Data Compression using Variable-to-Fixed Length Codes

Hokkaido University
Graduate school of Information Science and Technology, Division of Computer Science

Takuya Kida
kida@ist.hokudai.ac.jp
with Satoshi Yoshida, Hirohito Sasakawa, and Kei Sekine

Compact Data Structure for Bigdata @Shonan meeting

2013/9/29
Contents

- Background and aim
  - Related-works
- Variable-to-Fixed length codes (VF codes)
- Grammar-based compression
  - Re-Pair algorithm
- Proposed method (Re-Pair VF codes)
- Experimental results
- Summary
- Biography
Background and aim

Each data entry isn’t significant. The data are usually too redundant to store into an RDB system.

Develop an accessible data compression whose compressed data are reusable for data searching and analyzing.

Cloud Computing

Big Data

Various and Massive Stream Data

Internet SNS
- Road Information
- Information from Citizens

UGC (User Generated Content) Data

Open Smart Federation Architecture

Remote Sensing Data
- Meteorological satellite data
- Aerial photographs

GPS Logs & Life Logs
- GPS log
- Life log
**Problem**: Existing methods are inconvenient for reusing the compressed data, because they must follow the decompression process.

**Originality**: We have data compression techniques that can search keywords on compressed data directly without decompressing and reach a relatively good compression performance.
Compression method that splits the input text into variable length substring and then converts them into fixed length codewords.

- **Strong point**: Easy to handle the compressed data => Enables fast information retrieval or data mining
- **Weak point**: Hard to get a good compression ratio

Grammar Compressions model data very well

VF codes provide high accessibility

Satisfy both of high comp. ratio and accessibility!
## Variations of codes

<table>
<thead>
<tr>
<th>Compressed Text</th>
<th>Input Text</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fixed Length</td>
<td>Variable Length</td>
</tr>
<tr>
<td>Fixed Length</td>
<td><strong>FF Code</strong> (Fixed length to Fixed length code)</td>
<td><strong>VF Code</strong> (Variable length to Fixed length code)</td>
</tr>
<tr>
<td>Variable Length</td>
<td><strong>FV Code</strong> (Fixed length to Variable length code)</td>
<td><strong>VV Code</strong> (Variable length to Variable length code)</td>
</tr>
</tbody>
</table>

*Tunstall Code*
Grammar-based compression

Input text

ABDABCABCCABDABC

Grammar

E → ABC
F → ABDE
FECF

Construct a (context free) grammar that generates only the input text

Encode the grammar

Compressed text

0101110011101 000011100...
Replace most frequent bigram into a new symbol

**Re-pair algorithm**

Replace most frequent bigram into a new symbol

```
AAABACAAABCCAAAB
DABACDABCCDAB
EBACEBCCEB
FACFCCCF
FAGCG
```

**dictionary**

- D → AA
- E → DA
- F → EB
- G → CF

Encode with variable length code
(a variation of Huffman coding)
Proposed Method
(Re-Pair VF codes [Kida&Yoshida DCC2013])
In the Re-Pair algorithm, symbol replacement does not always imply improvement of compression ratio.

We want to encode with the best intermediate dictionary, which gives highest compression ratio.

We find the best dictionary and then expand the rules that do not included in it when translating to a grammar.

Finally, we encode the obtained grammar by Fixed length codes.
The number of sort of symbols: \( s + |\Sigma| \)

We need \( \lceil \log(s + |\Sigma|) \rceil \) bits for each symbol.

**Compressed sequence:**

\[ \text{EIKJBCFICEK} \]

**Dict.**

\begin{align*}
D & \rightarrow AA \\
E & \rightarrow DA \\
F & \rightarrow EB \\
G & \rightarrow CF \\
H & \rightarrow GE \\
I & \rightarrow FF \\
J & \rightarrow IG \\
K & \rightarrow HI
\end{align*}

**Sum:**

\[ (2s + n) \lceil \log(s + |\Sigma|) \rceil \text{ bits} \]

Calculate this value for each substitution to store the size of dictionary that gives the highest compression ratio.
Rewind the dictionary

Compressed sequence: EIKJBCEFICEK

AADAEBCF EFFGEFFFFFGBCEFFFFCEGEFF
We compared compression ratios, compression times, decompression times, and pattern matching speeds among these methods.

* **Method**
  * VF code via Re-pair (Re-Pair-VF)
  * Original Re-pair (Re-Pair)
  * Tunstall code
  * STVF code (proposed by Kida)
  * gzip (zgrep which decompress the data then search keywords)

* **Data**
  * dazai.utf.txt (Japanese text (UTF-8 encoded), 7MB)
  * dblp2003.xml (XML data, 90MB)
  * gbhtg119 (DNA data, 87MB)
  * reuters21578 (English text, 19MB)
Compression ratio
[Yoshida&Kida2013]

Re-Pair-VF overcomes gzip!

Re-Pair-VF
Re-Pair
Tunstall
STVF
gzip
Compression speed
[Yoshida&Kida2013]

Comparison of compression speed for different files and algorithms:

- Re-Pair-VF
- Re-Pair
- Tunstall
- STVF
- gzip

Files:
- dazai.utf.txt
- dblp2003.xml
- gbhtg119
- reuters21578

The graph shows the compression speed (MB/sec) for each file using various compression algorithms.
Decompression speed
[Yoshida&Kida2013]

![Decompression speed graph]

- dazai.utf.txt
- dblp2003.xml
- gbhtg119
- reuters21578

- Re-Pair-VF
- Re-Pair
- Tunstall
- STVF
- gzip
Search speed [Yoshida & Kida 2013]
(for reuters21578 [English news paper])

Throughput (MB/sec)

Pattern length

Throughput = \frac{\text{Original text length}}{\text{Pattern matching time}}
Our proposed method achieved a good compression ratio comparable to bzip2 (about 27%) for large texts (codeword length 19bit, block size 128MB, ratio of shared dictionary 3/8, sampling text size 128MB)
The proposed method takes much time for compressing, but it can decompress at high speed!
We focused attention to VF codes, which are useful for reusing compressed data, and developed an efficient VF coding based on Re-pair algorithm. The coding enables us to do fast keyword search, in addition to make partial decompression and re-compression easy.

Future works:
* Improvement of compression speed
* Realizing pinpoint access to compressed data with an original text position.
* Development an online coding algorithm
Thank you for your kind attention!
Takuya Kida received the B.S. degree in Physics, the M.S. and Dr. (Information Science) degree all from Kyushu University in 1997, 1999, and 2001, respectively. He was a Full-time Lecturer of Kyushu University Library from October 2001 to March 2004. He is currently an Associate Professor of Division of Computer Science, Graduate School of Information Science and Technology, Hokkaido University, since April 2004. His research interests include Text Algorithms and Data Structures, Information Retrieval, and Data Compression. He is a member of IEICE, IPSJ, and DBSJ.