Re-Pair Re-Visited

Francisco Claude, Rodrigo González, Gonzalo Navarro, Luís Russo

U. Chile, INESC Lisboa

Nov 1, 2007
Outline

1. Introduction

2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version

3. Dictionary Compression

4. Local Decompression

5. Theoretical Analysis

6. Experimental Results

7. Conclusions and Future Work
Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work
Outline

1. Introduction

2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version

3. Dictionary Compression

4. Local Decompression

5. Theoretical Analysis

6. Experimental Results

7. Conclusions and Future Work
Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work
Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work
Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work
Introduction

Methods
- Statistical
- Dictionary based

Differences
- Statistical methods usually achieve better compression ratio
- Dictionary based allow faster decompression

Statistical
- Huffman
- PPM

Dictionary based
- LZ77/78
- Re-Pair (Larsson & Moffat 1999)
Applications

Successful applications of Re-Pair

- Phrase-based compression and browsing (Moffat & Wan 2003)
- Compressing suffix arrays (González & N. 2006)
- Web Graph compression (Claude & N. 2007)

Other applications?

- We study the application of Re-Pair to normal text files
- We want to exploit its efficient local decompression
- Is there any guarantee on the compression ratio achieved?
Extending Re-Pair

**Re-Pair**
- New algorithms for compressing with Re-Pair
- New dictionary compression
- Experimental study of Re-Pair
- Theoretical analysis

**Results**
- New exact and approximate compression methods
- The dictionary is reduced to 50% of its size
- The dictionary is a compact data structure
- Re-Pair achieves good compression ratios
- It is very fast for decompressing
- Re-Pair is able to extract snippets from disk very efficiently
- It achieves $2nH_k + o(n \log \sigma)$ bits of space
**Re-Pair**

- Find the most frequent pair $ab$ in $T$
- Create the rule $s \rightarrow ab$
- Replace occurrences of $ab$ by $s$
- Iterate until every pair appears only once

**Original Re-Pair algorithm**

- Very fast, $O(n)$
- Consumes much memory (20$n$ bytes)
Re-Pair

\[ a \ a \ a \ a \ b \ c \ a \ a \ a \ b \ a \ a \ a \ b \ c \ a \ b \ d \ a \ b \ d \]

\[ a \ a \ 4 \quad \text{A} \rightarrow a \ b \]

\[ a \ b \ 5 \]

\[ b \ c \ 2 \]

\[ c \ a \ 2 \]

\[ b \ d \ 2 \]

\[ d \ a \ 1 \]

\[ a \ a \ A \ c \ a \ A \ a \ a \ A \ c \ A \ d \ A \ d \]
\[ a \ a \ \textcolor{green}{A} \ c \ a \ \textcolor{green}{A} \ a \ a \ \textcolor{red}{A} \ c \ a \ \textcolor{red}{A} \ d \ \textcolor{red}{A} \ d \]

\[ a \ a \ 2 \ \quad \{A \rightarrow a \ b\} \]
\[ a \ A \ 3 \ \quad \{B \rightarrow a \ A\} \]
\[ A \ c \ 2 \]
\[ c \ a \ 1 \]
\[ A \ d \ 2 \]
\[ c \ A \ 1 \]
\[ d \ A \ 1 \]

\[ a \ \textcolor{green}{B} \ c \ B \ a \ B \ c \ A \ d \ A \ d \]
Re-Pair

ea B c B a B c A d A d

\[
\begin{align*}
    a & \quad B \quad c \\
    B & \quad c \quad 2 \\
    c & \quad B \quad 1 \\
    B & \quad a \quad 1 \\
    c & \quad A \quad 1 \\
    A & \quad d \quad 2 \\
    d & \quad A \quad 1 \\
    \\
    a & \quad B \quad c \\
\end{align*}
\]

Claude, González, Navarro, Russo ()

Re-Pair Re-Visited

Nov 1, 2007 11 / 43
Re-Pair

\[ a \ B \ c \ B \ a \ B \ c \ C \ C \]

\[ a \ B \ 2 \quad A \rightarrow a \ b \]

\[ B \ c \ 2 \quad B \rightarrow a \ A \]

\[ c \ B \ 1 \quad C \rightarrow A \ d \]

\[ B \ a \ 1 \quad D \rightarrow B \ c \]

\[ c \ C \ 1 \]

\[ C \ C \ 1 \]

\[ a \ D \ B \ a \ D \ C \ C \]
Re-Pair

\[ a \ D \ B \ a \ D \ C \ C \]

\[ \begin{array}{cc}
    a & D \\
    D & B \\
    B & a \\
    D & C \\
    C & C \\
\end{array} \]

\[ \begin{array}{cc}
    A & \rightarrow & a \ b \\
    B & \rightarrow & a \ A \\
    C & \rightarrow & A \ d \\
    D & \rightarrow & B \ c \\
    E & \rightarrow & a \ D \\
\end{array} \]

\[ \text{dictionary} \]

\[ \text{compressed sequence} \]

Claude, González, Navarro, Russo ()
Outline

1 Introduction

2 Compressing with Re-Pair
   - Exact Version
   - Approximate Version

3 Dictionary Compression

4 Local Decompression

5 Theoretical Analysis

6 Experimental Results

7 Conclusions and Future Work
General Idea - Exact Version

- The frequency of pairs is tracked with a Treap
- Every replacement modifies the Treap
- Every pair points to its next occurrence
- It achieves $O(n \log n)$ time

Advantages/Disadvantages

- Less memory usage
- Slower compression
- It is an alternative to the exact method
Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work
General Idea - Approximate Version

- We place the text in a larger array
- The space left is used to count with a hash table
- We select $k$ pairs for replacing
- We replace them in one sweep over the text
- The space left in the array is used to track pairs

Advantages/Disadvantages

- Faster compression
- Tunable usage of memory
- Loses some compression
Outline

1. Introduction

2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version

3. Dictionary Compression

4. Local Decompression

5. Theoretical Analysis

6. Experimental Results

7. Conclusions and Future Work
Improving Dictionary Representation

- Represent the dictionary using compact data structures
- We reduce up to 50% the space for the dictionary
- Especially important for secondary memory
## Improving Dictionary Representation

### Rules

<table>
<thead>
<tr>
<th>A</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>d</td>
</tr>
<tr>
<td>D</td>
<td>B</td>
<td>c</td>
</tr>
<tr>
<td>E</td>
<td>a</td>
<td>D</td>
</tr>
</tbody>
</table>

### Rules

<table>
<thead>
<tr>
<th>A</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>7</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
</tr>
<tr>
<td>E</td>
<td>13</td>
</tr>
</tbody>
</table>

\[
B = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
a & b & a & 1 & 1 & d & 4 & c & a & 10 \\
\end{bmatrix}
\]
Improving Dictionary Representation

**Rules**

<table>
<thead>
<tr>
<th>A 1</th>
<th>B=</th>
<th>S=</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 1</td>
<td>1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0</td>
<td></td>
</tr>
<tr>
<td>B 4</td>
<td>a b</td>
<td>a 1</td>
</tr>
<tr>
<td>C 7</td>
<td>1 d</td>
<td>4 c</td>
</tr>
<tr>
<td>D 10</td>
<td>a 10</td>
<td></td>
</tr>
<tr>
<td>E 13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Rules**

<table>
<thead>
<tr>
<th>A 5</th>
<th>B=</th>
<th>S=</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 5</td>
<td>1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0</td>
<td></td>
</tr>
<tr>
<td>B 1</td>
<td>a 5</td>
<td>a b d</td>
</tr>
<tr>
<td>C 4</td>
<td>1 c</td>
<td></td>
</tr>
<tr>
<td>D 9</td>
<td>a 9</td>
<td></td>
</tr>
<tr>
<td>E 12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Claude, González, Navarro, Russo ()

Re-Pair Re-Visited

Nov 1, 2007 21 / 43
## Improving Dictionary Representation

### Rules

<table>
<thead>
<tr>
<th>Rules</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### B=

1. 100
2. 0
3. a
4. b
5. d
6. a
7. 1
8. c

### S=

A 5
B 1
C 4
D 11
E 9

### B=

1. 100
2. 0
3. a
4. b
5. d
6. a
7. a
8. 2
9. c

### S=

A 2
B 9
C 1
D 8
E 6
Limiting the Dictionary Size

- We stop forming pairs when the dictionary no longer fits in RAM
- Compression ratio depends on relative dictionary size, not absolute
- Preliminary results: good compression with dictionary as small as 5% of text
Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work
Adding structures

We add a bitmap $BR$ of length $n$ which marks every beginning of phrase.

$BR$ has to support rank and select operations

- $\text{rank}_{BR}(i)$: number of 1's in $BR[1..i]$
- $\text{select}_{BR}(i)$: position of the $i$-th 1 in $BR$
Decompressing

Using \textbf{BR}

Using \textbf{BR} we can decompress any substring expanding the optimal number of symbols.

Example

For extracting $T[2..5]$ we expand:

$$C[\text{rank}_{BR}(2)\ldots\text{rank}_{BR}(5)]$$
Squeezing \textbf{BR}

- If the text is compressible, \textbf{BR} is sparse
- It can be compressed to $H_0$
- We implemented two compressed structures
Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work
Theoretical Analysis

Theorem (KM99)

Let $y_1 \ldots y_t$ denote a parsing of the string $T[1, n]$ over an alphabet of size $\sigma$, such that each phrase $y_i$ appears at most $b$ times. For any $k \geq 0$ we have

$$t \log t \leq nH_k(T) + t \log(n/t) + t \log b + \Theta(t(1 + k \log \sigma))$$

Observation

After applying Re-Pair, every pair of consecutive symbols is unique. After replacing all the pairs that occur more than $b$ times, every pair of consecutive symbols appears at most $b$ times.
Theoretical Analysis

Theorem

Let \( T[1, n] \) be a text over an alphabet of size \( \sigma \) and having \( k \)-th order (classical or empirical) entropy \( H_k(T) \). Then, compression algorithm Re-Pair achieves a representation using at most \( 2nH_k(T) + o(n \log \sigma) \) bits for any \( k = o(\log \sigma n) \) (which implies \( \log \sigma = o(\log n) \) unless \( k = 0 \)).
Theoretical Analysis

The proof

- The size of the compressed sequence can be bounded using [KM99]
- We need to prove that every pair expands differently
- We prove that the dictionary does not grow too much
Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work
## Compression Ratio (as %)

<table>
<thead>
<tr>
<th>File</th>
<th>Repair</th>
<th>gzip -1</th>
<th>gzip -9</th>
<th>bzip2 -1</th>
<th>bzip2 -9</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNA</td>
<td>34.37</td>
<td>32.51</td>
<td>27.02</td>
<td>26.57</td>
<td>25.95</td>
</tr>
<tr>
<td>ENGLISH</td>
<td>29.06</td>
<td>44.99</td>
<td>37.64</td>
<td>32.83</td>
<td>28.07</td>
</tr>
<tr>
<td>PROTEINS</td>
<td>51.18</td>
<td>48.86</td>
<td>46.51</td>
<td>45.74</td>
<td>44.80</td>
</tr>
<tr>
<td>SOURCES</td>
<td>25.71</td>
<td>28.01</td>
<td>22.38</td>
<td>21.30</td>
<td>18.67</td>
</tr>
</tbody>
</table>
## Approximate Version (as %)

<table>
<thead>
<tr>
<th>File</th>
<th>Repair</th>
<th>appRepair</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNA</td>
<td>34.37</td>
<td>36.22</td>
</tr>
<tr>
<td>ENGLISH</td>
<td>29.06</td>
<td>34.30</td>
</tr>
<tr>
<td>PROTEINS</td>
<td>51.18</td>
<td>53.53</td>
</tr>
<tr>
<td>SOURCES</td>
<td>25.71</td>
<td>31.21</td>
</tr>
<tr>
<td>XML</td>
<td>13.93</td>
<td>17.10</td>
</tr>
</tbody>
</table>
Compressing BR (sizes in bytes)

| File              | |Br|   | impl1|   | impl2|   |
|-------------------|------------------|------------------|------------------|------------------|
| dblp.xml.200MB    | 27525152         | 12527918         | 11043041         |
| dna.200MB         | 27525152         | 18325588         | 17848550         |
| english.200MB     | 27525152         | 15473324         | 15226928         |
| pitches.50MB      | 6881312          | 4596241          | 4797069          |
| proteins.200MB    | 27525152         | 17605940         | 18759922         |
| sources.200MB     | 27525152         | 14809692         | 14209888         |
Decompression speed
Decompression speed
Disk Accesses for Decompression
Disk Accesses for Decompression
Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work
Conclusions

- Re-Pair achieves good compression ratio
- Very fast decompression, especially on disk
- Can be implemented on secondary storage
- Allows compression ratio/speed tradeoff
- Achieves high-order entropy
Future Work

- Analyze the approximate version
- \( k \) variable during compression
- Decompression speed with limited dictionary
Questions?