Efficiency of Data Structures for Detecting Overlap in Digital Documents

Krisztián Monostori, Arkady Zaslavsky, Heinz Schmidt

Overview

- Introduction
- Related Work
- Suffix Trees
- Modified Suffix Tree
- Directed Acyclic Graphs
- Conclusion and Future Work
Introduction

**Problems**
- Intellectual property
- Plagiarism
- Search results

**Copy-prevention**
- Special hardware
- Active documents

**Copy-detection**
- Digital watermarking
- String Comparison

Copy-Detection Architecture

- Parsing Module
- Registration Module
- Comparison Module
- Index on Documents
- Results
- Documents
- Suspicious Document
String-comparison

\[ T = t_1 s_1 t_2 s_2 t_3 s_3 ... t_k s_k t_{k+1} \]
\[ P = p_1 q_1 p_2 q_2 p_3 q_3 ... p_r q_r p_{r+1} \]

for each \( s_y \) there is an \( x \) \( s_y = q_x \) and \( \Sigma(|s_y|) \) is maximal

\[ T = abcdcadca \] and \( P = aabaca \)

\[ T = ()(ab)(cdb)(c)(adc)(a)() \] and
\[ P = ()(a)()(ab)(a)(c)(a) \quad \Rightarrow \quad \Sigma(|s_x|) = 3 \]

\[ T = ()(ab)()(c)()(b)()(ca)(d)(ca)() \] and
\[ P = ()(a)()(ab)(a)(ca)() \quad \Rightarrow \quad \Sigma(|s_x|) = 8 \]

Chunking Strategy

This text will be divided into chunks using different chunking primitives

- 20 consonants (Koala)
- 10 words (“shingling”)
- 50 bytes (sif)
- 5, 10 words; sentence; whole document (SCAM)
Matching Statistics Algorithm

Modified Suffix Tree
Directed Acyclic Graphs

Matching Statistics in Directed Acyclic Graphs
Handling Suffix Links

Suffix tree Directed Acyclic Graph

Converting a Suffix Tree into a DAG

Begin

Identify the set $Q$ of pairs $(p, q)$ such that there is a suffix link from $p$ to $q$ and the number of leaves in their respective subtrees is equal.

While there is a pair $(p, q)$ in $Q$ and both $p$ and $q$ are in the current DAG,

Merge node $p$ into $q$

End
Conversion Algorithm in Practice

Find number of leaves under each node
For each node
  Identify equivalent node
End For
For each node
  Divert suffix link to equivalent node
  Define offset value for suffix link
  For each edge running out of current node
    If equivalent exist
      Divert edge
      Define offset value for edge
      Add next node to remove_list
    End If
  End For
End For
For each node in the remove_list
  Remove subtree
End For

Performance Results

<table>
<thead>
<tr>
<th>Document</th>
<th>Size of Tree(bytes)</th>
<th>Size of DAG(bytes)</th>
<th>MS on Tree(ms)</th>
<th>MS on DAG(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genuine</td>
<td>309,192</td>
<td>246,124</td>
<td>530</td>
<td>410</td>
</tr>
<tr>
<td>60%</td>
<td>228,716</td>
<td>182,988</td>
<td>531</td>
<td>408</td>
</tr>
<tr>
<td>Huge chunk</td>
<td>41,573,992</td>
<td>32,234,564</td>
<td>1923</td>
<td>1611</td>
</tr>
</tbody>
</table>
Conclusion

- Copy-detection algorithms produce false matches
- Filter: suffix tree
- Utilising words
- Directed Acyclic Graph
  - Preserving suffix link information

Future Work

- Combining two approaches
- Building the Directed Acyclic Graph from scratch
- Alternative representations
  - Suffix arrays
  - Suffix vectors
- MatchDetectReveal System