Importance of Aho-Corasick String Matching Algorithms in Real World Applications

Franks Jhon Colque Zegarra[1] Luis Mamani Chirinos[2]

National University of San Agustin, School of Computer Science

czfranks@gmail.com

January 3, 2018

Overview

1. Introduction
   - Motivation
   - Set Matching Problem
   - Trie
2. Aho-Corasick
   - Aho-Corasick Automaton
   - Functions Of Aho-Corasick
   - Aho-Corasick algorithm
     - Phase I
     - Phase II
3. Aplications Of Aho-Corasick
   - Digital Forensics and Text Mining
   - Intrusion Detecting
   - Detecting Plagiarism
   - Bioinformatics

Motivation

- String Matching Algorithms we try to find the position where patterns are found within a larger string or text.
- String Matching can be performed in the text String through Single Pattern and Multipattern Pattern ocurrences.
- Aho-Corasick solve the multipattern matching efficiently.
- Aho-Corasick was invented by Alfred V. Aho and Margaret J. Corasick

Set Matching Problem.

In the Set Matching Problem we locate ocurrences of any pattern of set $\mathcal{P} = \{P_0, \cdots, P_k\}$,in target $\mathcal{T}[1 \cdots m]$
Set Matching can be solved in time

$$O(|P_1| + m + \cdots + |P_k| + m) = O(n + km)$$

where

$$n = \sum_{i=1}^{k} |P_i|$$

Aho-Corasick algorithm(AC) is a good solution to set Matching
Now set Matching can be solved in time

$$O(n + m + z)$$

where

$$z = number\ \ of\ \ pattern\ ocurrences\ in\ \mathcal{T}$$

Trie

A **Trie** (or a **keyword Tree**) is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings.

A Trie for a set of patterns $\mathcal{P}$ is a rooted tree $\mathcal{K}$ such that:

1. Each edge of $\mathcal{K}$ is labeled by a caracter.
2. Any two nodes out of a node have different labels.

$L(v)$: Label of node v as concatenation on the path from the root to v.S

3. For each $P \in \mathcal{P}$ $there's$ a node v with $\mathcal{L}(v) = P$
4. The label $\mathcal{L}(v)$ of any leaf v equals some $P \in \mathcal{P}$.

Example Trie

A Trie for $\mathcal{P} = \{he, she, his, hers\}$:

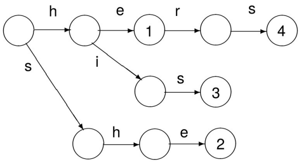

Figure: A Trie is an efficient implementation of a **dictionary** of strings.

Trie: Construction

Construction for $\mathcal{P} = \{he, she, his, hers\}$:

1) Begin with a root node only
2) Insert each pattern P_i
3) Starting a the root, follow the path labeled by chars pf P_i
4) If the path ends before P_i, continue it by adding new edges and nodes for the remaining charactes of P_i
5) Store identifier i of P_i at the terminal node of the path

This takes $O(|P_1| + \cdots + |P_k|) = O(n)$ time.

Lookup of a String P: $\cdots$

Aho Corasick(AC) automaton

Aho Corasick(AC) is the multipattern matching which locates all the ocurrences of set of patterns in a text of string. It first creates Deterministic automate for all predefined patterns.

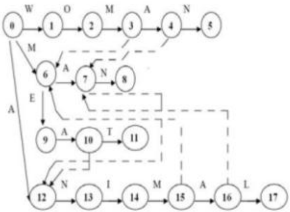

Figure: AC Automaton of patterns $\mathcal{P} = \{woman, man, meat, animal\}$

Goto Function

The **goto function** $g(q, a)$ gives the state entered from current state q by matching target char a

if edge(q, v) is labeled by a, then $g(q, a) = v$

$g(0, a) = 0$ for each a that does not label an edge out of the root

Otherwise $(q, v) = \phi$

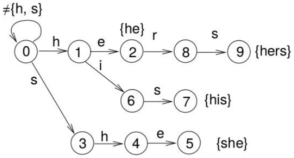

Figure: **Goto** function

Failure Function

The **failure function** $f(q)$ for $q \neq 0$:

$f(q)$ is the node labeled by the longest proper suffix w of $\mathcal{L}(q)$, w is a prefix of some pattern

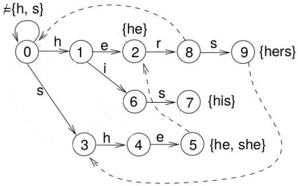

Figure: **Fail** transition of nodes $5, 8, 9$

Output Function

The **output function** $out(q)$ gives the set patterns recognized when entering state q $f(q)$ is the node labeled by the longest proper suffix w of $\mathcal{L}(q)$, w is a prefix of some pattern

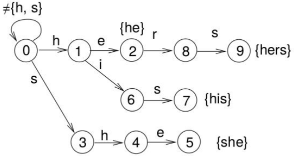

Figure: **Out** function

Aho-Corasick Automaton search of target $T[1 \cdots m]$

Example (code AC Search)

$q := 0;$
for $i := 1$ *to* m *do*
  while $g(q, T[i]) = \phi$ *do*
    $q := f(q);$
   $q := g(q, T[i]);$
   if $out(q) = \phi$ *then*
    print $i,\ out(q);$
endfor;

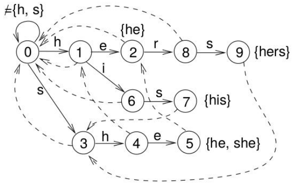

Figure: Search text "Ushers"

Complexity of Search in Aho-Corasick Automaton

Searching target $\mathcal{T}[1 \cdots m]$ with an AC automaton takes time $O(m + z)$

- Each **goto** either stay at the root or increased the depth of q by 1
- Each **fail** moves q closer to the root
- The ocurrences can be reported in $z \ x \ O(1) = O(z)$

Constructing an Aho-Corasick automaton

The Aho-corasick automaton can be constructed in two phases.

- Phase 1
- Phase 2

Constructing Phase 1

Phase 1

1. Construct the Trie Tree for $\mathcal{P}$
2. Complete the goto function for the root

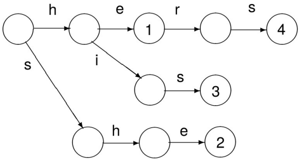

Figure: Trie of $\mathcal{P} = \{he, she, his, hers\}$

This Phase 1 of construction takes time $O(n)$

Constructing Phase 2

Q := emptyQueue();
for $a \in \Sigma$ **do**
  if $q(0, a) = q \neq 0$ **then**
    $f(q) := 0$; enqueue(q, Q);
while not isEmpty(Q) **do**
  r:= dequeue(Q);
  for $a \in \Sigma$ **do**
    if $g(r, a) = u \neq \emptyset$ **then**
      enqueue(u, Q); $v := f(r)$;
      while $g(v, a) = \emptyset$ **do** $v := f(v)$; // (*)
      $f(u) := g(v, a)$;
      out(u) := out(u) $\cup$ out($f(u)$);

Figure: Algorithm phase 2

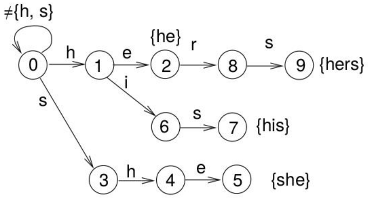

Figure: Trie of $\mathcal{P} = \{he, she, his, hers\}$

Constructing Phase 2

$Q :=$ emptyQueue();
for $a \in \Sigma$ **do**
    if $g(0, a) = q \neq 0$ **then**
        $f(q) := 0$; enqueue(q, Q);
while not isEmpty(Q) **do**
    r:= dequeue(Q);
    for $a \in \Sigma$ **do**
        if $g(r, a) = u \neq \emptyset$ **then**
            enqueue(u, Q); $v := f(r)$;
            while $g(v, a) = \emptyset$ **do** $v := f(v)$; // (*)
            $f(u) := g(v, a)$;
            out(u) := out(u) $\cup$ out($f(u)$);

Figure: Algorithm phase 2

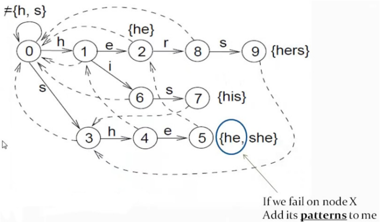

Figure: Trie of $\mathcal{P} = \{he, she, his, hers\}$

Constructing Phase 2

Q := emptyQueue();
for $a \in \Sigma$ **do**
  if $q(0, a) = q \neq 0$ **then**
    $f(q) := 0$; enqueue(q, Q);
while not isEmpty(Q) **do**
  r:= dequeue(Q);
  for $a \in \Sigma$ **do**
    if $g(r, a) = u \neq \emptyset$ **then**
      enqueue(u, Q); $v := f(r)$;
      while $g(v, a) = \emptyset$ **do** $v := f(v)$; // (*)
      $f(u) := g(v, a)$;
      out(u) := out(u) $\cup$ out($f(u)$);

Figure: Algorithm phase 2

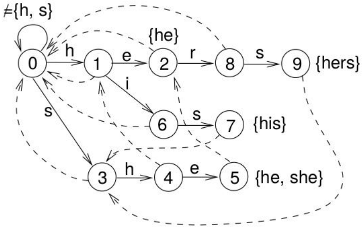

Figure: Trie of $\mathcal{P} = \{he, she, his, hers\}$

Constructing Phase 2

- Functions fail and output are computed for the nodes of the trie in a breadth-first order
- Then Phase 2 can be run in time $O(n)$
- Is also an $O(n)$ bound for the number that the f transitions on line that contains (*)?

Digital Forensics and Text Mining

- **Digital Forensics**: is a matematical scheme for demonstrating the authencity of a digital message or document.
- **Text Mining**: refers to the process of deriving high.quality information from text. Deriving patterns whitin the structured data, and finally evaluation and interpretation of the output.

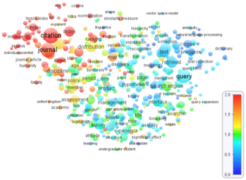

Figure: Term map of the Journal of the American Society for Information Science and Technology

Instrusion Detecting

- **Digital Signature**: is to compare one brand with a file virus database to identify matches
- **Heuristic detection**: is the scan code looking for patterns that resemble those used in the virus.

Figure: Digital Signature

Zero_Bug	Zharinov (Clam)	Zhengxi.7313 (Clam)	Zherkov-1882
Zherkov-1915	Zherkov-1915	Zherkov-2435	Zherkov-2968
Zherkov-2970	Zherkov-A	Zherkov-B	ZigZag
ZigZag-127 (B)	Zodiac #1	Zohra.4516 (Clam)	Zombie.2553 (Clam)
Zombie.667 (Clam)	Zombie.PM.4566 (Clam)	Zombie.PM.4592 (Clam)	_0017_0001_000
_0017_0001_001	_0017_0001_002	_0017_0001_003	_0017_0001_004
_0023_0004_003	_0023_0004_004	_0023_0005_000	_0023_0005_001
_0023_0005_002	_0025_0006_000	_0025_0006_001	_0026_0006_000

Figure: Index of virus detected by clamAV

Detecting Plagiarism

- Plagiarism Detecting is process of finding within a work or document.
- Plagiarism is the act of copying the idea of someone else.
- There are so many algorithms have been proposed for plagiarism detection.

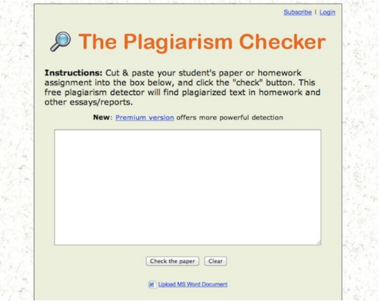

Figure: The Plagiarism Checker

Bioinformatics

Bioinformatics is the study of biological science which deals with the methods of storing, re**trie**ving and analyzing biological data, such nucleic acid(DNA,RNA) and protein sequence.

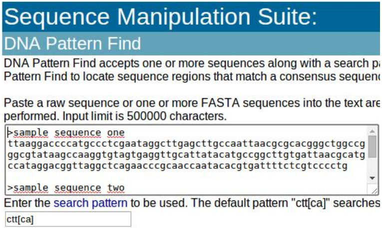

Figure: DNA Pattern Find

References

- Saima Hasib, Mahak Motwani, Amit Saxema (2013)
  Importance of Aho-Corasick String Matching Algorithm in Real World Applications
  (IJCSIT)International journal of Computer Science and Information Technologies, Vol. 4(3), 2013, 467-469
- Pekka Kilpelainen (2005)
  Set Matching and Aho-Corasick Algorithm
  Biosequence Algorithms, Spring 2005.

The End