IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 April 2023, accepted 15 May 2023, date of publication 23 May 2023, date of current version 1 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3279021

RESEARCH ARTICLE

Improving LZW Compression of Unicode Arabic Text Using Multi-Level Encoding and a Variable-Length Phrase Code

ENAS ABU JRAI[1], SHOROQ ALSHARARI[2], LAIALI ALMAZAYDEH[2], KHALED ELLEITHY[3], (Senior Member, IEEE), AND OSAMA ABU HAMDAN[4]

[1]Department of Basic Sciences, Ma'an University College, Al-Balqa Applied University, Salt 19117, Jordan
[2]Department of Software Engineering, Faculty of Information Technology, Al-Hussein Bin Talal University, Ma'an 71111, Jordan
[3]Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT 06604, USA
[4]Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA

Corresponding author: Laiali Almazaydeh (laiali.almazaydeh@ahu.edu.jo)

ABSTRACT This paper introduces a novel approach to enhance the efficiency of compressing Arabic Unicode text using the Lempel-Ziv-Welch (LZW) technique. This method includes two stages: transformation and compression. During the first phase, a multi-level scheme that works according to the level of words, syllables, and characters replaces multi-byte symbols with single-byte symbols, resulting in a binary output of 51%-75% smaller than the actual size and effective for compression. In the second phase: the outputs of the previous phase are received as inputs to the adaptive LZW technique, attached to a value representing the length of the initial phrase (minimum code length). This value is automatically determined according to the size of the data source to enhance the performance of LZW. The original data size is included in the compressed file to be used during the decompression process to detect the length of the initial phrase. The compression ratio achieved by the proposed method was compared to that of the traditional LZW technology that uses multi-byte encoded characters and a fixed initial length phrase, as well as two recent technologies, DEFLATE and Gzip. Experimental results indicate that our method achieves an average compression rate of about 71% and outperforms other methods for all forms of Arabic texts, with improved LZW able to compress an additional 7% or more or less of the files it compresses. Variable-length dictionary LZW continuously displays a significant difference in compression ratios for small files compared to modern methods, whether it uses a variable-width- encoding scheme only or with multi-level encoding as a precompression step. Multi-level schema can be used as a preprocess to other compression techniques, especially those that work efficiently with binary data. Also, the original data volume can be used as a private key within data security and encryption applications.

INDEX TERMS Adaptive initial dictionary, Lempel-Ziv-Welch (LZW), multi-dictionaries, multi-level mapping, private key, unicode Arabic text, variable-length phrase code.

I. INTRODUCTION

The volume of digital data in the current era is massive, and has grown rapidly. According to a study by International Data Corporation (IDC), the amount of digital data created, captured, and iterated is expected to reach 175 zettabytes by 2025 [1]. This represents a compound annual growth rate (CAGR) of 61%. This rapid growth in digital data has significant implications for storage space and transmission time, highlighting the urgent need to develop and improve data compression techniques.

Data compression is performed by representing the original symbols in the source data with the fewest possible bits [2]. Conversely, Data decompression is the process of restoring compressed data to its original state. Data compression techniques aim to reduce the amount of storage space

The associate editor coordinating the review of this manuscript and approving it for publication was Yilun Shang.

required, the time needed for transmission over the network, and errors that may occur during data transmission over communication channels. In addition to reducing costs and saving energy when using smart devices [2], [3], it enables recent technologies such as big data analytics and machine learning to become more workable and effective.

Data compression techniques are classified into lossy and Lossless [3], [4]. Lossy: The data are not recovered precisely, and this category is suitable for compressing audio and video files. Lossless: The original data are retrieved precisely from the compressed data; they are suitable for compressing critical data such as X-ray images and text files [5], [6].

Text compression is a sub-field of lossless data compression, and its techniques are categorized into two groups [2], [7]: statistical and dictionary techniques. Statistical techniques analyze the statistical properties of data to compress them [8], whereas dictionary techniques use a dictionary of strings and their corresponding symbols to compress data [9], [10].

The dictionary can be static, semi-static, or dynamic. (1) Static: An independent dictionary that does not need to be combined with the compressed data. It is pre-built and holds a fixed set of symbols and corresponding strings. Symbols are mapped to strings based on the order in which they appear in input data. Pictogram notation is one of the most common forms of static dictionary [11], [12]. (2) Semi-static: This analyzes the data source and builds a dictionary suitable for encoding data. The dictionary is merged with compressed data before transmission for decompression [13], [14]. 3) Dynamic or adaptive: The dictionary is constantly built and updated during the compression and decompression processes; it is built dynamically during both processes [15], [16]. An example, is LZW.

All text compression techniques despite their different methods have the same principle for data compression, which is to remove repetition [7], [17]. Repetition appears in texts at several levels, either at the level of bits [18], letters [19], [20], syllables [21], [22], or words [15], [23], [24], [25], [26], [27], [28], [29]. The repetition rate for each level varies from language to language.

The performance of text compression techniques depends on the compatibility of the compression technique method with text properties [26], [29]. For example, statistical techniques (e.g., Huffman) are more efficient for high-frequency text compression at the letter level [8] and dictionary techniques (e.g., Lempel-Ziv-Welch ''LZW'') are more efficient for high-frequency text compression at syllable and word levels [10]. The characteristics of text depending on the language to be compressed [27]. Each language has a grammar, morphological structure, coding system, and other characteristics that differ from other languages. For example, English as a European language is characterized by its high frequency at the word and syllable levels, since its letters do not include diacritics. The letter has two cases, uppercase and lowercase, and using a single-byte encoding system called ASCII. As for the Arabic language as a Semitic language [28], its text is characterized by a high frequency at the level of letters and bits because it is a derivation language, and its letters have one form, and consist of diacritics, in addition to its use of a multi-byte coding system called Unicode [30].

Unicode, specifically UTF-8, is used by 97.8% of all websites [38], which indicates the need to develop and improve compression techniques for natural languages in general, and Unicode UTF-8 scripts in particular.

Arabic is one of the most spoken Semitic languages ranking fifth in the world [28], [38], and Arabic Unicode UTF-8 scripts are used by 98.9% of all websites according to available usage statistics [39].

LZW is one of the simplest and most effective text compression techniques; however, it is inefficient in compressing the Arabic language [40], and its versatility and popularity as a standard technology make it a popular choice for improving Arabic language compression performance.

Several modifications and improvements have been made to LZW natural language compression technology to achieve the highest possible compression ratio. Some studies have modified this technique's working principle and ignored the language's characteristics so that it can be applied to texts in different languages [31], [41] (as shown in Table 1). In contrast, other studies take advantage of the inherent features of a particular language to compress text, by modifying or processing the text to fit the underlying principles of the technique [22], [25], [42], [43], [44], [45], [46], [47], [48], [49] (as demonstrated in Table 2 for a specific language).

This study combines the two methods to reach the maximum possible compression ratio, as it is based on denaturing the data and reducing its actual size before compression, to make it more compressible. Additionally, to setting the minimum initial dictionary size depending on the file size may enhance the performance of LZW when dealing with different sizes.

The proposed approach includes two phases: (transformation and compression). The first phase aims to convert the text into binary data of less size, regular repetition, and high compressibility, by exploiting several linguistic characteristics of the Arabic language, including the phenomenon of repetition at the level (word, syllable, and letter) to build several static dictionaries of different optimal sizes. The unique case of the Arabic character is to be encoded using unused ASCII code instead of Unicode.

The objective of the second stage, is compressing binary data using Dynamic LZW and enhancing its performance by determining the size of the initial dictionary based on the size of the data source and using the length of a variable length phrase.

The major contributions of the work are listed below, distinguishing it from the few works available in the literature.

1) The proposed multi-level data coding scheme has the capability to work not only with LZW, but also work as a pre-compression step in conjunction with other general compression techniques, particularly those that efficiently compress binary data.

TABLE 1. List of previous surveys on improvements made to LZW technology.

Ref	Year	Target Impact factor	Approach	Level	Result	Advantage
[31]	2022	-Structure of text -Text length	Incorporating LZW with Bit Reduction Algorithm	Word, char bit	Improves compression rates.	Simple and easy to implement
[32]	2021	Dictionary size	Proposes IDA-LZW, an innovative insignificant dictionary area	Char	Best average compression with reasonable resources.	Avoid compression ratio degradation
[33]	2021	Dictionary size	String is added if frequency meets threshold.	Char	Compression rate is 6% higher than LZW	Improves compression rates
[34]	2020	-Structure of text -Text length	Incorporating a b64pack and LZW coding techniques	Char	Outperforms the other methods under diverse dimensions	-
[35]	2019	-Structure of text -Phrase length -Repetition rate	Use reverse data codes, and reduce file size	Char	Improves compression rates.	Avoids dictionary overflow issue.
[36]	2019	Text size	Employs PDLZW with address space partitioning	Char	Optimize the compression rate of the PDLZW	Simplifies the parallel search in the dictionaries for hardware implementations
[37]	2012	Phrase length	Use a Variable-Length Phrase Code	Char	Suitable for compressing small files than large ones.	Applicable for image compression

TABLE 2. List of previous surveys on improvements to LZW technology to compress specific natural language.

Ref	Year	Language	Approach	Level	Result	Advantage
[22]	2020	Uyghur	-Constructed 12-bit and 16-bit syllable coding dictionaries. - processed additional language chars and spaces.	Syllable, char	- 12-bit scheme compressed <4KB Uyghur text by 0.3 on avg, - 16-bit scheme compressed <2KB text by 0.5 on avg.	Compression schemes outperformed Gzip, bzip2, LZW on short text
[43]	2020	For Malayalam and 3-4 byte encoded languages.	-Efficiently updates when max size reached, allows empty dictionaries. -Reloads full dictionary with new characters, clears when full. -Uses variable-width coding system.	Char	Higher compression ratio than Gzip for 3-4 byte encoded languages.	Reduce dictionary entries.
[42]	2019	Malayalam	Applied reversible transformation before using general-purpose compression.	Char	Malayalam docs saved at 75%.	Leverage internal source redundancy for file size reduction before compression.
[49]	2017	Bangla	A dictionary with Unicode ranges from 1-90 is used for Bangla characters.	Char	High compression rate approximately 3% for dictionary index and 33% for output sequence compared with LZW algorithm	Can be used in small memory devices and text communication
[25]	2015	Indonesian	Using dictionary to store words and codewords.	Word, char, affixes	- Compressed file to <50% of original size with codeword stream and supplement table output. - The processing time is much better than the reverse sequence of characters on the LZW method.	-
[48]	2010	Chinese	- The initialization character section in the code table - improve dictionary memory management - Use a more efficient dictionary renewal mechanism	Char	Improve the compression ratio effectively	Avoids dictionary overflow issue.

2) provide an up-to-date survey of the improved LZW technique to compress the natural languages.
3) Use repetition at multiple levels (words, phrases, letters, and bits) to increase the rate of regular repetition.
4) It uses two types of dictionaries that are the optimal size.
   a) static dictionaries
      i) Its contents do not exceed about 255 words and 255 Tri-grams, 30 letters.
      ii) It does not need to be indexed. since it uses the equivalent value storage location as a reference.
      iii) Unused ASCII character location is exploited.
   b) A dynamic dictionary deals with phrases of variable length, whose initial values depend on the size of the data source.

The optimal size of dictionaries is essential in reducing overhead costs, and the size of dictionaries positively affect compression and transmission time.

The following is the organization of this paper: Section I.(A) presents the main Arabic language features used in the proposed approach. Section I.(B) explains the working principle of the LZW technique and highlights the factors that impact its performance. In Section II, we provide a detailed review of existing survey papers related to implementing LZW specifically on Arabic text. Building on this foundation, Section III describes the proposed approach, including the multi-level dictionary construction. Section IV presents experiments and results. Finally, Section V presents the conclusion with some future directions.

A. ARABIC LANGUAGE CHARACTERISTICS

Arabic is among the most widely spoken languages, spoken by more than 467 million people [50], [51]. There are three types of Arabic texts [50]: Classical Arabic (CA), such as the language of the Qur'an; Modern Standard Arabic (MSA), such as the language of the media, education, and intellectuals, in addition to the Arabic Dialect, which is formed according to the geography of each region or country. Most compression techniques support only one of these types; for example, techniques for compressing Arabic text according to morphological analysis depend on specific patterns and roots, so they do not achieve a high compression ratio when dealing with slang texts, because they are not subject to morphological rules. Also, techniques of General compression will fail due to the letter appearance in different patterns (i.e., with and without diacritics), so does not achieve a satisfactory compression ratio compared to the rest of the languages [40]. The proposed approach is characterized by its ability to handle all types of Arabic languages.

Arabic words are classified into functional and content words. Functional words provide grammatical information rather than content. Functional words play a key role in the overall meaning of a sentence (e.g., prepositions, separate pronouns, and auxiliary verbs). The most important thing distinguishing them is that although their number is small, their repetition is high [52]{Citation}. Functional words consist of a small number of letters between two and five, especially in the absence of diacritics, as shown in Table 3 [47]. Therefore, the possibility of being part of another word is large; for example, (''kan-كان '' is part of the word ''makan-مكان''). Therefore, it is common for the function word to be repeated as an independent word or as a part of a word. Content words: any word that is not functional, including (nouns, verbs, and adjectives). The proposed approach replaces a word with a shorter binary value; the function words are encoded at the word level, and the content words are encoded at the word level if they appear as part of another word, at the syllable level, and at the letter level.

TABLE 3. Frequency of some function words.

Function Word without diacritics	
Word	Frequency
من	3.22.239
في	301.895
أي	132.635
و	130.809
على	119.639
إذا	115.842

The Arabic words consist of a series of alphabetic letters and diacritics [53]. The number of Arabic letters is 28. The most important characteristic of Arabic letters is that each letter has only one case, unlike English letters, that have two cases: capital or small letters. In addition, the number of Arabic letters is small and their frequency varies, as shown in Fig. 1 [54].

The 8 diacritics are connected to the letter so that, found above or below the letter (ً ٌ ٍ َ ُ ِ ّ ْ) [55]. The diacritics are optional. If they appear with the letters of the words (e.g., كَتَبَ-kataba), the text is termed (full vowels). In this case, the frequency of the diacritics is high, but if an appearance on some letters of the words (e.g., كَاتِبٌ-writer), the text is termed (partial vowels), and if the letters of the words are stripped from any diacritics (e.g., مكتوب -written) the text is termed (Un-vowels). The proposed approach deals with all forms of Arabic text and takes advantage of the letter case to encode it using unused ASCII code.

Arabic relies on affixing to generate and develop verbal wealth, through syllabic appendages that stick to their original material in the form of affixes (prefixes, infixes, and suffixes) [50], which consist of two letters (bigrams) or three letters (trigrams). Examples of these affixes are: ال، ات، ين، ون as shown in Table 4 Statistical studies have shown that the frequency of adhesions is high and their number is limited [56].

The proposed approach takes advantage of this feature to generate an optimally sized static dictionary that encodes the data source at the segment level.

Arabic characters and their diacritics are encoded according to the universal Unicode principle [57]. Unicode has

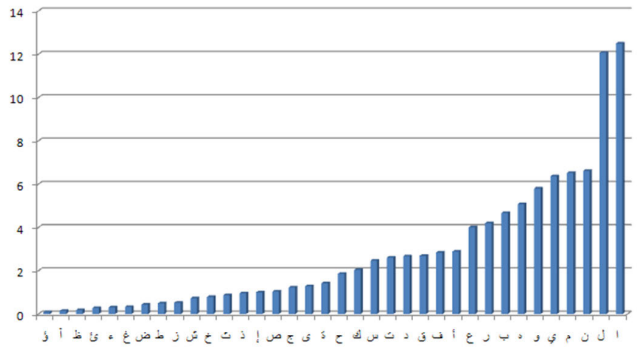


FIGURE 1. Arabic letter frequency distribution, sorted according to frequency of letters.

two main encodings, UTF-8 and UTF-16, both of which are designed for specific purposes. In the proposed approach, the test texts are stored according to the UTF-8 encoding system, which is currently the most widely used system globally and is a source for more than 92% of web pages. An un-vowel character is encoded as two bytes and a vowel character as four bytes, resulting in inefficient use of storage space. According to earlier studies, replacing ASCII code with Unicode code reduces the file size and improves compression by 50% [45], [58], [59], [60]. The proposed method uses a similar method for n-gram encoding [61], with the difference that the value of n represents a character, affixing, or a word.

TABLE 4. Example form of affixes.

Meaning	Suffix	Infix	Prefix	word
Informative	يه	ا	است	استعلاميه
they play	ونْ	-	يَ	يَلعبُونْ

The proposed approach exploits several linguistic characteristics of the Arabic language to make the text more compressible, characterized by its ability to handle all types and forms of Arabic language. It is based on the phenomenon of redundancy at the word level, then at the n-gram level, and then at the character level, taking advantage of the unique case to encode it with a non-ASCII code user instead of Unicode.

B. LZW TECHNIQUE

LZW is a universal lossless compression technique, and it is one of the most powerful compression techniques. Efficient for files with repetitive data [5], [12]. Simple, and does not require a priori knowledge of the file structure, data types, or usage statistics.

LZW Has a dynamic dictionary that includes strings and codes. Replaces strings of characters with single codes. The following steps show the pseudo-code for the LZW algorithm [37]:

1) Initialize the dictionary with all possible characters.
2) Read the first character from the input data and assign it to C.
3) Read the next character and save it in E, then collect C,E into string "S".
4) Check the dictionary for the string "S".
   a) If found,
      i) repeat steps 3.
   b) If not found,
      i) output code for "C",
      ii) save "S which is now called ***phrase***" to the dictionary
      iii) reset "C" to "E".
5) Repeat steps 3-5 until the end of the sequence.

The size of the initial dictionary is 2^ the length phrase (which is the code for the new entries). The dictionary adapts to the data being compressed, as the algorithm updates it dynamically during processing.

Several factors affect the performance of the LZW method when used for text compression:

(a) Structure of text [21], [29], [62]: LZW depends on text structure, target language characteristics, and pattern repetition rate.
(b) Text length [42], [45]: LZW is not ideal for compressing short texts because it limits repetition.
(c) Dictionary size [36], [63], [64]: The dictionary size affects LZW's performance by balancing the compression ratio and processing time. A larger dictionary leads to better compression but also uses more memory and processing power. A smaller dictionary reduces memory and processing requirements but may result in lower compression. The optimal dictionary size is determined by testing and balancing the different sizes. Commonly, a dictionary size of 4096 or 8192 entries provides good results. Their sizes can be adjusted according to specific needs.
(d) Phrase length [45], [63], [65]: If its length is short compared to the size of the data source, it creates a chain of new entries that are not used and takes up space in the dictionary. Conversely, if it is long, it produces a new code string with a larger number of equivalent bits.
(e) Text character set [49], [66]: LZW was initially designed to compress text encoded in the ASCII character set. Therefore, this technique is unsuitable for text containing characters from multiple bytes, such as Utf-8 used by 97.8% of character-encoded websites [67].

LZW works more efficiently for binary files with all their contents either 1 or 0 because it replaces sets of bits (instead of strings of characters in text files) with single codes [45], [40].

The LZW algorithm doesn't always provides an ideal compression result, especially when dealing with Arabic Text [40]. Therefore, many studies have been published to improve the performance of LZW compression in natural languages [45], [46], [47].

In this study, we aim to increase the compression ratio of Arabic texts of all types by compressing the data source that has been processed as binary data, setting the initial

dictionary size and code word length based on the size of the data source, and making its variable-width

II. RELATED WORK

In recent years, the LZW technique for compressing natural language texts has been widely discussed in the literature, resulting in various improvements being introduced and implemented. However, the focus on the Arabic language in these studies has been limited, with some based on word level [47], and others based on character [45] or segment level [46].

Our previous study [46] introduced a new hybrid technology for compressing Arabic texts, which effectively utilizes the morphological and grammatical features of Arabic to enhance the performance of compression ratio for LZW and BWT methods. The proposed method involves two stages: morphological analysis and compression.

During the morphological analysis stage, a multi-layered model is used to categorize the text into functional words, derivative words, and other words, along with a fourth layer to link words to their original positions in the input text. The root pattern dictionaries technique is then utilized to replace derived words with index values based on a dictionary of 4096 roots and 2048 patterns, while function words are encoded using a single byte based on a dictionary of 128 words. These dictionaries do not need to be transferred with compressed files, reducing transmission time. However, the morphological analysis stage requires more time and effective storage space.

In the compression stage, general compressors are used to compress the four layers separately. The LZW and BWT methods are applied, and the results are compared. The comparison showed an improvement in the performance of both techniques by 0.25 and 0.23, respectively.

In [45] a comparison of four techniques, namely Eilias, Golomb, Huffman, and LZW, was presented. These techniques were applied to Arabic text as a binary formula. LZW showed the best compression ratio and decoding time for large file sizes (above 300 KB), with a compression ratio of 24%. However, LZW was ineffective in compressing small files due to the fixed length of the code word (15), which prompted the researchers to recommend a variable codeword length based on the file size.

Moreover, the researchers introduced the Arabic character mapping technique to enhance the compression ratio of the three statistical techniques (Golomb, Elias, and Hofmann). Results indicated that implementing the Arabic character mapping technique improved the compression ratio of these techniques compared to the binary approach. Nonetheless, Huffman with character mapping outperformed other techniques on average sizes, including LZW, but only for small binary data. The authors proposed using an Arabic Character Mapping approach with LZW to increase the compression ratio for all file sizes.

In [46] introduced a new technique for compressing Arabic text files based on encoding text at the segment level. Its effectiveness is in using n-grams and window size to create a dynamic dictionary, which is then used to encode text with the smallest number of bits. It can be used to reduce the sizes of different text files, in English or Arabic in all their forms. The achieved compression ratio was 43.87%.

Authors in [43] changed the LZW (Lempel Ziv Welch) technique to compress the Unicode Arabic text to reduce the number of dictionary entries by using an initial blank dictionary and an additional dictionary that is later included with the zip file. The proposed modification was not only tested on Arabic texts and applied to other texts such as Bengali, Tamil, and Malayalam with varied sizes. The tests showed that the results of applying the proposed modification were superior to the traditional byte-based LZW compressors. Table 5 presents a comparison of a surveyed and reviewed research work on arabic text compression using LZW.

III. PROPOSED METHOD

This study proposes an approach to improve the LZW compression ratio for Arabic texts by transforming text files into binary files to make them more compressible using Adaptive LZW with a variable length phrase and setting its initial value based on the size of the data source to reinforce the compression ratio of texts.

The proposed approach works in two stages: transformation and compression, as shown in Fig. 2, in which the text is set with predefined shorter codes to reduce the actual size of the original file. It uses several static and separate dictionaries. In the second stage, the output of the first stage is compressed using the Dynamic LZW technique, because it works efficiently with binary data. It begins with reading a string of bits, whose initial length depends on the size of the data source, and changes based on the size of the dictionary.

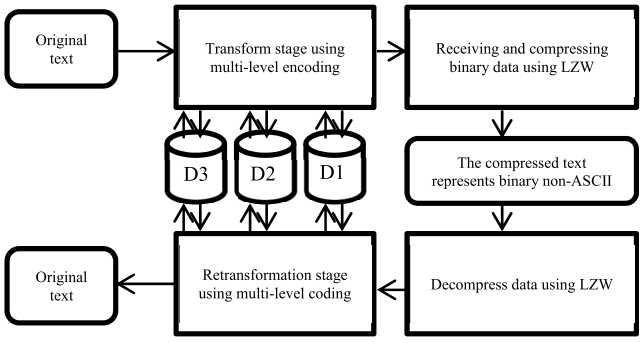


FIGURE 2. General description of the proposed approach.

To accurately decode from the three dictionaries, we implemented a flag code during the transforming phase to identify the reference dictionary for the original data. Fig. 3 shows an example of transforming the word "Alhamdulillah-الحَمدلله ".

A. GENERATE DICTIONARIES

Repetition is a linguistic phenomenon known as Arabic in the oldest texts. It appears in different forms at three levels. At the letter level, 16% of the letters in Arabic text were

TABLE 5. Comparison of related work on Arabic text compression using LZW.

Citation	Year	Target Impact factor	Approach	Level	Results	Advantage	Disadvantage
[43]	2020	Dictionary Size	-Efficiently updates when max size reached, allows empty dictionaries. -Reloads full dictionary with new characters, clears when full. -Uses variable-width coding system.	Character	-Average compression rate of 5% with 32 KB dictionaries and 4% with 64 KB dictionaries. -Higher compression ratio than Gzip for 3-4 byte encoded languages.	Reduce dictionary entries.	-Additional overhead. -Performance degradation while compressing files of 1 or 2 byte encodings
[45]	2016	-Structure of text	-Evaluated Eilias, Golomb, Huffman, and LZW on Arabic texts as binary data. -Introduced Arabic letter mapping to improve compression ratio for Golomb, Elias, and Huffman.	Character	- LZW outperforms other methods by 24% for large files only. -Huffman with character mapping outperforms other techniques at medium sizes, including LZW, but only for small binary data.	- It takes advantage of internal source redundancy by treating the text as binary data, ignoring the language of the text. -Enhancing compression of Arabic text using (Eilias, Golomb, and Huffman) methods with Arabic letter mapping	LZW was ineffective in compressing small files due to the fixed length of the code word (15)
[46]	2016	-Structure of text	Create dynamic dictionary with n-grams and window size.	syllable	Compression ratio was 43.87%.	- Combine the features of Huffman and Lempel Ziv algorithms. - Suitable for different sizes. - Deal with Arabic and English.	The compression ratio is minimal
[47]	2015	-Structure of text -Text Length	utilizes the morphological and grammatical features of Arabic to enhance the performance of compression ratio for LZW and BWT methods -The proposed method includes two stages: morphological analysis and compaction.	word, Character	The performance of LZW and BWT improved by 0.25 and 0.23, respectively	Dictionaries don't need to be transmitted with compressed files, reducing transmission time	-The morphological analysis stage requires more time and efficient storage space. -The compression ratio is minimal

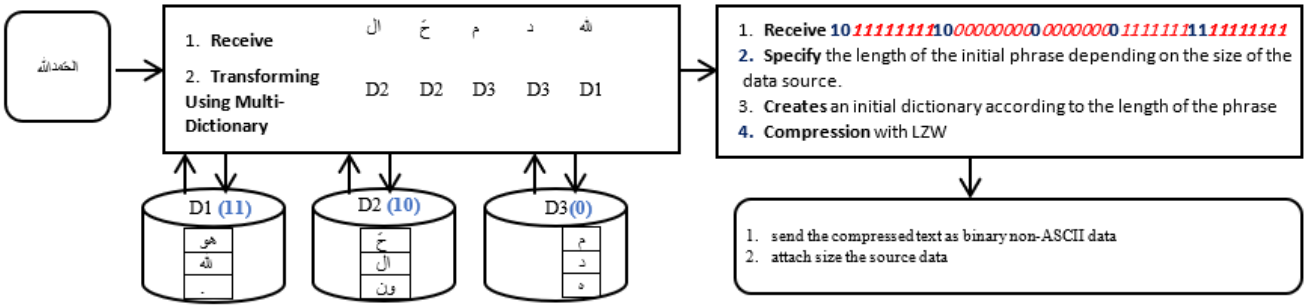


FIGURE 3. Example of transforming the word "Alhamdulillah-الحَمدلله".

spaces. At the word level, earlier studies have indicated that 40% of the words in Arabic texts are function words [52]. At the syllable level, it has several forms, including suffixes, distinguished by their limited numbers.

According to the three recurrence levels, the transformation phase was based on three independent dictionaries.

1) DICTIONARY OF WORDS

It is used for coding the function word by 10 bits. It has the 128 most common and often-used function words. We selected them from the OSAC corpus [56], [68] and arranged them in the dictionary descending according to their frequency.

Then we used a formal tool called "Shakili" [69] to form the same words complete formation and added them to the same dictionary and in the same order.

El-Khair [70] proved that if function words are replaced by fixed-length codes whose length is less than their average size, the size of the input text is reduced. Accordingly, we suggested that the functional word be represented by 1 byte, as shown in Fig. 4.

The un-vowel Functional words mainly consist of two or three letters. Accordingly, to that a functional word may appear as part of another word. Therefore, the designation is based on whether the match is full or partial. For example, the word to (God-لله) is part of the word (praise-الحَمدلله).

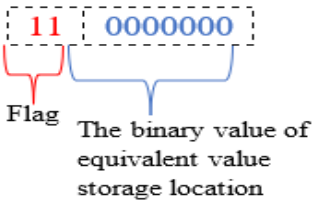


FIGURE 4. Functional word dictionary output structure.

Words without diacritics are 2-5 characters long and need 4-10 bytes of storage space. The proposed method replaces the function word with 10 bits, which saves storage space by 75%-90%.

2) DICTIONARY OF AFFIXES AND FORMED LETTERS

It replaces vowel characters (i.e. chars with a diacritic) or affixes by a ten of a bit. This dictionary contains two groups: The first group includes the most common and frequent suffixes, determined based on Arabic grammar books and arranged randomly. The second group includes Arabic letters in all their forms, for example If we symbolize the Arabic letter (ـ) It may appear in one of the following forms ـًـٌـٍـَـُـِـّـْ).

Usually, the character is encoded according to UTF-8 with four bytes (two bytes for characters and two bytes for movement); The proposed method replaces vowel characters or affixes with 10 bits Fig. 5, which reduces the character size by at least 69%.

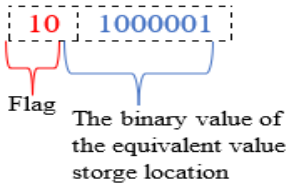


FIGURE 5. Affixes and formed letters dictionary output structure.

3) DICTIONARY OF SINGLE LETTERS

Unused ASCII character locations from 64 to 127 have been exploited to designate Arabic characters, single vowels, numbers, punctuation marks, and symbols. Therefore, the Arabic character is represented by 8 bits instead of 16 bits (as shown in Fig. 6) to obtain an effective storage space and reduce the data size by at least 50%, as many studies have proven the efficiency of this approach in improving the compression ratio [45], [58], [59], [60]. We arranged the Arabic letters in descending order of frequency.

We made sure to arrange the content of the dictionaries according to the most frequent ones to preserve redundancy in the data source because it is the main factor in raising the compression ratio in the next stage.

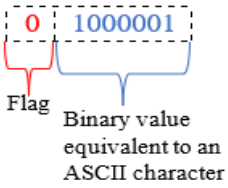


FIGURE 6. Single letters dictionary output structure.

B. TRANSFORMATION STAGE

Fig. 7 illustrates the flowchart of the first stage. The first step begins by reading one word at a time to search for its equivalent value starting in the dictionary of words (D1); if a perfect match is found, the binary value of the location of this word is returned, "11" was added at the beginning as a reference to the dictionary used when decoding and then moving to read the next word.

If no perfect match was found, it searched for the partial match with the largest number of partial character similarities and replaced the matching part with the binary value of its location, adding (11) at the beginning as a reference to the dictionary. It then returns to the first step to complete the search for the remainder of the non-matching part.

Suppose no match was found in the word dictionary. In that case, the segment dictionary (D2) is searched to replace the affixes or characters attached to the diacritics with the binary value of the matching location, adding ten at the beginning as a reference. Otherwise, the replacement is performed at the character level using the third dictionary (D3) to return the binary value of the location of the character it matches, noting that it is equivalent to one of the unused ASCII values consisting of seven bits preceded by 0 as a reference; otherwise, it returns the binary value of the ASCII value of the entered data. Spaces are not ignored because they are the boundaries between compound and simple words. Symbols, numbers, and punctuation were replaced with a 7-bit ASCII code, followed by 0 for reference.

C. COMPRESS STAGE

The LZW technique receives the output of the earlier step as a binary input for compression.

Typically, traditional LZW uses a fixed length code of 12 bits to output the code; therefore, the dictionary size is 4096, and when the dictionary is full, LZW becomes static. This approach may increase the encoded message length of small messages compared with the original text, especially UTF-8 encoding has variable length characters from 1 to 4 bytes, which leads to deterioration of the compression ratio, so our proposed approach uses dynamic phrase length, which changes every time the dictionary limit is reached.

The initial input values of the dictionary are based on the size of the initial phrase, which is determined according to the size of the data to be compressed and then included with the compressed data to be used in the decompression stage.

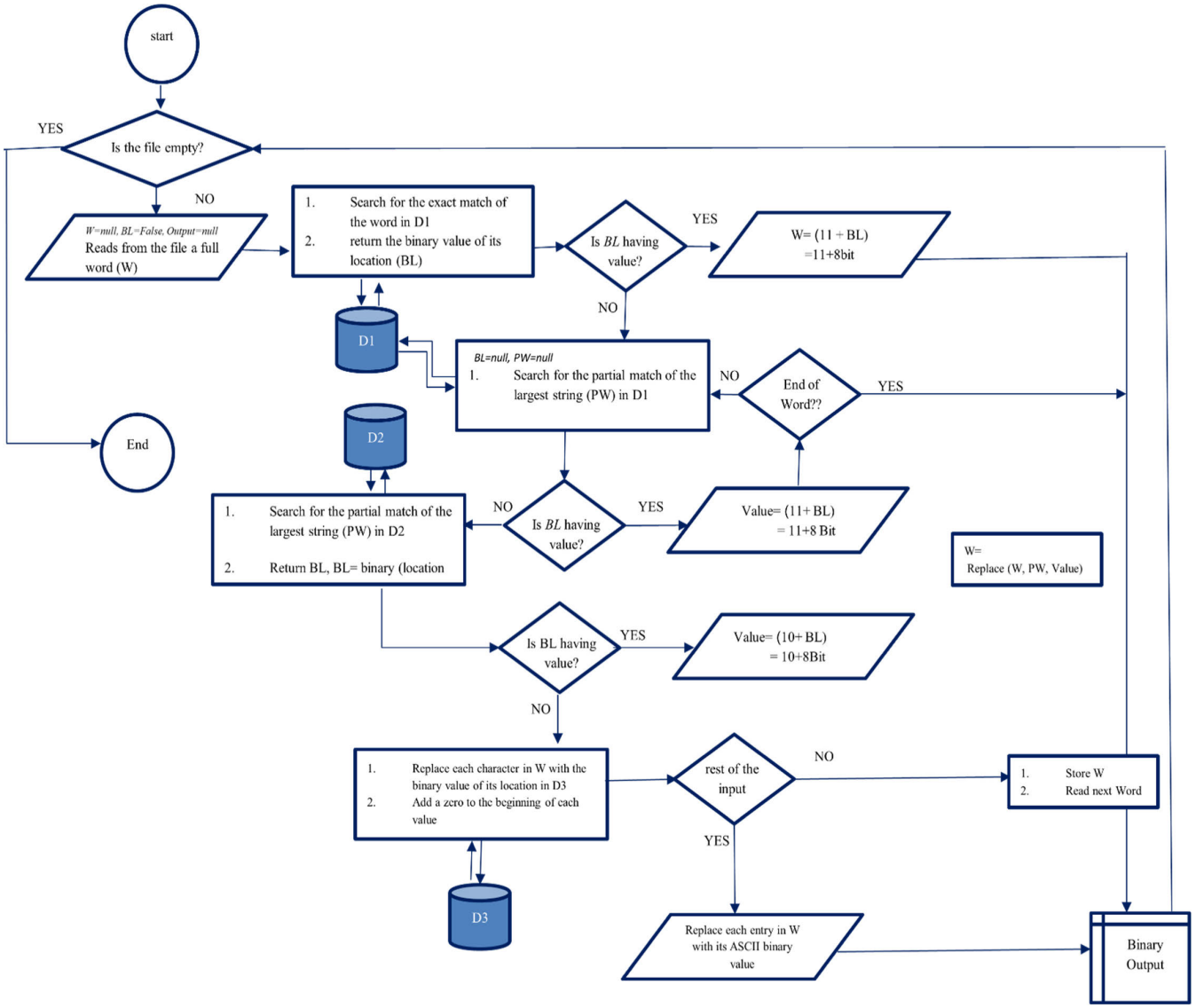


FIGURE 7. Flowchart for the first stage of the proposed approach.

It should be noted that traditional LZW compression replaces strings of characters with a single symbol. In contrast the proposed method implements LZW to replace strings of Bits with single symbols because; in the precompression stage, the nature of the data is changed to obtain a better compression ratio.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

To evaluate the performance of the proposed approach, we prepared three dictionaries according to the method described in the previous section, and shared them for both the encoder and decoder modules. We then performed three sets of experiments on Arabic text files, which were obtained from the OSAC Corpus [56], [68] removed the HTML tags from the text and stored them in UTF-8 encoding.

We used the ''Shakali'' tool [69] to form the same texts partially and completely; that is, we tested the same text in three shapes, as well as text files from the Internet. We conducted a second set of experiments to determine the length of the initial phrase code for which the compression stage relies on the generation of the first dictionary entries.

We used Python to implement the design and tested it on a 2.4GHz CPU, 4.00GB RAM, a 64-bit operating system, and Windows 10.

The compression ratio serves as a metric to gauge the efficiency of a compression method, with higher ratios indicating better quality. the compression ratio can be calculated by (1):

$$\text{Compression ratio (CR)} = \left(1 - \left(\frac{\text{number of compressed bits}}{\text{numbers of uncompressed bits}}\right)\right) \times 100\% \quad (1)$$

Table 6 compares the performance of the proposed technique (TLZW) with that of the traditional LZW (SLZW)

TABLE 6. (TLZW) VS (SLZW) in terms of bit compression ratio (CR).

Text Format	Test File	Size before (In Bit)	Size After Transformation (T) using Multi-Level Coding (In Bit)	T + LZW (TLZW) (In Bit)	Standard LZW (SLZW) (In Bit)	T-CR	CR TLZW	CR SLZW
Un-vowel	Sciences1.txt	124864	58115	40434	45651	53.46	67.62	63.44
	Health2.txt	140984	68100	45711	49005	51.70	67.58	65.24
	Sports3.txt	250000	117245	80742	87875	53.10	67.70	64.85
	Religion4.txt	157960	76850	46295	54127	51.35	70.69	65.73
	News5.txt	363720	170333	117747	124380	53.17	67.63	65.80
	Holy Quran6.txt	501096	237633	152215	160024	52.58	69.62	68.07
	Miscellaneous7.txt	636008	292276	195181	205832	54.05	69.31	67.64
Partial vowel	Miscellaneous8.txt	116808	49406	34584	42240	57.70	70.39	63.84
	Sciences9.txt	147544	62493	46358	56168	57.64	68.58	61.93
	Health10.txt	164144	73612	49977	55297	55.15	69.55	66.31
	Religion11.txt	188024	79554	50202	60523	57.69	73.30	67.81
	Sports120.txt	286120	127611	90129	97864	55.40	68.50	65.80
	News13.txt	432064	192467	136797	148628	55.45	68.34	65.60
	Miscellaneous14.txt	636416	288009	203644	213533	54.75	68.00	66.45
Full vowel	Holy Quran15.txt	94968	38886	26652	31896	59.05	71.94	66.41
	Sciences16.txt	216488	94905	62242	65905	56.16	71.25	69.56
	Health17.txt	256152	112820	73291	74719	55.96	71.39	70.83
	Religion18.txt	283736	123538	70057	78983	56.46	75.31	72.16
	Sports19.txt	431912	188743	122039	125318	56.30	71.74	70.99
	News20.txt	639032	273195	176954	177720	57.25	72.31	72.19

technique in terms of bit compression ratio. It should be noted that a variable dictionary width was used for both, the difference is the initial dictionary values and the length of the initial phrase code).

The proposed method showed better results than the traditional method for all types of Arabic text, as it did not depend on the structure of words or any grammatical or morphological rules and was better for all forms of text. These results could be attributed to several factors the nature of the data is changed from textual to binary using multi-level encoding to obtain the most regular frequency. However, the size of the original file was reduced prior to the compression.

The results also showed that the file reduction rate before compression ranged between 51% and 57%, depending on the text type and level of mapping used. This indicates that the proposed method can effectively reduce the size of Arabic text before compression, resulting in a higher overall compression ratio than the traditional method.

Fig. 8 shows the results of compressing Arabic texts with all three forms of the same size. The vowel-text compression results were better than those of the partial-vowel and un-vowel texts, respectively. Since diacritics are based on movements whose number is limited to seven [53], the probability that each movement appears with 29 characters increases the repetition at the syllable level, which is a major factor in increasing the compression ratio. This indicates that the proposed method depends on the content; therefore, it is the most effective when applied to text with vowels.

The results in Fig. 9 show an inverse relationship between the original file size and the optimization ratio (i.e., the compression ratio of the traditional method is subtracted from the compression ratio in our approach). It is also noted that the proposed approach achieves the best results for small texts compared to larger texts. This may be owing to the mapping of many unused bits in the dictionary, and the extension of the dictionary size. In the first set of pretests, the phrase was dynamic and had an initial length of 8 bits.

Table 7 shows the results of the second set of tests, which aimed to study the relationship between the initial phrase length (i.e., the minimum in the dictionary) and compression ratio. Experiments were conducted on several Arabic texts in three forms, with sizes ranging from (1) to (200) kilobytes. The same file was tested several times, in which the encoder length was manually specified and passed along with the

TABLE 7. Relation between the initial phrase length and the original data size on compression ratio.

File Name	Size File in Kbyte	TLZW PHARSE SIZE=1	TLZW PHARSE SIZE=2	TLZW PHARSE SIZE=4	TLZW PHARSE SIZE=6	TLZW PHARSE SIZE=7	TLZW PHARSE SIZE=8	TLZW PHARSE SIZE=9	TLZW PHARSE SIZE=10	the Most Efficient
File1	1	55.13	62.96	69.69	**70.63**	68.39	67.97	65.99	65.06	6
File2	5.9	55.75	62.86	67.03	**69.23**	68.08	68.27	65.74	65.74	6
File3	11.6	59.44	66.75	70.18	**72.3**	70.00	71.94	71.6	70.01	6
File4	12	56.33	63.53	67.26	69.55	68.08	**69.57**	66.95	67.9	8
File5	52.7	58.85	62.88	65.29	66.8	68.42	**68.74**	68.39	67.04	8
File6	100	60.04	64.22	67.31	69.06	69.66	**70.45**	70.32	70.28	8
File7	132	61.53	65.08	67.79	69.23	69.85	70.26	**70.32**	70.23	9
File8	150	59.95	63.84	63.84	66.96	68.53	69.4	**69.97**	70.16	9
File9	200	61.53	65.08	67.79	69.23	69.85	70.26	**70.32**	70.23	9

FIGURE 8. Compressing Arabic texts with three forms of the same size.

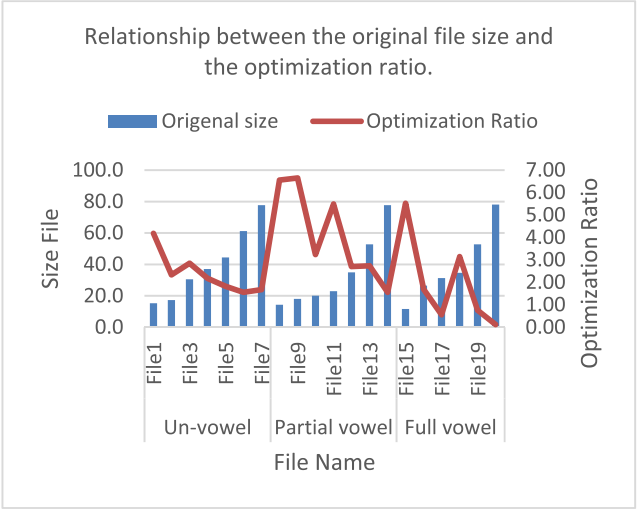


FIGURE 9. Relationship between the original file size and the optimization ratio.

encrypted data to the compression stage, where LZW created an initial dictionary of size (2^ phrase length) and began reading a string of bits from the data source of size equal to the phrase length.

The results indicate that the length of the initial phrase affects the compression ratio, and the appropriate length when compressing files ranging in size from 1 to 11 Kilobyte is six. For files with sizes of 12 to 100, it is 8, and for files with sizes of 100 to 200, the length of the most efficient phrase is 9. Based on these results, the programming of the proposed method was changed to automatically pass a phrase length to the LZW technique, depending on the data size. Create an initial dictionary with a size equal to 2^ ''phrase length'' before compression and decompression.

Fig. 10 comparison the static and dynamic modes of the initial phrase length and their impact on TLZW performance. The results indicate that limiting the initial phrase length according to the file size increases the compression ratio, which is considered an enhancement of our approach when dealing with large sizes. We recommend testing the proposed approach by using large files to determine the optimal phrase length.

In our final set of experiments, we sought to compare our method's compression ratio with that of two other modern compression methods: Deflate, which employs a combination of Huffman coding and LZ77 sliding window compression, and Gzip, which utilizes Deflate in combination with a file format containing additional metadata. Gzip is currently the most widely used and effective compression format on the web that can reduce the size of text files by up to 90%.

Table 8 presents a detailed comparison of our proposed method with traditional and modern compression techniques, while Fig.11 provides a summary of the results.

Our results show that our proposed method outperforms other methods for all forms of Arabic texts, which may be attributed to its dependence on two main principles:

First, the method takes advantage of the unique features of the Arabic language by proposing a multi-level coding system as a precompression step.

TABLE 8. Comparison of our proposed method with traditional and modern compression techniques.

FileName	Text Type	Original Size Kbyte	CR Standard LZW	CR DEFLATE	CR GZIP	Our approch (Multi-Level+LZW)
File10	Full Vowel	9KB	67.47	69.14	68.89	**73.36**
File11	Full Vowel	99KB	72.44	69.11	69.09	**72.61**
File12	Partial Vowel	2KB	58.21	60.8	59.63	**72.15**
File13	Partial Vowel	100KB	69.46	67.31	67.29	**70.45**
File14	Un Vowel	3KB	56.8	50.12	49.41	**69.67**
File15	Un Vowel	6KB	61.59	60.07	59.65	**71.13**
File16	Un Vowel	61KB	68.07	65.26	65.22	**69.62**
Average Compression Ratio			**64.86**	**63.12**	**62.74**	**71.28**

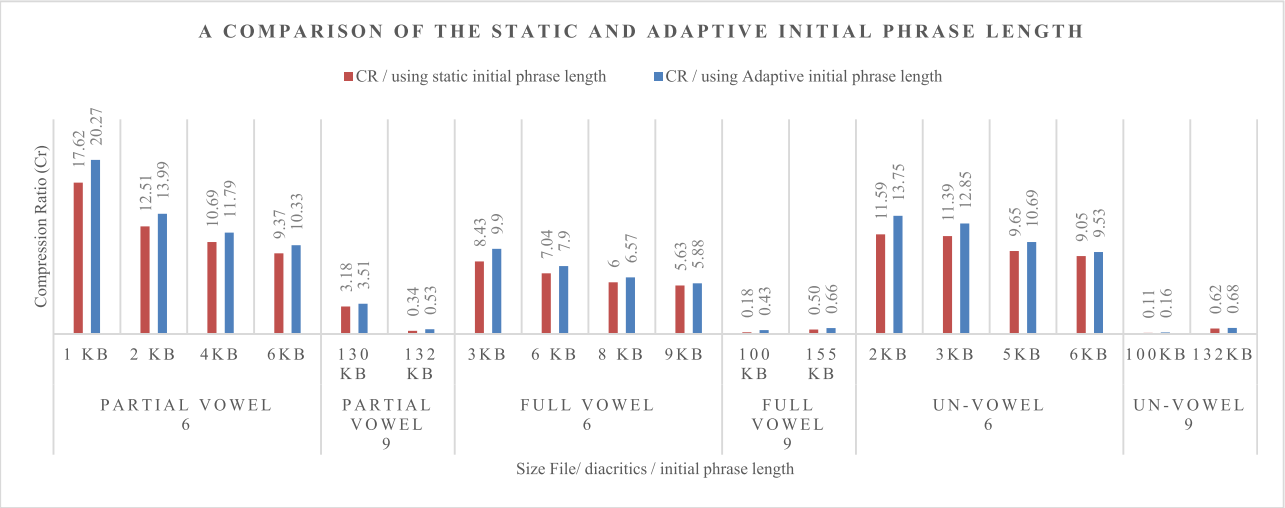


FIGURE 10. Comparison of the static and Adaptive modes of initial phrase length and their impact on TLZW performance.

This system modifies and aligns text with the basic principles of LZW technology, which include:

1) Converting text data into binary data, so that the performance of the technology is not affected by the structure of words or grammatical and morphological rules.
2) Reducing the size of the data before compression, by 51% - 57%, which reduces the deterioration of the size of the technical dictionary as much as possible.
3) Obtaining high-frequency and regular data through the use of fixed dictionaries that are optimal in size, containing no more than 255 words and 255 trigrams (30 letters), and do not require indexing since they use equivalent value storage locations as references. Additionally, unused ASCII character locations are exploited.

Second, adapt LZW's working principle to accommodate input data by setting initial values for the word length of the initial phrase, the initial dictionary data, and the use of variable dictionary size according to the size of the data source.

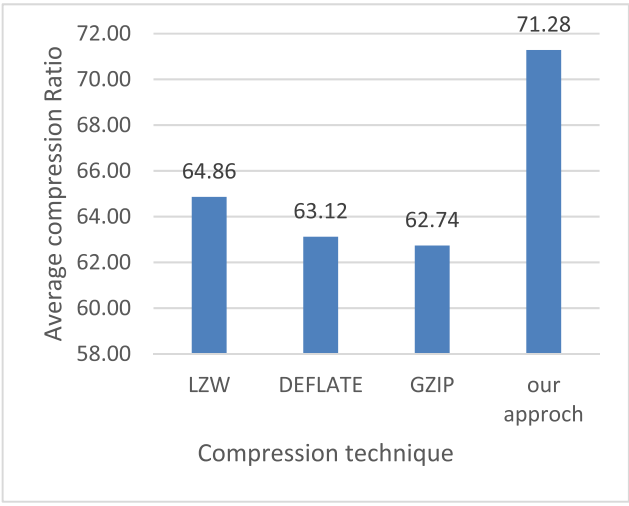


FIGURE 11. Compression ratio comparison between our method, Standard LZW, Deflate and GZIP.

Thus, our findings indicate that the multifactorial approach is effective in enhancing compression ratios. These results

have important implications for data compression in the storage, transmission and retrieval of Arabic language texts.

It should be noted that the performance of the proposed technique is better for texts with vowel characters than for non-vowel texts, and for small files compared to large files. Small files refer to texts that are small in size and complexity as they limit the key factor of compression technologies which is redundancy, and can be found in IoT devices, mobile applications and web applications. So small data compression is important to save storage and bandwidth, improve processing efficiency, and ensure data security.

V. CONCLUSION

In this paper, a new approach for compressing Arabic text using LZW is presented, which includes two stages: conversion and compression, the results of its implementation are discussed.

In the transition phase, the proposed approach effectively exploits the phenomenon of repetition that distinguished the Arabic language at various levels (words, syllables, and letters) to reduce the actual file size by 51%-75%. The text file is converted to binary as an initial step before compression using adaptive LZW technology.

An evaluation of the proposed approach in terms of the amount of compression demonstrated that its performance was better than that of the traditional adaptive LZW technique applied directly to the text. The results also indicate the importance of determining the first phrase's length based on the data source's size and using a variable-length dictionary to improve the compression ratio.

Although the proposed approach uses Adaptive LZW technology in the compression stage, the outputs of the first stage can be embedded with any other general compression technique, particularly the one that works efficiently with binary data.

The main advantage of the proposed approach is that there is no need for statistical or morphological analysis of the text coding. There is no need to send dictionaries to compressed files. The dictionaries used in the first stage were fixed and independent of the file. The dictionary used in the compression stage was adaptive and constructed during the compression and decompression processes. In addition, the size of fixed dictionaries is ideal as it does not exceed 640, and it uses the location of the equivalent value to represent the input as a binary value. In addition, it exploits the location of unused ASCII characteristics. The LZW deals with phrases of variable length, whose initial values depend on the size of the data source. The optimal size for dictionaries reduces public expenditure costs and may positively affect the time of compression and transmission. We propose this perspective as a direction for future research.

In conclusion, our research aimed to improve the performance of LZW compression in Arabic texts. Through a series of experiments on Arabic texts of various shapes and sizes, we found that the performance of the compression technique depends on the characteristics of the language, which must achieve a balance between the characteristics of the language and the style of the technique to be represented by the smallest possible size. Concerning LZW, the length of the initial string code, which varies depending on the size of the data, plays vital role in increasing the performance of LZW compression. The results of this study have important implications for data compression and pave the way for further research in this field.

VI. FUTURE WORK

In future research, we recommend combining a multi-level data encoding scheme with other compression techniques to improve the overall compression ratio. This improvement will contribute to enhancing the performance of mobile devices, saving energy, reducing network consumption, and improving communication efficiency during data transmission and reception in Internet applications and cloud applications.

In addition, it will be interesting to explore the optimal length of initial and variable phrase encryption when compressing large text files using LZW technology and using it as a private key in data security and cryptography applications to improve security and data protection.

REFERENCES

[1] D. Reinsel, J. Gantz, and J. Rydning, *The Digitization of the World From Edge to Core*, vol. 16. Framingham, MA, USA: International Data Corporation, 2018.

[2] D. A. Lelewer and D. S. Hirschberg, "Data compression," *ACM Comput. Surveys*, vol. 19, no. 3, pp. 261–296, Sep. 1987, doi: 10.1145/45072.45074.

[3] U. Jayasankar, V. Thirumal, and D. Ponnurangam, "A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 33, no. 2, pp. 119–140, Feb. 2021, doi: 10.1016/j.jksuci.2018.05.006.

[4] D. Salomon, "Basic techniques," in *Data Compression: The Complete Reference*, London, U.K.: Springer, 2007, pp. 17–46, doi: 10.1007/978-1-84628-603-2_2.

[5] A. Quddus and M. M. Fahmy, "A new compression technique for binary text images," in *Proc. 2nd IEEE Symp. Comput. Commun.*, Jul. 1997, pp. 194–198, doi: 10.1109/ISCC.1997.615995.

[6] K. Sharma and K. Gupta, "Lossless data compression techniques and their performance," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Greater Noida, May 2017, pp. 256–261, doi: 10.1109/CCAA.2017.8229810.

[7] S. Shanmugasundaram and L. Robert, "Text compression algorithms—A comparative study," *ICTACT J. Commun. Technol.*, vol. 2, no. 4, pp. 444–451, Dec. 2011, doi: 10.21917/ijct.2011.0062.

[8] D. Salomon, "Statistical methods," in *Data Compression: The Complete Reference*. London, U.K.: Springer, 2007, pp. 47–169, doi: 10.1007/978-1-84628-603-2_3.

[9] K. Sayood, "Dictionary techniques," in *Introduction to Data Compression* (The Morgan Kaufmann Series in Multimedia Information and Systems), 4th ed., K. Sayood, Ed. Boston, MA, USA: Morgan Kaufmann, 2012, pp. 135–161, doi: 10.1016/B978-0-12-415796-5.00005-3.

[10] D. Salomon, "Dictionary methods," in *Data Compression: The Complete Reference*. London, U.K.: Springer, 2007, pp. 171–261, doi: 10.1007/978-1-84628-603-2_4.

[11] A. Carus and A. Mesut, "Fast text compression using multiple static dictionaries," *Inf. Technol. J.*, vol. 9, no. 5, pp. 1013–1021, Jun. 2010, doi: 10.3923/itj.2010.1013.1021.

[12] B. Vijayalakshmi, "Lossless text compression technique based on static dictionary for unicode Tamil document," *Int. J. Pure Appl. Math.*, vol. 118, pp. 85–91, Jan. 2018.

[13] J. Adiego, M. A. Martinez-Prieto, and P. de la Fuente, "High performance word-codeword mapping algorithm on PPM," in *Proc. Data Compress. Conf.*, Mar. 2009, pp. 23–32, doi: 10.1109/DCC.2009.40.

[14] A. Fariña, G. Navarro, and J. R. Paramá, "Boosting text compression with word-based statistical encoding," *Comput. J.*, vol. 55, no. 1, pp. 111–131, Jan. 2012, doi: 10.1093/comjnl/bxr096.

[15] U. S. Bhadade and A. I. Trivedi, "Lossless text compression using dictionaries," *Int. J. Comput. Appl.*, vol. 13, no. 8, pp. 27–34, Jan. 2011, doi: 10.5120/1799-1767.

[16] C.-E. Wang, "Dynamic LZW for compressing large files," in *Proc. Int. Conf. Found. Comput. Sci. (FCS)*, Las Vegas, NV, USA, Jul. 2011, pp. 57–60.

[17] A. K. Bhattacharjee, T. Bej, and S. Agarwal, "Comparison study of lossless data compression algorithms for text data," *IOSR J. Comput. Eng.*, vol. 11, no. 6, pp. 15–19, 2013, doi: 10.9790/0661-1161519.

[18] H. Al-Bahadili and A. Rababa'a, "An adaptive bit-level text compression scheme based on the HCDC algorithm," *Int. J. Comput. Appl.*, vol. 32, no. 3, pp. 355–361, 2010, doi: 10.2316/Journal.202.2010.3.202-2914.

[19] H. Al-Bahadili and S. M. Hussain, "An adaptive character wordlength algorithm for data compression," *Comput. Math. Appl.*, vol. 55, no. 6, pp. 1250–1256, Mar. 2008, doi: 10.1016/j.camwa.2007.05.014.

[20] P. M. Long, A. I. Natsev, and J. S. Vitter, "Text compression via alphabet re-representation," in *Proc. Data Compress. Conf.*, 1997, pp. 161–170, doi: 10.1109/DCC.1997.582003.

[21] J. Lansky and M. Zemlicka, "Compression of small text files using syllables," in *Proc. Data Compress. Conf. (DCC)*, Snowbird, UT, USA, 2006, p. 458, doi: 10.1109/DCC.2006.16.

[22] W. Abliz, H. Wu, M. Maimaiti, J. Wushouer, K. Abiderexiti, T. Yibulayin, and A. Wumaier, "A syllable-based technique for Uyghur text compression," *Information*, vol. 11, no. 3, p. 172, Mar. 2020, doi: 10.3390/info11030172.

[23] R. S. J, L. College, R. L, and T. Lobo. F, "Dictionary based text filter for lossless text compression," *Int. J. Comput. Trends Technol.*, vol. 49, no. 3, pp. 143–149, Jul. 2017, doi: 10.14445/22312803/IJCTT-V49P122.

[24] J. Dvorsky, J. Pokorny, and V. Snasel, "Word-based compression methods for large text documents," in *Proc. DCC Data Compress. Conf.*, Snowbird, Utah, 1999, p. 523, doi: 10.1109/DCC.1999.785680.

[25] A. Sinaga, Adiwijaya, and H. Nugroho, "Development of word-based text compression algorithm for Indonesian language document," in *Proc. 3rd Int. Conf. Inf. Commun. Technol. (ICoICT)*, May 2015, pp. 450–454, doi: 10.1109/ICoICT.2015.7231466.

[26] B. Stecuła, K. Stecuła, and A. Kapczyłski, "Compression of text in selected languages—Efficiency, volume, and time comparison," *Sensors*, vol. 22, no. 17, p. 6393, Aug. 2022, doi: 10.3390/s22176393.

[27] B. O. Bush, "Language identification of tweets using LZW compression," in *Proc. 3rd Pacific Northwest Regional NLP Workshop, NW-NLP*, 2014.

[28] I. Zitouni, "Natural language processing of semitic languages," in *Theory and Applications of Natural Language Processing*. Berlin, Germany: Springer, 2014, doi: 10.1007/978-3-642-45358-8.

[29] M. A. Zayyat and M. S. Modabbes, "Study on im-pact of changing the nature of data on the overall file compression ratio," *Assoc. Arab Univ. J. Eng. Sci.*, vol. 29, no. 1, pp. 56–63, Apr. 2022, Art. no. 1.

[30] D. Ewell. (2004). *A Survey of Unicode Compression*. [Online]. Available: http://www.unicode.org/notes/tn14/

[31] N. S. Rao, M. P. Kumar T, and M. Naidu Y, "Two phase text data compression using new LZW approach with dynamic bit reduction," *Int. J. Mod. Trends Sci. Technol.*, vol. 8, no. S08, pp. 43–46, 2022.

[32] X. Cheng, B. Li, J. Wu, W. Li, C. Luo, and J. Su, "A novel dictionary management scheme of LZW compression algorithm based on insignificant dictionary area," in *Proc. IEEE 6th Int. Conf. Signal Image Process. (ICSIP)*, Oct. 2021, pp. 975–979, doi: 10.1109/ICSIP52628.2021.9688595.

[33] Y. Tsai and J. Ding, "An improved LZW algorithm for large data size and low bitwidth per code," in *Proc. TENCON - IEEE Region 10 Conf. (TENCON)*, Dec. 2021, pp. 203–208, doi: 10.1109/TENCON54134.2021.9707201.

[34] H. A. Basha, D. S. Arivalagan, and D. P. Sudhakar, "An enhanced low complexity and fast lossless compression technique for textual data," *Int. J. Sci. Technol. Res.*, vol. 9, no. 1, pp. 4013–4017, 2020.

[35] A. Deepa, Nitasha, and N. Chopra, "Intensification of Lempel-Ziv-Welch algorithm," *Int. J. Innov. Technol. Exploring Eng.*, vol. 8, no. 9S, pp. 587–591, Aug. 2019, doi: 10.35940/ijitee.I1092.0789S19.

[36] M. Safieh and J. Freudenberger, "Address space partitioning for the parallel dictionary LZW data compression algorithm," in *Proc. 16th Can. Workshop Inf. Theory (CWIT)*, Hamilton, ON, Canada, 2019, pp. 1–6, doi: 10.1109/CWIT.2019.8929928.

[37] U. Nandi and J. K. Mandal, "A compression technique based on optimality of LZW code (OLZW)," in *Proc. 3rd Int. Conf. Comput. Commun. Technol.*, Nov. 2012, pp. 166–170, doi: 10.1109/ICCCT.2012.40.

[38] *Usage Report of UTF-8 Broken Down by Content Languages*. Accessed: Feb. 13, 2023. [Online]. Available: https://w3techs.com/technologies/breakdown/en-utf8/content_language

[39] Omer, "Arabic short text compression," *J. Comput. Sci.*, vol. 6, no. 1, pp. 24–28, Jan. 2010, doi: 10.3844/jcssp.2010.24.28.

[40] 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," *Contemp. Eng. Sci.*, vol. 6, pp. 111–119, 2013, doi: 10.12988/ces.2013.13010.

[41] D. Barman and M. B. Ahamed, "Improved LZW compression technique using difference method," *Int. J. Innov. Technol. Exploring Eng.*, vol. 9, no. 5, pp. 87–92, Mar. 2020, doi: 10.35940/ijitee.E2216.039520.

[42] R. Ta and R. Ramachandran, "Preprocessed text compression method for Malayalam text files," *Int. J. Recent Technol. Eng.*, vol. 8, no. 2, pp. 1–5, Apr. 2021, doi: 10.35940/ijrte.B1806.078219.

[43] R. T. Anto and R. Ramachandran, "A compression system for unicode files using an enhanced lzw method," *Pertanika J. Sci. Technol.*, vol. 28, no. 4, Oct. 2020, doi: 10.47836/pjst.28.4.16.

[44] A. Musa, A. A. Dmour, O. A. Khaleel, and M. Irshid, "An efficient compression technique using Lempel-ziv algorithm based on dynamic source encoding scheme," *Int. J. Inf. Commun. Technol.*, vol. 2, no. 3, p. 210, 2010.

[45] H. G. Alshammar and D. Alghurair, "Improving compression methods for Arabic Text using dedicated character mapping," *Int. J. Sci. Res.*, vol. 5, no. 11, pp. 1379–1387, 2016.

[46] F. Thaher, "Dynamic with dictionary technique for Arabic text compression," *Int. J. Comput. Appl.*, vol. 135, no. 9, pp. 4–9, Feb. 2016, doi: 10.5120/ijca2016908299.

[47] A. Awajan and E. Abujari, "Hybrid technique for Arabic text compression," *Global J. Comput. Sci. Technol.*, vol. 15, pp. 1–6, Jan. 2015.

[48] C. Wei, X. Dewu, and F. Minfeng, "Chinese text compression algorithm based on LZW," in *Proc. Int. Conf. Comput. Appl. Syst. Model. (ICCASM)*, Oct. 2010, pp. 54–57, doi: 10.1109/ICCASM.2010.5620082.

[49] L. Barua, P. K. Dhar, L. Alam, and I. Echizen, "Bangla text compression based on modified Lempel-Ziv-Welch algorithm," in *Proc. Int. Conf. Electr., Comput. Commun. Eng. (ECCE)*, Feb. 2017, pp. 855–859, doi: 10.1109/ECACE.2017.7913022.

[50] I. Guellil, H. Saâdane, F. Azouaou, B. Gueni, and D. Nouvel, "Arabic natural language processing: An overview," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 33, no. 5, pp. 497–507, Jun. 2021, doi: 10.1016/j.jksuci.2019.02.006.

[51] K. Shaalan and A. Farghaly, "Introduction to the special issue on Arabic natural language processing," *ACM Trans. Asian Lang. Inf. Process.*, vol. 8, no. 4, p. 13, Dec. 2009, doi: 10.1145/1644879.1644880.

[52] M. S. S. Sawalha and E. S. Atwell, "Constructing and using broad-coverage lexical resource for enhancing morphological analysis of Arabic," in *Proc. 7th Conf. Int. Lang. Resour. Eval. (LREC)*, Valleta, Malta: European Language Resources Association (ELRA), May 2010, pp. 282–287.

[53] A. A. Neme and S. Paumier, "Restoring Arabic vowels through omission-tolerant dictionary lookup: Formation of words through computer resources," *Lang. Resour. Eval.*, vol. 54, no. 2, pp. 487–551, Jun. 2020, doi: 10.1007/s10579-019-09464-6.

[54] *A Study of Arabic Letter Frequency Analysis*. Accessed: Feb. 11, 2023. [Online]. Available: http://www.intellaren.com/articles/en/a-study-of-arabic-letter-frequency-analysis

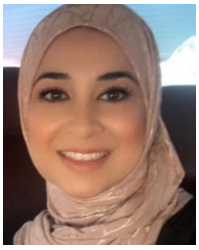
[55] M. Jarrar, F. Zaraket, R. Asia, and H. Amayreh, "Diacritic-based matching of Arabic words," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 18, no. 2, pp. 1–21, Jun. 2019, doi: 10.1145/3242177.

[56] M. K. Saad, "OSAC: Open source Arabic corpora," in *Proc. 6th Int. Conf. Electr. Comput. Syst. (EECS)*, Lefke, North Cyprus, Nov. 2010, pp. 118–123.

[57] A. Farghaly and K. Shaalan, "Arabic natural language processing: Challenges and solutions," *ACM Trans. Asian Lang. Inf. Process.*, vol. 8, no. 4, p. 14, Dec. 2009, doi: 10.1145/1644879.1644881.

[58] T. Abu Hilal, H. Abu Hilal, and A. Abu Hilal, "Multistage Arabic and Turkish text compression via characters encoding and 7-Zip," *J. Ubiquitous Syst. Pervasive Netw.*, vol. 15, no. 1, pp. 11–15, Mar. 2021, doi: 10.5383/JUSPN.15.01.002.

[59] B. Vijayalakshmi and D. N. Sasirekha, "Lossless Tamil compression using ASCII substitution and modified Huffman encoding technique," *Int. J. Recent Technol. Eng. (IJRTE)*, vol. 8, no. 6, pp. 2900–2906, Mar. 2020, doi: 10.35940/ijrte.F8177.038620.
[60] T. A. Hilal and H. A. Hilal, "Arabic text lossless compression by characters encoding," *Proc. Comput. Sci.*, vol. 155, pp. 618–623, 2019, doi: 10.1016/j.procs.2019.08.087.
[61] V. H. Nguyen, H. T. Nguyen, H. N. Duong, and V. Snasel, "*n*-gram-based text compression," *Comput. Intell. Neurosci.*, vol. 2016, Nov. 2016, Art. no. e9483646, doi: 10.1155/2016/9483646.
[62] P. M. Nishad and R. M. Chezian, "Behavioral study of data structures on Lempel Ziv Welch (LZW) data compression algorithm and ITS computational complexity," in *Proc. Int. Conf. Intell. Comput. Appl.*, Mar. 2014, pp. 268–274, doi: 10.1109/ICICA.2014.64.
[63] S. R. Namburi, P. A. V. Krishna Rao, P. K. Muvva, and V. N. Kumar, "An efficient method to reduce LZW algorithm OUPUT code length," *Int. J. Eng. Appl. Sci. Technol.*, vol. 04, no. 11, pp. 302–304, Apr. 2020, doi: 10.33564/IJEAST.2020.v04i11.054.
[64] U. Nandi and J. K. Mandal, "Modified compression techniques based on optimality of LZW code (MOLZW)," *Proc. Technol.*, vol. 10, pp. 949–956, 2013, doi: 10.1016/j.protcy.2013.12.442.
[65] Z. Chai and W. Chen, "An adaptive LZW compres-sion algorithm using changeable maximum-code-length," in *Proc. 4th Int. Conf. Comput. Inf. Technol.*, Wuhan, China, 2004, pp. 1175-1180, doi: 10.1109/CIT.2004.10002.
[66] J. V. S. R. S. H. Bolem, "Design and implementation of efficient LZW compression and decompression technique," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 8, no. 1, pp. 342–351, Jan. 2020, doi: 10.22214/ijraset.2020.1063.
[67] *Usage Statistics and Market Share of Character Encodings for Websites*. Accessed: Feb. 13, 2023. [Online]. Available: https://w3techs.com/technologies/overview/character_encoding
[68] M. K. Saad. (2010). *Open Source Arabic Language and Text Mining Tools*. [Online]. Available: https://sourceforge.net/projects/ar-text-mining/files/Arabic-Corpora/
[69] M. A. Sakhar. *Shakkalli*. Accessed: Feb. 13, 2023. [Online]. Available: https://Tashkeel.alsharekh.org
[70] I. A. El-Khair, "Effects of stop words elimination for Arabic information retrieval: A comparative study," *Inf. Sci.*, vol. 4, no. 3, pp. 119–133, 2006.



ENAS ABU JRAI received the bachelor's degree in software engineering from Al-Hussein Bin Talal University, Jordan, in 2008, and the master's degree in computer science from Middle East University, Jordan, in 2013. She is currently a Faculty Member with Al-Balqa Applied University, Jordan. Her research interest includes text compression.

SHOROQ ALSHARARI received the master's degree in computer science from Middle East University, Jordan, in 2013. She is currently a Lecturer in software engineering with Al-Hussein Bin Talal University (AHU), Jordan. Her current research interests include image processing and software design and implementation.

LAIALI ALMAZAYDEH received the Ph.D. degree in computer science and engineering from the University of Bridgeport, USA, in 2013, specializing in human–computer interaction. She is currently a Professor and the Dean of the Faculty of Information Technology, Al-Hussein Bin Talal University, Jordan. She has published more than 60 research papers in various international journals and conferences proceedings. Her research interests include human–computer interaction and pattern recognition. She received best paper awards in three conferences, ASEE 2012, ASEE 2013, and ICUMT 2016. Recently, she has been awarded two postdoctoral scholarships from European Union Commission and Jordanian–American Fulbright Commission.

KHALED ELLEITHY (Senior Member, IEEE) is currently the Dean of the College of Engineering, Business, and Education, the Associate Vice President of Graduate Studies and Research, and a Distinguished Professor in computer science and engineering with the University of Bridgeport. He has published more than 450 research papers in national/international journals and conferences in his areas of expertise. He is an editor or a co-editor for 12 books published by Springer. His research interests include wireless sensor networks, mobile communications, network security, quantum computing, and formal approaches for design and verification.

OSAMA ABU HAMDAN received the B.S. degree in computer engineering from the University of Jordan, in 2020. He is currently pursuing the Ph.D. degree in computer science and engineering with the University of Nevada, Reno. His research interests include software defined networking and programmable networks.