Accelerating Knuth-Morris-Pratt String matching over LZ77 Compressed Text

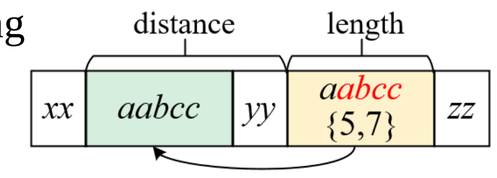
Xiuwen Sun, Di Wu, Da Mo Jie Cui, Hong Zhong
School of Computer Science and Technology, AnHui University, China

Introduction

- The heuristic string matching plays an important role in pattern matching owing to their excellent performance, and the increasing compressed text challenges string matching to achieve high-speed processing.
- Performance of compressed pattern matching depends on skipping more bytes and spending less for skipping, and there is no kmp-based compressed pattern matching that can achieve better performance on these two factors at present.

Basic Concept

- ***LZ77 Compress*:** replaces redundant string by a two-tuple of {***length, distance***}.

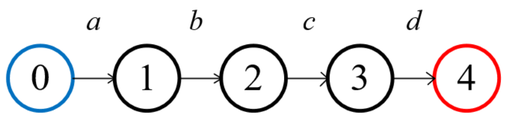


- ***auxiliary function σ(x)*:** the length of the longest prefix of pattern *P* that is also a suffix of string *x*.

$P = abcd$

$\sigma(aa) = 1,\ \sigma(aabc) = 3$

- ***A=(Q, Σ, δ, q0, F)*:** a string matching automaton that corresponds to pattern *P*[1....m].



$Q = \{0, 1, \cdots, m\}$, $q_0 = 0$, $F = \{m\}$, $\delta(q,a) = \sigma(P_q a)$ $q \in Q$, $P_q = P[1...q]$

for any string ***x*** and character ***a***, if q = σ(x), then $\sigma(xa)=\delta(q, a) =\sigma(P_q a)$

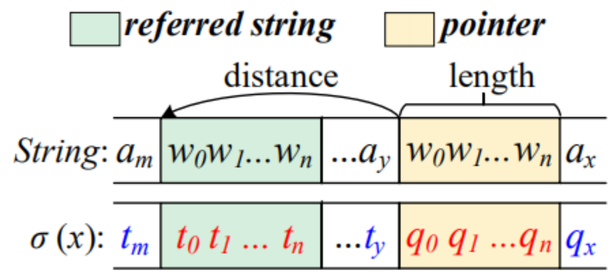
Pattern P=abcd string x=ab

$q=\sigma(ab)=2$

$\sigma(abc)=\delta(2,c)=\sigma(P_2 c)=3$

using ***σ*** value avoids precomputing the full transition function δ for an automaton-based KMP.

Algorithm



$t_y = t_m$

$q_0 = \sigma(P_{t_y} w_0) = \sigma(P_{t_m} w_0) = t_0$

$q_0 = t_0, q_1 = t_1 ... q_n = t_n$

copy $t_0 t_1 ... t_n$ to $q_0 q_1 ... q_n$

if q_i is accepting state, matching pattern

$t_m \neq t_y$

scan w_0 in *pointer*, assume get $q_0 = t_0$

$q_1 = \sigma(P_{q_0} w_1) = \sigma(P_{t_0} w_1) = t_1$

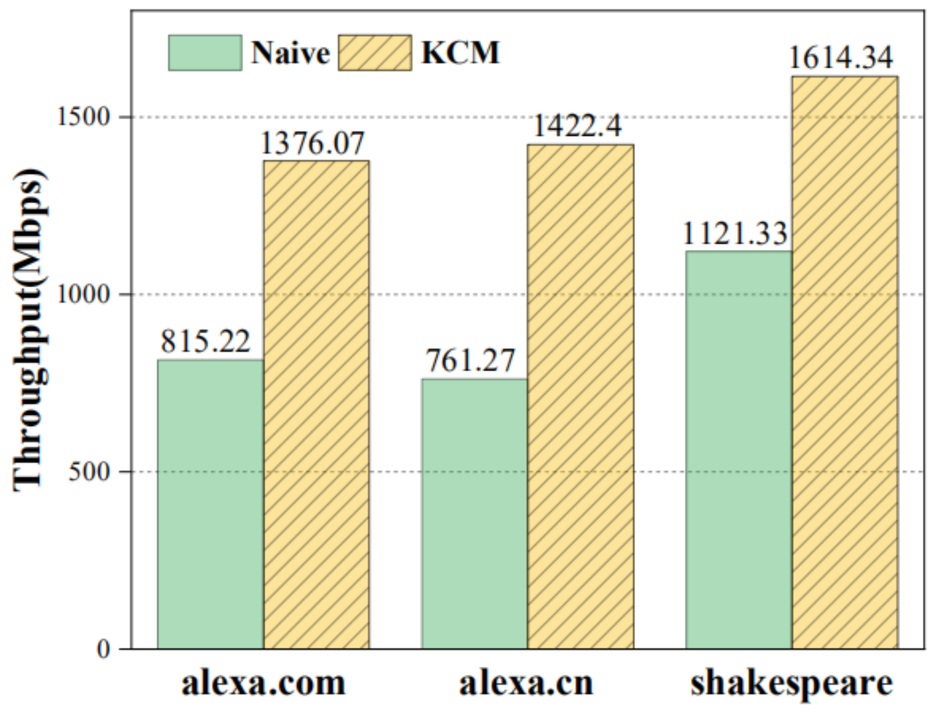
$q_1 = t_1, q_2 = t_2 ... q_n = t_n$

if q_i is accepting state, matching pattern

Example

offset:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
compressed:	x	a	b	a	b	c	x			{5, 6}			d	x
decompressed:	x	a	b	a	b	c	x	a	b	a	b	c	d	x
σ(x):	0	1	2	1	2	3	0	1	2	1	2	3	4	0
setp 1:	a	b	c	d										
step 2-4:		a	b	c	d									
setp 5-7:				a	b	c	d							
setp 8:								a	b	c	d			
step 9:								↑		a	b	c	d	
step 10:													↑	a

Evaluation



Throughput of the methods over three data sets

Characteristic	Alexa.com	Alexa.cn	Shakespeare
matched patterns	155,087	2,439,445	4976
skipped ratio	91.07%	91.85%	93.99%
Bytes represented by pointers	**91.21%**	**91.92%**	**94.06%**
throughput boost	1.69x	1.87x	1.44x

Metrics of KCM over three data sets

Conclusion

KCM
- A method for KMP string matching over compressed traffic. The skipped bytes approaches the theoretical upper bound.

Faster
- 1.58 Gbps on matching compressed text, which is 1.4~1.9 times improvement under the experiments with real traffic.