Text Compression using Lexicographic Permutation of Binary Strings

Sourya Basu, Shivam Chaturvedi and Rajesh Hegde

Dept of Electrical Engineering

Indian Institute of Technology Kanpur

SPCOM 2016

Bangalore

June 14, 2016

Outline

- Introduction
- Text Encoding Schemes based on Character Frequency models
- Unique Lexicographical Rank (ULR) and Maximum Rank(MR) of a Bit String
- Text Compression Algorithm using Lexicographic Permutation of Binary Strings
- Compression Ratio and Redundancy Rate Analysis
- Text Compression Experiments on Project Gutenberg and Calgary Corpus
- Conclusion

Introduction

- The paper proposes a novel method using combinatorics on binary strings
- Frequency based encoding models has been used to achieve lossless compression in text data
- In the binary string if the total number of $'0's$ and $'1's$ in the n bit binary string are known, then the total number of states the string can be in is $\frac{n!}{k!\,(n-k)!}$, where k is the number of $'0's$
- Thus information stored in the string can be represented using only $\lceil log_2 \binom{n}{k} \rceil$ bits

Text Encoding Schemes based on Character Frequency Models

- Higher the value of $|\frac{n}{2} - k|$, lower is the value of $\lceil log_2 \binom{n}{k} \rceil$, saving a huge number of bits for a good range of k
- Subsequently, an encoding scheme that forms a binary string with lower entropy is used to increase $|\frac{n}{2} - k|$
- Encoding scheme exploits the language specific characteristic
- Characters are encoded using models like uni-gram, bi-gram, and tri-gram to reduce entropy

Text Encoding Schemes based on Character Frequency Models

Encoded Character	Character frequency model	Preceding character	$\frac{k}{n}$ for the encoded character
u	Uni-gram model	ϕ	5/7
	Bi-gram model	q	6/7
	Tri-gram model	eq	1
y	Uni-gram model	ϕ	5/7
	Bi-gram model	z	6/7
	Tri-gram model	az	6/7
x	Uni-gram model	ϕ	4/7
	Bi-gram model	o	5/7
	Tri-gram model	bo	5/7
e	Uni-gram model	ϕ	6/7
	Bi-gram model	z	1
	Tri-gram model	pr	1

Variation in the $\frac{k}{n}$ ratio for encoding characters u, y, x and e. ϕ indicates lack of precedence in unigram

Unique Lexicographical Rank and Maximum Rank of a Bit String

- Unique Lexicographical Rank (ULR): Lexicographical rank of given binary string amongst all possible strings with same numbers of zeroes and ones

$$1 + \sum_{i=1}^{n-1} \delta(a_i) \cdot \delta(k - \sum_{l=1}^{i-1} \bar{\delta}(a_l)) \cdot \binom{n-i}{n-k-\sum_{l=1}^{i-1} \delta(a_l)} \tag{1}$$

where

$$\delta(x) = \begin{cases} 1 & \forall x > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \bar{\delta}(x) = 1 - \delta(x)$$

- Maximum Rank (MR): Total number of strings formed from given zeroes and ones

$$MR = \binom{n}{k} \tag{2}$$

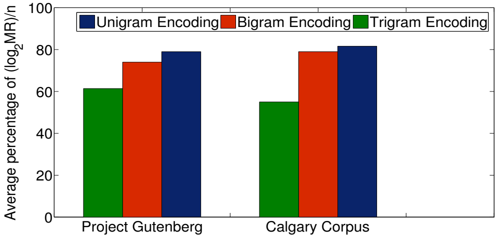
Unique Lexicographical Rank and Maximum Rank of a Bit String for various n and k

n	k	Decimal representation of a binary string	$\lceil log_2 MR \rceil$	$\lceil log_2 ULR \rceil$
5	3	3	4	0
5	3	5	4	1
10	4	754	8	8
20	13	40005	17	13
30	22	230023	25	16
40	23	4623578	37	27

- For fixed n and k, $\lceil log_2 MR \rceil$ is same, while ULR varies from 1 to MR
- Since their is one to one correspondence between ULR and given binary string, ULR can be sent for reliable decoding

Text Compression using Maximum Rank Reduction Method

- It is intended to send the values of n and k parameters of a binary string followed by its ULR which can be uniquely decoded at the receiving end
- Total number of bits sent $= \lceil log_2\binom{n}{k} \rceil + \lceil log_2(n) \rceil + \lceil log_2(n) \rceil$
- To increase compression ratio, the MR needs to be minimized, which can be done by maximizing $|n/2 - k|$ using the text encoding schemes mentioned earlier

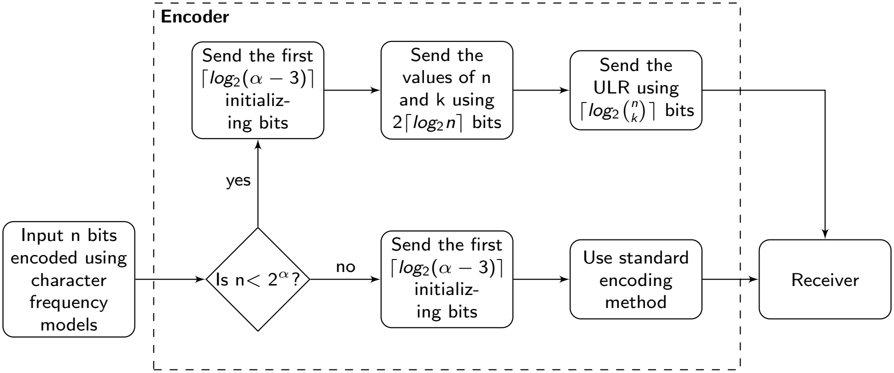


Text Compression using ULR method: Initialization Bits

- If upper limit to n is known to be of the order of 2^{α},then,for some message bit string of length n, we define its initial bits to be the $\lceil log_2(\alpha - 3)\rceil$ bit binary representation of $f(n)$
- $f(n)$ is defined as following

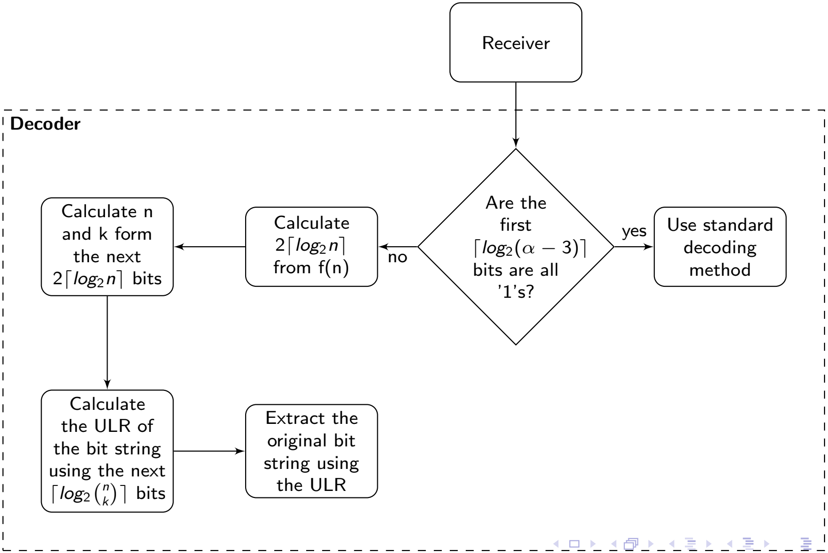
$$f(n) = \begin{cases} 0 & \text{if } n \leq 2^5 \\ \text{m-4} & \text{if } 2^m < \text{n} \leq 2^{m+1}, \text{such that } 5 \leq m < \alpha \\ \alpha - 4 & \text{if } n > 2^{\alpha} \end{cases}$$

Encoding using Lexicographic Permutation of Binary Strings



Encoder
Input n bits encoded using character frequency models
Is n< 2^{α}?
yes
no
Send the first $\lceil log_2(\alpha-3)\rceil$ initializing bits
Send the values of n and k using $2\lceil log_2 n\rceil$ bits
Send the ULR using $\lceil log_2\binom{n}{k}\rceil$ bits
Send the first $\lceil log_2(\alpha-3)\rceil$ initializing bits
Use standard encoding method
Receiver

Decoding using Lexicographic Permutation of Binary Strings



Receiver
Decoder
Are the first $\lceil log_2(\alpha - 3) \rceil$ bits are all '1's?
yes
Use standard decoding method
no
Calculate $2\lceil log_2 n \rceil$ from f(n)
Calculate n and k form the next $2\lceil log_2 n \rceil$ bits
Calculate the ULR of the bit string using the next $\lceil log_2 \binom{n}{k} \rceil$ bits
Extract the original bit string using the ULR

Encoding Algorithm

- **Input :** Original text message from the user
- **Encode:** Encode the given message using any one of the encoding models
- **Initial bits:**Send the first $\lceil log_2(\alpha - 3)\rceil$ initial bits using $f(n)$ as described above
- Following this we send 2x$\lceil log_2 n\rceil$ bits representing n and k each using $\lceil log_2 n\rceil$ bits
- **Send the ULR:**Send the next $\lceil log_2 \binom{n}{k}\rceil$ bits representing the ULR of the bit string
- **Output :** Compressed bit string representing the input text

Decoding Algorithm

- **Input :**Compressed bit string representing the input text
- **Receive:**Receive the first $\lceil log_2(\alpha - 3) \rceil$ bits
- If the first $\lceil log_2(\alpha - 3) \rceil$ bits are all '1's, use standard compression algorithms such as Huffman algorithm or Arithmetic coding to decode the original message
- Else if the first $\lceil log_2(\alpha - 3) \rceil$ bits are all '0's, then receive 5 more bits to get to know the exact values of n, followed by 5 more bits representing the value of k
- Else we get to know the range of n using the definition of f(n), i.e we get to know the value of $\lceil log_2 n \rceil$.Receive 2x$\lceil log_2 n \rceil$ more bits to get the exact values of n and k

Decoding Algorithm

- Since,the decoder now knows the values of n and k,it will calculate $\lceil log_2 \binom{n}{k} \rceil$, and will receive exactly this number of bits to get the ULR of the message with n bits having k $'0'$s
- **Original Message:**Using the ULR, find the actual encoded message of n bits having k $'0'$s Then, using the encoding methods find the actual string of bits, which is the actual message that is sent
- **Output :**Original text message

Encoding "BAT" using ULR Algorithm

- Assume that using one of the three encoding models discussed earlier, it is encoded as "000001100000010000010" i.e. 21 bits with $4'1's$ and $17'0's$
- The MR of this bit string would be $\binom{21}{4}$ i.e 5985.
- Hence we can send the ULR of given binary string in $\lceil log_2 5985 \rceil$ bits i.e 13 bits only
- We also need to send the number of $'0's$ and $'1's$ in the beginning using a predefined set of rules, which we ignore in this example as $log_2 n$ is asymptotically smaller than n

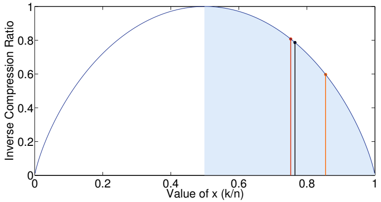
Compression Ratio Analysis

- Inverse compression ratio (*ICR*) is the ratio of compressed file size to original file size
- Lower the ICR, better the text compression obtained
- In [1], it has been shown that for large n, *ICR* is found to be dependent only on x

$$ICR \simeq -(x \log_2 x + (1-x) \log_2 (1-x)) \tag{3}$$

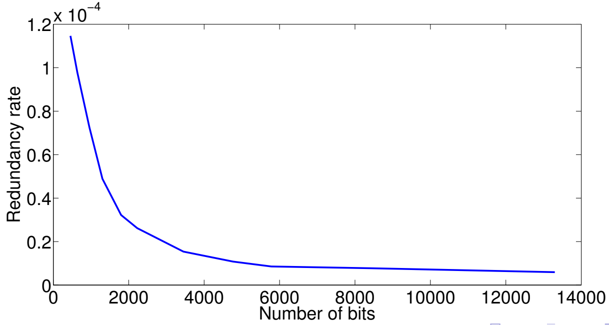
where, $x = \frac{k}{n}$

- From the figure below it can be seen that as the value of $|x - 0.5|$increase, the value of ICR decreases



Redundancy Rate Analysis

- In this algorithm, the redundant information sent to the receiver is equal to $log_2 MR - log_2 ULR$
- $$Redundancy \;\; rate = \frac{log_2 MR - log_2 ULR}{log_2 MR} \tag{4}$$
- The figure below shows that redundancy rate decreases rapidly with increase in total number of bits being sent



Databases : Project Gutenberg and Calgary Corpus

- Project Gutenberg and Calgary corpus are used in the experiments on text compression
- Project Gutenberg : The items in its collection are the full texts of public domain books. The releases are available in format like plain text, HTML, PDF, EPUB, MOBI and Plucker
- Calgary corpus : The Calgary corpus is a collection of text and binary data files, commonly used for comparing data compression algorithms. The corpus consists of 14 files totaling 3,141,622 bytes
- Three files are taken from the Standard Calgary Corpus namely, bib.txt, book2.txt, pap2.txt [2] and several books from Project Gutenberg[3] were used for performing the test
- It was also used for computing the frequency tables required for comparison with the static Huffman algorithm

Experimental Results on Text Compression

- Comparison performance evaluated as ICR, Redundancy rate for uni-gram, bi-gram and tri-gram encoding methods
- Tri-gram encoding scheme gives best ICR

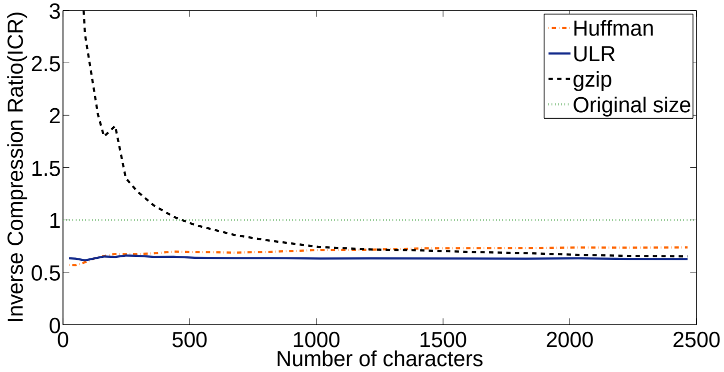
Encoding method	ICR	$\lceil log_2 MR \rceil$	Redundancy rate
unigram	0.76	435 KB	3.248×10^{-8}
	0.88	96 KB	1.488×10^{-7}
	0.81	483 KB	2.957×10^{-8}
	0.76	61 KB	2.341×10^{-7}
bigram	0.74	424 KB	3.369×10^{-8}
	0.86	94 KB	1.519×10^{-7}
	0.78	466 KB	3.065×10^{-8}
	0.74	60 KB	2.381×10^{-7}
trigram	0.53	304 KB	4.699×10^{-8}
	0.70	76 KB	1.879×10^{-7}
	0.58	346 KB	4.128×10^{-8}
	0.56	45 KB	3.174×10^{-7}

Comparison with Huffman and GZIP algorithms

- A comparison is made between the Huffman algorithm, gzip algorithm and ULR algorithm for unstructured text data, obtained from Twitter database as in [4]
- In Huffman algorithm, the character frequencies in the message must also be known to the decoder
- Such a table was made in this work using a large database of text taken from several books from Calgary and Project Gutenberg corpus
- This table was made available to both the encoder and the decoder, and was used for the purpose of comparison with other compression algorithms

Comparison with Huffman and GZIP algorithms

- For unstructured text data, the ULR algorithm gives consistent performance for varying text lengths over a wide range
- Performance is comparable to the static Huffman algorithm



Conclusion and Future Work

- Proposed compression method exploits redundancy in used language to form low entropy encoded binary string
- This string is then compressed by lexicographically arranging the permutations of the bits and sending the rank of required binary string
- We are assuming that both the coder and decoder knows some approximate distribution to the actual message being sent but it can be improved so that it learns the distribution on the fly

References

- Davisson, Lee D. “Universal noiseless coding.” Information Theory, IEEE Transactions on 19.6 (1973): 783-795.
- Witten, Ian, Timothy Bell, and John Cleary. “The Calgary corpus, 1987.”
- Hart, Michael S. “Project Gutenberg: Access to electronic texts.” Database 13.6 (1990): 6-9.
- Ankit Jalan, Ketan Rajawat, and Rajesh M. Hegde. “New Encoding Schemes for Efficient Multilingual Text Messaging”.Proceedings of The Twentieth Annual National Conference on Communications (NCC-2014),IIT Kanpur, March 2014