1. Background

Re-Pair [1]
- Lossless compression algorithm for text data
- Good compression ratio on repetitive text (Approximately 0.1% in repetitive text)
- Huge memory consumption (20 times as large as text length)

Difficult to apply several gigabyte of text
- Easy solution: divide into small blocks and compress one by one. This solution works well but makes compression ratio worse.

Re-Merge [2]
- A Variant of Re-Pair for large texts.
  It merges blockwise dictionary recursively. It achieves very good compression ratio, but the compression speeds are sacrificed.

Re-Use [3]
- Simple extension of Re-Pair for large texts.
  To reduce the total dictionary size, it shares a part of blockwise dictionaries among all blocks.

In this work, we developed more sophisticated method for sharing the entries of dictionary.


2. Re-Pair [1]
Substitute the most frequent bigram into a new symbol until all the bigrams are unique. Then, encode the compressed text and dictionary with an entropy code.

Compress
ENFOBOEFOBEEFOB
ENABOEABEEAB
ENCONECECECEC
ENCODED

2. Re-Pair [1]

3. Re-Use [3]
It constructs dictionary which shared in all blocks (shared dictionary) by sampling from the input text.

Shared DIC

Block1
Block2
Block3
Block4

Replace
Text1
Replace
Text2
Replace
Text3
And so on

Problem: Shared dictionary is static, some useless rules are possible to make compression ratio worse.

4. Proposed Method
Adaptive dictionary Sharing (ADS)
ADS shares the rules between two consecutive blocks adaptively.
Details are as follows:
1. Extract rules which appear in the nth block \( B[n] \) frequently from the dictionary for \( B[n-1] \) and \( U[n-1] \) and construct the set of extracted rules of \( D[n] \) \( (U[n]) \).
2. Replace \( B[n] \) by the rules in \( U[n] \), then obtain half-compressed text of \( B[n] \) \( (B'[n]) \).
3. Apply Re-Pair to \( B'[n] \), and then obtain the compressed text of \( B'[n] \) \( (C'[n]) \) and the set of local rules of \( B'[n] \) \( (L'[n]) \).
4. Combine \( L[n] \) and \( L'[n] \) into \( D[n] \).

5. Experiments
Performance comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Comp. ratio</th>
<th>Memory</th>
<th>Comp. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>0.89%</td>
<td>17,224 KB</td>
<td>67.5 sec</td>
</tr>
<tr>
<td>Re-Use</td>
<td>0.97%</td>
<td>16,432 KB</td>
<td>68.7 sec</td>
</tr>
<tr>
<td>Re-Merge</td>
<td>0.66%</td>
<td>602,596 KB</td>
<td>188.6 sec</td>
</tr>
<tr>
<td>Re-Pair</td>
<td>0.11%</td>
<td>5,065,924 KB</td>
<td>130.7 sec</td>
</tr>
<tr>
<td>Gzip</td>
<td>34.99%</td>
<td>1,680 KB</td>
<td>24.3 sec</td>
</tr>
<tr>
<td>Bzip2</td>
<td>5.16%</td>
<td>7,756 KB</td>
<td>55.4 sec</td>
</tr>
</tbody>
</table>

It shows the performance on highly repetitive text, einstein (650 MB) from Pizza & Chili corpus [3]. The proposed method reduces memory consumption without sacrifice of compression ratio. The proposed method is three times as fast as Re-Merge in compression time, and memory consumption of ours is 1/37 of Re-Merge.

Compression data sizes for various threshold

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Data Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>600 MB</td>
</tr>
<tr>
<td>5</td>
<td>450 MB</td>
</tr>
<tr>
<td>10</td>
<td>300 MB</td>
</tr>
<tr>
<td>20</td>
<td>150 MB</td>
</tr>
<tr>
<td>30</td>
<td>100 MB</td>
</tr>
<tr>
<td>40</td>
<td>50 MB</td>
</tr>
</tbody>
</table>

It shows the variation of the compression data sizes for various threshold on natural language text, english from Pizza & Chili corpus [3]. We can see that by reducing threshold, the size of dictionary could be reduced without increasing the size of the compressed text data.


6. Conclusion
We proposed a simple algorithm for dictionary sharing algorithm to reduce the memory consumption of Re-Pair algorithm. Experimental results show that proposed algorithm reduces memory consumption without sacrifice of compression ratio.

Our future work is to develop a method that determines the input parameters automatically.