International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Improving Compression Methods for Arabic Text Using Dedicated Character Mapping

Hedaya Ghanim Alshammar[1], Dina Hamad Alghurair[2]

[1]Higher Institute of Telecommunication and Navigation, HITN, PAAET, Ardia Block4, Street 602, Kuwait

[2]Higher Institute of Telecommunication and Navigation, HITN, PAAET, Ardia Block4, Street 602, Kuwait

Abstract: ***Natural Language Text Compression methods have been discussed thoroughly in the literature in the past years, different methodologies have been implemented and introduced, most however focused on English and European languages. Rather few studies have focused on Arabic Language, some methods used statistical approaches, other methods used dictionary based compression techniques, while some used features of the Arabic language and derivation rules in attempt to increase compression ratio. In this paper, we will introduce several statistical methods for natural language and apply it on Arabic text. We will also provide implementation for each of these methods and give a comparison between them in terms of performance, compression ratio, resource requirements for running the algorithms and areas and application and usage. Golomb , Elias Gamma Code, Huffman methods are to be implemented, and compared as a sample statistical algorithms, We will also introduce a dedicated Arabic Character Mapping technique to be used in the Elias, Golomb and Huffman algorithms, which will show through the results a major improvement to the compression ratio in comparison to the original methods when applied on binary data ignoring the language underneath, the improvement introduced will show that it can be superior even to LZW when used on small Arabic Sample Files, two sets of data will be tested, first set uses random Arabic text, the second set will use real texts from complete Arabic stories and books.***

Keywords: Arabic Text Compression; Golomb Code; Elias Code; Huffman Code; Improved Arabic Character Mapping

1. Introduction

Arabic is one of the most widely spoken languages around the world; hundreds of millions of people speaking Arabic worldwide as their native language, and hundreds of millions speaking it as a secondary language, still not many studies have focused on Arabic text compression and analysis, this makes it an open rich field of research, beginning with text analysis, language parsing, text compression, text recognition, speech recognition, and so goes the list.

The vast need and presence for Arabic content on the World Wide Web, and the continuous increase of Arabic based User Interfaces whether it was in machines or other daily services, makes such studies a necessity and an important field to improve application performance and response, and to save resources.

In today's technology, compression is one of the major aspects being focused on, whether the data is in text format, in signal form, multimedia or any other kind of data. The need for faster application response, better data transmission and space saving that stimulates such studies, so that there will be fewer chances for data loss and more reliable data storage and transmission.

Keeping in mind, this particular problem focuses on data compression in such a way that the encoded data after compression includes fewer bits than the original one.

Several techniques have been proposed yet to encode the original data that is to represent the data, so that it is compressed as compared to the original one.

Compression can be either lossy or lossless. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression. Lossy compression however reduces bits by identifying unnecessary information and removing it taking into account human perception or target application requirements. [1]

This paper aims to compare different compression techniques and their applications on Arabic text. The techniques followed in this proposed project are Elias' Gamma code, Huffman Code, Golomb code, LZW will also be tested.

We will also present an Arabic Character Mapping Technique which will be used in Golomb, Elias and Huffman coding methods, that will substantially improve the compression ratios of these methods in comparison to the rough binary approach, the mapping will include all Arabic characters, including the diacritics and numbers, the results will show that our Golomb method with the Arabic Character mapping is superior to Huffman on Binary data on all sample data files of all different sizes, whereas Huffman when using the Character Mapping will surpass all other methods in average including the rough LZW on binary data.

2. Text Compression Methodologies

Text compressions is another branch of general data compression focusing on natural language scripts, the research has been evolving and used in several areas including search engines; but it also helped in reducing the time to search for the words through the compressed texts (Turpin and Moffat 1997) [2], which helped in creating a new research area of searching compressed text and

compressed text databases where both indexes and text files are stored in a compressed format.

Golomb code is a lossless coding technique which was invented by Solomon W. Golomb [3] for languages or alphabets that have a geometric solution, the Golomb codes should provide optimal prefix code, it is most suitable for cases where small values are much used than higher values, thus providing different coding patterns where small values produce small codes while high values produce higher values, the method uses a very simple approach to produce these codes making it very suitable for applications which have limited resources. Several variations of Golomb encoding have been produced and proposed by several research society members.

The first step in computing Golomb Code is to calculate the three quantities [29]

$$q = \left\lfloor \frac{n-1}{b} \right\rfloor, r = n - qb - 1, c = \lfloor [\log_2 b] \rfloor \quad (1)$$

Imagine an input data stream that consists of positive integers where the probability of integer n appearing in the data is P(n) = (1−p)n−1p, for some 0 ≤ p ≤ 1. It can be shown that the Golomb code is an optimal code for this data if b is chosen such that

$$(1-p)b + (1-p)b+1 \leq 1 < (1-p)b-1 + (1-p)b. \quad (2)$$

Many techniques have been proposed in literature with an aim to utilize the data compression advantages. Normally coding a natural number n ∈ N = {1,2,3,...}; its binary representation needs a [log2(x)] + 1 bits, In [2], authors proposed that that if a large expected length of the Elias gamma code considers the entropy, thus it will be clearly non-optimal. The following formula shows this behavior

$$\lim_{N\to\infty} E[l\delta(X)]/\log N = 1 \quad (3)$$

It clearly shows that for very large N, the expected length of the Elias delta code approaches the Entropy, thus, is asymptotically optimal.

Authors in [4] proposed a research about all the encoding schemes depicted Gamma encoding technique to be a useful tool in case of posting lists, as it decreases the size of posting list 8 times smaller than the original one. This is useful in case of time saving as loading the compressed posting lists form the disk to the main memory, which requires lower time. This in turn decreased the response time of the program greatly. Time is the major factor in today's technology.

Authors in [5] proposed that besides dynamic slicing, the compact byte code generated by variable code encoding scheme finds its advantages in Code Optimization, Program Visualization, Record addresses of objects , Trace the sequence of target addresses

In [6], Byte length variation also results in a lossless compression that is able to achieve comparable efficiency as that of JPEG and is used after Discrete Cosine Transformation (DCT).

A hybrid technique was introduced in [7] where it employs several methods in attempt to achieve better coding results for Arabic text, where the researchers proposed that Using techniques borrowed from other languages or general data compression techniques while ignoring the proper features of Arabic has limited results in terms of compression ratio and speed. So they presented a hybrid technique that uses the features of Arabic language to improve the compression ratio of Arabic texts. The technique used works in several phases. In the first one, the text file is split into four different files using a multilayer model-based approach where semantics of the words are taken into account and their derivation is deduced to produce stem words or return words to their original form, while defining which rule used and save the entire process data. In the second phase, each one of these four files is compressed using the Burrows-Wheeler compression algorithm, which proved to be effective in terms of compression ration when applied to Text Files, which in turn uses a special technique, to represent text block into different format that is suitable for compression, it uses combined method dedicated for text where text letters are transformed into another format to reduce redundancy [7].

Huffman encoding was used in text compression as well, like it was used in general data compression, in [8], where the idea came from that not all characters or bytes require the same amount of space to be stored, thus variable length codes could be created to represent each variation with small value bytes requiring less space, Huffman encoding produces a Huffman table which contains the character encoding values that match each character, this table will become part of the encoded stream, at the beginning of the stream, so the decoder needs to read the Huffman table first before it begins decoding the text stream again, Huffman codes can also be fixed codes, sorted in a Trie where leafs are the final characters, the Trie data structure which is very suitable for text compression and storage, the Trie nodes are binary, 0 or 1, thus scanning the input stream as bits will have a one way traversal route to the leaf which contains the target character.

Dr. Huffman invented an algorithm to create this Trie, and to obtain an optima Trie as well, the generation of the Huffman table with optimal trei requires the scanning of the entire input file, and creating a weighted character list, so this is the first pass through the file, afterwards, the trie is created based on these weights, the algorithm is simple and straight forward, where each character becomes a node in the trie with initial weight that matches the number of occurrences, then the two least weights are combined into one node with the weight as the sum of both weights, the step is repeated tell we get one final node, thus the Trie is created.

Huffman encoding builds its trie based on the probability of character occurrence in the Text, without which you cant really employ this method, however in many circumstances this is not possible, for instance when we are dealing with network stream of text, so methods like Lempel-Ziv-Welch are employed to solve such problem, which uses adaptive algorithms as the encoding happens, LZW [9] which is a variable length coding method, it tries to avoid some of the drawbacks of Huffman encoding, the method also employs a string table that maps symbol sequences to/from an N -bit

index. The string table has 2^N entries and the transmitted code can be used at the decoder as an index into the string table to retrieve the corresponding original symbol sequence. The sequences stored in the table can be arbitrarily long. The algorithm is designed in such a way that the string table can be reconstructed at the decoder end based on information in the encoded stream, the table itself, while is central to the encoding and decoding, is never sent with the data, this is the most important thing to understand in the LZW algorithm, the algorithm, begins with an initial table which represents the entire character spectrum then keeps changing it as it goes by and as needed the algorithm is The encoder algorithm is designed to find the longest possible match in the string table before it makes a transmit statement.

In [10] A morphological compression system is built for Arabic text files, which makes use of the morphological structure of the Arabic language and its features. This concept of compression technique reduces the size of data by replacing some words in the Arabic text by their morphological representation. In Arabic, this representation consists of a root or so called stem word and a pattern combination, which is coded, into the compressed file. The word is morphologically handled by isolating it from all its prefixes and suffixes first, returning it back to its origin in text such as ((يعملون - (عمل) which means (working-> work) pronounced as (Yamaloon- Amel), then the result is reduced to its root and or stem pattern form. A cascaded arrangement of both Word-Based and Character-Based techniques is applied. This mixing is to enhance the compression ratio, where one of the word-based techniques is used to compress the word, and the character-based technique is used for the words that could not be reduced (i.e., compressed) using any of the word-based techniques, such as Huffman, LZW or any traditional methods.

In [11] a multilayer model-based approach is applied for text compression. It uses linguistic features and information to develop a multilayer decomposition model of the text in order to achieve better compression ratios. This new method is illustrated for the of the Arabic language, where it can be utilized by any other language on similar principles and approach, where the majority of words are generated according to the Semitic root-and-pattern scheme by retrieving the stim or the word, similar to the method discussed earlier in [19]. Text is then split into three linguistically homogeneous layers representing the three categories of words: derivative such as(سرحان, يكتبان, يفعلون), nonderivative such as (كرسي, حارة, بيت) and functional words such as (ان, كان, لذلك). A fourth layer, called the Mask layer, it is introduced to aid with the reconstruction of the original text from the three layers at the decoder end. Suitable traditional compression techniques are then applied to the different layers in order to maximize the compression ratio. The proposed approach has been evaluated and tested in terms of the compression ration it provides and its execution time or performance and resources usage. Results are viewed along with sample Arabic texts to illustrate the performance of this new approach. The novelties of the compression technique presented in this article are that (first) the morphological structure of words may be used to support better compression ratio and to improve the performances of traditional compression techniques by making use of the specific language features(Arabic in such case); (2) search for words can be done on the compressed text directly through the appropriate one of its layers; and (3) applications such as text mining and document classification can be performed directly on the compressed texts.

In [12] the author proposed a method for short text compression (suited for message communication, mobile text messages, data control and interfacing) combining the benefits of pre-processing of the data, entropy statistical method reduction through splitting of files and hybrid dynamic coding: A new technique is proposed in this study that uses the fact that Arabic texts have single case letters and one form to write. Experimental tests had been performed on short Arabic texts and a comparison with the well-known plain Huffman compression was made to measure the performance of the proposed schema for Arabic short text.

In [13] evaluation of the efficiency of LZW and BWT techniques for different categories of Arabic text files of different sizes has been applied and briefed. Comparing these techniques on Arabic and English text files is introduced. Additional to exploiting morphological features of the Arabic language to boost the performance of LZW techniques. They found that the enhanced LZW was the best one for all categories of the Arabic texts, then the traditional LZW standard and BWT respectively.

[14] presented another study of Morphological analysis of Arabic words allows decreasing the storage requirements of the Arabic dictionaries, a more efficient method for encoding of diacritical Arabic text, suggestion of faster spelling and efficient Optical character recognition. All these factors allow efficient storage and archival of multilingual digital libraries that include Arabic texts.

The paper presented a lossless compression algorithm based on the affix analysis that takes advantage of the statistical studies of the diacritical Arabic morphological features. The algorithm decomposes a given Arabic word into its stem and its affixes. The affixes which can either e prefix, infix and suffix are the redundant elements of the word which are coded independently using patterns. The roots are stored in the root dictionary which comes as part of the algorithms and only their codes are stored and transferred. Also, they maintained categorized affix dictionaries and their valid combinations to validate and generate the morphological forms during encoding and decoding using a list of patterns and codes. Since the goal is lossless reproducible Arabic text from the patterns and codes, stemming is not an option and noise words (high frequency words) cannot be filtered out.

3. Arabic Character Mapping for Improving Compression Ratio

We present a character mapping technique for mapping Arabic Characters (in our implementation we introduce 56 different Arabic Characters, to include the Alphabet, the Diatrics, the Arabic letters, we also include four more characters the '\t' '\r' '\n' ' ' which also have high statistical

presence in the Arabic Text as well), more characters could also be included like stop marks to reflect the highest statistical distribution of each character.

The mapping was tested in two tiers, the first one where characters are alphabetically ordered according to the Arabic language, followed by the diacritics and the Arabic Numbers, which yielded great results in comparison to the binary approach.

Table 1: ordered character mapping, first tier of tests

[' ',1]	['ر',11]	['ف',21]	['ً',31]	['ء',41]	['٤',51]
['ا',2]	['ز',12]	['ق',22]	['ٌ',32]	['ى',42]	['٥',52]
['ب',3]	['س',13]	['ك',23]	['ٍ',33]	['ة',43]	['٦',53]
['ت',4]	['ش',14]	['ل',24]	['ّ',34]	['أ',44]	['٧',54]
['ث',5]	['ص',15]	['م',25]	['ْ',35]	['\t',45]	['٨',55]
['ج',6]	['ض',16]	['ن',26]	['ٓ',36]	['\r',46]	['٩',56]
['ح',7]	['ط',17]	['ه',27]	['ٔ',37]	['٠',47]	['57,'إ]
['خ',8]	['ظ',18]	['و',28]	['ٰ',38]	['١',48]	['\n',58]
['د',9]	['ع',19]	['ي',29]	['آ',39]	['٢',49]	
['ذ',10]	['غ',20]	['ٖ',30]	['ؤ',40]	['٣',50]	

This mapping is essential for the improved Golomb, Elias and Huffman Custom implementations, All remaining characters that are found in the input stream will be mapped to a higher value numbers, by adding its unsigned character value to the maximum of the mappings which is 61 in our implementation.

Thus allowing our algorithm to adapt to encoding all sorts of characters while still giving the priority to Arabic Characters since they will be found most in the input stream thus expected to have higher frequency count.

The second tier of tests were on running the character mapping while giving the priority to the characters which have higher frequency in the Arabic Text, following a statistical study over the Arabic Character, over a huge set of Arabic texts, we have also made our own statistical data analysis of the Arabic character frequency and compared it to [15], we however used a larger dataset of over 10million characters of Arabic texts(file set in [18] through [28]), which showed a slight difference in the results in [15] for few characters, as shown in figure 1, we also included the space character since it has the highest frequency in any text, thus giving smaller numbers to the most frequent characters, while higher numbers for the less frequent numbers, whereas diacritics and Arabic numbers will be given sequential random frequencies that follows the Arabic characters, considering they are used less often.

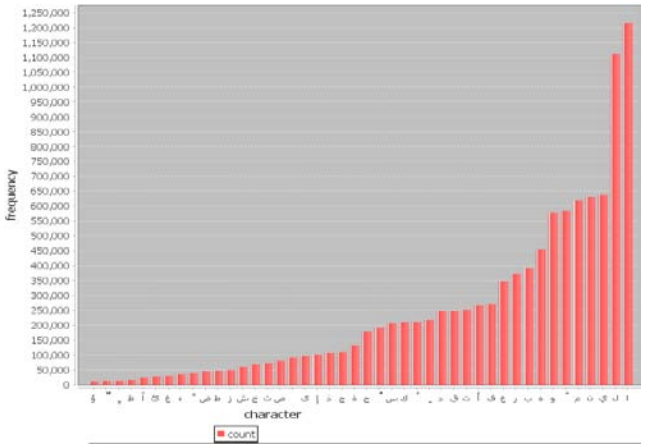

Figure 1: Arabic Character frequencies using our analysis program

Table 2:improved arabic character mapping values

['',1]	['ع',11]	['ج',21]	['ض',31]	['ً',41]	['٠',51]
['ا',2]	['ف',12]	['ذ',22]	['ء',32]	['ّ',42]	['١',52]
['ل',3]	['أ',13]	['إ',23]	['غ',33]	['ٍ',43]	['٢',53]
['ي',4]	['ت',14]	['ى',24]	['ى',34]	['ٌ',44]	['٣',54]
['ن',5]	['ق',15]	['ص',25]	['آ',35]	['ٓ',45]	['٤',55]
['م',6]	['د',16]	['ث',26]	['ظ',36]	['ْ',46]	['٥',56]
['و',7]	['ك',17]	['خ',27]	['ؤ',37]	['ٰ',47]	['٦',57]
['ه',8]	['س',18]	['ش',28]	['ٖ',38]	['\t',48]	['٧',58]
['ب',9]	['ح',19]	['ز',29]	['ٔ',39]	['\r',49]	['٨',59]
['ر',10]	['ة',20]	['ط',30]	['ُ',40]	['\n',50]	['٩',60]

The character mapping previously mentioned will boost the Elias, Golomb, and Huffman substantially, especially Elias Method which the frequency transformation of characters is the essence of the Elias encoding method, this will show the using such statistical mapping will boost Elias Encoding up to 3 times in some experiments, where as boosts its compression ratio by at least two times in the remaining data files in comparison to the rough binary approach of encoding the Arabic Unicode text files.

The usage of the previous Arabic Character mapping method, allows to create a dedicated statistical method for transforming the Arabic characters into numbers of low values, thus allowing the statistical methods like Golomb, Elias and Huffman to make use of this transformation to improve compression ratio, especially the Elias Code.

4. Experiments and Results

Our test results are based upo Unicode Arabic text files ,the average compression ration improvement over traditional binary encoding of the same file is 1:2 and some times 1:3 for Elias, the major improvement over the binary approach is the Elias Encoding, where creating the mapping gives Elias encoding method exactly what it requires to perform at its best.

We have created 2 data sets, first one is a group of 6 Unicode Arabic text files, generated randomly using the website in [16], where different texts were generated and stored in a text file, creating different files of different volumes ranging from 1KB to 1007KB, the second test set, are text files of real

Arabic stories, taken from different textual sources and saved in Unicode text format, so they can be used in our comparative study and analysis, this set includes 11 test files of increasing volume from 9KB to 1317KB.

We have created a separate implementation for Elias and Golomb, where as the Huffman and LZW are borrowed from [17] and created another adaptation for Huffman to include the Arabic Character Mapping for the Huffman algorithm only.
So Golomb, Elias and Huffman all have two different implementations, one that works on Binary files and another that work on Unicode Arabic text files, and the comparison thus is given for 7 different algorithms, the Bit stream reading library is unified amongst the 7 algorithms to have a fair performance comparison.

The following diagrams will show a comparison between the different compression algorithms.

The first set is for Normal mapping, where characters are mapped according to their alphabetical order, tests are applied over two data sets, the mapping is as described in table1 for the V2 version of the algorithm.

V2 name of the algorithm denotes the usage of Arabic Character mapping (Elias V2, Golomb V2, Huffman V2), the file size in the tables bellow is in Bytes.

Table 3: Data set 1, random arabic texts, traditional methods vs. Normal mapping, compression ratio comparison

Compression Ratio Overview								
File Name	File Size	Golomb	Elias	Huffman	LZW	Golomb V2	Elias V2	HuffmanV2
r1.txt	322	71%	69%	66%	72%	28%	41%	56%
r2.txt	5670	80%	98%	47%	57%	46%	42%	30%
r3.txt	16110	80%	98%	47%	46%	45%	42%	29%
r4.txt	51590	80%	98%	46.00%	37%	45.00%	42%	28%
r5.txt	154812	79%	98%	46.00%	30%	45.00%	42%	28%
r6.txt	1030884	80%	98%	46.00%	23%	45.00%	42%	28%
Average		78%	93%	50%	44%	42%	42%	33%

The above table shows how the V2 version of the algorithms have improved after using the character mapping, where Elias took the most advantage of this.

Table 4: Data set 1, random arabic texts, traditional methods vs. Improved mapping, compression ratio comparison

Compression Ratio Overview								
File Name	File Size	Golomb	Elias	Huffman	LZW	Golomb V2	Elias V2	HuffmanV2
r1.txt	322	71%	69%	66%	72%	28%	42%	56%
r2.txt	5670	80%	98%	47%	57%	43%	35%	30%
r3.txt	16110	80%	98%	47%	46%	43%	35%	29%
r4.txt	51590	79%	98%	46.00%	30%	43.00%	35%	28%
r5.txt	154812	79%	98%	46.00%	30%	45.00%	42%	28%
r6.txt	1030884	80%	98%	46.00%	23%	43.00%	35%	28%
Average		78%	93%	50%	43%	41%	37%	33%

The above table shows how changing the character mapping order according to the highest frequency boosts the Elias by additional 5% in average and the Golomb by additional 1% in the average results, but it didn't change the Huffman ratios.

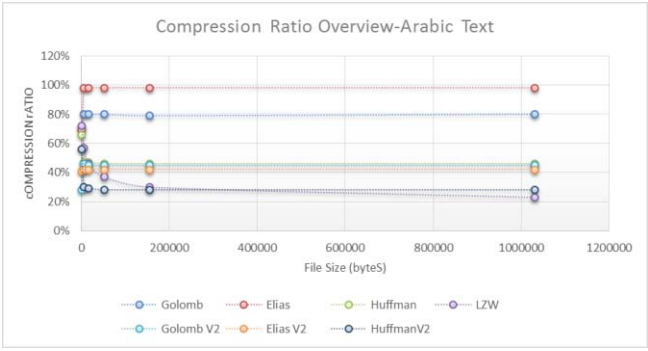

Figure 2: Data set 1, Random Arabic Texts, Traditional Methods vs. Normal Mapping, Compression ratio comparison

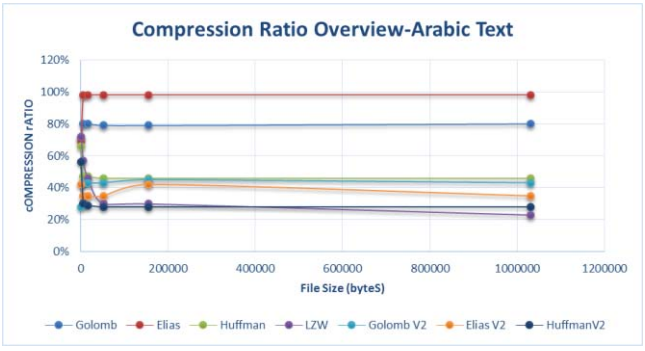

Figure 3: Data set 1, Random Arabic Texts, Traditional Methods vs. Improved Mapping, Compression ratio comparison

Figure 2 and 3 draws the table 3 and 4 and only coveys the information graphically.

The figure bellow, shows the encoding time for the different algorithms, it clearly shows that all algorithms have practically very close encoding time except for the LZW, which increases encoding time dramatically with the increase of the file size, but at the same time, its compression ratio also increases due the fact that the more data comes in, the longer character lists are created and new codes are added to the coding table, thus creating new codes for longer lists as the steam scanning moves forward.

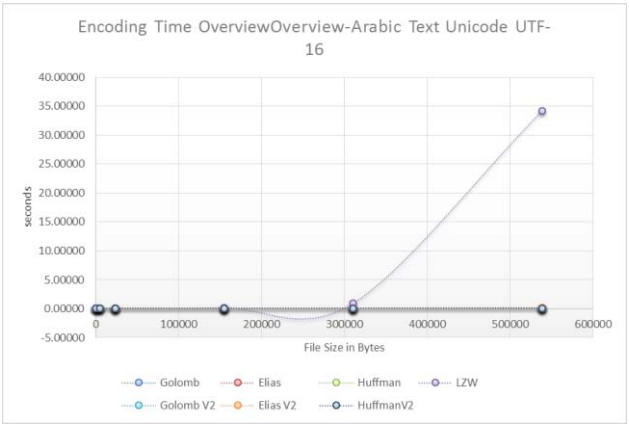

Figure 4: Data set 1, Random Arabic Texts, Normal Mapping, Encoding Time comparison

Since the LZW has a special curve, we have created another diagram without the LZW, so that we can take a closer look at the remaining algorithms performance, thus the figure bellow is only for Huffan,Golom, Elias and their V2 counterparts.

The 6 algorithms performs well, and their performance

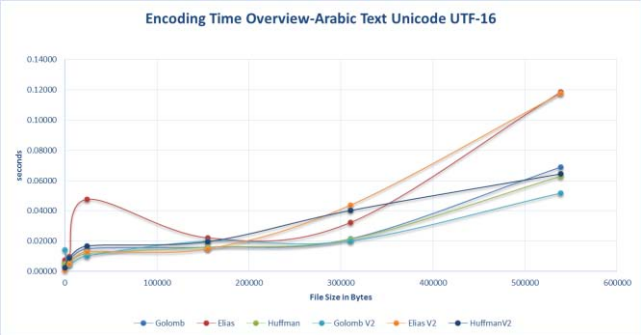

Figure 5: Data set 1, Random Arabic Texts, Normal Mapping, Encoding Time comparison (without LZW), a closer overview

The figure and table bellow, shows that LZW has the best decoding time, whereas the Golomb, Elias and Huffman have similar curves, with Huffman surpassing Golomb and Elias in decoding time, as the sample file size increases the decoding time generally in increasing uniformly.

Table 5: Decoding time for the algorithms, lzw average the best

Decoding Time Overview								
File Name	File Size	Golomb	Elias	Huffman	LZW	Golomb V2	Elias V2	HuffmanV2
r1.txt	322	0.0026	0.0011	0.0013	0.0009	0.0015	0.0011	0.0010
r2.txt	4832	0.0080	0.0086	0.0121	0.0026	0.0109	0.0045	0.0040
r3.txt	23976	0.0238	0.0098	0.0176	0.0067	0.0113	0.0067	0.0093
r4.txt	155118	0.0102	0.0216	0.0052	0.0055	0.0222	0.0118	0.0091
r5.txt	310242	0.0270	0.0582	0.0185	0.0102	0.0140	0.0323	0.0084
r6.txt	538814	0.0684	0.1143	0.0386	0.0469	0.0424	0.0861	0.0449
average		0.0233	0.0356	0.0156	0.0121	0.0170	0.0238	0.0128

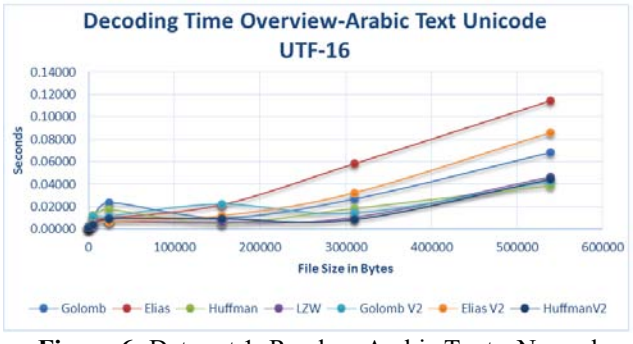

Figure 6: Data set 1, Random Arabic Texts, Normal Mapping, Decoding Time comparison

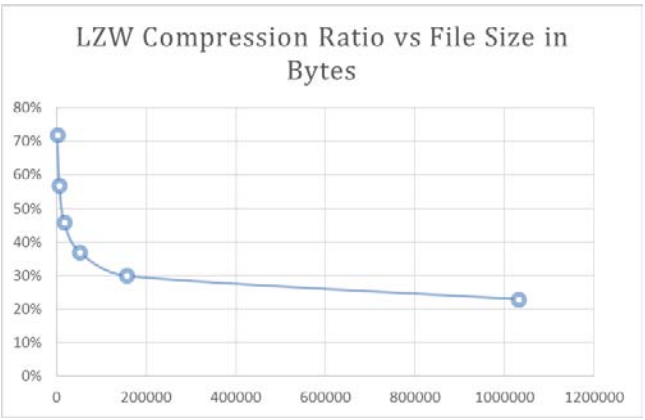

Figure 7: Data set 1, Random Arabic Texts, Normal Mapping, LZW compression ratio vs. file size.

The above figure shows how LZW compression ratio increases as the file size increases, which was discussed earlier and explained why this happens in this method.

Following we will present tests over the second data set, which contains some real Arabic texts from the literature, stories, books, this set contains 11 files.

Table 6: 1 data set 2, real arabic texts, traditional methods vs. Normal mapping, compression ratio comparison

Compression Ratio Overview								
File Name	File Size (bytes)	Golomb	Elias	Huffman	LZW	Golomb V2	Elias V2	HuffmanV2
1.txt	8620	80%	99%	45%	50%	42%	39%	28%
2_story1.txt	15436	80%	99%	47%	47%	90%	46%	30%
3_pubdoc_1_11756_281.txt	20242	81%	99%	46%	43%	41%	41%	29%
4_alef-ibn-arabi.txt	34372	80%	99%	45%	38%	42%	40%	28%
5_lellah_0.txt	265512	80%	98%	46%	30%	45%	42%	29%
6_ar_Book_online_Ale3tesam.txt	326316	80%	99%	47%	29%	49%	43%	29%
7_bukhla.txt	515288	80%	99%	45.00%	28%	40.00%	40%	28%
8_tahfut.txt	540200	80%	100%	45.00%	26%	40.00%	40%	27%
9_najat.txt	895734	80%	99%	45.00%	25%	40.00%	40%	27%
10_ilhyat.txt	975142	80%	99%	45%	24%	45%	42%	28%
11_ar_Fath_Almajeed.txt	1348564	80%	98%	45%	25%	44%	40%	27%
Average		80%	99%	46%	35%	48%	41%	28%

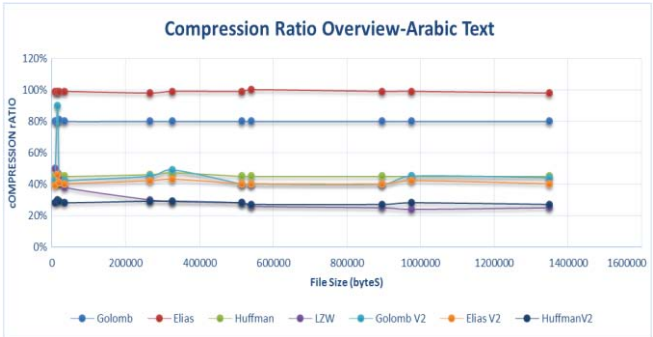

Figure 8: Data set 2, Real Arabic Texts, Traditional Methods vs. Normal Mapping, Compression ratio comparison

Table 7: data set 2, real arabic texts, traditional methods vs. Improved mapping, compression ratio comparison

Compression Ratio Overview								
File Name	File Size (bytes)	Golomb	Elias	Huffman	LZW	Golomb V2	Elias V2	HuffmanV2
1.txt	8620	80%	99%	45%	50%	37%	32%	28%
2_story1.txt	15436	80%	99%	47%	47%	88%	39%	30%
3_pubdoc_1_11756_281.txt	20242	81%	99%	46%	43%	40%	34%	29%
4_alef-ibn-arabi.txt	34372	80%	99%	45%	38%	40%	32%	28%
5_lellah_0.txt	265512	80%	98%	46%	30%	42%	34%	29%
6_ar_Book_online_Ale3tesam.txt	326316	80%	99%	47%	29%	46%	36%	29%
7_bukhla.txt	515288	80%	99%	45.00%	28%	38.00%	32%	28%
8_tahfut.txt	540200	80%	100%	45.00%	26%	38.00%	31%	27%
9_najat.txt	895734	80%	99%	45.00%	25%	37.00%	32%	27%
10_ilhyat.txt	975142	80%	99%	45%	24%	42%	33%	28%
11_ar_Fath_Almajeed.txt	1348564	80%	98%	45%	25%	41%	32%	27%
Average		80%	99%	46%	35%	45%	34%	28%

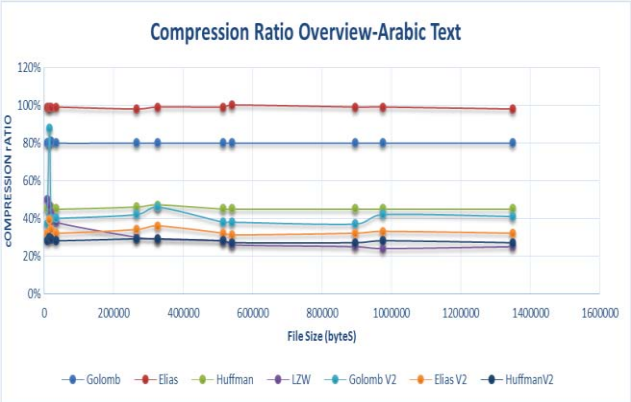

Figure 9: Data set 2, Real Arabic Texts, Traditional Methods vs. Improved Mapping, Compression ratio comparison

5. Arabic Text Compression Algorithms Applications & Analysis

Our work includes comparison between four methods, LZW, Elias, Golomb, Huffman, where we only applied the improvement to the later three, according to our tests, each encoding algorithm has several different characteristics that makes it more unique in certain areas than the other, thus makes it more applicable in that particular scenario.

Golomb and Elias encoding methods are very simple in implementation, produced great results with small files of less than 1000bytes, with compression ratios almost as equal to Huffman and LZW methods, when applied in Binary, where as Golomb and Elias were far more superior over small sample files when applying Arabic Character Mapping technique, it can be easily implemented in hardware, and on mobile handsets with low memory and low processing capabilities.

Binary Elias encoding performed the worse in terms of compression ratio with larger text file, when applied in its Binary form, which practically no longer reduced the file size, it became more of an encoding method that only changed the character encoding without affecting the size ratio, whereas Binary Golomb stabilized at 80% compression ratio for larger file sizes and no longer improved or became worse, while still maintaining good execution time and memory usage in comparison to the other methods such as Huffman and LZW.

Elias and Golomb coding methods cant perform properly without using the Character Mapping of the Arabic Language, using it made both algorithms superior in every way to Binary methods of Huffman, LZW, Elias and Golomb over small samples under 2000 Bytes, this makes them perfect for mobile and devices with low resources, our implementation provided a great balance between ratio-speed-memory usage and simplicity of encoding.

The Huffman encoding method practically had a good balance of both performance and compression ratio, stabilizing at 46% for larger file sizes over 20Kbytes, when applied as Binary algorithm, despite having a more complicated encoding concept, still the encoding and decoding implementation only needed few lines of code and few to encode after building the trei, drawback of this algorithm is that it needs to know the input file in advance in order to perform properly, so it doesn't suite streaming applications where only potions of data is being streamed.

Improving Huffman using Arabic Character Mapping has put Huffman in the lead of all other algorithms, the compression ratio stabilizes around 28% for larger files.

LZW showed the best compression ratio, superior to any other method tested here when dealing with larger file sizes. which stabilized at 24% compression at larger sample files sizes 300KB+, despite having a randomly generated Arabic text files and different files with completely different Arabic content with diacritics, the encoding time however is increasing exponentially with the increase of input volume.

LZW however showed the best decoding time, the reason goes to the fact that LZW can encode several characters from the input stream as one Code-Word, thus decoding one Codeword could result in several characters, allowing the decoder to increase the speed of the decoding process as well as writing the characters to the output stream, the Codeword length is a vital parameter where 15 showed to give the best encoding speed and compression ratio of 24% thus reducing the file size more than 4 times, however the bigger the code word is the more memory resources the application needs to withstand the 2^w entries in the symbol table.

LZW didn't perform well on small file sizes when using large code words like 15, thus this algorithm needs to make the code word dependent on the size of the file to perform properly.

LZW tests here have been applied with the Arabic Text File as Binary data, its compression ratio might improve a lot if we used the Arabic Character Mapping the same way we did with Golomb Elias, and Huffman, also when making the

Codeword parameter variable with the file size, we will get better results for smaller file sizes than the ones we got before, however we won't for now, in the future we might do.

So the previous facts shows that depending on the target application, one might need to use LZW if compression ratio is of most important, or if decoding time is the criteria, when dealing with large file sizes, whereas Golomb-Elias can be the best choice if a balance between decoding, encoding, compression ratio is required.

Golomb and Elias with Arabic Character Mapping produced the best result in terms of balance of compression ration-encoding-decoding time, Golomb and Elias surpassed Huffman in small files less than 1Kilo Bytes.

Huffman with Character Mapping is the best method in average, the average test results showed that it surpassed the other methods in average over various sample file sizes, our custom implementation proved that dedicated methods of Arabic Text Compression can widely improve the compression ratios of the algorithms.

6. Future Improvements

This study can be improved by introducing the concept of stemming, removing the prefixes and suffixes of the Arabic words, then applying character mapping over the stems of the words, while encoding the affixes by a special order, several layers for encoding and decoding each word, that depends on the word and its derivatives, the addition of such information will require a more in depth study of the Arabic language and in cooperation of di]:ctionary and additional information at the encoder and decoder ends.

7. Acknowledgment

This paper was created based upon a research on Arabic Text Compression algorithms, in *Higher Institute for Communication and Training, PAAET*

References

[1] G. Wade, Signal coding and processing, Cambridge University Press, 1994
[2] A. Moffat, A. Turpin. (1997). On the implementation of minimum redundancy prefix codes.IEEE Transactions on Communications,45, 1200–1207
[3] S.W. Golomb, (1966). , Run-length encodings. IEEE Transactions on Information Theory, IT--12(3):399--401
[4] A. Bagherzandi , K. Y. Oktay , "Enhancing Wikipedia Search Performance Using Elias Gamma Code", 2010
[5] T. Wang, A. Roychoudhury, "Using Compressed Bytecode Traces for Slicing Java Programs", 2004
[6] M. Gorev, P. Ellervee, "Variable byte-length data compression algorithm", Electronics Conference (BEC), 2010 12th Biennial Baltic, pp. 353 – 356, 2010
[7] A. Awajan,, A. Enas, Hybrid Technique for Arabic Text Compression, Global Journal of Computer Science and Volume 15 Issue 1 Version 1.0 Year 2015
[8] Z., Julie, Huffman Encoding and Data Compression, May 23, 2012
[9] MIT 6.02 DRAFT, Compression Algorithms: Huffman and Lempel-Ziv-Welch (LZW),2012
[10] S. Al-Fedaghi, B, Al-Sadoun, "Morphological Compression of Arabic Texts," Computer Journal of Information Processing & Management, vol. 26, no. 2, pp. 303-316, 1990.
[11] A. Arafat, Multilayer Model for Arabic Text Compression, the International Arab Journal of Information Technology, Vol. 8, No. 2, April 2011
[12] O. man , K. Khalaf, Arabic Short Text Compression, Journal of Computer Science 6 (1): 24-28, 2010
[13] A. Essam, B. Abdullah, a Framework to Automate the Parsing of Arabic Language Sentences, December 5, 2007
[14] A. Daoud, "Morphological Analysis and Diacritical Arabic Text Compression," Computer Journal of the International Journal of ACM Jordan, vol. 1, no. 1, pp. 41-47, 2010.
[15] *[Online]* www.intellaren.com/articles/en/a-study-of-**arabic-letter-frequency**-analysis*, 2016*
[16] [Online]: http://generator.lorem-ipsum.info/_arabic , 2016
[17] S. Robert, W. Kevin, Princeton University, Algorithms 4th Edition 2011
[18] [Online] : http://qurancomplex.gov.sa/Quran/tafseer/Tafseer.asp?t=KATHEER&TabID=3&SubItemID=1&l=arb , Tafseer Ibn Katheer, Second Edition, 2016
[19] [Online] http://studentform.mediu.edu.my/public/upload/439f4bb88aee644d85d080a8b9304725.docx, 2016
[20] [Online]: bukhlahttp://www.muslimphilosophy.com/ip/bukhla.doc. 2016
[21] [Online]:http://www.muslimphilosophy.com/books/najat.doc, 2016
[22] [Online]:http://www.muslimphilosophy.com/books/ilhyat.doc, 2016
[23] [Online]: http://ghazali.org/works/tahfut.doc, 2016
[24] [Online]: https://d1.islamhouse.com/data/ar/ih_books/single3/ar_Book_online_Ale3tesam.doc, 2016
[25] [Online], http://d1.islamhouse.com/data/ar/ih_books/single/ar_Fath_Almajeed.doc, 2016
[26] [Online]:http://www.tasavof.ir/books/download/arabic/ibn-arabi/alef-ibn-arabi.doc, 2016
[27] [Online]: http://www.ibs.edu.jo/files/lellah_0.doc, 2016
[28] [Online]: http://www.ucas.edu.ps/Units/Research_Unit/specialForms/8.doc, 2016
[29] S. David, Data Compression: The Complete Reference, 2nd Edition, p. 53, 2000.

Author Profile

Hedaya Ghanim Alshammar is Specialist Trainers in Higher Institute for Communication and Training, PAAET, Kuwait, having Masters of Science in Computer Engineering and Information

Systems, Gulf University, Bahrain. Hedaya Ghanim Alshammar is working in the field of education and research.

Dina Hamad Alghurair is Specialist Trainers in Higher Institute for Communication and Training, PAAET, Kuwait, having Masters of Science in Computer Engineering and Information Systems, Gulf University, Bahrain. Dina Hamad Alghurair is working in the field of education and research