Lossless Compression without Coding and Decoding using Arabic Ligature Characters Unicode

Huda Ragheb Kadhim[1,*], Rand Abdulwahid Albeer[1], Dhamyaa A. Nasrawi[1] , Huda Hallawi[1], Muthanna Medin Nasser[1], Ibrahim Haider Jabbar[1], Burhan Karar Abbas[1]

[1]College of Computer Science and Information Technology, University of Kerbala, Iraq

Emails: huda.raghib@uokerbala.edu.iq; rand.a@uokerbala.edu.iq; dh.alnasrawy@uokerbala.edu.iq; huda.f@uokerbala.edu.iq; e17201231@s.uokerbala.edu.iq; e17201209@s.uokerbala.edu.iq; e17201243@s.uokerbala.edu.iq

Abstract

Data compression technologies play a big role in various areas where efficient data storage and transmission are essential. Data compression is the science of reducing redundant data to a compact form, which used to safely store files or information. On the other side, Unicode is a global standard for the representation of text and symbols in computers. The basic elements of the Unicode standard are code points, which represent a specific symbol. Unicode provides a unified way to map and manage these points to ensure consistent representation and interpretation of text data across different systems, platforms, and languages. This paper proposes a method to compress texts in Arabic, based on Unicode ligatures, which typically join characters together. This method replaces two or more Unicode Arabic ligature characters with a single Unicode Arabic ligature based on their appearance in the Arabic text file, eliminating the need for coding or decoding. The size of the original and output text files has been compared to show the percentage of compression. The selected dataset: Modern Standard Arabic text involves Arabic news, and Classical Arabic text involves Arabic Holy and Honorific text collected from Kaggle. The percentage of compression depends on the frequency of ligature characters in Arabic documents. Unfortunately, the results were not promising, as the method was only able to compress the file to a very small percentage (6.71 %and 12.82 %, respectively, for Arabic news and Arabic Holy text). We think that the proposed method can be improved by using a hybrid technique of text compression in the future; in addition, consider other properties of Arabic Unicode. Programming can express competency concepts in a well-defined mathematical model for a particular.

Keywords: Arabic ligatures characters Unicode; Compression; Decompression; Redundant data; Text compression

1. Introduction

As the volume of data in the world of technology increases, scientists are dedicating more time to developing and improving data management and storage than in previous eras. One of the most important techniques used is data compression, which works to reduce the number of bits that are needed to represent the data in a safe way for basic information. Consequently, the utilization of resources, such as transmission capacity or data storage space, is reduced by the reduction of data size [1].

Finding an effective algorithm to eliminate different types of redundancy from a given type of data is a compression problem; the amount of redundancy that can be retrieved from the original data will also determine how effective the approach is. Different methods may be needed for different types of data in order to detect and eliminate redundancy. The internal structure of the data would determine how a compression technique behaved. A compression algorithm may work better if the supplied data contains more redundancy[2].

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

Lossless and lossy compression, two fundamental data compression techniques, differ in their coding to accommodate different types of data. Lossless compression guarantees that the decompressed data is identical to the original, without any loss of information, in situations where data integrity is essential, such as in text files, executable files, and specific image formats (such as PNG) for medical purposes.

Often used in multimedia files (images, video, and audio), lossy compression results in the loss of some less important data, leading to a reduction in quality. However, this method prioritizes achieving smaller file sizes over exact accuracy[3].

Several popular algorithms are used in lossless and lossy compression, like Huffman coding, Run-Length Encoding (RLE), Lempel-Ziv-Welch (LZW), and the Shannon-Fano algorithm, etc. While JPEG (Joint Photographic Experts Group), MPEG (Moving Picture Experts Group), and MP3 (MPEG Audio Layer III) require the return of the original data, lossless decompression necessitates the use of more intricate algorithms. Lossy compression achieves significantly higher compression ratios, resulting in faster processing times[4], [5].

Text compression is the process of reducing the size of a text file by encoding the file information more efficiently. This method minimizes the number of bits and bytes required to represent the data, guarantees that no information is lost during decompression, and restores the original file.

Numerous benefits of text compression include its ability to handle large datasets of Arabic and English text with high redundancy in natural language, faster data transfer over networks using less bandwidth, and enhanced performance by reducing the load on storage and expediting access to textual data (databases and log files), which is frequently compressed in the healthcare industry. Additionally, text compression plays a crucial role in data mining, which employs data analysis techniques to identify valid patterns and relationships and enables efficient data refinement [3], [4] .

Arabic text compression is more complex than other languages due to the interconnection of Arabic letters within a word and the presence of related punctuation, which can alter the meaning of two identical words. Moreover, Arabic is sparser than English, resulting in a higher repetition rate of English words for a given text length[6].

In Arabic text, many ligature characters have a good frequency, thus, high significant benefits to utilize Unicode of these ligature characters in the compression process.

In order to leverage the Arabic ligature characteristics of Unicode, a new method for Arabic text compression is proposed without the need for coding or decoding unlike previous methods. This method replaces two or more Arabic ligature characters in Unicode with a single Unicode Arabic ligature based on their appearance in the Arabic text file. This work contributes these areas to the pertinent body of literature:

1. The leveraging of the Arabic ligature characters Unicode and their appearance to compress Arabic text documents.
2. This method does not require coding or decoding. Thus, the efficiency of time for a compression algorithm is high.
3. Enhancement of compression percentage of by selecting high frequency patterns of Arabic ligature characters.

The other subsections include previous studies in section 2, an elaboration on the Arabic ligature characters Unicode in section 3, the proposed method explained in section 4, results, conclusions and future works in sections 5, and 6.

2. Related Works

Text compression is a subfield of lossless data compression, and its techniques are divided into two categories: dictionary and statistical techniques. Dictionary techniques employ a dictionary of sequences and their corresponding symbols, while statistical techniques analyse the statistical properties of data to compress it[7].

The principle of data compression is consistent across all text compression techniques, despite their distinct methodologies. It concentrates on repetition, manifesting in texts at various levels (bits, letters, syllables, or words). The characteristics of text are contingent upon the language that is compressed. Arabic's text is distinguished by a high frequency at the letter and bit levels because of its derivation nature, which is the case for all Semitic languages. It employs a multi-byte coding system known as Unicode, and its letters are composed of diacritics and have a singular form. There are three categories of Arabic texts: Classical Arabic (CA), the language of the Holy books and or text; Modern Standard Arabic (MSA), the language of the media, education, news, and intellectuals; and Arabic dialects, which are used in accordance with the geography of each region or country[8].

The Arabic language received limited attention in the field of data compression, particularly with the introduction of Arabic Unicode. Some studies address the importance and challenges of the Arabic language. This section reviews some relevant works of Arabic text compression.

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

Tariq Abu Hilal and Hasan Abu Hilal proposed a sequential encoding technique that efficiently converts Arabic characters from UTF-8 to ANSI, reducing storage space by 50%. The technique retains the original format of the encoded text, resulting in lossless text compression[9].

Sarah Abdulkareem Al-Busaeed and Umut İnan produce a novel Arabic letter encoding system based on features of the Arabic language to address coding issues in Arabic. This system reduces the text size by half and allows one-byte representation of Arabic letters and accent marks[10].

Majed AbuSafiya presents a new method of compressing natural language text, which speeds up text search by deleting the first byte of the letters' Unicode. According to this method, the text's letters are assumed to be part of the same natural language's alphabet[11].

ENAS ABU JRAI et al. present a novel method for compressing Arabic Unicode text using the Lempel-Ziv-Welch (LZW) technique, involving transformation and compression stages. The first phase replaces multi-byte symbols with single-byte ones, while the second phase uses the outputs as inputs for adaptive LZW [7].

Ashraf Saleh Mohammad Alomoush and Norita Md Norwawi introduced an innovative version of the Digital Quran model by addressing the duplication of Quranic terms. A lookup database lists every distinct Quranic word along with its unique ID. This method maximizes memory space because it presents the unique ID using integers. Rather than concatenating Unicode with a hexadecimal representation for each letter in a word[4].

Our proposed method focused on Arabic ligature characters in Unicode to compress Arabic text documents based on redundant data. The proposed method implements easy and clear model without need of coding and decoding. Therefore, the proposed method has no complexity in time and space.

3. Arabic Ligature Characters Unicode

Computer processing uses the international character-encoding system known as Unicode to deliver texts. The Unicode standard employs 16-bit encoding, which is capable of encoding all the characters used in the writing of the world's languages. This encoding supports 65,000 characters in a variety of formats, such as letters, numerals, symbols, and a significant number of the current characters in the world's languages. Every character in Unicode has a unique number that is unaffected by the language, software, or platform[12].

Arabic is the fourth most prevalent language among internet users and the fifth most spoken language in the globe, with a worldwide population of over 420 million native and non-native speakers in the Arab world. Additionally, more than 2 billion Muslims use it as their liturgical language. Used mostly in writing, Standard Arabic is directly derived from the language of the Quran. Standard Arabic, using the abjad script, is written from right to left[13]. Arabic, like Urdu and Farsi, is composed of 28 characters that are written in a calligraphy style. The structure of an Arabic letter is contingent upon its position within a word. It may be situated in the initial, intermediate, terminal, or isolated position within a word. Table 1 provides an explanation of the forms and samples of Arabic alphabets[14].

Table 1: Arabic alphabets forms

Name	Unicode	Shapes			
		Isolated	Final	Medial	Initial
HAMZA	0621	ء			
ALEF WITH MADDA ABOVE	0622	آ	ـآ		
ALEF WITH HAMZA ABOVE	0623	أ	ـأ		
WAW WITH HAMZA ABOVE	0624	ؤ	ـؤ		
ALEF WITH HAMZA BELOW	0625	إ			
YEH WITH HAMZA ABOVE	0626	ئ	ـئ	ـئـ	ئـ
ALEF	0627	ا	ـا		
BEH	0628	ب	ـب	ـبـ	بـ

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

TEH MERBUTA	0629	ة	ـة		
THE	062A	ت	ـت	ـتـ	تـ
THEH	062B	ث	ـث	ـثـ	ثـ
JEEM	062C	ج	ـج	ـجـ	جـ
HAH	062D	ح	ـح	ـحـ	حـ
KHAH	062E	خ	ـخ	ـخـ	خـ
DAL	062F	د	ـد		
THAL	0630	ذ	ـذ		
RAH	0631	ر	ـر		
ZAIN	0632	ز	ـز		
SEEN	0633	س	ـس	ـسـ	سـ
SHEEN	0634	ش	ـش	ـشـ	شـ
SAD	0635	ص	ـص	ـصـ	صـ
DHAD	0636	ض	ـض	ـضـ	ضـ
TAH	0637	ط	ـط	ـطـ	طـ
ZAH	0638	ظ	ـظ	ـظـ	ظـ
AIN	0639	ع	ـع	ـعـ	عـ
GHAIN	063A	غ	ـغ	ـغـ	غـ
FEH	0641	ف	ـف	ـفـ	فـ
QAF	0642	ق	ـق	ـقـ	قـ
KAF	0643	ك	ـك	ـكـ	كـ
LAM	0644	ل	ـل	ـلـ	لـ
MEEM	0645	م	ـم	ـمـ	مـ
NOON	0646	ن	ـن	ـنـ	نـ
HEH	0647	ه	ـه	ـهـ	هـ
WAW	0648	و	ـو		
ALEF MAKSOUR	0649	ى	ـى		
YEH	064A	ي	ـي	ـيـ	يـ
HAMZA	0621	ء			
ALEF WITH MADDA ABOVE	0622	آ	ـآ		
ALEF WITH HAMZA ABOVE	0623	أ	ـأ		

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

When two or more characters appear together as a single entity, it's known as a ligature. Ligatures have their roots in cursive handwriting, which typically combines characters. Instead of encoding each distinct ligature as a separate Unicode letter in the Arabic language, Unicode primarily uses rendering technologies to control context and generate the required glyphs (graphical representation of character). This preserves a small character set while enabling the rendering of rich text. Thus, these ligatures are frequently utilized to improve the aesthetics and fluidity of Arabic letters in addition to saving space. There are certain constructs in Unicode that are specifically related to ligatures. For example, there is a small ligature 'ﺿﻢ'(U+FC25), which is defined as equivalent compatibility-wise to 'ض' (U+0636) followed by 'م' (U+0645) [12].

In general, the types of Arabic ligatures characters in Unicode Standard Version 16.0 [15] are shown in figures 1,2 and 3:

- Arabic Ligatures characters (two elements): joining two Arabic letters into a single glyph to enhance the readability and visual appeal of written text are shown in figure 1.
- Arabic Ligatures characters (three elements): joining three Arabic letters into a single glyph to enhance the readability and visual appeal of written text are shown in figure 2.
- Arabic Word Ligatures (Honorific Word): they are special characters that are specified in the (Arabic Presentation Forms A) segment dedicated to commonly used phrases in cultural and religious situations. These ligatures aid in maintaining conventional expressions without necessitating the complete spelling of the words. Such as the ALAYHAA AS-SALAAM ligature and others found across various scripts, as shown in figure 3.

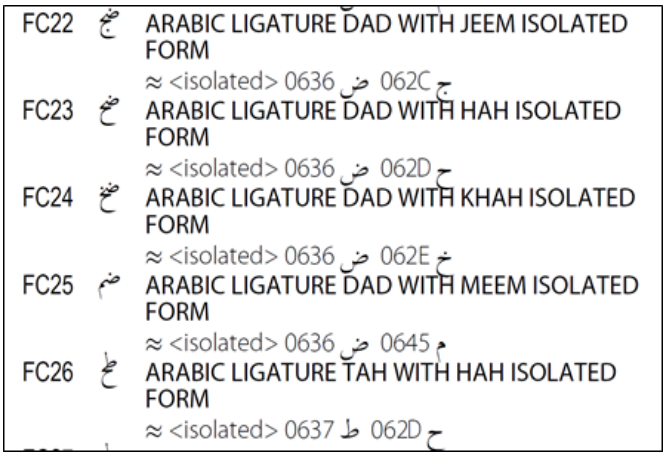

Figure 1. Sample of Arabic Ligatures characters(two elements) [15]

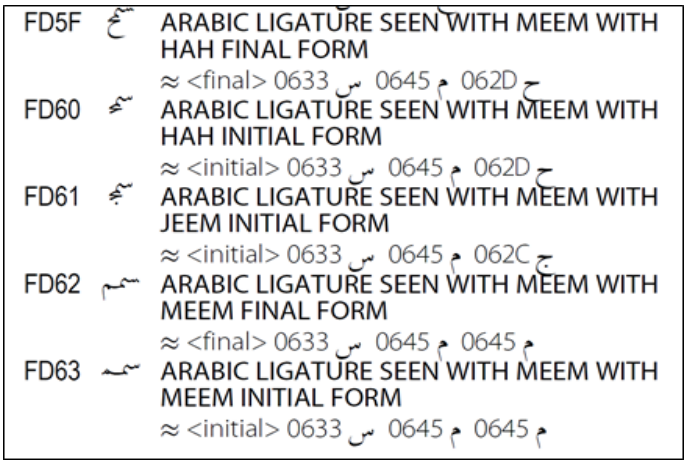

Figure 2. Sample of Arabic Ligatures characters (three elements) [15]

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

Figure 3. Sample of Arabic Honorific Word Ligatures [15]

4. Methodology

The proposed method replaces two or more Arabic ligature characters in Unicode with a single character, depending on their appearance in the Arabic text file.

This section elaborates on the methodology followed for Arabic data compression in details.

4.1 Description of the Problem

The aim of proposed lossless compression method is to identify a working technique that eliminates redundant information from Arabic text. Finding any redundancies in the original data is the first step in the data compression process. Redundancy has a broad definition. From the perspective of saving storage, it can be some similar structures in nature, some overlapped information, some common base data, or some comparable qualities.

The input and output are Arabic text without diacritical marks (Unicode) collected from internet. Some details about inputs dataset explained in section 6.

4.2 Probability Modelling

In proposed lossless compression method, model is used to describe the redundant characteristics of the source data based on a probability distribution of the redundant.

The redundant data in proposed method represent Arabic ligature characters Unicode (two or more characters appear together as a single entity) which replaces with single character Unicode, depending on their appearance in the Arabic text file.

The Arabic ligature characters include those with two elements, three elements and Arabic Honorific Word Ligatures. The Arabic Presentation Forms-A block Range: FB50–FDFF contains Arabic ligatures that start from isolated ligature Unicode FBEA "ﺋ". In this paper, we called the selected Arabic ligature characters Unicode as redundant data, which are highlighted in yellow, as shown figure 4.

While the frequency of redundant data illustrated in figure 5 and 6 with the most frequently occurring ligature characters in various Modern Standard Arabic text and Classical Arabic text of varying sizes.

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

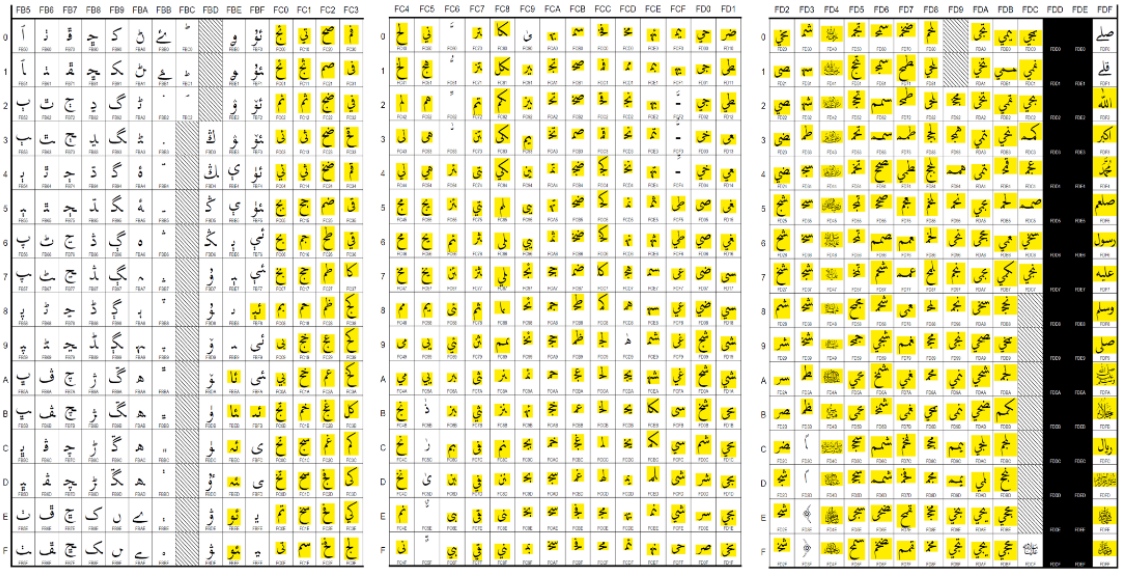

Figure 4. Redundant data: Arabic ligature Characters (two elements, three elements and Arabic Honorific Word) [15]

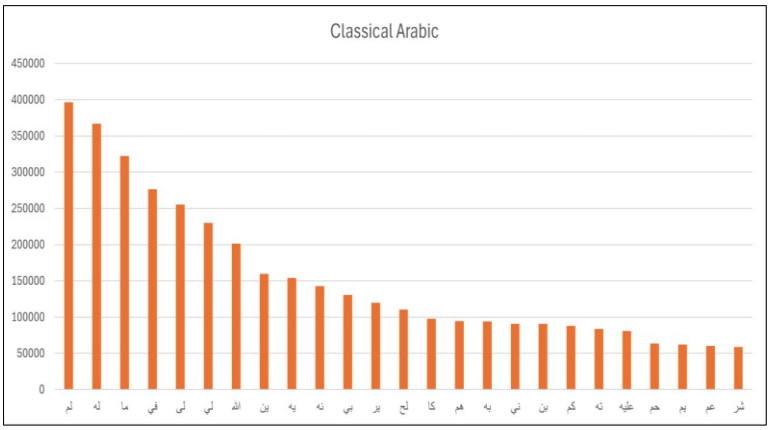

Figure 5. The most frequent ligature characters in much Classical Arabic text

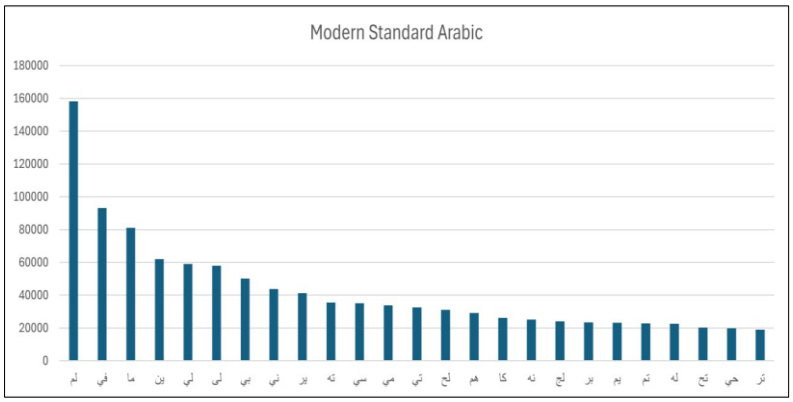

Figure 6. The most frequent ligature characters in much Modern Standard Arabic text

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

4.3 Compression Algorithm

Now, three steps were executed for the proposed compression method:

- **Create Dictionary of Redundant Data**

First, create a dictionary of Arabic ligature characters called (redundant data); in this method, the Arabic ligature characters were chosen (two elements, three elements and honorific word ligatures). Programmatically, the Arabic Ligatures Characters are stored as dictionary data structure in Python. Figure 7 demonstrate the dictionary of Arabic Ligatures Characters.

```
Arabic_Ligatures_Characters_dictionary = {
    'ﻧﺎ' : '\uFBEA',    'ﺑﺞ': '\uFC00',    'ﺑﺢ': '\uFC01',
    'ﺑﻢ' : '\uFC02',    'ﺑﻰ': '\uFC03',    'ﺑﻲ' : '\uFC04',
    'ﺗﺞ' : '\uFC05',    'ﺗﺢ': '\uFC06',    'ﺗﺦ': '\uFC07',
    'ﺗﻢ': '\uFC08',    'ﺗﻰ': '\uFC09',    'ﺗﻲ' : '\uFC0A',
    'ﺛﺞ': '\uFC0B',    'ﺛﺢ': '\uFC0C',    'ﺛﺦ': '\uFC0D',
    'ﺛﻢ': '\uFC0E',    'ﺛﻰ': '\uFC0F',    'ﺛﻲ' : '\uFC10',
    'ﺟﺞ': '\uFC11',    'ﺟﻢ': '\uFC12',    'ﺟﻰ': '\uFC13',
    'ﺟﻲ': '\uFC14',    'ﺟﺢ': '\uFC15',    'ﺟﻢ' : '\uFC16',
    'ﺻﻲ': '\uFD08',    'ﺷﺞ': '\uFD09',    'ﺷﺢ': '\uFD0A',
    'ﺷﺦ': '\uFD0B',    'ﺷﻢ': '\uFD0C',    'ﺷﺮ': '\uFD0D',
    'ﺳﺮ': '\uFD0E',    'ﺻﺮ': '\uFD0F',    'ﺿﺮ': '\uFD10',
    'ﻧﻮ': '\uFBEE',    'ﻧﻪ': '\uFBEC'
}
```

Figure 7. The dictionary of Arabic Isolated Ligatures Characters (two-element example)

- **Searching for Redundant Data**

The second step involves searching for the redundant data within the document, a process that can be time-consuming for large Arabic text files.

- **Replacing Process**

Finally, the redundant data are replaced with their corresponding Arabic ligatures Unicode. After that, the compressed document was saved, and the compression percentage and elapsed time of compression were computed. The details of proposed Arabic text compression method are summarized in the following algorithm:

Algorithm 1 Arabic text compression
Input: Arabic document file
Output: Arabic compressed document file
Create dictionary of Arabic Ligatures Characters called *redundant data Dictionary* Open Arabic document file. Scan document to find *redundant data* by check the *key* of Dictionary Compute the frequency of each character in *redundant data*

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

For each character in *redundant data Dictionary*: If characters match the *key* in *redundant data Dictionary* then replace by corresponding Arabic Ligatures Unicode *(value* in *redundant data Dictionary)* Return Arabic compressed document file

In another side, the decompression process is the opposite operation of the compression process; the details are explained in the following algorithm:

Algorithm 2 Arabic text decompression
Input: Arabic compressed document file
Output: Arabic document file
Use dictionary of Arabic Ligatures Characters called *redundant data Dictionary* Open Arabic compressed document file. Scan document to find Arabic Ligatures Characters by check the *value* of Dictionary For each character in Arabic Ligatures Characters: If characters match the *value* in *redundant data Dictionary* then replace by corresponding *key* in *redundant data Dictionary* Return Arabic document file

5. Compression Quality Measurement

The parameters for time and space can be calculated to determine the compression efficiency.

- Efficiency of space represented by:

1. Compression ratio CR the ratio of a compression algorithm has output to input file size, computed using Eq. (1).

$$CR = \frac{size\ of\ compressed\ file}{size\ of\ original\ file} \quad (1)$$

2. Compression Factor CF the opposite of CR computed using Eq. (2).

$$CF = \frac{size\ of\ original\ file}{size\ of\ compressed\ file} \quad (2)$$

3. Compression Percentage CP, this displays the proportion of shrinking computed using Eq. (3).

$$CP = \frac{size\ original\ file - size\ compressed\ file}{size\ original\ file} * 100 \quad (3)$$

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

- Efficiency of time: Additionally of space, time efficacy for a compression algorithm is determined by the time required to compress and decompress the file [16].

6. Experimental Results

This section evaluates the performance of the proposed Arabic text compression method using performance measurement. The proposed models also undergo a comparison with related works.

In this study, two datasets were used to collect Arabic texts from Kaggle. The first covers Modern Standard Arabic text used in newspapers articles from Arabic online newspapers, which involves cultural, financial, political, medical, athletic, technical, and religious topics. The second is related to Classical Arabic text: Hadith, Bin Baz Fatwas, Fatwa, arabic_answers. All datasets will be converted to text files.

In proposed Arabic compression method, there is no coding or decoding in compression. Thus, the time required to compress the file is low. Table 2 show the result of proposed method, while table 3 explains the comparison with related works.

Table 2: Space and time efficiency of proposed Arabic compression method

File No.	Text type	Original file size (Byte)	Compressed file size (Byte)	CP	Compression time seconds
1	Modern Standard Arabic (News)	10,997,015	10,255,171	6.75 %	4.5046475
2		19,543,261	18,287,253	6.43 %	9.8380357
3		33,293,024	31,090,349	6.62 %	16.5498738
4		9,465,338	8,862,367	6.37 %	4.6770423
5		19,795,394	18,480,783	6.64 %	15.1525353
6		38,404,632	35,741,548	6.93 %	16.7748303
7		25,013,599	23,202,762	7.24 %	11.7712718
Avg.		22,358,895	20,845,748	6.71 %	11.3240338
8	Classical Arabic	21,483,247	18,258,437	15.01 %	8.7214051
9		33,924,222	30,141,694	11.15 %	16.0361054
10		239,886,474	211,776,59	11.72 %	114.8012682
11		68,549	59,377	13.38 %	0.0325926
Avg.		73,840,623	65,059,027	12.82 %	35

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

Table 3: Proposed Arabic compression method comparison with related works

Ref. and Year	Level	Method	CP	Advantage	Disadvantage
[20]2019	Characters	sequential encoding that converts Arabic characters from UTF-8 to ANSI	45%	No limitations on text contents, no overhead, high compression percentage	-
[21]2019	Characters	Arabic letter encoding system based on features of the Arabic language	57%	high compression percentage	Implement in just Arabic scripts with diacritical marks.
[22]2021	Syllables	deleting the first byte of the letters' Unicode	50%	high compression percentage	Implement in Arabic letters only without diacritical marks, need overhead, need coding and decoding.
[11]2023	Syllables, words	transformation and compression stages. The first stage replaces multi-byte symbols with single-byte ones, then uses the outputs as inputs for adaptive LZW in the second stage	71.28%	high compression percentage, no need for statistical or morphological analysis, no need to send dictionaries.	Best for small files, need coding and decoding.
[4]2023	Words	A lookup database lists every distinct Quranic word along with its unique ID	53.09% result just for surah al-Fatihah	reliability, validity, and storage management efficiency	Implement in Quranic word, need coding and decoding
Our model	Syllables, words	Replaces two or more Unicode Arabic ligature characters with a single Unicode Arabic ligature based on their appearance in the Arabic text file	12.82 %	No overhead, no coding and decoding, low time complexity.	Implement in Arabic letters only without diacritical marks, Low compression percentage

Table 2 explain the space and time efficiency of proposed Arabic compression method by compute the Compression Percentage CP using equation(3) which need original file size and compressed file size. In addition to compute the compression time in seconds.

Table 3 demonstrate the proposed Arabic compression method comparison with related works in terms of level, method, compression percentage, advantages and disadvantages.

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025

7. Conclusion And Future Work

This paper aimed to leverage the Arabic ligature characteristics of Unicode in Arabic text compression. The proposed method involves three steps to implement the compression: first, create a dictionary of redundant data; then, search for the redundant data in the document; and finally, replace with their corresponding Arabic ligatures in Unicode. The proposed method does not require coding or decoding. Therefore, the low time complexity, as well as no overhead in compressed file. Despite this, the results were not as promising as those from previous works, primarily due to the potential limitations of the proposed method. These limitations include the influence of redundant data frequency, document size on the compression rate. Therefore, the proposed method only achieved an average compression rate of 6.71% and 12.82% for Modern Standard Arabic text and Classical Arabic text, respectively. This means the frequency of selected Arabic ligatures is low in general. However, their frequency increases in Classical Arabic text. In future work, the proposed method can be improved using hybrid technique of text compression. Furthermore, the many properties of Arabic Unicode can be considered in any further work.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] M. J. Haque and M. N. Huda, "Study on data compression technique," *International Journal of Computer Applications*, vol. 159, no. 5, pp. 6-13, 2017.

[2] I. M. Pu, *Fundamental data compression*, Butterworth-Heinemann, 2005.

[3] H. Jani and J. Trivedi, "A survey on different compression techniques algorithm for data compression," *International Journal of Advanced Research in Computer Science and Technology*, vol. 2, no. 3, pp. 1-5, 2014.

[4] N. M. Norwawi and A. S. M. Alomoush, "LIGHTWEIGHT VERSION FOR DIGITAL QURAN MODEL BY HANDLING DUPLICATION," *PERINTIS eJournal*, vol. 13, no. 1, pp. 69-76, 2023.

[5] P. Raundale, "Comparative Study of Data Compression Techniques," *International Journal of Computer Applications*, vol. 178, no. 28, pp. 1-10, 2019.

[6] Z. M. Alasmer, B. M. Zahran, B. A. Ayyoub, M. A. Kanan, A. I. Hammouri, and J. Ababneh, "A Comparison between English and Arabic text compression," *Journal of Contemporary Engineering Sciences*, vol. 6, no. 3, pp. 111-119, 2013.

[7] E. A. Jrai, S. Alsharari, L. Almazaydeh, K. Elleithy, and O. Abu-Hamdan, "Improving LZW Compression of Unicode Arabic Text Using Multi-Level Encoding and a Variable-Length Phrase Code," *IEEE Access*, vol. 11, pp. 51915-51929, 2023.

[8] I. Guellil, H. Saâdane, F. Azouaou, B. Gueni, and D. Nouvel, "Arabic natural language processing: An overview," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 5, pp. 497-507, 2021.

[9] T. A. Hilal and H. A. Hilal, "Arabic text lossless compression by characters encoding," *Procedia Computer Science*, vol. 155, pp. 618-623, 2019.

[10] S. A. Al-Busaeed and U. A. İnan, "A New Arabic Coding Scheme," *International Journal of Engineering and Natural Sciences*, vol. 2, no. 3, pp. 22-28.

[11] M. AbuSafiya, "Speeding up Natural Language Text Search using Compression," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 4, 2021.

[12] M. Needleman, "The unicode standard," *Serials Review*, vol. 26, no. 2, pp. 51-54, 2000.

[13] T. D. Kamusella, "The Arabic language: A Latin of modernity?," *Journal of Nationalism, Memory and Language Politics*, 2017.

[14] D. A. AL-Nasrawi, A. F. Almukhtar, and W. S. AL-Baldawi, "From Arabic Alphabets to Two Dimension Shapes in Kufic Calligraphy Style Using Grid Board Catalog," *Communications in Applied Sciences*, vol. 3, no. 2, 2015.

[15] Archived Code Charts, "CodeCharts_16.0".

[16] B. Vijayalakshmi and N. Sasirekha, "Comparative Analysis of Lossless Text Compression Methods with Novel Tamil Compression Technique," *International Journal of Research in Engineering and Science (IJRES)*, vol. 9, no. 7, pp. 38-44, 2021.

DOI: https://doi.org/10.54216/FPA.200102
Received: December 14, 2024 Revised: February 01, 2025 Accepted: April 02, 2025