Pattern Matching on Simple Collage System using Finite Automata

Marek Hanuš    Bořivoj Melichar

The Prague Stringology Club
Department of Computer Science & Engineering
Czech Technical University in Prague

London Stringology Days & London Algorithmic Workshop
2008
Introduction

Pattern Matching in Compressed Text
Collage Systems
Finite Automata

Pattern Matching on Simple Collage Systems

Finite Translation Automaton
Pattern Matching
Time and Space Complexities
Experimental Results

Conclusion

Conclusion and Future Work
Pattern Matching in Compressed Text

- Decompress-then-search approach.

\[
\text{decompression} \quad \rightarrow \quad \text{original text} \quad \rightarrow \quad \text{occurrence(s) of pattern(s)}
\]

Decompress-then-search approach.

\[
\text{compressed text} \quad \rightarrow \quad \text{original text} \quad \rightarrow \quad \text{occurrence(s) of pattern(s)}
\]
Pattern Matching in Compressed Text

- Decompress-then-search approach.

\[
\text{decompression} \quad \text{pattern matching}
\]

\[
\text{compressed text} \quad \rightarrow \quad \text{original text} \quad \rightarrow \quad \text{occurrence(s) of pattern(s)}
\]

- Pattern matching in compressed text without decompressing it first.

\[
\text{compressed text} \quad \rightarrow \quad \text{compressed pattern matching} \quad \rightarrow \quad \text{occurrence(s) of pattern(s)}
\]
Previous Work

- Several methods for pattern matching on collage systems have been presented so far.
Collage Systems

- General framework for representing various dictionary-based compression methods.
Collage Systems

- General framework for representing various dictionary-based compression methods.

- **Collage system** is a pair $C = (D, S)$, where:
  - $D$ is a dictionary in the form of a sequence of assignments $X_1 = expr_1; X_2 = expr_2; \ldots; X_l = expr_l$, where each $X_p$ is a variable and $expr_p$ is any of the form:
    - $a$ for $a \in A \cup \{\varepsilon\}$, (primitive assignment)
    - $X_i X_j$ for $i, j < p$, (concatenation)
    - $[i] X_i$ for $i < p$ and an integer $j$, (prefix truncation)
    - $X_i[j]$ for $i < p$ and an integer $j$, (suffix truncation)
    - $(X_i)^j$ for $i < p$ and an integer $j$. (j times repetition)
  - $S$ is a sequence of variables $X_{i_1} X_{i_2} \ldots X_{i_n}$ defined in $D$. 
Collage Systems

- General framework for representing various dictionary-based compression methods.
- **Collage system** is a pair $C = (D, S)$, where:
  - $D$ is a dictionary in the form of a sequence of assignments $X_1 = expr_1; X_2 = expr_2; \ldots; X_l = expr_l$, where each $X_p$ is a variable and $expr_p$ is any of the form:
    - $a$ for $a \in A \cup \{\varepsilon\}$, \hspace{1cm} (primitive assignment)
    - $X_i X_j$ for $i, j < p$, \hspace{1cm} (concatenation)
    - $[j] X_i$ for $i < p$ and an integer $j$, \hspace{1cm} (prefix truncation)
    - $X_i[j]$ for $i < p$ and an integer $j$, \hspace{1cm} (suffix truncation)
    - $(X_i)^j$ for $i < p$ and an integer $j$. \hspace{1cm} ($j$ times repetition)
  - $S$ is a sequence of variables $X_{i_1} X_{i_2} \ldots X_{i_n}$ defined in $D$.
- **Simple collage system** uses only primitive assignment and concatenation with $|X_i| = 1$ or $|X_j| = 1$ (LZW, LZ78).
Example – LZW

Text $T = ababaccccbab$ over alphabet $A = \{a, b, c\}$. 
Example – LZW

- Text $T = ababaccccbab$ over alphabet $A = \{a, b, c\}$.
- Collage system $C = (D, S)$ where
  - $D$ is
    
    $$
    X_1 = a; \ X_2 = b; \ X_3 = c; \\
    X_4 = X_1 b; \ X_5 = X_2 a; \ X_6 = X_4 a; \ X_7 = X_1 c; \\
    X_8 = X_3 c; \ X_9 = X_8 c; \ X_{10} = X_3 b; \ X_{11} = X_5 b; 
    $$
  - $S$ is $X_1X_2X_4X_1X_3X_8X_3X_5X_2$. 
Example – LZW

- Text $T = ababaccccbab$ over alphabet $A = \{a, b, c\}$.
- Collage system $C = (D, S)$ where
  - $D$ is
    \[
    X_1 = a; \quad X_2 = b; \quad X_3 = c;
    \]
    \[
    X_4 = X_1b; \quad X_5 = X_2a; \quad X_6 = X_4a; \quad X_7 = X_1c;
    \]
    \[
    X_8 = X_3c; \quad X_9 = X_8c; \quad X_{10} = X_3b; \quad X_{11} = X_5b;
    \]
  - $S$ is $X_1X_2X_4X_1X_3X_8X_3X_5X_2$.
- Variables $X_6$, $X_7$, $X_9$, $X_{10}$ and $X_{11}$ are not used.
Example – LZW

Coverage of $T$ by $S$.

$S = X_1 X_2 X_4 X_1 X_3 X_8 X_3 X_5 X_2$

$T = a b a b a c c c c b a b$
Finite Automata

- **Finite automaton** is a general tool for representing various pattern matching problems:
  - exact matching of one pattern or a set of patterns,
  - approximate matching of one pattern or a set of patterns,
  - matching of a set of patterns defined by regular expressions.
  - ...
Example

- Set of patterns $P = \{ac, cb\}$ over alphabet $A = \{a, b, c\}$
Example

- Set of patterns $P = \{ac, cb\}$ over alphabet $A = \{a, b, c\}$
- Nondeterministic finite automaton $M_N = \{\{0, 1, 1', 2, 2'\}, \{a, b, c\}, \delta_N, 0, \{2, 2'\}\}$:

```
START 0

0 -> 1 [a]
0 -> 1' [c]
1 -> 2 [c]
1' -> 2' [b]
```
Example

- Deterministic finite automaton
  \[ M = \{\{0, 1, 1', 2, 2'\}, \{a, b, c\}, \delta, 0, \{2, 2'\}\}: \]

\[
\text{\begin{tikzpicture}
    \node[state, initial] (0) {0};
    \node[state] (1) at (1, 1) {1};
    \node[state] (1p) at (1, -1) {1'};
    \node[state] (2) at (2, 1) {2};
    \node[state] (2p) at (2, -1) {2'};
    \draw (0) edge[loop above] node {$a$} (0);
    \draw (0) edge[loop below] node {$b$} (0);
    \draw (0) edge[above] node {$b$} (1);
    \draw (0) edge[below] node {$c$} (1p);
    \draw (1) edge[above] node {$a$} (2);
    \draw (1) edge[below] node {$c$} (2p);
    \draw (1p) edge[below] node {$c$} (2);
    \draw (1p) edge[above] node {$b$} (2p);
    \draw (2) edge[above] node {$a$} (2p);
    \draw (2) edge[below] node {$b$} (0);
    \draw (2p) edge[above] node {$a$} (2);
    \draw (2p) edge[below] node {$b$} (0);
\end{tikzpicture}}
\]
Finite Translation Automaton

- **Deterministic finite translation automaton**
  \[ M_T = (Q_T, A_T, B_T, \delta_T, q_{T0}, F_T) \] will be constructed while searching (i.e. while reading the sequence \( S \)), where
  - \( Q_T \) is a set of states,
  - \( A_T = \{ X_1, X_2, \ldots, X_l \} \) is an input alphabet,
  - \( B_T = \{ t, f \} \) is an output alphabet,
  - \( \delta_T : Q_T \times A_T \rightarrow Q_T \times B_T \) is a transition function,
  - \( q_{T0} \) is an initial state,
  - \( F_T \) is a set of final states (always empty because matches are reported using output symbols).
Finite Translation Automaton

▶ **Deterministic finite translation automaton**

\[ M_T = (Q_T, A_T, B_T, \delta_T, q_{T0}, F_T) \]

will be constructed while searching (i.e. while reading the sequence \( S \)), where

▶ \( Q_T \) is a set of states,
▶ \( A_T = \{X_1, X_2, \ldots, X_l\} \) is an input alphabet,
▶ \( B_T = \{t, f\} \) is an output alphabet,
▶ \( \delta_T : Q_T \times A_T \rightarrow Q_T \times B_T \) is a transition function,
▶ \( q_{T0} \) is an initial state,
▶ \( F_T \) is a set of final states (always empty because matches are reported using output symbols).

▶ Mealy type automaton.
Let $M$ be a pattern matching automaton, $M_T$ be a finite translation automaton and $C = (D, S)$ be a collage system.
Let $M$ be a pattern matching automaton, $M_T$ be a finite translation automaton and $C = (D, S)$ be a collage system.

For every variable $X$ in sequence $S$ of collage system $C$: 
Let $M$ be a pattern matching automaton, $M_T$ be a finite translation automaton and $C = (D, S)$ be a collage system.

For every variable $X$ in sequence $S$ of collage system $C$:

- If a transition in $M_T$ for variable $X$ and current state is defined, then use this transition.
- Otherwise determine its target state using $M_T$, $M$ and expression representing $X$, define the transition in $M_T$ and use it (this may be recursive).
- If the output symbol of the used transition is $t$ then report an occurrence.
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_1 = a \]

\[ T = aabaccccbab \]
\[ P = \{ac, cb\} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_1 = a \]

\[ T = ababaccccbab \]
\[ P = \{ ac, cb \} \]

Outline

- Introduction
- Pattern Matching on Simple Collage Systems
- Conclusion

Finite Translation Automaton

Pattern Matching

Time and Space Complexities

Experimental Results

Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_1 = a \]
Example – LZW

\[ S = X_1 X_2 X_4 X_1 X_3 X_8 X_3 X_5 X_2 \]
\[ X_2 = b \]

\[ T = a b a b a c c c b a b \]
\[ P = \{ a c, c b \} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_2 = b \]

\[ T = ababaccccbab \]
\[ P = \{ac, cb\} \]
Example – LZW

\[
S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \\
X_4 = X_1b \\nX_1 = a
\]

\[
T = ababaccrcbab \\
P = \{ac, cb\}
\]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_4 = X_1b \quad X_1 = a \]

\[ T = ababaccccbab \quad P = \{ac, cb\} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_1 = a \]

\[ T = ababaccccbab \]
\[ P = \{ ac, cb \} \]
Example – LZW

\[ S = X_1 X_2 X_4 X_1 X_3 X_8 X_3 X_5 X_2 \]
\[ X_1 = a \]

\[ T = ababaccccbab \quad P = \{ ac, cb \} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_3 = c \]

\[ T = ababaccccbab \]
\[ P = \{ ac, cb \} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_3 = c \]

\[ T = ababaccccbab \]
\[ P = \{ac, cb\} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_8 = X_3c \quad X_3 = c \]

\[ T = ababaccccbbab \quad P = \{ac, cb\} \]

"ac" found
Example – LZW

\[ \begin{align*}
S &= X_1X_2X_4X_1X_3X_8X_3X_5X_2 \\
T &= ababacccccbab \\
P &= \{ac, cb\}
\end{align*} \]

\[ X_8 = X_3c \quad X_3 = c \]

"ac" found
Example – LZW

\[ S = X_1 X_2 X_4 X_1 X_3 X_8 X_3 X_5 X_2 \]
\[ X_8 = X_3 c \quad X_3 = c \]

\[ T = \text{ababacc}c\text{cbab} \quad P = \{ \text{ac, cb} \} \]

\[ S = X_1 X_2 X_4 X_1 X_3 X_8 X_3 X_5 X_2 \]
\[ X_8 = X_3 c \quad X_3 = c \]

\[ T = \text{ababacc}c\text{cbab} \quad P = \{ \text{ac, cb} \} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_8 = X_3c \quad X_3 = c \]

\[ T = ababaccccbab \]
\[ P = \{ ac, cb \} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_3 = c \]

\[ T = ababacccccbab \]
\[ P = \{ ac, cb \} \]
**Example – LZW**

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_3 = c \]

\[ T = ababaccccbab \]
\[ P = \{ac, cb\} \]

"ac" found
Example – LZW

\[ S = X_1 X_2 X_4 X_1 X_3 X_8 X_3 X_5 X_2 \]
\[ X_5 = X_2 a \quad X_2 = b \]

\[ T = ababacccccbab \quad P = \{ac, cb\} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ T = ababacccc bab \]
\[ P = \{ac, cb\} \]

\[ X_5 = X_2a \quad X_2 = b \]

\[ X_3/t \quad X_3/f \]

"ac" found

Marek Hanuš, Bořivoj Melichar

Pattern Matching on Simple Collage System using FA
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_5 = X_2a \quad X_2 = b \]

\[ T = ababaccccbab \quad P = \{ ac, cb \} \]
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]

\[ X_5 = X_2a \quad X_2 = b \]

\[ T = ababacccccbab \quad P = \{ac, cb\} \]

Marek Hanuš, Bohívoj Melichar

Pattern Matching on Simple Collage System using FA
Example – LZW

\[ S = X_1X_2X_4X_1X_3X_8X_3X_5X_2 \]
\[ X_2 = b \]

\[ T = ababaccccbab \]
\[ P = \{ac, cb\} \]

"ac" found
"cb" found
Example – LZW

\[ S = X_1 X_2 X_4 X_1 X_3 X_8 X_3 X_5 X_2 \]
\[ X_2 = b \]

\[ T = ababacccccbab \]
\[ P = \{ac, cb\} \]
Time and Space Complexities

- Time complexity is $O(n_U)$, where $n_U$ is the length of the uncompressed text.
- Space complexity is $O(|D||Q_T|)$. 
Experimental Results

- Comparison of the decompress-then-search approach and our algorithm on 20MB of English text.

![Graph showing comparison of decompress-then-search approach and our algorithm](chart.png)
Conclusion and Future Work

- **Conclusion:**
  - Solves many pattern matching problems (approximate matching, regular expressions, ...).
  - Pattern matching in text compressed by several compression methods (LZW, LZ78, ...).
  - Faster than decompress-then-search approach for shorter patterns (smaller pattern matching automata).

- **Future Work:**
  - Extend the algorithm for more general collage systems.
The End

Thank you for your attention.