The Most Contributory Substring Problem:

Definition and Solution

Ben Stephenson
Department of Computer Science
University of Western Ontario
Outline

- Background
- Problem Definition
- Purpose and Motivation
- Solution
- Summary
Background

• A *string* is a ordered sequence of zero or more characters
  – *ab, hello, computer science*

• A *substring* is a string that is present within another string
  – Some substrings of *computer science* are:
    • *computer, science, put, comp, ience, ...*
  – The number of substrings is very large
The Most Contributory Substring Problem: Definition

- Given a set of strings, $L$, find the substring $w$ such that the number of occurrences of $w$ multiplied by its length is maximal.

- Equivalently: Given a set of strings, $L$, find the substring $w$ that reduces the number of characters in $L$ by the greatest amount when all occurrences of $w$ are removed.
Purpose, Hypothesis and Motivation

• **Purpose:** Find an efficient solution to this problem

• **Hypothesis:** An efficient algorithm exists

• **Motivation:** Several problems can be reduced to the most contributory substring problem
Applications

- Evaluating Profile Data
  - Record events that occur when a program runs
  - Determine which sequence of events contributes the most to the execution of the program

- Data Compression
  - Find the most contributory substring for the data being compressed
  - The most contributory substring should be encoded using the shortest code word
Finding the Most Contribution Substring

1. Building a suffix tree
2. Transform the suffix tree
3. Perform Scoring
4. Traverse the suffix tree to identify the node with the highest score – it represents the most contributory substring
1: Build a Suffix Tree

- A data structure that represents a set of strings
  - Constructed by merging the strings in \( L \), placing a unique sentinel character between each string
  - Can be constructed efficiently
Suffix Tree for \textit{ababab}$\text{abab}$#
Suffix Tree for $ababab$\$abab#$

String 1

String 2

Sentinel Character
Suffix Tree for \textit{ababab}\$abab\#\}

- root
- #
- ab
- b
- $\textit{abab}\#
- #
- ab
- $\textit{abab}\#
- #
- $\textit{abab}\#$
- #
- $\textit{abab}\#
- #
- $\textit{abab}\#$
- #
- $\textit{abab}\#$
Suffix Tree for \textit{ababab$\text{abab}$#}
Suffix Tree for \textit{ababab$\#abab$ab}$

The diagram represents a suffix tree for the string \textit{ababab$\#abab$ab}. Each node in the tree represents a suffix of the input string, with the path from the root to a node indicating the suffix. The tree structure is designed to efficiently store and search for all suffixes of the given string.
Suffix Tree for \textit{ababab$abab\#}
2: Split Leaf Nodes

- Strings that contain a sentinel character are not of interest
  - Sentinel characters are not part of the original data
- Leaf node branches that start with a non-sentinel character must be split
  - Once split, all leaf node branches begin with a sentinel character
3: Scoring

• Identify the starting position for the suffix represented by each leaf node
• Traverse the tree from the leaves toward the root, merging the lists of starting positions
• Eliminate overlapping strings
• Compute score
Identify Start Positions

ababab$abab#$

```
root

#     ab     b    $abab$
12     5       7

#   ab   $abab$
10     11     6

#   $abab$   ab
8       3

$abab$
1

$abab$
2
```
Merge Start Positions

```
# 12  ab  b  $abab#
  10  5  #  11  6
 #  $abab#  ab  ab  #  $abab#
  8  3  1  2  9  4
# $abab#  ab  ab  # $abab#
  1  2
```

ababab$abab#
Merge Start

Positions

ababab$abab#$
Merge Start
Positions

- root
  - #
  - ab
    - 12
    - 1, 3, 5, 8, 10
  - b
    - 2, 5, 6, 9, 11
  - $abab#$
    - 7

- #
  - ab
    - 10
    - 1, 3, 8
  - $abab#$
    - 5
    - 2, 5, 9

- #
  - $abab#$
    - 11
    - 6
  - ab
    - 10
    - 1, 3, 8
  - $abab#$
    - 5
    - 2, 5, 9

- #
  - $abab#$
    - 11
    - 6
  - ab
    - 8
    - 3
  - $abab#$
    - 1
    - 2

- #
  - $abab#$
    - 8
    - 3
  - ab
    - 1
    - 2

- Merge Start
- Positions

- $ababab$abab#$
Eliminate Overlapping

ababab$abab#
Eliminate Overlapping

`ababab$abab#$`

**Diagram:**

```
   root
  /   \
#   ab   #
12 - 1, 3, 5, 8, 10 - 7
 /   \
#   ab   #
10 - 1, 8 - 11 - 6
 /   \
#   ab   #
  8 - 3 - 9 - 4
```

1. Eliminate overlapping.
2. Overlapping.
Compute Score

$ababab$aba$

\[
\begin{array}{c}
\text{root} \\
\text{#} \quad \text{ab} \quad \text{b} \quad \text{$abab$} \\
12 \quad 5 \times 2 = 10 \quad 5 \times 1 = 5 \quad 7 \\
\text{#} \quad \text{ab} \quad \text{$abab$} \\
10 \quad 2 \times 4 = 8 \quad 5 \quad 3 \times 3 = 9 \quad 11 \quad 6 \\
\text{#} \quad \text{$abab$} \quad \text{ab} \quad \text{ab} \quad \text{$abab$} \\
8 \quad 3 \quad 1 \times 6 = 6 \quad 1 \times 5 = 5 \quad 9 \quad 4 \\
\text{$abab$} \quad \text{$abab$} \\
1 \quad 2
\end{array}
\]
Compute Score

$abababababab$
The most contributory substring is $ab$ with score 10

- Occurs 5 times in $ababab$#\text{abab}#
- Removing all occurrences completely removes both strings in this example
Summary

• The most contributory substring, \( w \), for a set of strings, \( L \), is the string that reduces the number of characters in \( L \) the most when all occurrences of \( w \) are removed.

• Finding this string is valuable for:
  – Evaluating profile data
  – Data compression

• Suffix tree can be used to identify this string.
Questions?