# Position Index Preserving Compression for Text Data

**Md. Nasim Akhtar**
Department of Information Technology
Moscow State Academy of Fine Chemical Technology, Moscow, Russia

**Md. Mamunur Rashid**
Department of Information Technology
Moscow State Academy of Fine Chemical Technology, Moscow, Russia

**Md. Shafiqul Islam**
Department of Information Technology
Moscow State Academy of Fine Chemical Technology, Moscow, Russia

**Mohammod Abul Kashem**
Department of Computer Science & Engineering
Dhaka University of Engineering and Technology, Gazipur, Bangladesh
**and**
**Cyrll Y. Kolybanov**
Department of Information Technology
Moscow State Academy of Fine Chemical Technology, Moscow, Russia

## ABSTRACT

Data compression offers an attractive approach to reducing communication cost by using available bandwidth effectively. It also secures data during transmission for its encoded form. In this paper an index based position oriented lossless text compression called PIPC (Position Index Preserving Compression) is developed. In PIPC the position of the input word is denoted by ASCII code. The basic philosophy of the secure compression is to preprocess the text and transform it into some intermediate file which can be compressed with better efficiency and which exploits the natural redundancy of the language in making the transformation. The proposed algorithm compresses the data 35% to 50% of its original size depending on occurrence of repeated word.

**Keywords:** Compression, Index, IDBE, LIPT, PIPC.

## 1. INTRODUCTION

In the last decade, we have seen an unprecedented explosion of textual information through the use of Internet, Digital Library and information retrieval system. It is estimated that by the year 2002 the National service provider, backbone will have an estimated traffic around 27645 Gbps and the growth will continue 100% every year. The text data occupies 45% of the total internet traffic but no lossless compression standard for text has yet been proposed [1].

Lossless compression researchers have developed highly sophisticated approaches, such as Huffman encoding, Arithmetic encoding, the Lempel-Ziv (LZ) family, Dynamic Markov Compression. Prediction by partial matching (PPM) and Burrows Wheeler Transform (BWT) based algorithms. However, none of these methods has been able to reach theoretically best-case compression ratio consistently, which implies that better algorithm may be possible.

There were efforts earlier to apply different compression techniques to compressed text on server but only a few of the compression techniques like LZ algorithm have been successful. A number of sophisticated algorithms have been proposed for lossless text compression, of which (Intelligent Dictionary Based Encoding) IDBE and BWT out perform the classical algorithms.

The IDBE algorithm follows steps:

1) Make an intelligent dictionary.
2) Encode the input text data.

In the proposed algorithm the intelligent dictionary of IDBE has been omitted and we get a significant file size reduction using the proposed technique. In this paper, a Position Index Preserving Compression (PIPC) for compressing English document is proposed. It uses sophisticated compression algorithm, proposed for lossless text compression namely BZIP2 based on Burrow Wheelers Transform (BWT) [2], GZIP based on Lempel-Ziv (LZ) family[3] and (Length Index Preserving Transformation) LIPT for preserving English text[4]. Preprocessing of text archives higher compression by encoding a word in input file by word in to a compress file. The basic idea of the actual module is to compress the text into some intermediate form, which can be compressed with better efficiency. The compress text is provided to a back end data compression module, which compresses the actual text. However, execution time performance and run time memory expenditure of this compression system have remained high compared with the backend compression algorithm, such as LIPT and IDBE [5].

Basic concept of proposed algorithm is discussed in section 2 and the proposed algorithm is implemented in section 3. A input and output file size equation has been developed for the proposed algorithm in section 4.The result applying proposed algorithm is analyzed comparing with the existing algorithm in section 5.In section 6 time complexity of proposed algorithm has been calculated and section 7 concluded the paper with suggestion of future development [6].

## 2. BASIC CONCEPT OF THE PROPOSED ALGORITHM

In this paper a new text compression technique, called Position Index preserving Compression (PIPC) is proposed which makes the text better compressible than most of the above methods. The philosophy of compression is to compress the text into some source text file which compressed when the natural redundancy of the language is occurring. PIPC encoding scheme makes use of recurrence of some words in the English text. Where position is different indexed to create context in the compress text that their corresponding ASCII code can exploit. PIPC use ASCII code to denote index position of the words hence two or more lengths of words are repeated again and again in the compressed text resulting better context [7][8].
In addition to say that PIPC also uses the ASCII code denoted the offset of the word in the English dictionary having the same words. This serves to include additional context in the compressed text. To make efficient the algorithm, the repetition of words, that is word frequency in English text is main factor from our point of view. Huffman compression methods also need sharing of the some static dictionary at both the sender and receiver end. Also the IDBE method requires the dictionary both sender and receiver end [9].

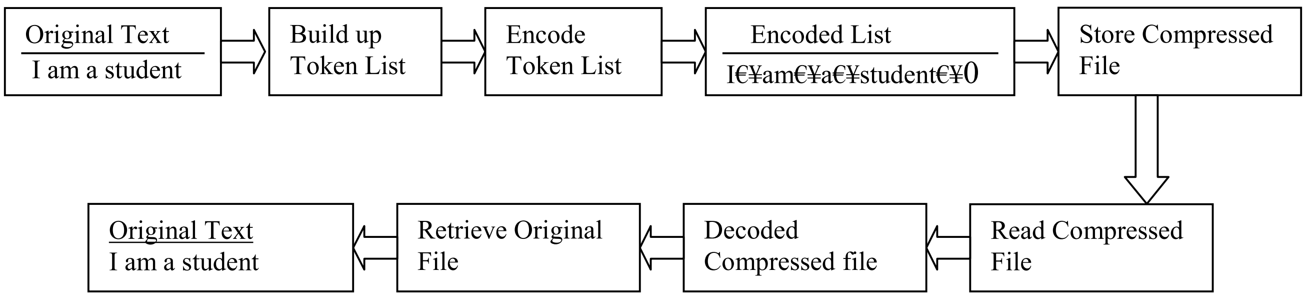


Fig: 1 Text compression paradigm incorporating a lossless strategy.

The algorithm has been developed following two strategies, one is Encoding strategy and other is Decoding strategy.

## 3. ENCODING ALGORITHM

```
MSL_Compression () {
    While (! end of file) {
            Read one word from input file;
            Enqueue ();
            } //Enter word into list
     Write compress-file to total number of
word;
     While (! end of list) {
            If (! null of current token) {
               Write current token;
               Write multiply bit;
               While (! end of list) {
          If (current token ==next current
token)

       Write their position and multiply bit;
               }//End of while
             }//End if
         }//End of while
```

## 4. DECODING ALGORITHM:

```
Algorithm Decoding ()
        {
        Read total no of word;
Create list and all token set null initialize;
While (! end of file)
        {
         Retrieve token;
         Read input position;
         If (Test (W)) Enqueue (W);
      Else {
           For expected word find next null
           token in list and put into token;
                                     }
      Read multiply bit;
```

```
        Read word position;
        Make actual position;
        Enqueue true position
        }
Dequeue ();      }

Int Test (int position)
{
If (position = position1 & token == NULL)

        Return 1;
    Else
        Return 0;
}
```

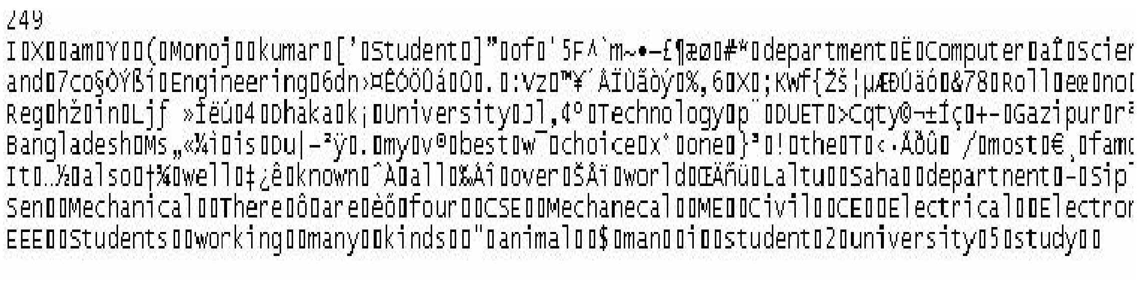
## 5. IMPLEMENTATION OF THE PROPOSED ALGORITHM

An English text document file takes as input file for implementation of proposed algorithm. As an example, the original text is:

*I am Monoj Kumar Student of department of Computer Science and Engineering.*
*Roll no Reg no in Dhaka University of Engineering and Technology DUET.*
*Gazipur Bangladesh DUET is my best choice DUET is one of the most famous university in Bangladesh It also well known all over the world.*
*I am Laltu Kumar Saha Student of department of Computer Science and Engineering Roll no Reg no in Dhaka University of Engineering and Technology DUET Gazipur Bangladesh DUET is my best choice DUET is one of the most famous University in Bangladesh It also well known all over the world.*
*I am Siplob Kumar Sen Student of department of Mechanical Engineering Roll no Reg no in Dhaka University of Engineering and Technology DUET Gazipur Bangladesh DUET is my best choice DUET is one of the most famous University in Bangladesh It also well known all over the world.*

*There are four Engineering departments in DUET Computer Science and Engineering CSE Mechanical Engineering ME Civil. Engineering CE and Electrical and Electronics Engineering EEE. Students of DUET are working well in Bangladesh and all over the world. There are many kinds of creation in the world Man is one kind of creational .I am student of DUET.DUET is the Engineering university .Student study in university.*

## 6. ENCODED FORM OF SOURCE TEXT



```
249
I□X□□am□Y□□(□Monoj□□kumar□['□Student□]"□□of□'5FΛ`m~•–£¶æØ□#*□department□Ë□Computer□aÎ□Scier
and□7co§ÒYßí□Engineering□6dn›¤ÊÓÖ□á□O□.□:vz□™¥´ÅÏÙãòý□%,6□X□;Kwf{Žš¦µÆÐÚäó□&78□Roll□eœ□no□
Reg□hž□in□Ljƒ »Îëú□4□Dhaka□k¡□University□J],¢°□Technology□p¨□DUET□>Cqty©¬±Íç□+-□Gazipur□r²
Bangladesh□Ms„«¼ì□is□Du|-²ÿ□.□my□v®□best□w¯□choice□x˚□one□}³□!□the□T□‹·ÅÕÛ□ /□most□€¸□famo
It□…½□also□†¾□well□‡¿ê□known□ˆÀ□all□‰Âî□over□ŠÂï□world□ŒÄñü□Laltu□□Saha□□department□-□Sip
Sen□□Mechanical□□There□ô□are□èő□four□□CSE□□Mechanecal□□ME□□Civil□□CE□□Electrical□□Electror
EEE□□Students□□working□□many□□kinds□□"□animal□□$□man□□i□□student□2□university□5□study□□
```

## 7. ORIGINAL TEXT

Applying decoding algorithm the original text has been retrieved.

## 8. CALCULATION OF FILE SIZE

**Input File Size Equation**

This equation calculates the exact input file size**.**

$$\sum_{i=0}^{N-1} f(i) \times word\_length(i) + f(sp\_cha)$$

Where,

N = Total no words

i = Instantaneous word

f (i) = frequency of Instantaneous words

word_length = word_length of Instantaneous words f (sp_cha) = total no of word- frequency of dot- frequency of Comma - frequency of (?)

**Output File Size Equation**

This equation calculates the compressed file size.

$$2 \times m + \sum_{i=1}^{N} word\_length(i) + f_{224}(f(i))$$

$$+ \sum_{j=1}^{N-1} (f_{j \times 225-(j+1) \times 225}(f(1)) + f_{j \times 225-(j+1) \times 225}(f(i)))$$

Where,

m = No of individual words

$f_{224}$=No of frequency of 0 to 224 interval f j*225 –(j+1)*225 f(1)=At least one time of j to j+1 interval.

fj*225 –(j+1)*225 f(i))=No of frequency of j*225 to (j+1)*225 interval.

## 9. RESULT ANALYSIS

In this section the performance of PIPC using position index is evaluated. The measurements have compression results in terms of frequency of words with word's length which measures the minimum and maximum range in between which the proposed algorithm is effective. The result is given in the Table-1 and Figure-1.

**Words Frequency Distribution with Length**

Table-1: Word frequency Distribution

| File Name | Original file size (Byte) | word length with frequency | | | | | | | | | | | Com-pressed file size (Byte) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | |
| Solar | 4410 | 81 | 158 | 147 | 137 | 69 | 89 | 56 | 43 | 29 | 13 | 16 | 3220 |
| Student | 900 | 24 | 52 | 37 | 36 | 25 | 6 | 14 | 4 | 2 | 21 | 13 | 452 |
| Bible | 700 | 20 | 11 | 62 | 16 | 15 | 12 | 5 | 3 | 3 | 0 | 0 | 409 |
| Bible1 | 700 | 20 | 7 | 59 | 22 | 18 | 11 | 5 | 3 | 3 | 0 | 0 | 484 |
| Repeat | 5000 | 696 | 348 | 348 | 348 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1840 |

In Table-1 a minimum and maximum range for PIPC algorithm is calculated. The PIPC works properly on an average of minimum 4% repetition of word. Maximum 65% compression will be obtained when huge number of repetition will occur.

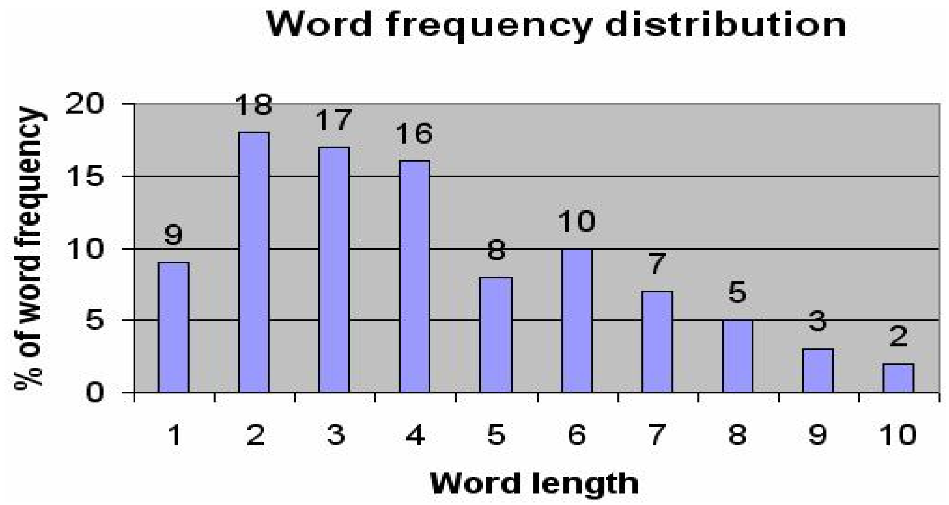


Fig:2 The graph of Solar.txt, its word frequency distribution

**Table2:** Percentage of Compression Using PIPC

| File Name | Original File size (Byte) | PIPC compressed file size(Byte) | Percentage of compression |
|---|---|---|---|
| Solar.txt | 4410 | 3220 | 27% |
| Student.txt | 900 | 452 | 50% |
| Bibble.txt | 700 | 409 | 42% |
| Paper1.txt | 3000 | 2290 | 24% |
| Paper2.txt | 2000 | 1480 | 26% |

**Comparison Graph between Original and PIPC file size:**

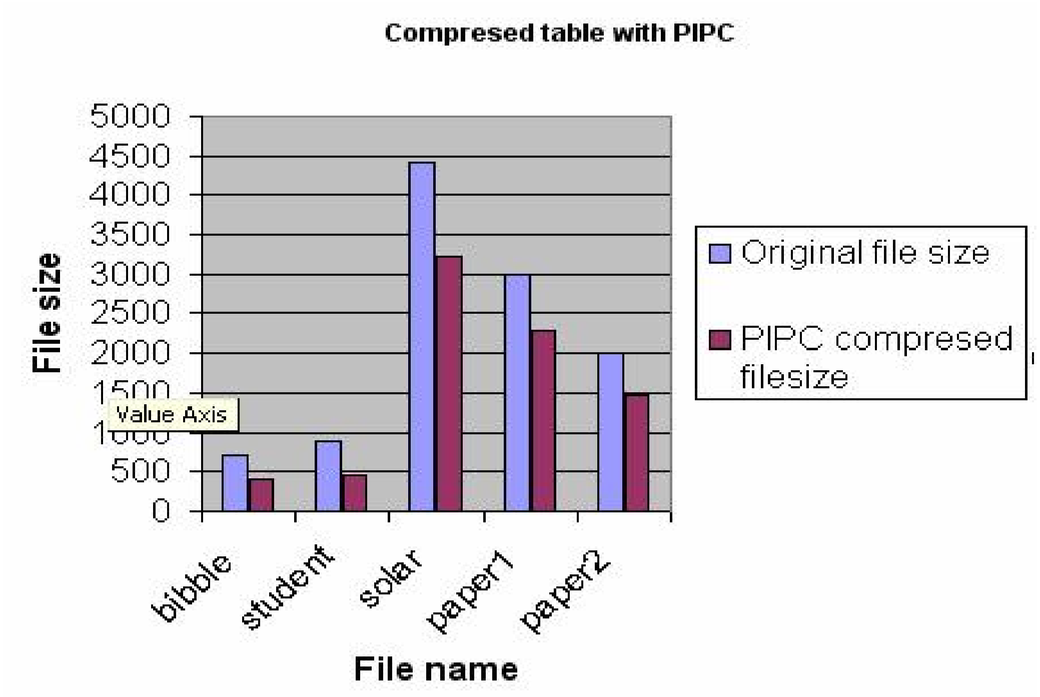


**Fig 3:** Comparison Graph between Original and PIPC file size

**Table 3:** Comparison Table between IDBE and PIPC in respect to File Size

| File Name | Original File size(Byte) | IDBE compressed file size(Byte) | PIPC compressed file size(Byte) | Percentage of improvement |
|---|---|---|---|---|
| Solar.txt | 4410 | 3290 | 3220 | 2.1 % |
| Student.txt | 900 | 540 | 452 | 16.25% |
| Paper1.txt | 3000 | 2420 | 2290 | 5.3% |
| Paper2.txt | 2000 | 1600 | 1480 | 7.5% |

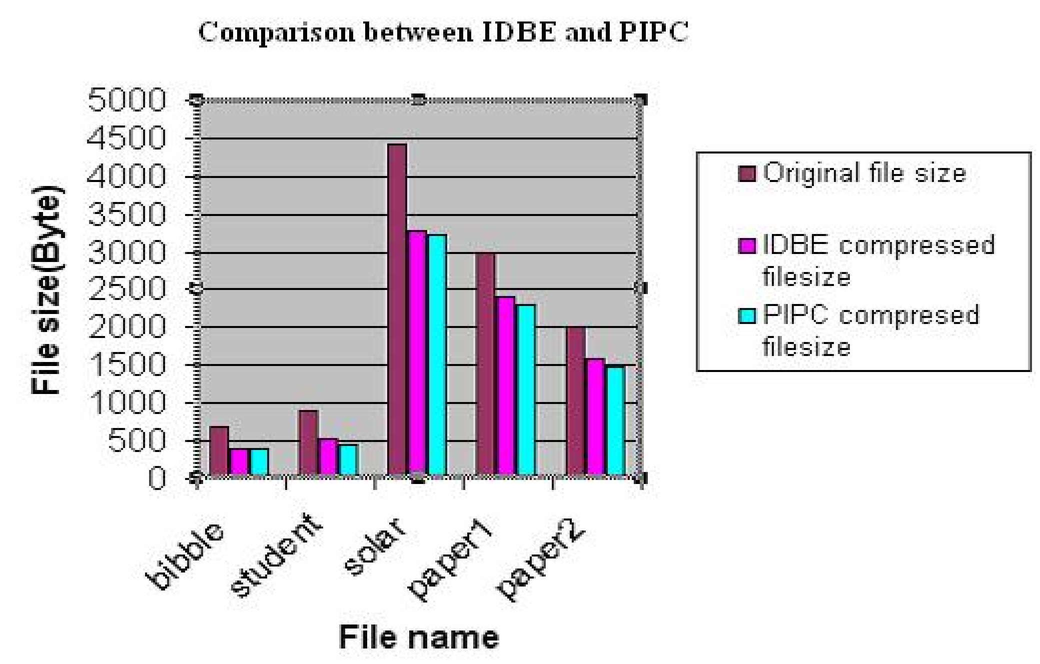


**Fig 4:** Comparison graph between IDBE and PIPC in file size

## 10. TIME COMPLEXITY OF ENCODING ALGORITHM

In the encryption module the time complexity depends on the two strategy, one is separating the word as token, which required time for retrieved total number of lines from input file and number of character per line, written as, **No of line *(No of character per line)$^2$**. Other is for writing output file depends on the total number of node which are equal to the total number of word consists the input file, written as, **No of Node*(No of Node - 1)**.
So, the Time complexity of the encoding algorithm is: **$O(L*C^2+N*(N-1))$**

### Time Complexity of Decoding Algorithm

In the decryption module the time complexity is reverse process of the two encoding strategy, one is creating **the total number of node equal to the total number of words** and other is for writing output file depends on the total number of line from intermediate file and number of character per line, written as,

**No of line * (No of character per line)$^2$**.

So, the Time complexity of the decoding algorithm is: **$O(N+L*C^2)$**
Where,

L = No. of line,
C = No. of character per line,
N = Total No. of Node.

## 11. CONCLUSION

A novel lossless text compression called Position Index Preserving Compression (PIPC) has been implemented. The proposed algorithm compresses data without any intelligent dictionary that makes it different from existing algorithms. Moreover the proposed techniques reduce the file size significantly compare to the exiting algorithm. For further development we proposed advanced technique of PIPC algorithm that is to be implemented in some additional strategy. This strategy could follow a double indexing technique, where PIPC algorithm will be used first, followed by encoding the intermediate file by binary indexing.

## 12. REFERENCES

[1] Suzanne Bunton, "On-Line Stochastic Processes in Data Compression", Doctoral Dissertation, University of Washington, Dept. of Computer Science and Engineering, 1996, pp 3-4.
[2] M. Burrows and D. J. Wheeler, "A Block-sorting Lossless Data Compression Algorithm", SRC Research Report 124, Digital Systems Research Center, 2000, pp 12-14.
[3] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. Information Theory*, IT-23, 1999, pp.237-243.
[4] F. Awan, A. Mukherjee, "LIPT: A lossless Text Transform to Improve Compression", Proceeding of International Conference on Information and Theory: Coding and computing, IEEE Computer Society Less Vegas Nevada,, 2001, pp 47-51.
[5] H. Kruse and A. Mukherjee "Data Compression Using Text Encryption", *Proc Data Compression Conference*, 1997, IEEE Computer Society Press, , pp 447-448..
[6] D.A.Huffman, "A Method for the Construction of Minimum Redundancy Codes", *Proc. IRE,* 40(9),1952, pp.1098-1101.
[7] H. Kruse and A. Mukherjee, "Preprocessing Text to Improve Compression Ratios", *Proc. Data Compression Conference*, IEEE Computer Society Press, 1998, pp 556-557.
[8] Fred Halsall, "Multimedia Communications." Second Edition, 2002, pp.116-120.
[9] Horowitz, Sahni, Rajasekaran, "Fundamentals of Computer algorithms." Edition 2003-04, pp. 18.

Received: March 2010. Accepted: December 2010.