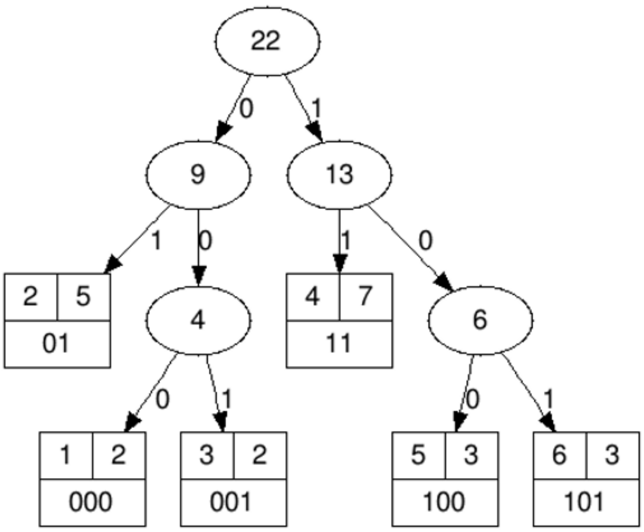

Figure 1: Huffman Tree

in the encoded matrix [22-23]. However, Huffman coding provides better compression than Shannon-Fano coding. A detailed analysis of Huffman and Shannon-Fano coding with a numeric example has been given in [3]. For the dataset A=[1 2 2 1 2 2 2 6 6 6 4 4 4 3 3 4 4 4 4 5 5 5], Huffman tree is shown in Figure 1. The figure shows that Huffman provides 2.4545 bits averagely for the data list A, which is 0.0122% less storage than Shannon-Fano coding.

Lempel–Ziv–Storer–Szymanski (LZSS), developed by James A. Storer and Thomas Szymanski, is a dictionary-based text compression technique and a derivative of LZ77 [24]. Deflate was created by Phil Katz in 1951 to compresses data using LZSS and Huffman coding together [25]. Deflate was covered by patents that led to developing another lossless text compression algorithm for its widespread use. For this reason, a deflate-based data compression algorithm called Gzip was developed for free use by Jean-loup Gailly and Mark Adler. The pseudocode of LZ77 is shown in the following Algorithm 1.

Generative Pre-trained Transformer 2 (GPT-2) is a language model that uses Byte Pair Encoding (BPE) techniques for compression. BPE technique chooses subwords instead of words or characters in a neural network reported in [26-28]. The pseudocode of Byte Pair Encoding (BPE) is demonstrated in the following Algorithm 2.

3 Proposed method

The most popular text compression algorithms are Gzip, Bzip2, Lempel–Ziv–Markov chain algorithm (LZMA), Brotli, Etc. Many of them concentrate on the compression ratios, while others focus on speed. In this article, we mainly focus on compression ratios. Burrows-Wheeler transform (BWT) takes a long time for a large file transformation reported in [18]. To solve the issue, we split a large text file into a set of small files and apply the BWT to each file to speed up the conversion in our proposed technique. Secondly, each converted text is reduced by keys. It has been analyzed

Algorithm 1: The pseudo-code of LZ77 algorithm

```
1  Take a text (input);
2  while input ≠ '' do
3      Find the longest prefix of the input in a window and assign it to PF;
4      if prefix ≠ '' then
5          Calculate the distance (X) from where the prefix has started and assign
             it to X;
6          Calculate the length of the prefix (Y);
7          Assign the character to Z that follows prefix in the input;
8      else
9          Assign 0 to X and Y;
10         Assign the first character of the input to Z;
11     end
12     Output a triple (X, Y, Z);
13     Move the cursor Y+1 positions to the right;
14 end
```

Algorithm 2: The pseudo-code of Byte Pair Encoding (BPE) algorithm

```
1  Split each word into a series of characters;
2  Find out the highest frequency patterns and perform the joining operation
     with the highest frequency patterns;
3  Repeat step 2 until it gets the predefined highest number of subword of
     iterations;
```

that GPT-2 model produces a Hangul Syllables list as an output that contains much fewer Hangul characters than its input text. We save the number of Hangul characters produced in each section for later reconstruction. Lastly, we combine the outputs of the GPT-2 segments and apply Huffman coding for encoding. Figure 2 shows the general encoding procedure of the proposed technique.

4 Experimental results and analysis

This section shows the experimental results of some of the most commonly used text compression methods and the proposed technique and explains the methods' overall performance. The most important thing is to determine the evaluation parameters. We evaluate the methods mentioned in this article based on the compression ratio defined in equation 1. There are many text compression methods used in real applications. However, we select Brotli, Bzip2, LZMA, and Gzip for comparison. We choose ten different text samples and apply the state-of-the-art and proposed techniques to them. We show the samples' compression ratios in Table 1, and Figure 3 demonstrates a graphical representation of the compression ratios for quick comparison.

Table 1 shows that, on average, Bzip2, Gzip, LZMA, Brotli, and the proposed techniques give 2.91, 2.5954, 2.8924, 2.7791, and 2.9066 compression ratios for the ten samples. The table demonstrates that Bzip2 and Gzip provide the highest and the lowest compression ratios. It is calculated that the proposed technique averagely provides 10.707%, 0.489%, and 4.387% better compression than Gzip, LZMA, and

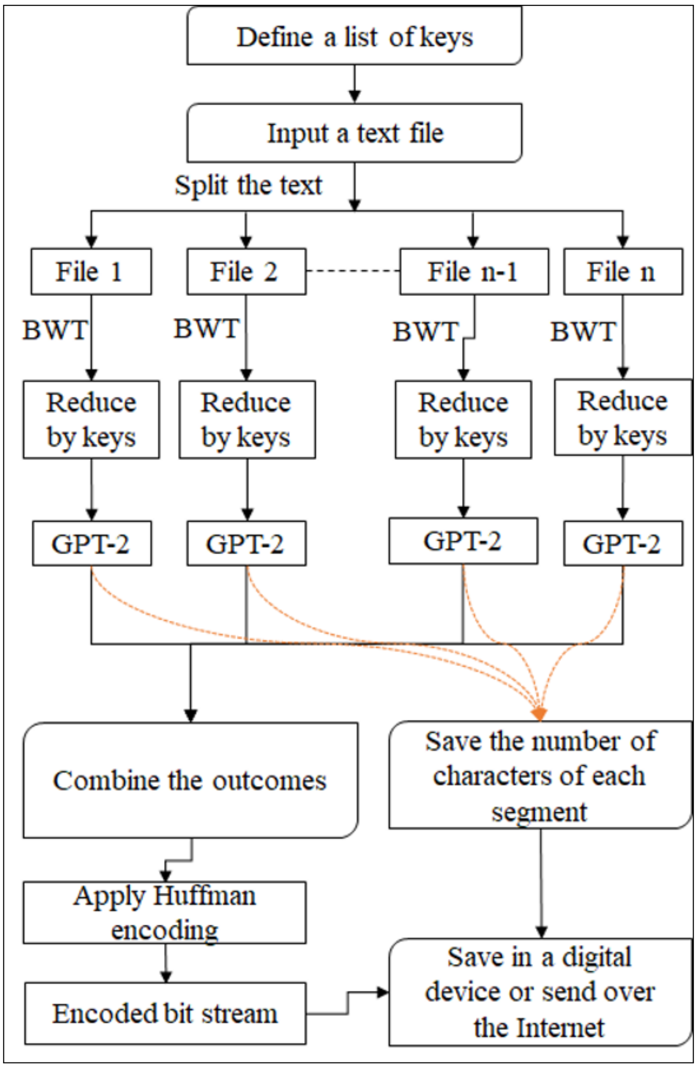

Figure 2: Proposed encoding technique

Brotli, respectively. However, Bzip2 shows 0.117% better results than the proposed technique. On average, though Bzip2 provides better compression than the proposed technique, the proposed technique sometimes shows a better result than Bzip2. As an example, the proposed technique provides 0.741%, 4.159%, and 0.133% more compression than Bzip2 for the text samples 2, 3, and 9, respectively.

$$CR = \frac{Number\ of\ bits\ of\ an\ original\ text}{Number\ of\ bits\ of\ compressed\ text} \qquad 1$$

5 Conclusions

This work proposes an easy yet effective text compression procedure using Burrows-Wheeler transform, GPT-2 language model, and Huffman coding. It is inspired as the

Table 1: Experimental compression ratios

Sample Texts	Bzip2	Gzip	LZMA	Brotli	Proposed
1	3.642	2.768	3.601	3.283	3.472
2	2.948	2.731	2.928	2.853	2.97
3	3.503	3.503	3.503	3.503	3.655
4	2.895	2.582	2.878	2.765	2.889
5	3.169	2.827	3.15	3.027	3.157
6	3.219	2.871	3.199	3.074	3.208
7	1.957	1.745	1.945	1.869	1.947
8	2.932	2.615	2.914	2.8	2.932
9	2.999	2.675	2.981	2.864	3.003
10	1.836	1.637	1.825	1.753	1.833
Average	**2.91**	**2.5954**	**2.8924**	**2.7791**	**2.9066**

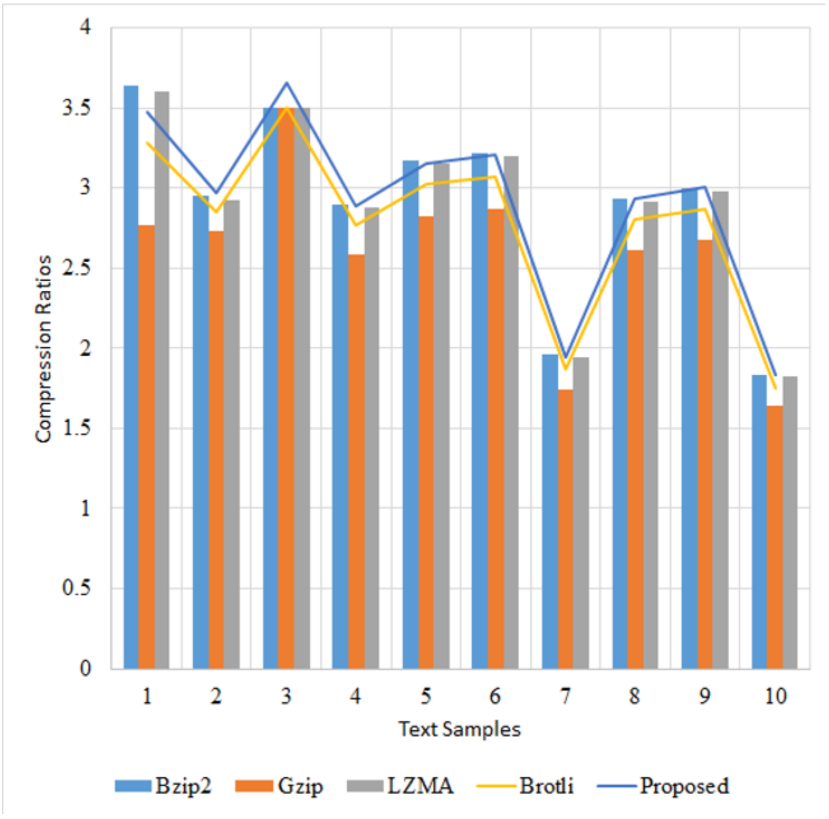

Figure 3: Comparison of compression ratios

GPT-2 works with Byte Pair Encoding and provides much fewer Hangul characters than the original one, and Huffman coding provides a better result for a small number of symbols. The experimental results show that the proposed technique averagely provides better compression than state-of-the-art techniques without Bzip2. However, the proposed approach shows better reduction for at least 30% of text samples than the Bzip2. Finally, we conclude that the proposed technique sometimes outperforms the state-of-the-art methods.

References

1 Domo.com. 2020. Becoming A Data-Driven CEO — Domo. [online] Available at: *https://www.domo.com/solution/data-never-sleeps-6 [Accessed 12 June 2020].*

2 Pan, W., Li, Z., Zhang, Y. and Weng, C., 2018. The new hardware development trend and the challenges in data management and analysis. *Data Science and Engineering, 3(3), pp.263-276.*

3 Rahman, M. and Hamada, M., 2019. Lossless Image Compression Techniques: A State-of-the-Art Survey. *Symmetry, 11(10), p.1274.*

4 Rahman, M.A., Shin, J., Saha, A.K. and Islam, M.R., 2018, June. A Novel Lossless Coding Technique for Image Compression. *In 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR) (pp. 82-86). IEEE.*

5 Sadchenko, A.; Kushnirenko, O.; Plachinda, O. Fast lossy compression algorithm for medical images. *In Proceedings of the 2016 International Conference on Electronics and Information Technology (EIT), Odessa, Ukraine, 23–27 May 2016; pp. 1–4.*

6 Pandey, M.; Shrivastava, S.; Pandey, S.; Shridevi, S. An Enhanced Data Compression Algorithm. *In Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Tamil Nadu, India, 24–25 February 2020; pp. 1–4.*

7 Bovik, A.C. ed., 2009. The essential guide to image processing. *Academic Press.*

8 Rahman, M.A. and Hamada, M., 2019, October. A Semi-Lossless Image Compression Procedure using a Lossless Mode of JPEG. *In 2019 IEEE 13th International Symposium on Embedded Multicore/Manycore Systems-on-Chip (MCSoC) (pp. 143-148). IEEE.*

9 Rahman, M., Hamada, M. and Shin, J., 2021. The Impact of State-of-the-Art Techniques for Lossless Still Image Compression. *Electronics, 10(3), p.360.*

10 Oswald, C.; Sivaselvan, B. An optimal text compression algorithm based on frequent pattern mining. *J. Ambient. Intell. Humaniz. Comput. 2018, 9, 803–822.*

11 Portell, J.; Iudica, R.; Garc´ıa-Berro, E.; Villafranca, A.G.; Artigues, G. FAPEC, a versatile and efficient data compressor for space missions. *Int. J. Remote Sens. 2018, 39, 2022–2042.*

12 Rahim, R. Combination of the Blowfish and Lempel-Ziv-Welch Algorithms for Text Compression; *OSF Storage: STMIK Triguna Dharma, Universiti Malaysia Perlis, 2017.*

13 Welch, T.A. A technique for high-performance data compression. *Computer 1984, 17, 8–19.*

14 Storer, J.A. (Ed.) Image and Text Compression; *Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 176.*

15 Salomon, D. A Concise Introduction to Data Compression; *Springer Science & Business Media: Berlin/Heidelberg, Germany, 2007.*

16 Nelson, M.; Gailly, J.L. The Data Compression Book, 2nd ed.; *M & T Books: New York, NY, USA, 1995.*

17 Gupta, A.; Bansal, A.; Khanduja, V. Modern lossless compression techniques: Review, comparison and analysis. *In Proceedings of the 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 22–24 February 2017; pp. 1–8.*

18 Rahman, M. and Hamada, M., 2020. Burrows–Wheeler Transform Based Lossless Text Compression Using Keys and Huffman Coding. *Symmetry, 12(10), p.1654.*

19 Burrows, M.; Wheeler, D.J. A Block-Sorting Lossless Data Compression Algorithm; *Systems Research Center: Palo Alto, CA, USA, 1994.*

20 Patel, R.A.; Zhang, Y.; Mak, J.; Davidson, A.; Owens, J.D. Parallel lossless data compression on the GPU. *In Proceedings of the 2012 Innovative Parallel Computing (InPar), San Jose, CA, USA, 13–14 May 2012; pp. 1–9.*

21 Sharma, M., 2010. Compression using Huffman coding. *IJCSNS International Journal of Computer Science and Network Security, 10(5), pp.133-141.*

22 Rufai, A.M., Anbarjafari, G. and Demirel, H., 2013, April. Lossy medical image compression using Huffman coding and singular value decomposition. *In 2013 21st Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.*

23 Rahman, M.A., Rabbi, M.F., Rahman, M.M., Islam, M.M. and Islam, M.R., 2018, September. Histogram modification based lossy image compression scheme using Huffman coding. *In 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT) (pp. 279-284). IEEE.*

24 Storer, J.A. and Szymanski, T.G., 1982. Data compression via textual substitution. *Journal of the ACM (JACM), 29(4), pp.928-951.*

25 Deutsch, P., 1996. RFC1951: DEFLATE compressed data format specification *version 1.3.*

26 Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I., 2019. Language models are unsupervised multitask learners. *OpenAI blog, 1(8), p.9.*

27 Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I., 2018. Improving language understanding by generative pre-training.*https://s3-us-west-2.amazonaws.com/openaiassets/research-covers/language-unsupervised/languageunderstandingpaper.pdf*

28 Sennrich, R., Haddow, B. and Birch, A., 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909.*