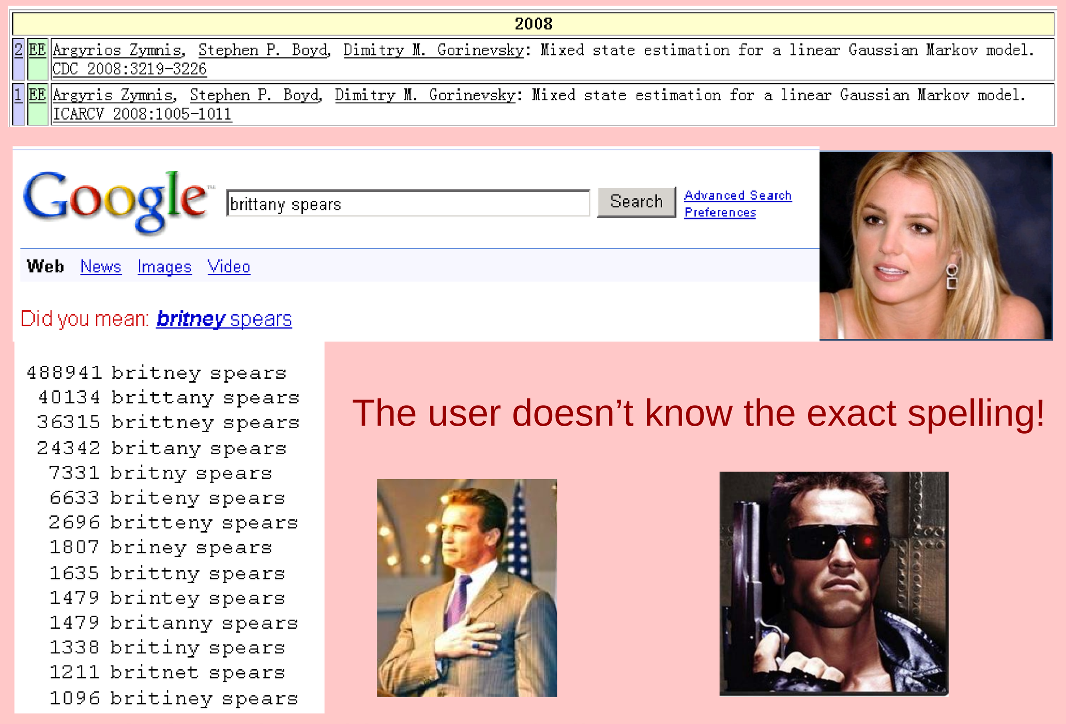
Top-k String Similarity Search with Edit-Distance Constraint

Dong Deng[$], Guoliang Li[$], Jianhua Feng[$], Wen-Syan Li[^]
[$]Department of Computer Science, Tsinghua University, Beijing, China
[^]SAP Labs, Shanghai, Beijing

Motivation



Find movies with a star "similar to" Schwarzenger

Star	Title	Year	Genre
Keanu Reeves	The Matrix	1999	Sci-Fi
Samuel Jackson	Iron man	2008	Sci-Fi
Schwarzenegger	The Terminator	1984	Sci-Fi
Samuel Jackson	The man	2006	Crime

Top-k String Similarity Search

#1: Data in real world is dirty

Edit Distance: minimum # of single character transformations
e.g. ED(srajit, seraji) = 2

#2: Hard to define a threshold

#3: Many real applications

- ☐ Information retrieval
- ☐ Molecular biology
- ☐ Bioinformatics
- ☐ Data Quality, Data Cleaning

Problem Definition

Top-k String Similarity Search: *Given a string set S and a query string q, top-k string similarity search returns a string set R ⊆ S such that |R|=k and for any string r ∈ R and s ∈ S − R, ED(r, q) ≤ ED(s, q).*

TABLE I

A STRING SET $\mathcal{S}$ AND A QUERY q = "srajit"

ID	s_1	s_2	s_3	s_4	s_5	s_6
String	sarit	seraji	suijt	suit	surajit	thrifty

the top-3 similar strings of *srajit*

Progressive Framework

(a) Traditional method

		0	1 s	2 r	3 a	4 j	5 i	6 t
0		0	1	2	3	4	5	6
1	s	1	0	1	2	3	4	5
2	e	2	1	1	2	3	4	5
3	r	3	2	1	2	3	4	5
4	a	4	3	2	1	2	3	4
5	j	5	4	3	2	1	2	3
6	i	6	5	4	3	2	1	2

(b) Progressive method

		0	1 s	2 r	3 a	4 j	5 i	6 t
0		0	1	2				
1	s	1	0	1	2			
2	e	2	1	1	2			
3	r		2	1	2			
4	a			2	1	2		
5	j				2	1	2	
6	i					2	1	2

(a) E_0 = FINDMATCH(−1, −1) = {⟨0, 0⟩, ⟨1, 1⟩}

(b) Computing E_1 based on E_0

E_0	⟨0,0⟩			
	Substitution	Insertion	Deletion	Substitution
EXTENSION	~~⟨1, 1⟩~~	⟨1, 0⟩	⟨0, 1⟩	⟨2, 2⟩
E_1	⟨1, 0⟩, ⟨0, 1⟩, ⟨2, 2⟩, ⟨2, 1⟩, ⟨3, 2⟩,			

(c) Computing E_2 based on E_1

E_1		⟨1,0⟩	⟨0,1⟩	⟨2,2⟩	⟨2,1⟩
EXTENSION	Substitution	~~⟨2,1⟩~~	~~⟨1,2⟩~~	~~⟨3,3⟩~~	~~⟨3,2⟩~~
	Insertion	⟨2,0⟩	~~⟨1,1⟩~~	~~⟨3,2⟩~~	⟨3,1⟩
	Deletion	~~⟨1,1⟩~~	⟨0,2⟩	⟨2,3⟩	~~⟨2,2⟩~~
E_2		⟨2,0⟩, ⟨0,2⟩, ⟨2,3⟩, ⟨3,1⟩, ⟨4,2⟩, ⟨3,			

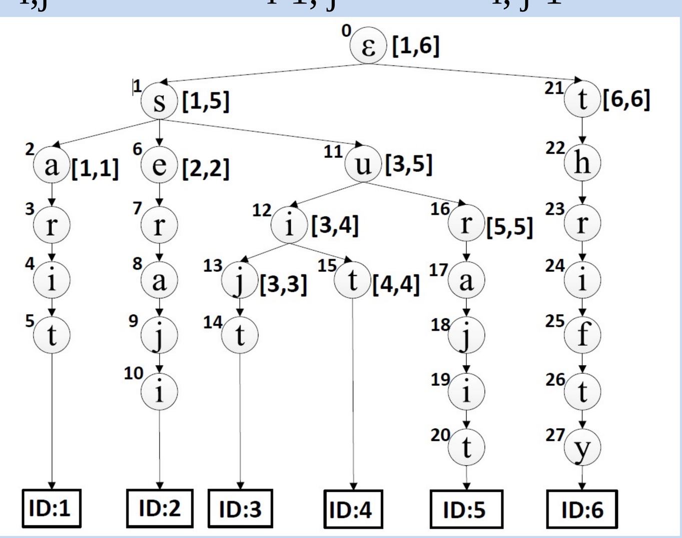
Traditional Method: Dynamic Programming

$D_{i,0} = i$, $D_{0,j} = j$, *Insertion* *Deletion* *Match/Subsitition*

$D_{i,j} = \min\{D_{i-1,j} + 1, D_{i,j-1} + 1, D_{i-1,j-1} + 0/1\}$

Progressive Method: Smallest Cell First
Two Operations: Find Match / Extend

A trie for strings in table 1



Top-3 Query: *srajit*

(n_i, j) { *i*-th node of trie; *j*-th character of Q }

Tx: node and char with ED=x

(a) T_0 = ⟨n_0, 0⟩ ∪ FINDMATCH(0, 0) = {⟨n_0, 0⟩, ⟨n_1, 1⟩}

(b) Computing T_1 based on T_0

T_0	⟨n_0, 0⟩		
	Substitution	Insertion	Deletion
EXTENSION	~~⟨n_1, 1⟩~~ ⟨n_{21}, 1⟩	⟨n_1, 0⟩ ⟨n_{21}, 0⟩	⟨n_0, 1⟩
T_1	⟨n_{21}, 1⟩ ⟨n_1, 0⟩ ⟨n_{21}, 0⟩ ⟨n_0, 1⟩ ⟨ … ⟨n_9, 4⟩ ⟨n_{10}, 5⟩ ⟨n		

(c) Computing T_2 based on T_1

T_1		⟨n_{21}, 1⟩	⟨n_0, 1⟩	⟨n_1, 0⟩	⟨n_2
EXTENSION	Substitution	⟨n_{22}, 2⟩	~~⟨n_1, 2⟩~~ ~~⟨n_{21}, 2⟩~~	~~⟨n_2, 1⟩~~ ~~⟨n_6, 1⟩~~ ~~⟨n_{11}, 1⟩~~	~~⟨n_2~~
	Insertion	⟨n_{22}, 1⟩ ⟨n_{23}, 2⟩	~~⟨n_1, 1⟩~~ ~~⟨n_{21}, 1⟩~~	⟨n_2, 0⟩ ⟨n_6, 0⟩ ⟨n_{11}, 0⟩	⟨n_2
	Deletion	⟨n_{21}, 2⟩	⟨n_0, 2⟩	~~⟨n_1, 1⟩~~	~~⟨n_2~~
T_2		⟨n_{22}, 2⟩ ⟨n_{21}, 2⟩ … ⟨n_7, 3⟩ ⟨n_6, 3⟩ ⟨n_{12}, 3⟩ ⟨			

Pivotal Entry-based Method

Definition(Pivotal Entry): An entry ⟨*i, j*⟩ in E*x* is called a pivotal entry, if $D[i + 1][j + 1] > D[i][j]$.

		0	1 s	2 r	3 a	4 j	5 i	6 t
0		0						
1	s		0	1	2			
2	e		1	1	2			
3	r			1	2			
4	a				1			
5	j					1		
6	i						1	2

TABLE IV

PIVOTAL ENTRIES TO COMPUTE EDIT DISTANCE("srajit","seraji")

(a) E^p_0 = {⟨1, 1⟩}

(b) Computing E^p_1 based on E^p_0

E^p_0	$E^p_0[0]$ = ⟨1, 1⟩		
	Substitution	Insertion	Deletion
EXTENSION	$E^p_1[0]$ = ⟨2, 2⟩	~~$E^p_1[1]$ = ⟨2, 1⟩~~ $E^p_1[1]$ = ⟨6, 5⟩	$E^p_1[-1]$ = ⟨1, 2⟩
E^p_1	⟨1, 2⟩ , ⟨2, 2⟩ , ⟨6, 5⟩		

(c) Computing E^p_2 based on E^p_1. E^p_2={⟨1, 3⟩, ⟨2, 3⟩, ⟨6, 6⟩, ⟨7, 5⟩, ⟨7, 6⟩}

E^p_1	$E^p_1[-1]$ = ⟨1, 2⟩		
	Substitution	Insertion	Deletion
EXTENSION	$E^p_2[-1]$=⟨2, 3⟩	$E^p_2[0]$=~~⟨2, 2⟩~~ $E^p_2[0]$=⟨6, 6⟩	$E^p_2[-2]$=⟨1, 3⟩
E^p_1	$E^p_1[0]$ = ⟨2, 2⟩		
	Substitution	Insertion	Deletion
EXTENSION	$E^p_2[0]$=~~⟨3, 3⟩~~ $E^p_2[0]$=⟨6, 6⟩	$E^p_2[1]$=~~⟨3, 2⟩~~ $E^p_2[1]$=⟨7, 6⟩	$E^p_2[-1]$=~~⟨2, 3⟩~~ $E^p_2[-1]$=⟨2, 3⟩

Definition (Pivotal Triple): Given an entry ⟨*n, j*⟩, one of *n*'s children n_c, and a query *q*, triple ⟨*n, j,* n_c⟩ is called a pivotal triple, if ED(n_c, q[1, j+1]) >ED(*n*, q[1, j]).

(a) T^p_0 = {⟨n_0, 0, n_{21}⟩, ⟨n_1, 1, n_2⟩, ⟨n_1, 1, n_6⟩, ⟨n_1, 1, n_{11}⟩}

(b) Computing T^p_1 based on T^p_0

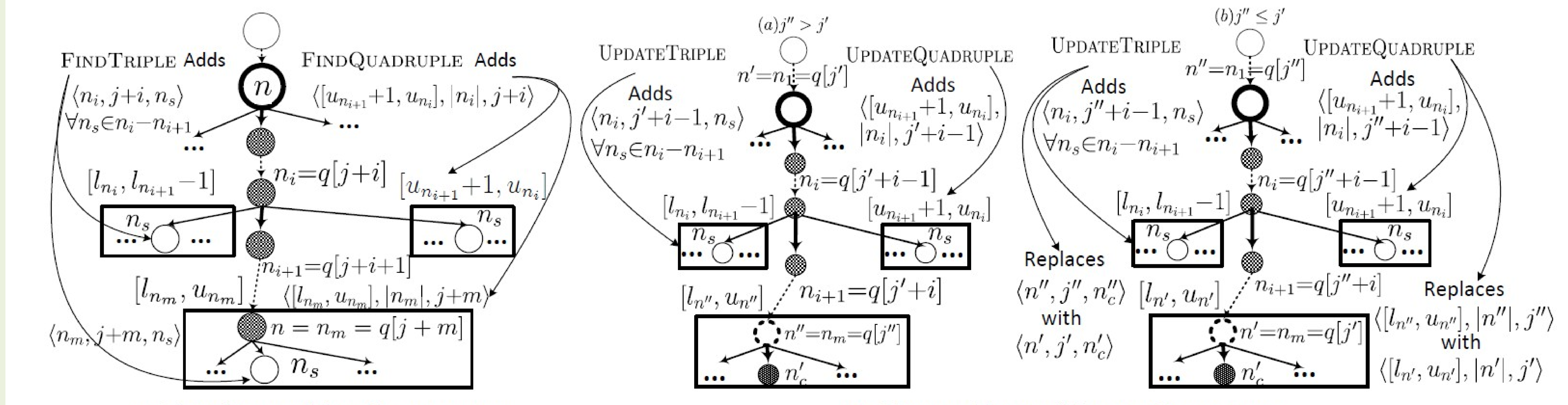
T^p_0		T^p_0 [0]	⟨n_0, 0, n_{21}⟩	⟨n_1, 1, n_2⟩	
EXTENSION	Substitution	$T^p_1[0]$:	⟨n_{21}, 1, n_{22}⟩	⟨n_2, 2, n_3⟩	
	Insertion	$T^p_1[1]$:	⟨n_{21}, 0, n_{22}⟩	~~⟨n_2, 1, n_3⟩~~ ⟨n_3, 2, n_4⟩	~~⟨n~~
	Deletion	$T^p_1[-1]$:	⟨n_0, 1, n_{21}⟩	~~⟨n_1, 2, n_2⟩~~ ⟨n_2, 3, n_3⟩	
T^p_1			⟨n_{21}, 1, n_{22}⟩ ⟨n_2, 2, n_3⟩ ⟨n_6, 2, n ⟨n_{21}, 0, n_{22}⟩ ⟨n_{11}, 1, n_{12}⟩ ⟨n_{11}, 1		

(c) Computing T^p_2 based on T^p_1

T^p_1		T^p_1 [0]	⟨n_{21}, 1, n_{22}⟩	⟨n_2, 2, n_3⟩
EXTENSION	Substitution	$T^p_2[0]$:	⟨n_{22}, 2, n_{23}⟩	⟨n_3, 3, n_4⟩
	Insertion	$T^p_2[1]$:	~~⟨n_{22}, 1, n_{23}⟩~~ ⟨n_{23}, 2, n_{24}⟩	~~⟨n_3, 2, n_4⟩~~
	Deletion	$T^p_2[-1]$:	⟨n_{21}, 2, n_{22}⟩	~~⟨n_2, 3, n_3⟩~~
T^p_2			⟨n_{22}, 2, n_{23}⟩ ⟨n_3, 3, n_4⟩ ⟨n_{11}, 3, n_{12}⟩ ⟨n_{20}, 7, ϕ⟩ ⟨n_{12}, 2, n_{13}⟩	

Range-based Method

Definition 4 (Pivotal Quadruple): A quadruple ⟨[*l*, *u*], *d*, *j*⟩ is a pivotal quadruple, if it satisfies (1) ⟨*l*, *u*⟩ is a sub-range of a *d*-th level node's range;
(2) for any string s with ID in [l, u], ED(s[1, d+1], q[1, j+1])>ED(s[1, d], q[1, j]);
(3) strings with ID l − 1 or u + 1 do not satisfy conditions (1) or (2).



(a) FINDTRIPLE/FINDQUADRUPLE (b) UPDATETRIPLE/UPDATEQUADRUPLE

Fig. 4. Operations of the pivotal triple based method and the range-based method

Experiments

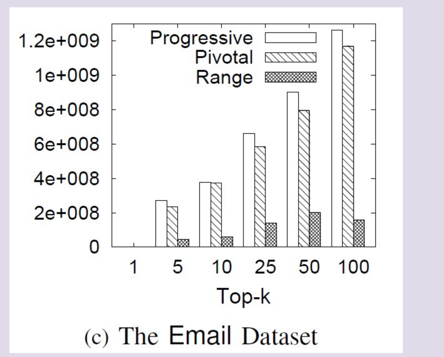
implemented in C++, Ubuntu: Intel Xero X5670 2.5GHz CPU and 4 GB memory

Datasets

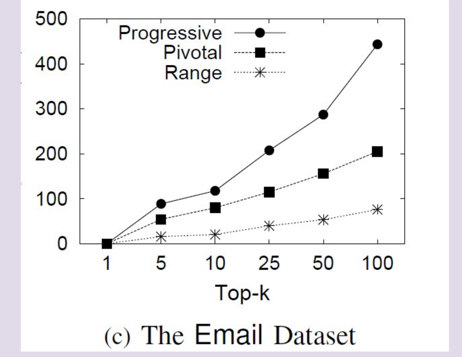
TABLE VII

DATASETS

Datasets	Cardinality	Avg Len	Max Len	Min Len
Word	146,033	16.01	35	1
Author	10.27 million	22.02	383	8
Email	6.4 million	26.58	57	7

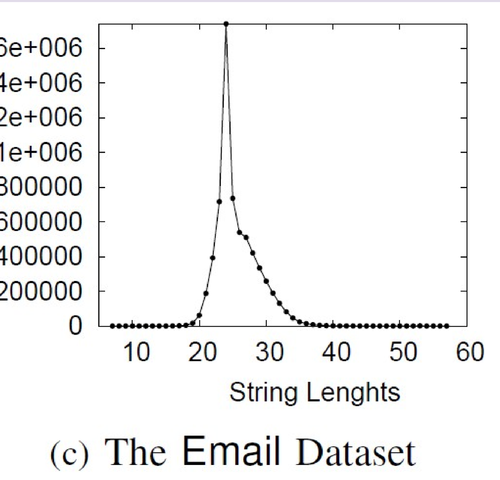


Number of calculated entries

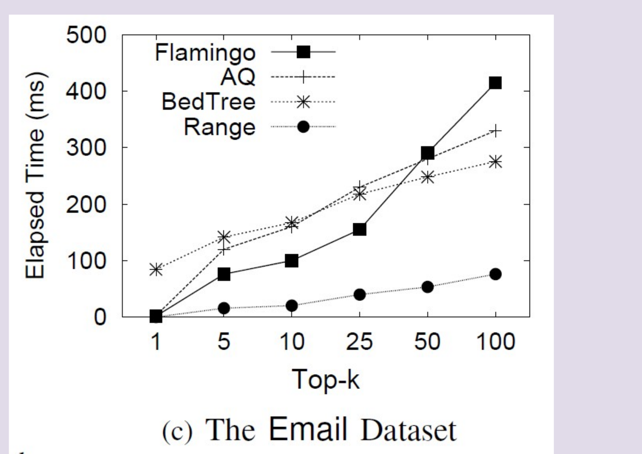


Running Time of Each Methods

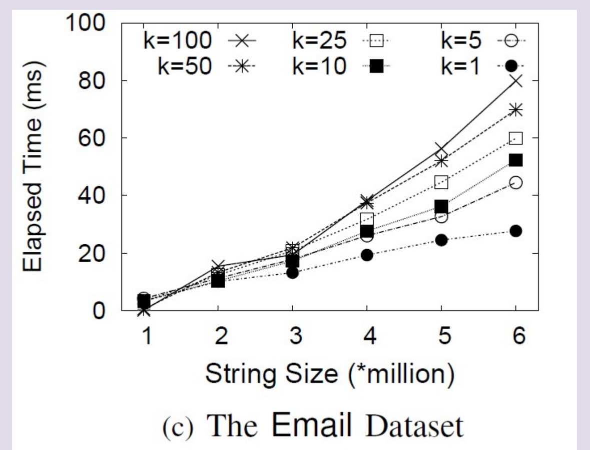
Email Dataset



Length Distribution



State-of-the-art methods



Scalibility

http://dbgroup.cs.tsinghua.edu.cn/dd/projects/topksearch