information

*Article*

# A Syllable-Based Technique for Uyghur Text Compression

**Wayit Abliz [1,2], Hao Wu [2], Maihemuti Maimaiti [1,2], Jiamila Wushouer [1,2], Kahaerjiang Abiderexiti [1,2], Tuergen Yibulayin [1,2] and Aishan Wumaier [1,2,*]**

1 School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China; wayit@xju.edu.cn (W.A.); mahmutjan@xju.edu.cn (M.M.); jamila@xju.edu.cn (J.W.); kaharjan@xju.edu.cn (K.A.); turgun@xju.edu.cn (T.Y.)

2 Xinjiang Laboratory of Multi-Language Information Technology, Xinjiang University, Urumqi 830046, China; wuhao94@aliyun.com

* Correspondence: hasan1479@xju.edu.cn; Tel.: +86-136-599-13514

Received: 23 February 2020; Accepted: 18 March 2020; Published: 23 March 2020

**Abstract:** To improve utilization of text storage resources and efficiency of data transmission, we proposed two syllable-based Uyghur text compression coding schemes. First, according to the statistics of syllable coverage of the corpus text, we constructed a 12-bit and 16-bit syllable code tables and added commonly used symbols—such as punctuation marks and ASCII characters—to the code tables. To enable the coding scheme to process Uyghur texts mixed with other language symbols, we introduced a flag code in the compression process to distinguish the Unicode encodings that were not in the code table. The experiments showed that the 12-bit coding scheme had an average compression ratio of 0.3 on Uyghur text less than 4 KB in size and that the 16-bit coding scheme had an average compression ratio of 0.5 on text less than 2 KB in size. Our compression schemes outperformed GZip, BZip2, and the LZW algorithm on short text and could be effectively applied to the compression of Uyghur short text for storage and applications.

**Keywords:** text compression; Uyghur; syllable; code table

## 1. Introduction

Network data on the internet continues to increase significantly each year. In 2018, for the mobile internet only, access traffic reached 71.1 billion GB in China. Text messaging and instant messaging consume huge amounts of storage and communication resources. Text compression is a type of lossless compression technology that improves storage space utilization and text transmission efficiency. Text compression technology mainly employs statistics-based and dictionary-based methods. These methods have distinct advantages and disadvantages, depending on the specific application, and they also operate differently. Statistics-based methods use the statistical information of characters (or other basic units of the language) to generate shorter-length codes, such as run-length coding, Shannon–Fano coding, and Huffman coding [1–3]. Run-length coding has better compression performance than the other two when several consecutively repeated elements occur. Shannon–Fano coding uses a top-down building tree, which has low coding efficiency and long average coding length. It is rarely used in practical applications. Huffman coding encodes the sequence according to the probability of character occurrence, so that the average code length is the shortest. This method has average compression efficiency for those characters with average probability of occurrence. The dictionary-based methods, such as the LZ77 and LZ78 [4,5] algorithms, perform compression-decompression by constructing a dictionary mapping table. The LZ77 algorithm uses a sliding dynamic dictionary to store local historical information and replaces duplicate content with an index of the elements in the dictionary for encoding. The LZ78 algorithm also uses a dynamic dictionary to store information, extracts character strings from the character stream, and represents them by numbers and encodes the repeated character strings.

## 2. Related Research

Most text compression algorithms are character or word based, excluding language texts with other word forms [6–9]. Previous researchers have proposed a syllable-based text compression method and a scheme for lossless compression of short data [10–16]. Nguyen et al. [15] proposed a syllable-based Vietnamese text compression algorithm. This method first constructs syllable and consonant dictionaries, and then it uses fixed code bits to represent the information, such as syllables, tones, and capitalization. Akman et al. [16] proposed a syllable-based Turkish text compression algorithm. On the basis of the analysis of the nature of syllables, this method implements encoding by counting repeated syllables and setting special symbols, such as spaces and English characters. Oswald et al. [17] proposed an algorithm that further reduces text redundancy by finding patterns in the text and using these patterns with the LZ78 algorithm. Bharathi et al. [18] proposed an incremental compression method. This method implements variable-length encoding to perform data compression and can perform searches on compressed data. Several other researchers have proposed compression techniques for different languages [19–24]. At present, little research has been conducted on text compression in the Uyghur language. Xue et al. [25] tested Uyghur compression using Huffman and LZW algorithms. In a text containing only Uyghur characters, the Unicode-based Uyghur text is converted into Latin characters and then is compressed. The drawback of this method is that it is not capable of compressing text mixed with other language characters.

In this study, we used a large amount of Uyghur corpora for syllable statistical analysis, performed reverse ordering according to the frequency of occurrence of syllables, and selected high-frequency syllables to construct syllable code tables. On the basis of two fixed-length coding schemes, we constructed 12-bit and 16-bit syllable coding dictionaries. We also processed other language characters and spaces accordingly. We achieved good coding performance with this technique.

## 3. Syllables of Uyghur

Uyghur is a typical agglutinative language. It has strong derivational ability and rich morphological variations. Uyghur has 32 letters in total, with 24 consonants and 8 vowels. A word consists of a combination of syllables with no special signs between the syllables. The inherent syllable structure is (initial sound) + nucleus sound + (final sound), where the nucleus sound must be a vowel. There can be no initial sound or final sound in the syllable, but there must be a nucleus sound [26–28]. The present-day Uyghur is classified into 12 syllabic types. The syllable classification is shown in Table 1, where C represents a consonant and V a vowel.

**Table 1.** Uyghur syllable classification.

| No. | Syllabic Structure | Example | No. | Syllabic Structure | Example |
|---|---|---|---|---|---|
| 1 | V | ئا/A | 7 | CCV | ستالىن/Stalin |
| 2 | VC | ئات/At | 8 | CCVC | فرانكا/Franka |
| 3 | CV | كالا/Kala | 9 | CCVCC | تىرانسپورت/Tiransport |
| 4 | CVC | نان/Nan | 10 | CVV | خۇا/Hua |
| 5 | VCC | ئەرز/Ärz | 11 | CVVC | تۈەن/Tüän |
| 6 | CVCC | خەلق/Hälq | 12 | CCCV | ستروميتىر/Strometir |

In Table 1, the examples include the present-day Uyghur and the Latinized transition forms. The syllabic structures of nos. 7–12 are used mainly to record words that have foreign origin. Each syllable of the syllabic structures for no. 10 and no. 11 have two vowels, which are used mainly to record words with two vowels from Chinese and other languages, such as *Zhonghua* and *Guangdong*. The inherent feature of Uyghur syllables is that a syllable contains only one vowel and may contain no consonants, and thus the number of vowels in a word is theoretically equal to the number of syllables in the word. The following three problems need to be solved to implement this syllable segmentation:

1. Some loanwords from Chinese have two vowels, such as *tüän* and *hua*.
2. No more than one consonant should appear in front of the vowel, but some loanwords from foreign languages have more than one consonant in front of a vowel, such as *Stalin* and *Strategiyä*.
3. When syllables are segmented, the syllabic structure of two vowels of certain loanwords from Chinese and the syllabic structure of multiple consonants of certain words from foreign languages are prone to making the segmentation algorithm ambiguous, such as syllabic type 11 (CVVC), which structurally is a combination of syllabic type 3 (CV) and type 2 (VC). When a character string that has the CVVC structure occurs in a word, identifying whether the string has one syllable or two is a key issue for the syllable segmentation.

## 4. Syllable Segmentation and Selection

### *4.1. Syllable Segmentation and Analysis*

The corpus constructed in this paper included a collection of 52,718 articles from a variety of journals, government documents, scientific and literary works, and short documents such as social media posts. We performed syllable segmentation on all 713,716 unique words and expressions appearing in these articles, using the segmentation method described by Wayit et al. [29]. We found a total of 8621 different syllables belonging to the 12 syllabic structures using this syllable segmentation. We counted these syllables for each syllabic structure, and the statistics are given in Table 2.

**Table 2.** Statistics of Uyghur syllable.

| Structure | Number of | | Frequency | Structure | Number of | | Frequency |
|---|---|---|---|---|---|---|---|
| | Theoretical | Actual | | | Theoretical | Actual | |
| V | 8 | 8 | 31,653 | CCV | 4608 | 425 | 1358 |
| VC | 192 | 172 | 34,002 | CCVC | 110,592 | 688 | 2829 |
| CV | 192 | 184 | 593,850 | CCVCC | 2,654,208 | 151 | 287 |
| CVC | 4608 | 2992 | 441,376 | CVV | 1536 | 260 | 1358 |
| VCC | 4608 | 294 | 1319 | CVVC | 36,864 | 394 | 2194 |
| CVCC | 110,592 | 2956 | 11,667 | CCCV | 110,592 | 97 | 180 |

In Table 2, a theoretical syllable number reflects all of the syllables that can be composed of 24 consonants and 8 vowels in this structure. For example, the CCV syllabic structure theoretically can generate 24 * 24 * 8 = 4608 syllables. The actual number is the number that occurs based on the structure when counting 713,716 words during the syllable segmentation. For example, the CCV structure has only 425 syllabic structures. Frequency of occurrence is the number of occurrences of all syllables based on the type of syllabic structure among all of the words. From the statistical results, we found that the number of occurrences for the six inherent syllabic structures accounted for the majority of the syllables. According to the statistical results given in Table 2, the average syllable length (ASL) calculated by Equation (1) was 2.4 characters.

$$ASL = \frac{\sum(Syllable\ Length \times frequency)}{\sum frequency} \tag{1}$$

### *4.2. Selection of High-Frequency Syllables*

Figure 1 shows the Zipf's law distribution of 8621 syllables in the corpus. All syllables are sorted in descending order of frequency *f*. The highest syllable ranks as *r = 1*. In the figure, the *x*-axis is the logarithm of the ranking *r*, and the *y*-axis is the logarithm of the frequency *f*. Figure 2 shows the syllable coverage. The $x$-axis is the top $n$ syllables, and the $y$-axis is the logarithm of the sum of the frequencies of the top $n$ syllables. The calculation of *log fn* by Equation (2) follows

$$log f_n = log(\sum_{r=1}^{r=n} f_r) \quad (2)$$

For example, *log r = 2.0* on the $x$-axis corresponds to *log f = 0.605* on the $y$-axis. This means that the first 100 syllables with the highest frequency can cover 60% of the words in the corpus. After the 2000th syllable, the increase in the syllables does not significantly increase the coverage.

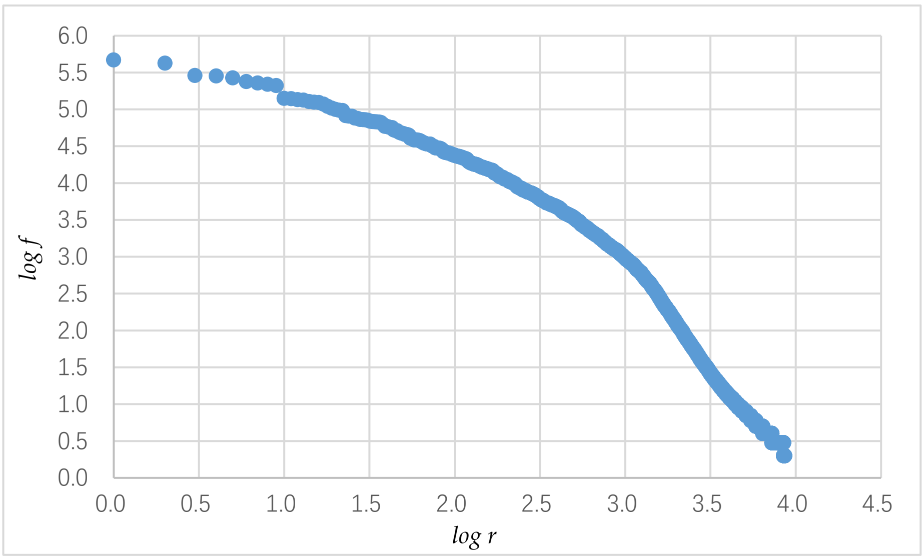


**Figure 1.** Zipf's law distribution of syllables.

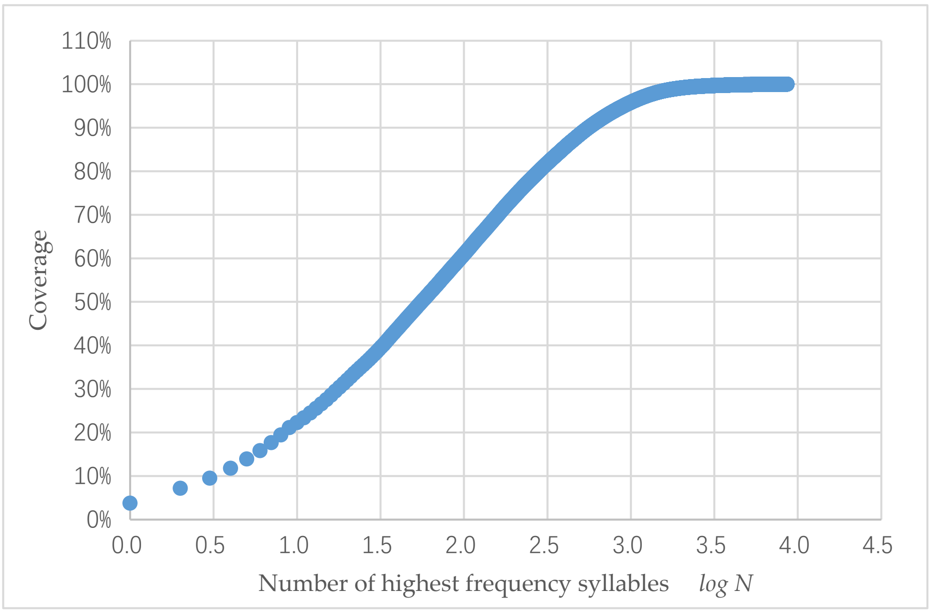


**Figure 2.** Coverage of high-frequency syllables.

## 5. Data Compression Coding

### *5.1. Syllable Coding*

In this study, we used two coding schemes to encode the syllables: the 12-bit short coding scheme B12 and the 16-bit long coding scheme B16. The B12 scheme included an 11-bit syllable code table. The B16 scheme included a 16-bit syllable code table. We used these two coding schemes in conjunction with Unicode encoding, which we used to encode the characters in the nonsyllable coding table. In actual application, to identify the characters not included in the code tables, we had to add some identification flag to these Unicode codes.

In this paper, we used dicChar and dicSyllb to represent the encoded characters and syllables in the dictionary and used xChar and xSyllb to represent the uncoded characters and syllables in the dictionary. xSyllb was composed of several dicChars, and SP was a space (U0020).

#### 5.1.1. B12 Coding Scheme

The most common word division symbol in Uyghur is a space (U0020). In the B12 scheme, we set the first bit to the space flag bit. A flag bit of "1" indicated that the current syllable was followed by a space (i.e., the end syllable of the word). A flag bit of "0" indicated that no space appeared after the current syllable was encoded (i.e., the first syllable or intermediate syllables of the word).

Excluding the space flag bit, we called the remaining 11 bits of the 12-bit short code bits the syllable code bits. The 11-bit syllable code bits contained 2048 code positions—that is, the 11-bit syllable code table contained 2048 syllable codes. We classified the code positions as follows:

1. ASCII characters: The frequency of ASCII characters in Uyghur was higher than that of other symbols, such as Chinese characters, so we treated each ASCII character as a syllable, and thus we left the first 128 encoding positions for ASCII characters (0x00–0x7F).
2. Uyghur characters: The code range of Uyghur characters was in the Unicode basic block (U0600–U06FF). This block occupied 33 code positions in the syllable coding, including 24 consonant characters "[ل], [ڭ], [گ], [ك], [ق], [ف], [غ], [ش], [س], [ژ], [ز], [ر], [د], [خ], [چ], [ج], [ت], [پ], [ب], [ي], [ۋ], [ھ], [ن], [م] " and eight vowel characters "[ۆ], [ۇ], [ۈ], [و], [ى], [ې], [ە], [ا] " and special character HAMZE [ئ] (U0626).
3. Uyghur commonly used punctuation marks: Common punctuation marks included "،" (U060C), "؛" (U061B), "؟" (U061F), and "-" (U0640). They occupied four code positions.
4. Other commonly used punctuation marks in Uyghur: "«" (U00AB), "»" (U00BB), " … " (U2026), four-per-em space (U2005), left-to-right mark (U200E), and right-to-left mark (U200F). They occupied six code positions.
5. We reserved 15 positions to flags, for describing various situations in the data stream.
6. High-frequency Uyghur syllables: Excluding the previously mentioned syllable codes, 1862 code positions remained. As shown in Figure 1, the coverage of 1862 syllable codes was around 98%, which contained the more commonly used high-frequency syllables.

These characters belonged to dicChar, and the remaining characters belonged to xChar. The xChar also included the non-Uyghur characters in (U0600–U06FF) and the other Uyghur characters in the Unicode extension areas (UFE70–UFEFF) and (UFB50–UFDFF).

#### 5.1.2. B16 Coding Scheme

In the B16 coding scheme, the code length of a syllable was exactly equal to the length of the Unicode character code. This facilitated subsequent research on syllable-based text retrieval of compressed text. We selected the Private Use Area (UE000–UF8FF) with 6400 positions as the 16-bit long code block. The number of syllables occurring in the previously noted corpus was 8621. This block was large enough to accommodate most of the syllables that occurred. This scheme did not use the space flag.

In this scheme, if the text encountered xSyllb, we used the Unicode source code directly. If it was a character in the Private Use Area, we resolved it by attaching an identifier flag.

#### 5.1.3. Code Block Division

The code blocks of the previous two coding schemes are shown in Table 3.

**Table 3.** Code ranges of the two coding schemes.

| **No.** | **Encoding Entity** | **Encoding Range** | | |
|---|---|---|---|---|
| | | **Unicode** | **B12** | **B16** |
| 1 | ASCII characters | U0000–U007F | 0x000–0x07F | Unchanged |
| 2 | Uyghur characters | U0600–U06FF | 0x080–0x0A0 | Unchanged |
| 3 | Uyghur punctuation marks | U0600–U06FF | 0x0A1–0x0A4 | Unchanged |
| 4 | Other punctuation marks | U00AB, U00BB, U2005, U200E, U200F, U2026 | 0x0A5–0x0AA | Unchanged |
| 5 | Selected syllables | NULL | 0x0AB–0x7F0 | UE003–UF8FF |
| 6 | Flags | NULL | 0x7F1–0x7FF | UE000–UE002 |

The B12 scheme contained 1862 high-frequency syllables. The B16 scheme using the Private Use Area contained 6400 code positions.

### *5.2. Design of Flag Coding*

#### 5.2.1. B12 Scheme Flags

The purpose of the identification flag is to identify the xChar and xSyllb that appeared in the data stream. We first selected some of the corpora shown in Table 4 to make statistics based on the length of the xChar string. The length probability is shown in Figure 3. From the xChar length, the number of xChars with string lengths of 1 and 2 were the highest, and the probability that the length was greater than 8 was very low. A length of 1 was found primarily in Chinese characters and other symbols (①(1)√ VI ‰, etc.), and a length of more than 2 was found mainly in Chinese characters. The flag and their meanings based on the statistical results are shown in Table 5.

**Table 4.** Corpus information for determining xChar length.

| **No.** | **Type** | **Content** | **Size (MB)** | **No.** | **Type** | **Content** | **Size (MB)** |
|---|---|---|---|---|---|---|---|
| 1 | Hot news | CCTV Focus Interview Program (2018/7 to 2018/12) | 8.36 | 7 | Book | *Roosevelt Biography* | 2.33 |
| 2 | Natural sciences | CCTV Human and Nature Program (2015 to 2018) | 5.57 | 8 | Novel | *Farmer's Son Wang Leyi* | 1.43 |
| 3 | General news | CCTV News Hookup (2018/7 to 2018/12) | 19.8 | 9 | Other | Glossary of scientific terms | 1.6 |
| 4 | Agricultural technology | CCTV Get Rich Program (51st-94th, 2018) | 4.76 | 10 | Other | Name list of various institutions in Xinjiang | 0.6 9600 in total |
| 5 | Educational books | *Elementary and Secondary Child Psychology* | 1.6 | 11 | Short text (mobile terminal) | Web message, comment, SMS, WeChat chat data | 1 2908 in total |
| 6 | Popular science books | *The World's Most* | 0.6 | Corpus total size 46.8 MB; unique words 136,523; and unique syllables 7434. | | | |

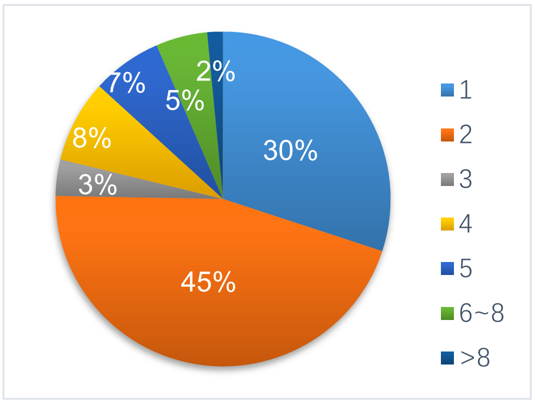


**Figure 3.** Length probabilities of xChar strings.

**Table 5.** Flag code for B12 coding scheme.

| No. | Flag | Code | Meaning | Note |
|---|---|---|---|---|
| 1 | SDB | 0xE000 | Syllable Data Begin | Start decoding |
| 2 | fXC | 0x7F1–0x7F9 | xChar begin | Will encounter xChar sequences of length 0 < n < 10, each xChar encoding length is 16 bits, n = fXC-0x7F0 |
| 3 | fXBB | 0x7FA | xChar Block Begin | Will encounter xChar sequences with length n > 9, start to intercept 16bit data until fXBE is encountered |
| 5 | fXBE | 0xE002 | xChar Block End | The end of the xChar sequence, starting to intercept 12-bit data |
| 4 | ESD | 0x7FF | End of syllable data | Stop decoding |

If one string *S*, including 2 ASCII characters a1, a2 (encoding in the encoding table was A1, A2); 2 Russian characters r1, r2 (Unicode R1, R2); 10 Chinese characters c1–c10 (Unicode C1–C10); 1 dicSyllb composed of 2 Uyghur characters u1, u2 (encoding in the dictionary was $S_1$); 1 xSyllb composed of 3 Uyghur characters u3, u4, u5 (Unicode U3, U4, and U5); and 3 Private Use Area characters e1, e2, e3 (Unicode was E1 = 0xE000, E2 = 0xE001, E3 = 0xE002), then the encoding result of string S was

$$S_{string} = \text{a1a2r1r2c1}\sim\text{c10u1u2u3u4u5e1e2e3}$$
$$S_{encoding} = \textbf{SDB} + \text{A1A2} + \textbf{fXC}\ (\mathit{0x7F2}) + \text{R1R2} + \textbf{fXBB}\ (\mathit{0x7FA})\ \text{C1-C10}\textbf{fxBE}(\mathit{0xE002}) + S_1 + \textbf{fXC}(\mathit{0x7F3})\ \text{U3U4U5} + \textbf{fxC}(\mathit{0x7F3})\text{E1E2E3}.$$

In the B12 scheme, the length of fXC and fXBB was 12 bits, and the length of fXBE was 16 bits. The advantage of this design was that it was easy to identify the fXC when it started to read 12 bits continuously and read n xChars directly, according to the value of fXC-0x7F0. When it encountered fxBB, it started to read 16 bits. When it read fXBE, it indicated the end of the xChars sequence. Then it restarted reading 12 bits.

### 5.2.2. B16 Scheme Flags

The B16 scheme identification flags and meaning are shown in Table 6.

**Table 6.** Flag code for B16 coding scheme.

| No. | Flag | Code | Meaning | Note |
|---|---|---|---|---|
| 1 | SDB | 0xE000 | Syllable data begin | Start decoding |
| 2 | fXC | 0xE001 | Private use area | Identifies that the next character is a private area character |
| 3 | ESD | 0xE002 | End of syllable data | Stop decoding |

If string *S* contained 2 ASCII characters a1, a2 (Unicode A1, A2); 2 Russian characters r1, r2 (Unicode R1, R2); 10 Chinese characters c1–c10 (Unicode C1–C10); 2 dicSyllb (encoding in the dictionary $S_3$(U1,U2) = 0xE003, $S_5$(U3,U4) = 0xE005); 1 xSyllb (three Uyghur characters with Unicode U3, U4, U5); and 4 private area characters e3, e5, e3, and e5 (Unicode E3 = 0xE003, E5 = 0xE005), then the encoding result of *S* was

$$\mathrm{S}_{\mathrm{string}} = \mathrm{a1a2r1r2c1\text{-}c10u1u2u3u4u3u4u5e3e5e3e5}$$
$$\mathrm{S}_{\mathrm{encoding}} = \mathbf{SDB} + \mathrm{A1A2} + \mathrm{R1R2} + \mathrm{C1\text{-}C10} + \mathrm{S}_3(\mathit{0xE003}) + \mathrm{S}_5(\mathit{0xE005}) + \mathrm{U3U4U5} +$$
$$\mathbf{fxC}(\mathit{0xE001})\ \mathrm{E3} + \mathbf{fxC}(\mathit{0xE001})\mathrm{E5} + \mathbf{fxC}(\mathit{0xE001})\mathrm{E3} + \mathbf{fxC}(\mathit{0xE001})\mathrm{E5}.$$

We used the B16 to research syllable-based text retrieval. When we retrieved syllable $S_3S_5$ from the encoded string $S_{encoding}$, the $S_3S_5$ code 0xE003 and 0xE005 appeared three times in total, and the last two times had the identifier flag fXC, which could be excluded. When retrieving e3e5, we encoded e3e5 into fXC + E3 + fXC + E5 before retrieval according to the input content. This excluded the encoding of $S_3S_5$ without an identifier. If e3e5e3e5 was encoded with fXBB + E3E5E3E5 + fXBE, then when the e3e5 was retrieved, the system encoded e3e5 into fXBB + E3E5+ fXBE. There was no matching content in $S_{encoding}$. This was why the B16 scheme did not design the fXBB and fXBB flags of the B12 scheme.

### 5.2.3. Datagram and File Format

The format design of the compressed datagram and compressed file is shown in Figure 4. SDB was a 2-byte identifier, which indicated that the compressed data stream started from this scheme. CodetabID was a code table ID in a dictionary 1-byte long. A dictionary could contain 255 code tables. A coding table represented a language or a language's different coding scheme. After receiving the data stream, the decoder performed decoding according to the encoding table ID. Sdata was the data stream to be decoded, and ESD was the end of the data stream. The ESD length in the B12 scheme was 12 bits, and the ESD length of the B16 scheme was 16 bits.

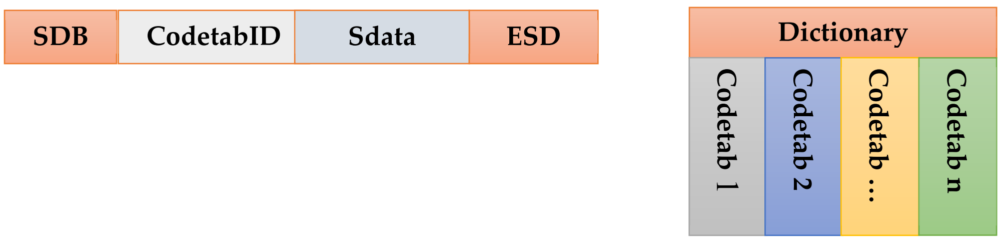


**Figure 4.** Datagram and compressed file format design.

In the adaptive dictionary method, the encoder generated a dictionary based on the text content and compressed it. Finally, the compressed data stream and file came with dictionary information. We read the data stream during decoding to determine whether the current item was identified or uncompressed data, and we looked up the final output for the decompressed data in the dictionary based on the identification. Because a string *S* had different frequencies and positions in the text *T*, the encoding in the compressed text *Z* was also different. In this study, we proposed a method to use the syllable characteristics of natural language to generate a general static dictionary for compression and decompression. The actual compressed data did not have a dictionary. Regardless of the frequency and position of a string *S* in the text *T*, its encoding in the compressed text *Z* was the same.

## *5.3. Data Compression Process*

Taking the B12 scheme as an example, the implementation process of the compression method proposed in this paper is shown in Figure 5. The process functions in the flowchart are as follows:

CheckXBlockEnd: Check the previous code first. If the previous encoding was xChar and the number of this xChar and the previous consecutive xChars exceeded 9, we added an fXBE identifier to the current data stream.

SetXCharFlag: Append the fXC or fXBB flag to the data stream according to the current xChar/xSyllb length.

AddSPCode: Check the previous code first. If it was dicSyllb/dicChar and no space flag bit had been added, we added a space flag bit to the previous code. If it was not, we added a space 12-bit code (0x020) to the data stream.

Finally, we generated a bit sequence of text. If the total length of the text bit was not divisible by 8, then we added several "0" bits, so that the sequence length was divisible by 8. Then, we converted the bit sequence into a byte sequence for storage.

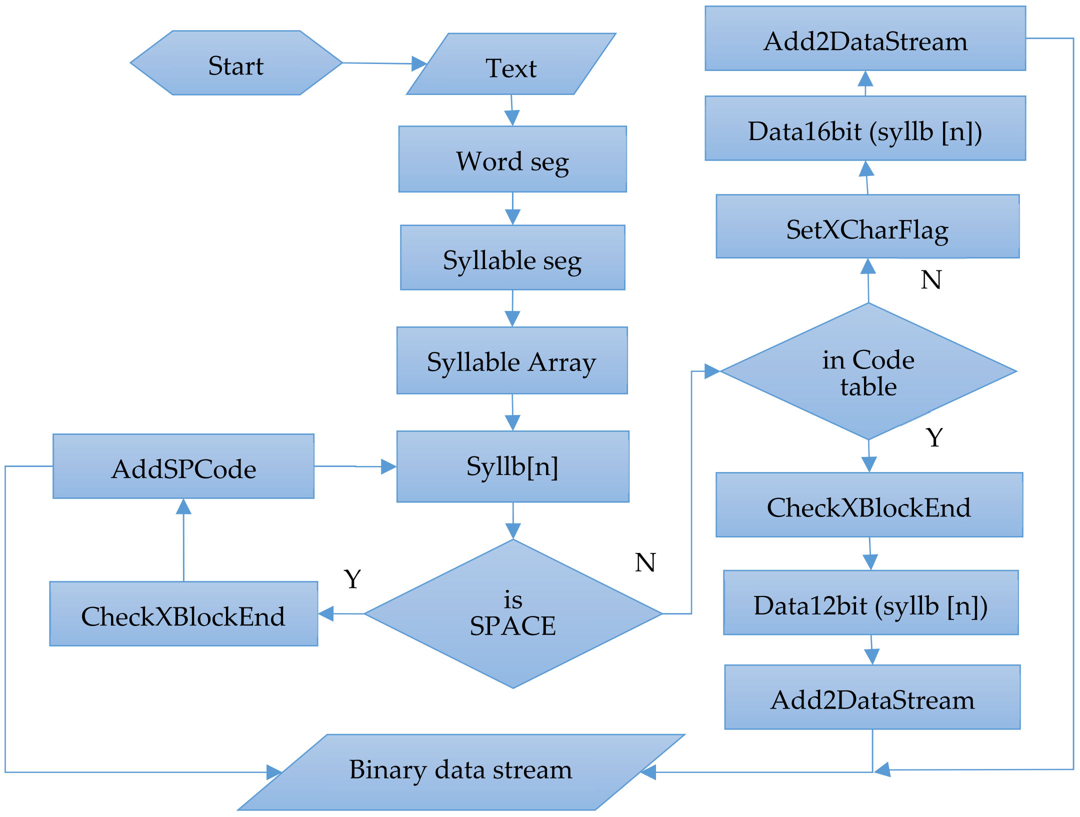
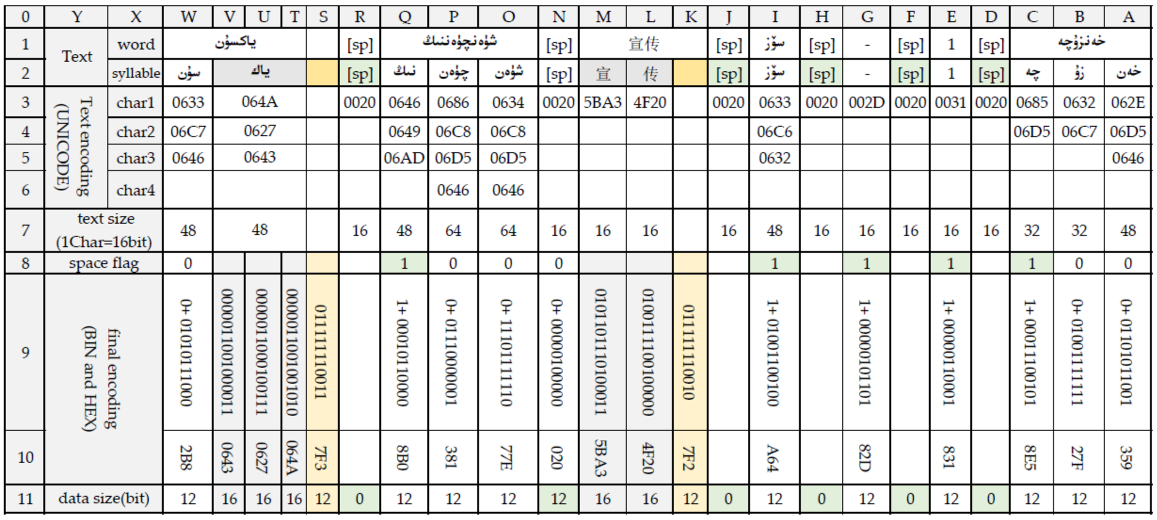


**Figure 5.** B12 scheme flowchart.

Figure 6 shows an example of B12 coding. The reading direction of Uyghur text in the text was from right to left, and the original reading direction of other characters in the text remained unchanged. The reading order of the text in Figure 6 was A, B . . . →K→M→L→O, P . . . →W. The original text to be compressed had two xChar (two Chinese characters L2, M2) and one xSyllb (three characters T2, U2, V2). Line 1 was the result of the original word segmentation; Line 2 was the syllable segmentation of words, Line 3 to Line 6 were the Unicode encoding of the inner characters of these syllables, and Line 9 and Line 10 were the final encoding results. Line 8 was a space flag bit.



| 0 | Y | X | W | V | U | T | S | R | Q | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Text | word | ياكسۇن | | | | | [sp] | شۈەنچۈەننىڭ | | | [sp] | 宣传 | | | [sp] | سۆز | [sp] | - | [sp] | 1 | [sp] | خەنزۇچە | | |
| 2 | | syllable | سۇن | ياك | | | | [sp] | نىڭ | چۈەن | شۈەن | [sp] | 宣 | 传 | | [sp] | سۆز | [sp] | - | [sp] | 1 | [sp] | چە | زۇ | خەن |
| 3 | Text encoding (UNICODE) | char1 | 0633 | 064A | | | | 0020 | 0646 | 0686 | 0634 | 0020 | 5BA3 | 4F20 | | 0020 | 0633 | 0020 | 002D | 0020 | 0031 | 0020 | 0685 | 0632 | 062E |
| 4 | | char2 | 06C7 | 0627 | | | | | 0649 | 06C8 | 06C8 | | | | | | 06C6 | | | | | | 06D5 | 06C7 | 06D5 |
| 5 | | char3 | 0646 | 0643 | | | | | 06AD | 06D5 | 06D5 | | | | | | 0632 | | | | | | | | 0646 |
| 6 | | char4 | | | | | | | | 0646 | 0646 | | | | | | | | | | | | | | |
| 7 | text size (1Char=16bit) | | 48 | 48 | | | | 16 | 48 | 64 | 64 | 16 | 16 | 16 | | 16 | 48 | 16 | 16 | 16 | 16 | 16 | 32 | 32 | 48 |
| 8 | space flag | | 0 | | | | | | 1 | 0 | 0 | 0 | | | | | 1 | | 1 | | 1 | | 1 | 0 | 0 |
| 9 | final encoding (BIN and HEX) | | 0+ 01010111000 | 0000011001000011 | 0000011000100111 | 0000011001001010 | 011111110011 | | 1+ 00010110000 | 0+ 01110000001 | 0+ 11101111110 | 0+ 00000100000 | 0101101110100011 | 0100111100100000 | 011111110010 | | 1+ 01001100100 | | 1+ 00000101101 | | 1+ 00000110001 | | 1+ 00011100101 | 0+ 01001111111 | 0+ 01101011001 |
| 10 | | | 2B8 | 0643 | 0627 | 064A | 7F3 | | 8B0 | 381 | 77E | 020 | 5BA3 | 4F20 | 7F2 | | A64 | | 82D | | 831 | | 8E5 | 27F | 359 |
| 11 | data size(bit) | | 12 | 16 | 16 | 16 | 12 | 0 | 12 | 12 | 12 | 12 | 16 | 16 | 12 | 0 | 12 | 0 | 12 | 0 | 12 | 0 | 12 | 12 | 12 |

**Figure 6.** Example of B12 coding scheme. (Reading order A, B . . . →K→M→L→O, P . . . →W).

## 5.4. Data Compression Process

### 5.4.1. Compression Ratio

The compression ratio was an indicator used to evaluate the performance of a compression method. The compression ratio (*CR*) is calculated by Equation (3)

$$CR = \frac{Compressed\ file\ size}{orginal\ file\ size} = \frac{S_c}{S_o} \tag{3}$$

where $S_O$ is the size of the original text, $S_C$ is the size of compressed data, and its value consists of a Unicode encoding portion $P_U$ and a binary encoding portion $P_B$.

1. Compression Ratio of the B12

Any element in the $P_B$ was a binary code obtained by matching a 12-bit short code table. The length of $P_B$ is calculated by Equation (4)

$$P_B = SyllCount \times L_{enc} \tag{4}$$

where $P_U$ consists of xChar and xSyllb. When one element of the $P_U$ was *Pi = char1char2... charn*, there were two types of coding sequences. When the number of characters in *Pi* was *CharCount (Pi) < 10*, $P_{i\,(n<10)}$ = ***fCX****char1char2... charn*, and when *CharCount (Pi) > 9*, $P_{i\,(n>9)}$ = ***fXBB****char1char2... charn****fXBE***. The calculation formula of the $P_U$ length is shown in Equations (5) and (6), and its unit is a bit

$$P_{U(char\ count<10)} = \sum_{i=1}^{i=n} (12bit + CharCount(P_i) \times 16bit)\ ,\ \text{and} \tag{5}$$

$$P_{U(char\ count>9)} = \sum_{i=1}^{i=m} (12bit + CharCount(P_i) \times 16bit + 16bit). \tag{6}$$

According to Figure 6, the original text size was *So = 592bit*. The original text had 19 syllables, of which *dicSyllb = 10* and a space (N2 unit) that could not be marked with space flag, $P_B = 10 \times 12bit + 12bit = 132bit$, two Chinese characters (L2, M2 unit), *xChar = 2*, $P_{i1} = 2 \times 16 + 12 = 44bit$, and one xSyllb (T2, U2, V2 unit) with three characters $P_{i2} = 3 \times 16 + 12 = 60bit$, total $Sc = P_B + P_{i1} + P_{i2} = 132 + 44 + 60 = 236bit$. The *CR* calculation result was *CR = 0.399*.

2. Compression Ratio of the B16

The calculation formula of $P_B$ part was the same as in equation (4), *Lenc = 16 bit*. When xSyllb appeared in $P_U$, its structure was $P_{is}$ = *uchar1uchar2... ucharn*, as follows

$$P_U = \sum_{i=1}^{i=n} (CharCount(P_{is}) \times 16bit). \tag{7}$$

When the character string *Pix = char1char2... charn* encoded in the Private Use Area (UE000-UF8FF) appeared, the encoding sequence was *Pix =* ***fXC****char1****fXC****char2...* ***fXC****charn*, so the formula for calculating the PU length was

$$\mathrm{P_U} = \sum_{\mathrm{i=1}}^{\mathrm{i=n}} (\mathrm{CharCount}(\mathrm{P_{ix}}) \times 2 \times 16\mathrm{bit}). \tag{8}$$

According to Figure 6, compressed data size *Sc = (8 dicSyllb + 6 space + 2 ASCII character)* × *16 + (3 xSyllb character + 2 Chinese character)* × *16 = 256 + 80 = 336 bit*. The CR calculation result was *CR = 0.57*.

### 5.4.2. Average Coding Length

Average coding length was another indicator used to evaluate text compression performance. The shorter the average coding length, the more efficient the compression method. $S_C$ was the size of the compressed text in bits. The average encoding length *BPC* (*bits pec character*) was calculated by Equation (9)

$$BPC = \frac{Sc}{CharCount}. \tag{9}$$

According to Figure 6, the compressed text size was *Sc = 236* bit and had 37 characters, and the *BPC* calculation result was *BPC = 6.378* bits, which was 9.622 bits fewer than when Unicode code was used.

## *5.5. Data Decompression*

In the data decompression process, a compressed text file is decompressed. The decompression process is the reverse of the compression process. We used the following specific decompression process.

### 5.5.1. Decompression of B12 Scheme

We read the original text in bytes, converted each byte into an 8-bit binary, and generated a continuous bit stream.

1. Intercept 12 bit; if it was dicSyllb, read the next 12 bits.
2. If the decoding result was fXC, n Unicode characters would be intercepted continuously, where *n = fXC-0x7F0*, and each character was 16 bits in length.
3. If the decoding result was fXBB, then the Unicode characters would be intercepted continuously until fXBE was read.
4. If there was no remaining bit data stream, the decompression was completed; otherwise, repeat Step 1. The decoding algorithm is shown in Algorithm 1:

**Algorithm 1** B12 scheme decoding algorithm

```
Input: Binary data stream
Output: Decoded text stream
while start_position < Binary_data_length do
    data12bit = get12bitData (start_position);
    if data12bit is a syllable in the dictionary then //one syllable
        text = text + Decode_with_dictionary (data12bit);
        start_position = start_position + 12 bit;
        continue;
    end if
        if xChar_Flag < data12bit < xChar_Block_Begin_Flag then //xChars count < 10
        xChars_number = code12bit - xChar_Flag;
        text = text + Get_Unicode_characters (xChars_number);
        start_position = start_position + xChars_number *16 bit;
        continue;
    end if
        if data12bit == xChar_Block_Begin_Flag then //xChars count > 9
        xChars_block_length = Find_next_xChar_block_end_flag (start_position);
        text= text + Get_Unicode_characters (xChars_block_length/16bit);
        start_position = start_position + xChars_block_length;
    end if
end while
```

#### 5.5.2. Decompression of B16 Scheme

1. Read the original text in Unicode characters.
2. Intercept 1 character (2 bytes); if the encoding range is in the range 0xE003-0xF8FF, then use the dictionary encoding table to decode.
3. Intercept 1 character (2 bytes); if the encoding is equal to fXC (0xE001), then read the next Unicode character directly. Repeat Step 2. If no character data stream remains, the decompression is complete.

## 6. Experiment and Analysis

### *6.1. Experimental Corpus and Comparison Methods*

1. In this study, we randomly selected 15 texts according to size as the experimental corpus. We based the corpus on Unicode encoding.
2. We used the short text in Table 4 as the experimental corpus. The corpus shared 2908 short texts with a total size of 907,108 bytes. We stored each short text twice in UTF8 and UTF16 encoding formats. The corpus-related information is shown in Table 7.

**Table 7.** Short-text corpus information.

| Size (Bytes) | File Count | | | |
|---|---|---|---|---|
| <50 | 138 | dicChar | unique dicChar | 129 |
| 50–100 | 315 | | average frequency | 3480 |
| 101–200 | 633 | dicSyllb | unique dicSyllb | 1608 |
| 201–300 | 588 | | average frequency | 94 |
| 301–400 | 421 | xchar | unique xChar | 631 |
| 401–500 | 312 | | average frequency | 3 |
| 501–1000 | 466 | xSyllb | unique xSyllb | 658 |
| >1000 | 35 | | average frequency | 2 |
| | | words | unique word | 13,631 |
| Total | 2908 | | average frequency | 4 |

The comparison method used GZip, LZW, and BZip2 compression. The LZW algorithm used Mark Nelson [30], GZip used the Microsoft. NET GZipStream class [31], and BZip2 used ICSharpCode.SharpZipLib.BZip2 (©Mike Kruger Version: 0.86.0.518).

## *6.2. Experimental Results*

### 6.2.1. Compression of Text of Different Sizes

Table 8 gives the comparison experiments of the five compression methods for text compression. Among them, $S_O$ indicated the size of the compressed corpus, and the data unit was a byte. We compared the compression method used in this study with other compression ratios, as given in Table 8 and shown in Figure 7.

**Table 8.** Comparison of Uyghur text compression methods.

| **So** | **CR** | | | | | **BPC** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **(Bytes)** | **B12** | **B16** | **GZip** | **LZW** | **BZip2** | **B12** | **B16** | **GZip** | **LZW** | **BZip2** |
| 32 | 0.34 | 0.56 | 1.50 | 1.31 | 2.09 | 5.50 | 9.00 | 24.00 | 21.00 | 33.50 |
| 58 | 0.30 | 0.50 | 1.20 | 1.18 | 1.50 | 4.80 | 8.00 | 19.20 | 18.93 | 24.00 |
| 102 | 0.32 | 0.54 | 0.93 | 0.98 | 1.01 | 5.08 | 8.62 | 14.92 | 15.69 | 16.15 |
| 160 | 0.28 | 0.51 | 0.71 | 0.84 | 0.74 | 4.50 | 8.20 | 11.40 | 13.50 | 11.80 |
| 176 | 0.28 | 0.49 | 0.77 | 0.87 | 0.75 | 4.49 | 7.91 | 12.31 | 13.93 | 12.04 |
| 284 | 0.26 | 0.48 | 0.59 | 0.79 | 0.57 | 4.20 | 7.61 | 9.40 | 12.59 | 9.17 |
| 876 | 0.30 | 0.52 | 0.44 | 0.67 | 0.40 | 4.79 | 8.27 | 7.00 | 10.75 | 6.47 |
| 1586 | 0.29 | 0.51 | 0.40 | 0.58 | 0.36 | 4.63 | 8.18 | 6.46 | 9.24 | 5.78 |
| 2538 | 0.29 | 0.50 | 0.34 | 0.51 | 0.30 | 4.64 | 7.92 | 5.47 | 8.16 | 4.77 |
| 5022 | **0.30** | **0.51** | **0.26** | **0.43** | **0.23** | **4.81** | **8.19** | **4.22** | **6.88** | **3.66** |
| 7630 | 0.29 | 0.51 | 0.27 | 0.40 | 0.23 | 4.71 | 8.17 | 4.36 | 6.45 | 3.65 |
| 13,694 | 0.30 | 0.50 | 0.20 | 0.36 | 0.18 | 4.73 | 7.98 | 3.21 | 5.72 | 2.83 |
| 39,372 | 0.29 | 0.50 | 0.16 | 0.35 | 0.14 | 4.59 | 7.97 | 2.64 | 5.66 | 2.24 |
| 61,980 | 0.30 | 0.51 | 0.21 | 0.36 | 0.16 | 4.73 | 8.14 | 3.30 | 5.79 | 2.61 |
| 93,584 | 0.30 | 0.51 | 0.19 | 0.35 | 0.14 | 4.79 | 8.12 | 3.01 | 5.61 | 2.25 |

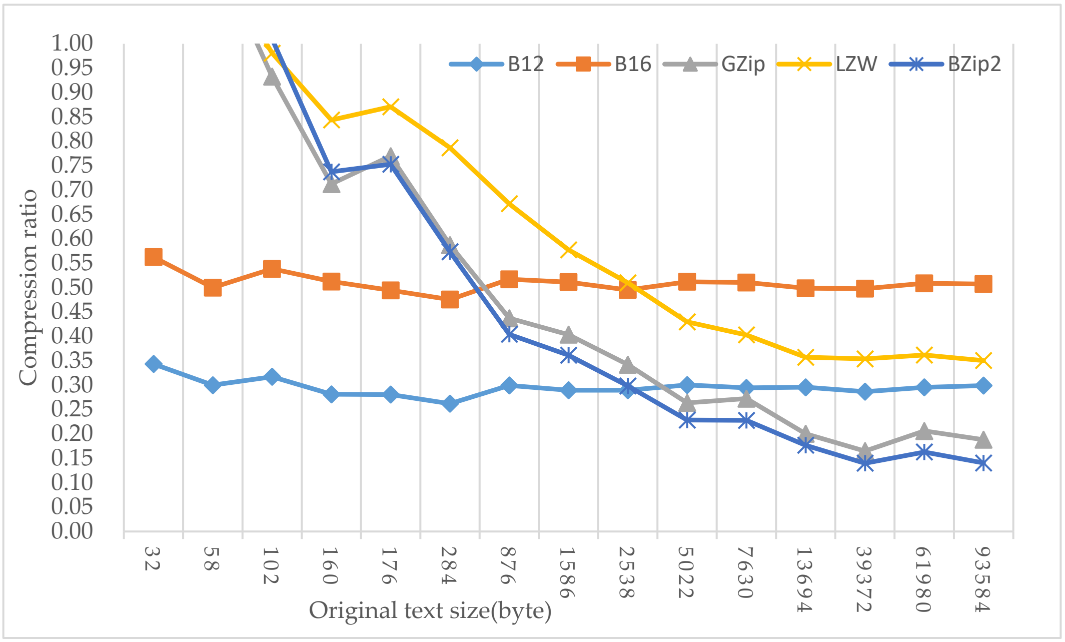


**Figure 7.** Compression ratio of the five methods.

### 6.2.2. Short-Text Compression

We selected 2908 pieces of short text and stored each piece of text once with UTF8 and UTF16 encoding. We compressed each piece of text using the five methods. So represented the sum of 2908 files of the same encoding type, and Sc was the sum of the 2908 compressed files. The compression time and decompression time were the sums of the compression and decompression time of the 2908 files, respectively. The specific results are given in Table 9 and shown in Figure 8.

**Table 9.** Short-text compression ratio of five methods

| Method | Text Code | Time (ms) | | So (byte) | Sc (byte) | CR | BPC |
|---|---|---|---|---|---|---|---|
| | | Compression | Decompression | | | | |
| B12 | UTF8 | 3595 | 14,511 | 845,127 | 280,261 | 0.33 | 4.94 |
| | UTF16 | 3306 | 15,016 | 907,108 | 280,261 | 0.31 | 4.94 |
| B16 | UTF8 | 2846 | 2909 | 845,127 | 475,984 | 0.56 | 8.40 |
| | UTF16 | 2775 | 3096 | 907,108 | 475,984 | 0.52 | 8.40 |
| BZip2 | UTF8 | 5646 | 2975 | 845,127 | 524,630 | 0.62 | 9.25 |
| | UTF16 | 6364 | 2619 | 907,108 | 529,350 | 0.58 | 9.34 |
| GZip | UTF | 2788 | 1889 | 845,127 | 553,315 | 0.65 | 9.76 |
| | UTF16 | 4326 | 2185 | 907,108 | 534,337 | 0.59 | 9.42 |
| LZW | UTF8 | 18,104 | 1580 | 845,127 | 705,984 | 0.84 | 12.45 |
| | UTF16 | 16,737 | 1505 | 907,108 | 693,398 | 0.76 | 12.23 |

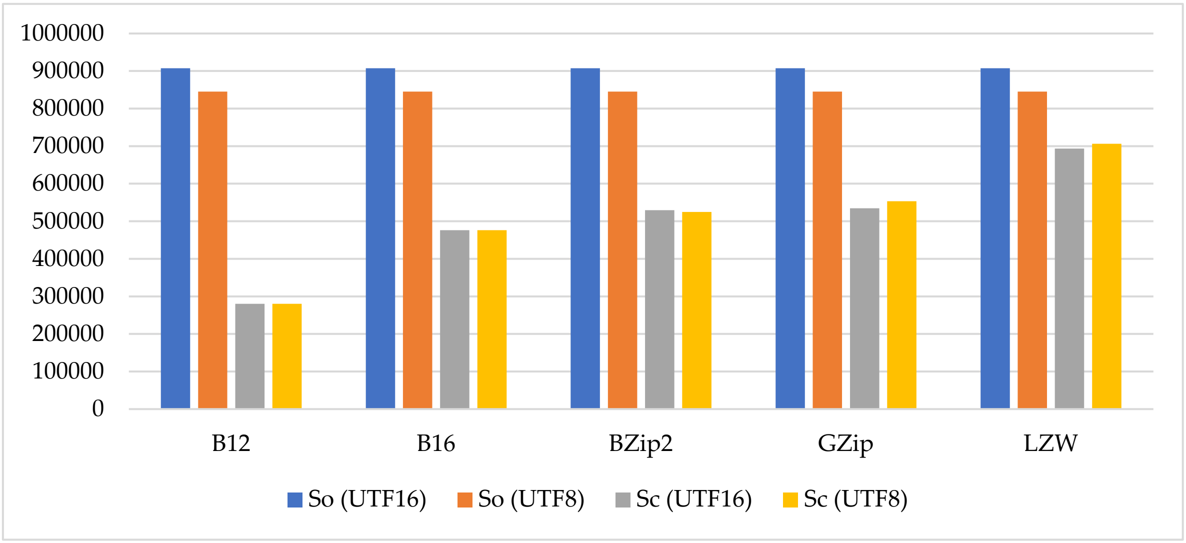


**Figure 8.** Comparison of short-text compression effects of the five methods.

The distribution of Zipf's law for dicSyllb and xSyllb in the corpus is shown in Figure 9. The *x*-axis shows the ranking of syllables in descending order according to the syllable frequency, *r = 1* indicates the syllable with the highest frequency, and the *y*-axis shows the frequency of the syllable corresponding to r. To facilitate comparison and observation, the frequency of xSyllb was represented by *(−1) logf*.

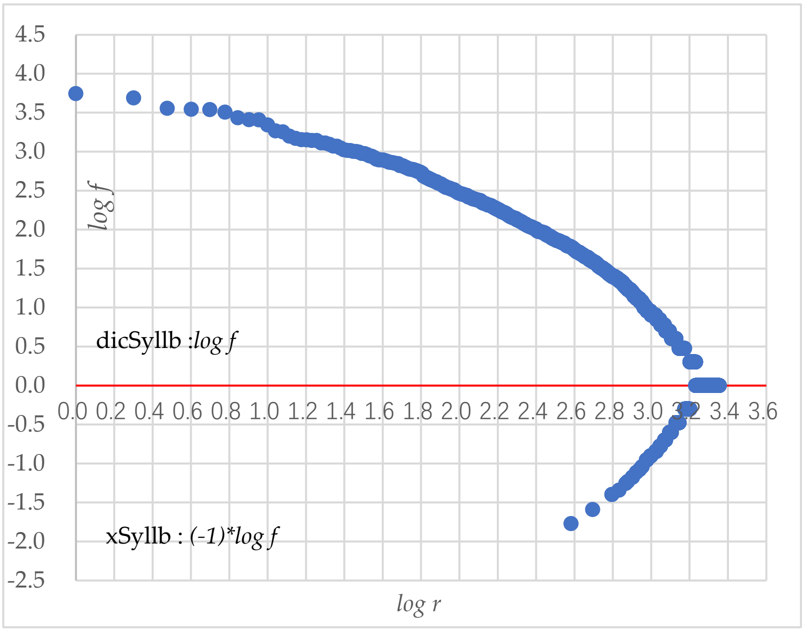


**Figure 9.** Zipf's Law distribution of syllables in short-text corpus.

## *6.3. Experimental Analysis*

The two coding schemes discussed in this paper offered certain advantages in short text. The B12 coding scheme performed best on text with a size of less than 4 KB. The effect was obvious when the text was less than 1 KB. The compression ratio was always stable at about 0.3 and 0.5. For compression of short texts smaller than 200 bytes, GZip, LZW, and BZip2 algorithms had compression ratios exceeding 1. As the text grew, the compression efficiency of GZip, BZip2, and LZW gradually improved and exceeded B12 and B16.

The LZW, GZip, and BZip2 compressed files appended the dictionary data needed for decompression. When the text was large, the dictionary data had little effect on the compression ratio. This is why the compression efficiency of short texts smaller than 200 bytes was greater than 1. The B12 and B16 schemes were based on syllable encoding. According to Equation (1), the average length of a Uyghur syllable was 2.4 characters; therefore, theoretically, no matter how large the text size was, the compression ratio was stable at $CR_{B12} = 12/(2.4 \times 16) = 0.31$ and $CR_{B16} = 16/(2.4 \times 16) = 0.42$ or so. The occurrence of xChar, XSyllb in the text, and units of one-character characters in the encoding table affected CR. The purpose of selecting high-frequency syllables and using the space flag bit in the B12 coding scheme was to reduce this effect and to further increase the compression ratio.

In the general compression method, to obtain the best compression efficiency, the additional dictionary data was related to the current text. The same character string may have been represented by different encodings in different compressed data because of different frequencies. Further processing of compressed content required decompression. In this syllable-based compression scheme, the encoding value and length were fixed. The basic unit of the compressed data stream changed from characters to syllables. The research and application of speech synthesis and speech recognition based on syllables has offered certain advantages. When processing content based on short data obtained from big data (e.g., WeChat and SMS), a theoretically faster and more convenient retrieval speed could be obtained with a compression ratio of 0.3 to 0.5. The advantages of the B16 solution in this regard were obvious. Third-party retrieval tools would not need to decompress to perform syllable-based retrieval of content.

## *6.4. $CR_{best}$ and $CR_{worst}$ of the B12 Scheme*

The formula for calculating the compressed file size from the B12 scheme was $Sc = P_B + P_U$. In the formula, $P_B$ was the compressed content, and $P_U$ was the original code reserved part with the flag.

*So*, when $P_U = 0$, $S_C$ was directly related to $P_B$. Because $P_B = SyllCnt \times Lenc$, and $Lenc = 12$, so $P_B = SyllCnt \times 12$. Currently, the formula for calculating the compression ratio was as follows (assuming the original text is in Unicode format)

$$CR = \frac{Sc}{So} = \frac{SyllCnt \times 12bit}{CharCount \times 16bit}. \tag{10}$$

As shown in Tables 1 and 2, the longest syllable structure was CCVCC, and the syllable length had five characters. Then the content structure of a text file was continuous CCVCC + SP (5-character syllable + 1 space), SP was exactly marked with a space flag bit, and the number of Sc characters was exactly divided by 6. Thus, the best $CR_{best}$ calculation result is

$$CR_{best} = \frac{Sc}{So} = \frac{SyllCnt \times 12bit}{CharCount \times 16bit} = \frac{SyllCnt \times 12bit}{SyllCnt \times (5+1) \times 16bit} = \frac{12bit}{6 \times 16bit} = \frac{1}{8} = 0.125. \tag{11}$$

Similarly, according to the design of the B12 data structure, the CR effect was the worst when the text appeared with one dicChar and one xChar consecutively. Currently, the size of this encoding structure was as follows: *dicChar + fXC + xChar* $\rightarrow$ *12 bit + 12 bit + 16 bit = 40 bit*, and an average character was 20 bits. If the number of characters in the file was exactly a multiple of 2, $CR_{worst}$ was calculated as

$$CR_{worst} = \frac{Sc}{So} = \frac{P_U}{CharCount \times 16bit} = \frac{CharCount \times 20bit}{CharCount \times 16bit} = \frac{5}{4} = 1.25 \tag{12}$$

$CR_{best}$ and $CR_{worst}$ of the B12 scheme currently had no relationship with the original file size So.

The B12 solution was a special compression method used in specific environments. It used the syllable characteristics of natural language for encoding and decoding. The current compression ratio of Uyghur natural text was about 0.3, and the compression ratio was not directly related to the file size. The client must have the same syllable encoding dictionary when compressing and decompressing (the current dictionary size is <20 KB). It was suitable for the compression of natural language short texts. For example, when using the B12 solution of the product (e.g., pharmaceutical) instructions using a two-dimensional code, it theoretically could accommodate two times more information. WeChat MSG and SMS transmission saved two-thirds of available communication resources. If it was not a text based on natural language (such as a random meaningless string, i.e., "aaaaaabbbbbb"), the compression ratio was significantly reduced ($CR_{worst} = 1.25$), which was a disadvantage of this scheme and one of the subjects of our next research.

## 7. Conclusions

Compression is a fundamental area of research in the field of computer communications, with important theoretical significance and application value. Few studies have examined Uyghur text compression methods and current compression techniques have some shortcomings. We proposed a Uyghur text compression method based on syllable features, using B12 and B16 encoding schemes for experiments. Compared with other algorithms, such as LZW, GZip, and BZip2, the B12 scheme had higher compression efficiency when the text size was less than 4 KB. It could be applied effectively to compressed transmission of short texts and QR codes. The advantage of B16 scheme was that it could quickly retrieve syllable-based information in a compressed state. The compression method proposed in this paper could be applied to other agglutinative languages in the same language family with high similarity, such as Uzbek, Kazakh, and Kirgiz [32]. Future work will include studies of the general syllable-based text compression methods for agglutinative languages and their applications. Future research will examine full-text retrieval technology of short text based on syllables without decompression.

**Author Contributions:** Conceptualization, W.A.; Formal analysis, W.A. and H.W.; Funding acquisition, A.W.; Investigation, J.W.; Methodology, W.A. and M.M.; Project administration, A.W.; Resources, J.W.; Software, W.A., H.W., and K.A.; Writing—original draft, W.A.; Writing—review and editing, M.M., T.Y., and A.W. All authors have read and approved the final manuscript.

**Funding:** This work was supported by the Opening Foundation of the Key Laboratory of Xinjiang Uyghur Autonomous Region of China (grant number 2018D04019), the National Natural Science Foundation of China (grant numbers 61762084, 61662077, 61462083), the Scientific Research Program of the State Language Commission of China (grant number ZDI135-54), and the National Key Research and Development Project of China (grant number 2017YFB1002103).

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

## References

1. David, S.; Le-Nan, W. *Data Compression*, 2nd ed.; Publishing House of Electronics Industry: Beijing, China, 2003; pp. 9–163.
2. Shannon, C.E. A mathematical theory of communication. *Bell Labs Tech. J.* **1948**, *27*, 379–423. [CrossRef]
3. Huffman, D.A. A method for the construction of minimum-redundancy codes. *Resonance* **2006**, *11*, 91–99. [CrossRef]
4. Ziv, J.; Abraham, L. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory* **1977**, *23*, 337–343. [CrossRef]
5. Ziv, J.; Abraham, L. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory* **1978**, *24*, 530–536. [CrossRef]
6. Akman, K.I. A new text compression technique based on language structure. *J. Inf. Sci.* **1995**, *21*, 87–94. [CrossRef]
7. Moffat, A. Word-based text compression. *Softw. Pract. Exp.* **1989**, *19*, 185–198. [CrossRef]
8. Moffat, A.; Isal, R.Y. Word-based text compression using the Burrows-Wheeler transform. *Inf. Process. Manag.* **2005**, *41*, 1175–1192. [CrossRef]
9. Crochemore, M.; Langiu, A.; Mignosi, F. Note on the greedy parsing optimality for dictionary-based text compression. *Theor. Comput. Sci.* **2014**, *525*, 55–59. [CrossRef]
10. Gardner-Stephen, P.; Bettison, A.; Challans, R.; Hampton, J.; Lakeman, J.; Wallis, C. Improving Compression of Short Messages. *Int. J. Commun. Netw. Syst. Sci.* **2013**, *6*, 497–504. [CrossRef]
11. Platos, J.; Snasel, V.; Elqawasmeh, E. Compression of small text files. *Adv. Eng. Inform.* **2008**, *22*, 410–417. [CrossRef]
12. Rein, S.; Guhmann, C.; Fitzek, F.H. Compression of Short Text on Embedded Systems. *J. Comput.* **2006**, *1*, 1–10. [CrossRef]
13. Kalajdzic, K.; Ali, S.H.; Patel, A. Rapid lossless compression of short text messages. *Comput. Stand. Interfaces* **2015**, *37*, 53–59. [CrossRef]
14. Adubi, S.A.; Misra, S. Syllable-Based Text Compression: A Language Case Study. *Arab. J. Sci. Eng.* **2016**, *41*, 3089–3097. [CrossRef]
15. Nguyen, V.H.; Nguyen, H.T.; Duong, H.N.; Snasel, V. *A Syllable-Based Method for Vietnamese Text Compression. International Conference on Ubiquitous Information Management and Communication*; ACM: Danang, Vietnam, 2016.
16. Akman, I.; Bayindir, H.; Ozleme, S. A lossless text compression technique using syllable based morphology. *Int. Arab J. Inf. Technol.* **2011**, *8*, 66–74.
17. Oswald, C.; Ajith, K.J.; Avinash, J. A Graph-Based Frequent Sequence Mining Approach to Text Compression. In *Mining Intelligence and Knowledge Exploration, Proceedings of the International Conference on Mining Intelligence and Knowledge Exploration, Hyderabad, India, 13–15 December 2017*; Springer: Hyderabad, India, 2017.
18. Bharathi, K.; Kumar, H.; Fairouz, A. A Plain-Text Incremental Compression (PIC) Technique with Fast Lookup Ability. In Proceedings of the 2018 IEEE 36th International Conference on Computer Design (ICCD), Orlando, FL, USA, 7–10 October 2018; IEEE: Orlando, FL, USA, 2018.
19. Vijayalakshmi, B. Lossless Text Compression Technique Based on Static Dictionary for Unicode Tamil Document. *Int. J. Pure Appl. Math.* **2018**, *118*, 85–91.

20. Sinaga, A.; Adiwijaya Nugroho, H. Development of word-based text compression algorithm for Indonesian language document. In Proceedings of the 2015 3rd International Conference on Information and Communication Technology, ICoICT, Nusa Dua, Indonesia, 27–29 May 2015; IEEE: Bali, Indonesia, 2015.
21. Farhad Mokter, M.; Akter, S.; Palash Uddin, M.; Ibn Afjal, M.; Al Mamun, M.; Abu Marjan, M. An Efficient Technique for Representation and Compression of Bengali Text. In Proceedings of the 2018 International Conference on Bangla Speech and Language Processing, ICBSLP, Sylhet, Bangladesh, 21–22 September 2018; IEEE: Sylhet, Bangladesh, 2018.
22. Hilal, T.A.; Hilal, H.A. Arabic text lossless compression by characters encoding. *Procedia Comput. Sci.* **2019**, *155*, 618–623. [CrossRef]
23. Chen, Q.; Chen, X.; Han, D. Compression algorithm LZW on Chinese text. *Comput. Eng. Appl.* **2014**, *50*, 112–116.
24. Satoh, N.; Morihara, T.; Okada, Y.; Yoshida, S. Study of Japanese text compression. In Proceedings of the Data Compression Conference, DCC '97, Snowbird, UT, USA, 25–27 March 1997; IEEE: Snowbird, UT, USA, 1997.
25. Zhong-Qi, X.; Winira, M. Uyghur text compression technique research. *J. Xinjiang Univ.* **2012**, *29*, 9–12.
26. Maimaiti, M.; Wumaier, A.; Abiderexiti, K.; Yibulayin, T. Bidirectional long short-term memory network with a conditional random field layer for Uyghur part-of-speech tagging. *Information* **2017**, *8*, 157. [CrossRef]
27. Osman, T.; Yang, Y.; Tursun, E.; Cheng, L. Collaborative Analysis of Uyghur Morphology Based on Character Level. *Acta Sci. Nat. Univ. Pekin.* **2019**, *55*, 47–54.
28. Nurmemet, Y.; Wushour, S.; Reyiman, T. Syllable based language model for large vocabulary continuous speech recognition of Uyghur. *J. Tsinghua Univ.* **2013**, *53*, 741–744.
29. Wayit, A.; Jamila, W.; Turgun, I. Modern Uyghur automatic syllable segmentation method and its implementation. *China Sci.* **2015**, *10*, 957–961.
30. LZW Data Compression. Available online: http://dogma.net/markn/articles/lzw/lzw.htm/ (accessed on 10 January 2020).
31. GZipStream Class (System.IO.Compression). Available online: https://docs.microsoft.com/zh-cn/dotnet/api/system.io.compression.gzipstream?view=netframework-4.8/ (accessed on 10 January 2020).
32. Tuergen, I.; Kahaerjiang, A.; Aishan, W.; Maihemuti, M. A Survey of Central Asian Language Processing. *Chin. Inf. Process* **2018**, *32*, 1–13, 21.