International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 7, pp 92-100, July-2026

https://doi.org/10.47001/IRJIET/2026.107010

Analysis of the Efficiency of Huffman and LZW Compression Algorithms on Vietnamese Text Data

[1]Nguyen Minh Hoang, [2]*Trinh Van Hung

[1]FPT High School - Vinhomes Metropolis, Hanoi, 100000, Vietnam. E-mail: fhl31829@gmail.com

[2]Hanoi Polytechnic College, Hanoi, 100000, Vietnam. E-mail: haphuong20132016@gmail.com

*Corresponding Author: Trinh Van Hung

Abstract **- This paper analyzes and compares the effectiveness of two representative lossless compression algorithms, Huffman coding and LZW, on Vietnamese text data encoded in UTF-8. A web-based text compression application was developed in JavaScript, implementing byte-level canonical Huffman coding and LZW with variable code width from 9 to 16 bits. The two algorithms were evaluated on a corpus of approximately 360 KB covering four text categories (encyclopedic, literary, legal–administrative, and scientific–technological) collected from Vietnamese Wikipedia and Wikisource, through three experiments: by text category, by data size, and by the influence of Vietnamese diacritics. The results show that LZW achieves compression ratios of 29.3–48.2% of the original size and becomes increasingly effective as data grows, clearly outperforming Huffman coding (64.3–68.6%), which is lower-bounded by the symbol entropy of the data (about 5.2 bits/byte). In contrast, Huffman compresses roughly four times faster (about 80 MB/s versus 20 MB/s). The experiments also indicate that the multi-byte UTF-8 encoding of Vietnamese script significantly reduces the effectiveness of byte-level Huffman coding while having almost no effect on LZW. Overall, these findings indicate that dictionary-based coding is better suited to Vietnamese UTF-8 text, whereas byte-level statistical coding is best reserved for speed-critical settings or as an entropy-coding stage after another transform; the compression tool and the four-category corpus built here are offered as reusable resources.**

Keywords: Huffman, LZW, lossless data compression, Vietnamese text, UTF-8, entropy.

I. INTRODUCTION

The rapid growth of electronic information systems in Vietnam—such as online newspapers, public-service portals, digital libraries, and repositories of legal documents—has driven an ever-increasing need to store and transmit ever-larger volumes of Vietnamese text. Lossless compression reduces storage size and transmission bandwidth while still recovering every byte of the original data exactly, and therefore plays a foundational role in most text-processing systems, databases, and communication protocols [6].

Among lossless compression algorithms, the two most classic and influential approaches are statistical coding and dictionary coding. The representative of the first approach is Huffman coding [1], which assigns short codewords to frequently occurring symbols and long codewords to rare ones, achieving optimality within the class of per-symbol prefix codes. The representative of the second approach is LZW [2] — a practical variant of the Lempel–Ziv family [3] — which replaces repeated symbol strings with indices into a dictionary built dynamically during compression. These two algorithms are still widely taught and remain core components of many common formats (GIF, TIFF, PDF, Unix compress...).

Prior work on Vietnamese text compression has mainly explored syllable- and word-level methods, such as syllable-based coding [10] and trigram-based compression [11], which exploit linguistic units above the byte level. However, to the best of our knowledge, no study has systematically compared the two classic byte-level algorithms, Huffman and LZW, on Vietnamese UTF-8 text using the same implementation, the same experimental platform, and a controlled corpus. Most existing quantitative evaluations of these two algorithms have been carried out on standard English benchmark corpora such as Calgary and Canterbury [8], whose statistical characteristics differ substantially from those of Vietnamese text. Vietnamese text has its own characteristics: the Vietnamese script uses a rich system of tone marks and diacritics, requiring most accented characters to be encoded with 2-3 bytes in UTF-8 [5], [9]. This multi-byte encoding changes the statistical distribution at the byte level and may influence the relative performance of statistical and dictionary-based compression algorithms. Consequently, it remains unclear whether conclusions drawn from English benchmark corpora can be directly generalized to Vietnamese UTF-8 text. This research gap motivates the present study and raises the following question: which algorithm is truly better suited to Vietnamese UTF-8 text, and how does UTF-8 encoding affect the performance of each algorithm?

ISSN (online): 2581-3048
Volume 10, Issue 7, pp 92-100, July-2026
https://doi.org/10.47001/IRJIET/2026.107010

To answer that question, this paper makes three main contributions: (i) building a web-based text compression application that fully implements the two byte-level algorithms Huffman and LZW, with automatic data-integrity verification; (ii) building a Vietnamese test corpus of four text categories collected from real-world sources; (iii) quantitatively analyzing compression efficiency along three dimensions: text category, data size, and the effect of Vietnamese diacritics. The remainder of the paper is organized as follows: Section 2 presents the theoretical background; Section 3 describes the software and the experimental method; Section 4 presents the results and discussion; and Section 5 concludes.

II. THEORETICAL BACKGROUND

2.1 Lossless compression and the entropy limit

According to Shannon's information theory [4], the average amount of information of a discrete symbol source is measured by the entropy

$$H = -\sum_{i=1}^{256} p_i \log_2 p_i$$

(bits/symbol), where p_i is the probability of occurrence of the i-th symbol. In this study, each symbol corresponds to a UTF-8 byte value (0–255); therefore, entropy is evaluated over the byte distribution of the input data rather than over Unicode code points or linguistic characters.

$$p_i = \frac{f_i}{N},$$

Where f_i is the frequency of the i-th byte value in the input data and $N$is the total number of bytes. Entropy was computed over UTF-8 byte values rather than Unicode code points so that it directly reflects the statistical properties observed by the byte-oriented Huffman and LZW implementations.

Entropy is the lower bound on the average code length of any method that encodes each symbol independently: no prefix code can achieve an average below H bits/symbol [7]. For text data considered at the byte level (256 symbols), an algorithm such as Huffman can only approach H bits/byte; achieving deeper compression requires exploiting the dependencies between adjacent symbols — this is precisely the approach of dictionary algorithms such as LZW.

2.2 The Huffman algorithm

The Huffman algorithm [1] builds an optimal binary tree from the symbol frequency table: the two nodes with the smallest weights are merged into a parent node, repeated until a single root remains. The path from the root to a leaf defines the prefix codeword of each symbol; the more frequently a symbol occurs, the shorter its codeword. In this study, Huffman is implemented in canonical form (canonical Huffman): codewords are reassigned in order of length and symbol value, so that the header of the compressed file needs to store only 256 bytes of code lengths instead of the entire tree structure, reducing the header cost to a fixed 264 bytes. The time complexity is O(n + k·log k), where n is the number of data bytes and k = 256 symbols.

The inherent drawbacks of static Huffman are: (i) it must scan the data twice (once to gather statistics, once to encode); (ii) each symbol is encoded independently, so the compression ratio is lower-bounded by the zero-order entropy of the source; (iii) the fixed header cost may cause small files to grow after compression.

2.3 The LZW algorithm

LZW [2] initializes the dictionary with 256 single-byte strings. During compression, the algorithm finds the longest string W already in the dictionary that matches the current data, emits the index of W, then adds the string W plus the next character to the dictionary. The decompressor reconstructs an identical dictionary without any need to transmit it, including the special case of encountering an index not yet present in the dictionary (the KwKwK situation). The implementation in this study uses variable code width: starting at 9 bits and increasing gradually to 16 bits as the dictionary fills up (a maximum of 65,536 entries); when the dictionary is full, a CLEAR code is emitted to reset it, helping the algorithm adapt as the data statistics change. Because the decompressor always updates its dictionary one entry behind the compressor, the point at which the code width increases must occur one step earlier at the decompressor — a classic implementation detail that is often overlooked. LZW needs only a single pass over the data, has complexity O(n), incurs no header beyond an 8-byte identifier, and in particular exploits the repetition of phrases — something Huffman cannot do.

2.4 Characteristics of Vietnamese data in UTF-8

The Vietnamese script uses 134 accented letters (â, ầ, ế, ộ, ữ, đ...). In UTF-8, ASCII characters occupy 1 byte, but most accented Vietnamese letters require 2 bytes (the Latin-1/Latin Extended range, such as â, ê, ô, đ) or 3 bytes (the Latin Extended Additional range, such as ế, ộ, ữ, ạ...) [5], [9]. The consequences are: (i) for the same content, accented Vietnamese text is significantly longer at the byte level than its unaccented version; (ii) with byte-level Huffman, one accented character is split into 2–3 symbols, making the byte distribution more dispersed and raising the byte-level entropy;

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 7, pp 92-100, July-2026
https://doi.org/10.47001/IRJIET/2026.107010

(iii) with LZW, the byte string of an accented character (or even an entire syllable) is quickly added to the dictionary as a single unified entry, so the multi-byte cost is paid only in the initial phase. Vietnamese also has high repetition at the syllable and phrase level (especially in administrative texts with fixed formulas), which is favorable for dictionary compression.

III. SOFTWARE DEVELOPMENT AND EXPERIMENTAL METHOD

3.1 Text compression software

We developed the software in pure JavaScript with a two-layer architecture. The core layer consists of two modules, huffman.js and lzw.js, which fully implement the two algorithms (compression and decompression) on Uint8Array byte arrays and can be shared by both the browser and the Node.js environment. The application layer consists of: (i) a web interface (index.html) that lets the user select a file or enter text directly, compress with each algorithm or compare both, display a results table (size, compression ratio, time, entropy) together with charts, download the compressed file, and decompress .huf/.lzw files; (ii) a benchmark.js program that automatically runs the entire measurement scenario on Node.js and exports the results in CSV/JSON format.

A Huffman compressed file (.huf) consists of 4 identifier bytes, 4 bytes for the original data length, 256 bytes for the code-length table, and the data bitstream; an LZW file (.lzw) consists of an 8-byte header and the bitstream of codes. After each compression, the software automatically decompresses and compares each byte with the original data to verify losslessness. We confirmed the correctness of the two core modules through 22 automated test cases, including boundary cases: empty data, a single repeated symbol, the KwKwK string, incompressible random data, and a 3 MB text that overflows the LZW dictionary multiple times.

3.2 Test data

The corpus consists of four real-world Vietnamese text files, encoded in UTF-8, with a total size of 361,528 bytes, collected from Vietnamese Wikipedia (CC BY-SA license), Vietnamese Wikisource, and the Legal Library (Table 1).

Table 1: Test dataset

Category	Source	Size (bytes)	Entropy (bits/byte)
Encyclopedic	Wikipedia: article “Việt Nam”	52,336	5.259
Literary	Wikisource: Cung oán ngâm khúc, Lục Vân Tiên	27,002	5.278
Legal	2013 Constitution, 2019 Education Law, 2015 Civil Code	212,242	5.104
Sci-Tech	Wikipedia: Artificial Intelligence, Internet, Machine Learning	69,948	5.184

3.3 Measurement method

We designed three experiments: Exp. 1 — comparison by text category, with samples truncated to the same 25 KB (at UTF-8 character boundaries) to eliminate the effect of size; Exp. 2 — comparison by data size, with the combined corpus truncated at 10, 25, 50, 100, 200, and 350 KB; Exp. 3 — comparison of the same 100 KB content in two forms, with diacritics and with diacritics removed (NFD normalization followed by stripping of diacritic characters). The measured metrics include: compression ratio (compressed size / original size), average bits per byte and per character, byte-level entropy, and compression and decompression time.

We repeated each timing measurement 5 times and took the median, in the Node.js environment (V8 engine) with the high-resolution clock process.hrtime. All measurements were performed on a machine with an Apple M1 Pro processor and 16 GB of RAM, running macOS 14.5 and Node.js 22.22. Every compression in all experiments was verified to decompress to a 100% match with the original data. Absolute throughput figures are platform-dependent; only the relative comparison between the two algorithms should be generalized.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 7, pp 92-100, July-2026

https://doi.org/10.47001/IRJIET/2026.107010

IV. RESULTS AND DISCUSSION

4.1 Compression efficiency by text category

Table 2: Compression results by category (25 KB samples, Exp. 1)

Category	Entropy (bits/byte)	Huffman (bytes)	LZW (bytes)	Huffman / LZW ratio
Encyclopedic	5.270	17,234 (67.3%)	11,406 (44.6%)	1.51×
Literary	5.278	17,228 (67.3%)	12,081 (47.2%)	1.43×
Legal	5.155	16,857 (65.9%)	10,066 (39.3%)	1.67×
Sci-Tech	5.043	16,511 (64.5%)	10,678 (41.7%)	1.55×

Note: Entropy values differ slightly from Table 1 because the samples were truncated to 25 KB.

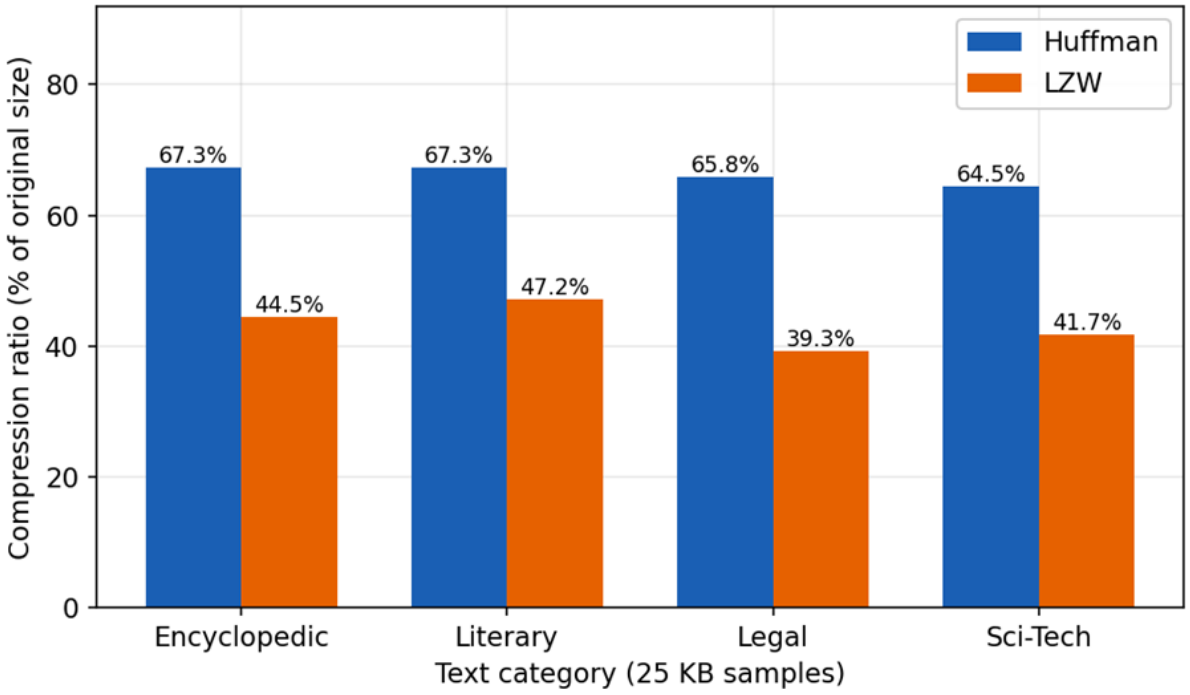

Figure 1: Compression ratio by text category (25 KB samples)

Table 2 and Figure 1 show that across all four categories, LZW clearly outperforms Huffman by a factor of 1.4–1.7. Huffman varies narrowly around 64.5–67.3% because its compression ratio depends almost entirely on the byte-level entropy, which is fairly stable across Vietnamese text categories (5.04–5.28 bits/byte). In contrast, LZW differentiates clearly according to the repetitiveness of the text: legal text compresses best (39.3%) thanks to densely repeated administrative formulas ("Chính phủ quy định chi tiết", "theo quy định của pháp luật"...), whereas literary text compresses worst (47.2%) due to its diverse vocabulary and few repeated phrases.

When each file is compressed at its full size (without truncation to 25 KB), LZW's advantage is even more pronounced for large files: the 207 KB legal file is compressed by LZW down to 29.3%, whereas Huffman remains almost unchanged at 64.3%. This shows that the longer the LZW dictionary is "trained", the more effective it becomes — analyzed in detail in Section 4.2.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 7, pp 92-100, July-2026

https://doi.org/10.47001/IRJIET/2026.107010

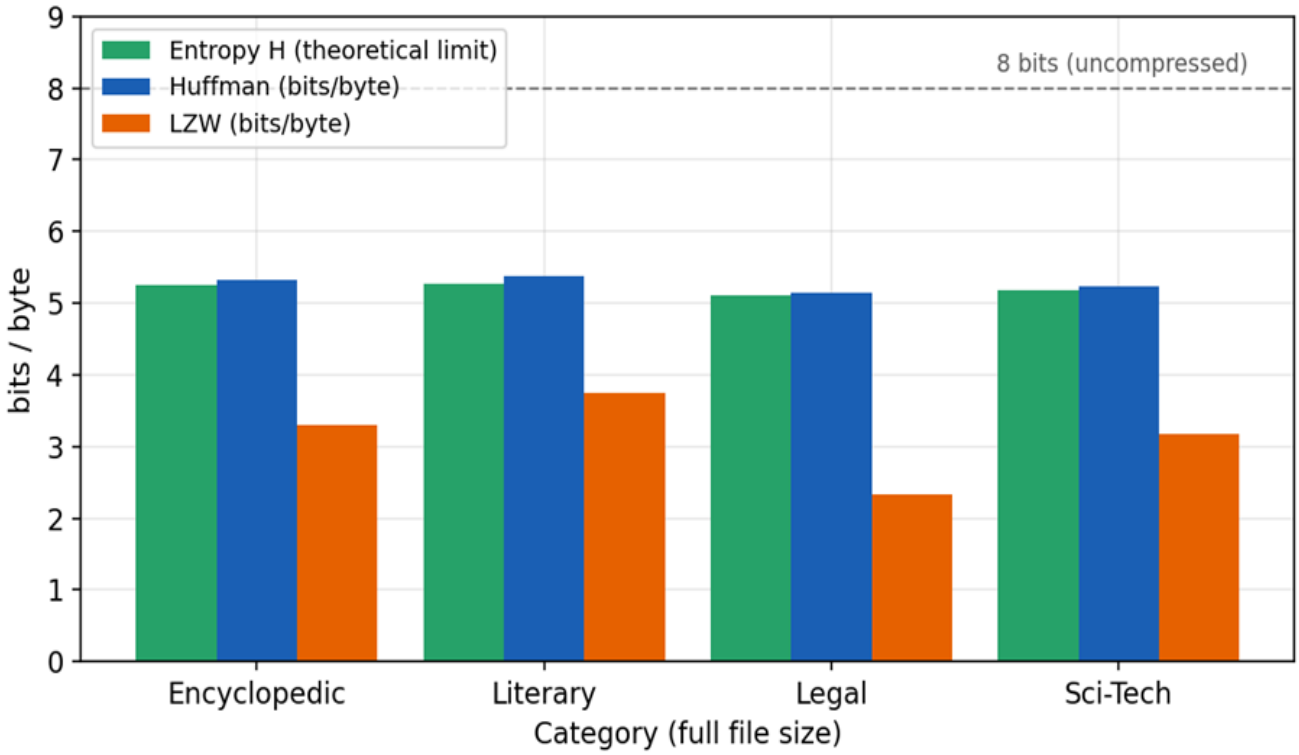

Figure 2: Average bits per byte versus the entropy limit (full file size)

Figure 2 compares coding performance with the theoretical limit. Huffman reaches 5.15–5.38 bits/byte, only 1–2% above the entropy of the data, meaning it is already close to optimal within the class of independent per-symbol coding — exactly as theory predicts [1], [7]. However, it is precisely the entropy limit (~5.2 bits/byte) that prevents Huffman from compressing below about 65%. LZW reaches 2.34–3.76 bits/byte, far below the zero-order entropy limit, demonstrating that the main redundancy of Vietnamese text lies in the dependencies between symbols (repeated syllables, words, and phrases) rather than in the frequency distribution of individual bytes.

4.2 Compression efficiency by data size

Table 3: Compression results by data size (Exp. 2)

Size	Huffman ratio	comp. (ms)	decomp. (ms)	LZW ratio	comp. (ms)	decomp. (ms)
10 KB	68.6%	0.15	0.28	48.2%	0.59	0.96
25 KB	67.3%	0.37	0.64	44.6%	1.19	1.20
50 KB	66.6%	0.63	1.24	41.3%	2.21	2.16
100 KB	66.6%	1.24	2.50	40.8%	4.81	4.09
200 KB	65.5%	2.35	4.87	34.7%	9.60	7.66
350 KB	65.3%	4.18	8.35	33.0%	18.09	11.24

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 7, pp 92-100, July-2026

https://doi.org/10.47001/IRJIET/2026.107010

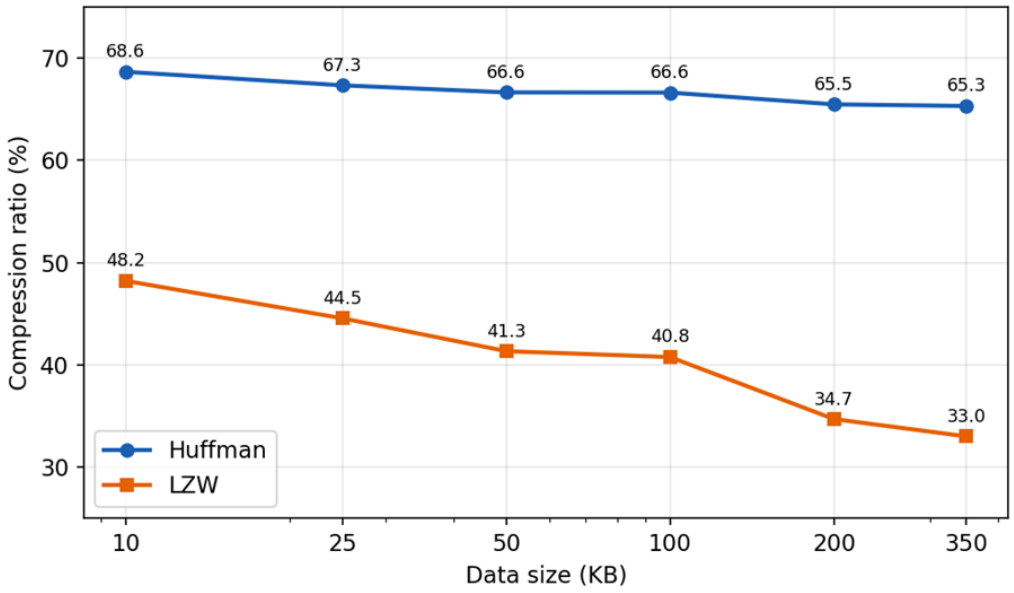

Figure 3: Compression ratio by data size

Figure 3 shows two opposing trends. Huffman's compression ratio is almost flat (68.6% → 65.3%): the small improvement is mainly due to the 264-byte header cost being amortized as files grow. Meanwhile, LZW improves continuously and significantly (48.2% → 33.0%): in the early phase the dictionary is still sparse, so the algorithm must emit many codes for short strings; as the data grows, the dictionary accumulates complete Vietnamese syllables, words, and phrases, and each emitted code replaces an increasingly long string. With 10 KB of data, the gap between the two algorithms is 20.4 percentage points; at 350 KB, the gap widens to 32.3 points.

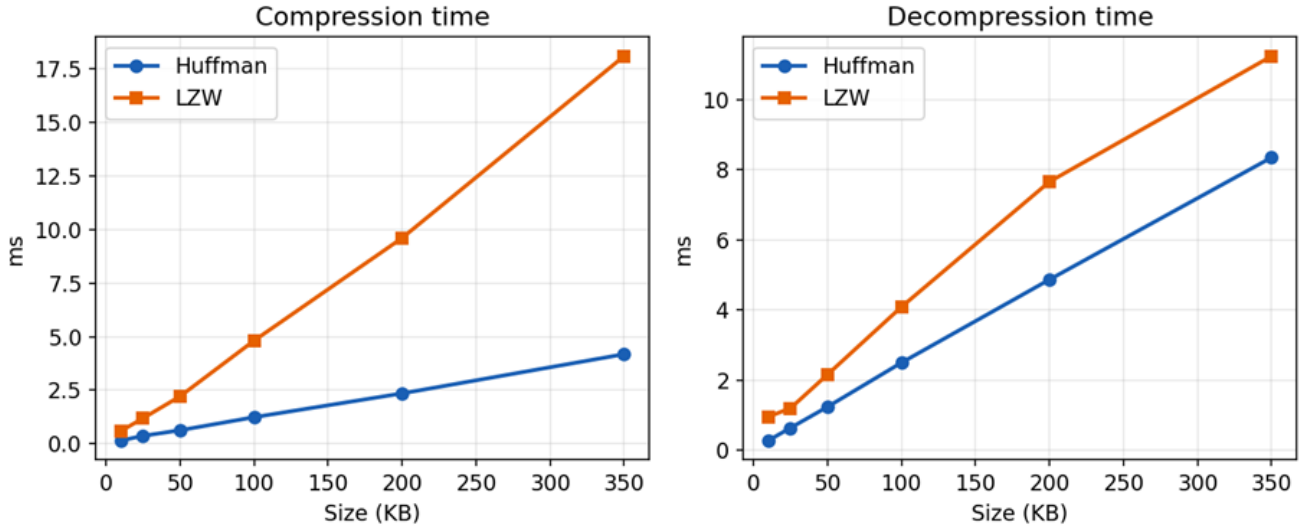

Figure 4: Compression and decompression time by data size

Regarding speed (Table 3, Figure 4), both algorithms have times that grow linearly with data size, consistent with O(n) complexity. Huffman compresses about 4 times faster than LZW (on average 78–83 MB/s versus 17–22 MB/s at sizes of 50 KB and above) because operating on each byte is much simpler than looking up and updating LZW's hash dictionary. On the decompression side, the gap narrows to about 1.3–1.5 times. Even so, for a 350 KB text both process it in under 20 ms on an ordinary computer, meaning that speed is not a real barrier for typical text data.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 7, pp 92-100, July-2026

https://doi.org/10.47001/IRJIET/2026.107010

4.3 Effect of Vietnamese diacritics and UTF-8 encoding

Table 4: Comparison of the same content with and without diacritics (Exp. 3)

Metric	Unit	With diacritics (UTF-8)	Without diacritics (ASCII)
Original size	bytes	102,400	78,279 (−23.6%)
Characters	chars	78,176	78,176
Entropy	bits/byte	5.283	4.432
Huffman: compressed	bytes (ratio)	68,209 (66.6%)	44,002 (56.2%)
Huffman: efficiency	bits/char	6.98	4.50
LZW: compressed	bytes (ratio)	41,737 (40.8%)	32,478 (41.5%)
LZW: efficiency	bits/char	4.27	3.32

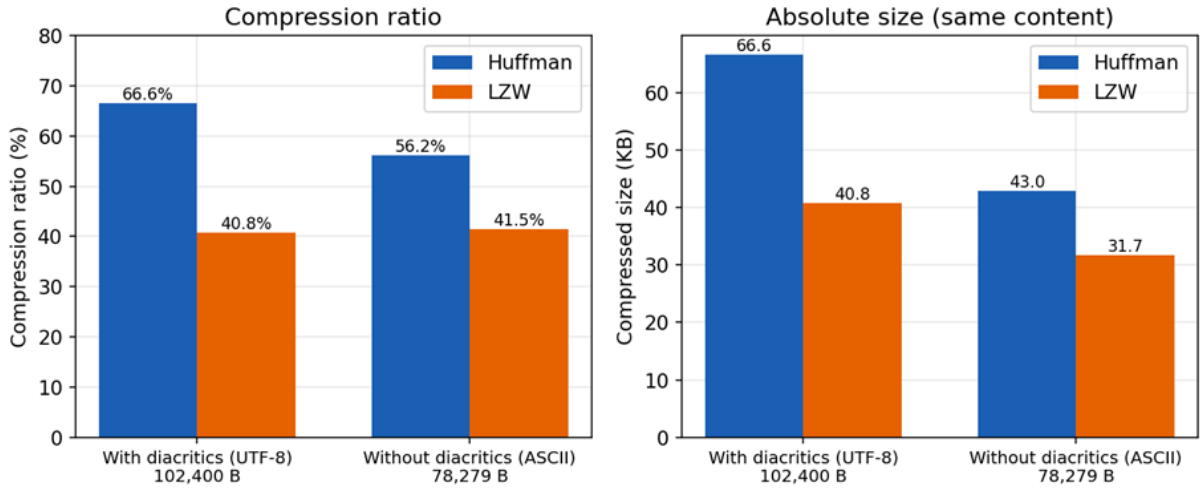

Figure 5: Compression ratio and absolute size: with versus without diacritics (same content, 78,176 characters)

Exp. 3 clarifies the impact of the UTF-8 encoding layer. For the same content of 78,176 characters, the accented version occupies 102,400 bytes — 30.8% more than the unaccented version — because accented letters require 2–3 bytes. For Huffman, Vietnamese diacritics cause a double disadvantage: the number of bytes to encode increases, while at the same time the byte-level entropy is higher (5.28 versus 4.43 bits/byte), so each byte also needs more bits; as a result, the coding cost per character rises from 4.50 to 6.98 bits — that is, 55% purely because of the character encoding layer. In contrast, LZW is almost immune in terms of compression ratio (40.8% versus 41.5%): the 2–3-byte strings of accented characters are learned by the dictionary very early and thereafter treated no differently from single-byte characters. This shows that dictionary algorithms adapt naturally to multi-byte text such as Vietnamese, whereas byte-level statistical algorithms suffer directly from the character encoding.

4.4 General discussion

Combining the three experiments, the following conclusions can be drawn. First, for Vietnamese text data, LZW is a better choice than Huffman in terms of compression ratio in every scenario examined, and its advantage grows with data size as well as with the formulaic nature of the text (strongest for administrative–legal text). Second, Huffman remains valuable when compression speed is a priority, when memory resources are very limited (no need to maintain a 65,536-entry dictionary), or when serving as an entropy-coding stage after another transform — precisely its role in DEFLATE or JPEG [6]. Third, the results of Exp. 3 suggest that if Vietnamese were processed at the character or syllable level instead of the byte level, the effectiveness of statistical methods could improve significantly — a natural extension of this study. The overall ordering we observe — dictionary coding clearly ahead of per-symbol statistical coding — is

ISSN (online): 2581-3048
Volume 10, Issue 7, pp 92-100, July-2026
https://doi.org/10.47001/IRJIET/2026.107010

consistent with the relative behavior of these algorithm families reported on standard English benchmark corpora such as Calgary and Canterbury [6], [8]; the distinctive feature of Vietnamese is that the multi-byte UTF-8 layer widens this gap, inflating the byte-level entropy that bounds Huffman while being absorbed into the LZW dictionary after an initial learning phase. Finally, the scope should be noted: a 361 KB corpus of four categories is enough to reveal consistent trends, but the quantitative conclusions (for example LZW's figure of 33%) should be regarded only as estimates for similar classes of data.

4.5 Threats to Validity

This study has several limitations. First, the corpus size is relatively small (361 KB) compared with large-scale benchmark corpora. Second, only four categories of Vietnamese text were considered. Third, Huffman was implemented in its static canonical form rather than adaptive variants. Fourth, only two compression algorithms were compared; modern compressors such as DEFLATE, Brotli, and Zstandard were outside the scope of this work. Therefore, the numerical compression ratios reported here should be interpreted as representative of similar Vietnamese UTF-8 text rather than universal values. The reported ratios should therefore be read as a comparison between the two classic algorithm families rather than against the state of the art.

V. CONCLUSION

This paper has built a web-based text compression application implementing two algorithms, canonical Huffman and variable-code-width LZW, and has quantitatively evaluated their efficiency on a four-category Vietnamese corpus. The experimental results — with 100% of compressions verified to decompress to a match with the original data — show that: (i) LZW compresses Vietnamese text down to 29.3–48.2% of the original size, outperforming Huffman (64.3–68.6%) by 1.4–1.7× on equal-size 25 KB samples, and up to 2.2× on the full-size legal file (64.3% versus 29.3%); (ii) Huffman has already approached the byte-level entropy limit (a 1–2% gap) and thus cannot improve further, whereas LZW exploits the repetition of Vietnamese syllables and phrases; (iii) Huffman compresses about 4 times faster, suitable for throughput-priority scenarios; (iv) the multi-byte UTF-8 encoding of the Vietnamese script markedly reduces the efficiency of byte-level Huffman but has almost no effect on LZW. Future directions include: experimenting on larger and more diverse corpora, implementing adaptive Huffman and more modern algorithms (LZ77/DEFLATE, BWT, arithmetic coding) for comparison, and studying compression at the syllable/word level to exploit the linguistic characteristics of Vietnamese.

CODE AND DATA AVAILABILITY

The source code and the test corpus are available from the author upon reasonable request. The corpus texts retain their original licenses (Wikipedia and Wikisource: CC BY-SA; legal documents: public domain).

REFERENCES

[1] D. A. Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[2] T. A. Welch, “A Technique for High-Performance Data Compression,” *Computer*, vol. 17, no. 6, pp. 8–19, 1984, doi: 10.1109/MC.1984.1659158.

[3] J. Ziv and A. Lempel, “Compression of Individual Sequences via Variable-Rate Coding,” *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[4] C. E. Shannon, “A Mathematical Theory of Communication,” *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[5] The Unicode Consortium. The Unicode Standard, Version 15.0. *Mountain View, CA: The Unicode Consortium*, 2022.

[6] D. Salomon, Data Compression: The Complete Reference, *4th ed., London: Springer*, 2007.

[7] T. M. Cover and J. A. Thomas, Elements of Information Theory, *2nd ed., Hoboken, NJ: Wiley-Interscience*, 2006.

[8] R. Arnold and T. Bell, "A Corpus for the Evaluation of Lossless Compression Algorithms," *Proceedings DCC*, 1997.

[9] F. Yergeau, “UTF-8, a transformation format of ISO 10646,” *IETF RFC* 3629, Nov. 2003.

[10] V. H. Nguyen, H. T. Nguyen, H. N. Duong, and V. Snasel, “A syllable-based method for Vietnamese text compression,” *in Proc. 10th Int. Conf. Ubiquitous Information Management and Communication (IMCOM), Danang, Vietnam*, 2016.

[11] V. H. Nguyen, H. T. Nguyen, H. N. Duong, and V. Snasel, “Trigram-Based Vietnamese Text Compression,” *in Recent Developments in Intelligent Information and Database Systems, Cham: Springer*, 2016, pp. 297–307.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 7, pp 92-100, July-2026

https://doi.org/10.47001/IRJIET/2026.107010

Citation of this Article:

Nguyen Minh Hoang, & Trinh Van Hung. (2026). Analysis of the Efficiency of Huffman and LZW Compression Algorithms on Vietnamese Text Data. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 10(7), 92-100. Article DOI https://doi.org/10.47001/IRJIET/2026.107010

****

