

Flexible String Matching Against Large Databases in Practice

Divesh Srivastava

AT&T Labs–Research

<http://www.research.att.com/~divesh/>

Joint work with Nick Koudas and Amit Marathe

Thesis

- String data prevalent in databases
 - customer names, addresses, circuit ids, ...
 - need to search, correlate tables based on string data
- But string data often of poor quality
 - multiple conventions, errors, embedded comments
 - equality, prefix, substring matching of limited use
- Solution: Use “flexible” string matching
 - string edit distance of limited use
 - cosine similarity with tf.idf of great practical use

Outline of Talk

- Thesis
- Background: matching on one string-valued attribute
- Functionality need: match on n attributes, use equivalences
- Performance need: enhance efficiency, trade off recall

Background: Need for Flexible String Matching

- Poor quality string data
 - “MICROSOFT INC.” vs “MICROSOFT CORPORATION”
 - “11810 WILLS ROAD ALPHARETTA GA30004” vs “BLDG 110 FLR 1
RM 110 11810 WILLIS RD ALPHARETTA GA 30004205”
- Equality, substring, string edit distance matchings of limited use

Background: Single Attribute TF.IDF Matching

- Cosine similarity with (token freq, inverse DB freq)
 - with string s , associate a weight vector v_s
 - $v_s[i]$ is tf.idf weight of token t_i in string s
 - similarity between strings $s1$ and $s2$, $sim(s1, s2) = \sum_i (v_{s1}[i] * v_{s2}[i])$
- Correlating two database tables using one string attribute
 - find all pairs with similarity \geq threshold
 - conceptually: thresholded matrix multiplication
 - practically: use SQL for flexibility, efficiency

Background: SQL for Single Attribute Matching

- Pre-processing: extract tokens, compute tf.idf weights, normalize

```
Base(tid, sva, ...)
BaseSize(size)
BaseTF(tid, token, tf)
```

```
insert into BaseLength(tid, len)
select T.tid, SQRT(SUM(I.idf*I.idf*T.tf*T.tf))
from BaseTF T, BaseIDF I
where T.token = I.token
group by T.tid
```

```
insert into BaseIDF(token, idf)
select T.token, LOG(S.size) - LOG(COUNT(T.tid))
from BaseTF T, BaseSize S
group by T.token
```

```
insert into BaseWeights(tid, token, weight)
select T.tid, T.token, T.tf*I.idf/L.len
from BaseTF T, BaseIDF I, BaseLength L
where T.token = I.token and T.tid = L.tid
```

- Query time: above steps for query string, correlate & aggregate

```
Search(tid, sva, ...)
SearchTF(tid, token, tf)
SearchLength(tid, len)
SearchWeights(tid, token, weight)
```

```
select S.tid, B.tid, SUM(S.weight*B.weight)
from SearchWeights S, BaseWeights B
where S.token = B.token
group by S.tid, B.tid
having SUM(S.weight*B.weight) > T
```

Functionality Need: Multiple Attribute Matching

- Need to match simultaneously on name and address
 - attribute concatenation: loses per-attribute statistical information
 - static weights: how to choose relative weights?
- Solution: dynamic weights for per-tuple attribute importance
 - normalize weight vectors jointly, using $L(n, a) = \frac{L(n)}{\sqrt{L(n)^2 + L(a)^2}}$
 - run flexible matches on disjoint union of name, address
 - similarity scores are still between 0 and 1

Multiple Attribute Matching: Experiments

- Query: (“Worldcom”, “Wall St Manhattan NY”)
- Results: dynamic weighting has better quality answers

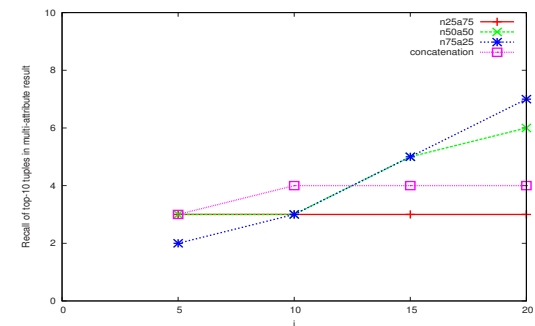
sim	name	address
0.75	WORLDCOM	600 s federal st chicago il
0.75	WORLDCOM	400 international pkwy dallas tx
0.75	WORLDCOM	300 renaissance ctr detroit mi
0.75	WORLDCOM	111 8th ave new york ny
0.75	WORLDCOM	165 boulevard se atlanta ga
0.75	WORLDCOM	165 boulevard ne atlanta ga
0.75	WORLDCOM	910 15th st denver co
0.75	WORLDCOM	401 fieldcrest dr greenburgh ny
0.75	WORLDCOM	1102 grand blvd kansas city mo
0.75	WORLDCOM	1102 grand blvd kansas city mo

sim	name	address
0.5	WORLDCOM	600 s federal st chicago il
0.5	WORLDCOM	400 international pkwy dallas tx
0.5	WORLDCOM	300 renaissance ctr detroit mi
0.5	WORLDCOM	111 8th ave new york ny
0.5	WORLDCOM	165 boulevard se atlanta ga
0.5	WORLDCOM	165 boulevard ne atlanta ga
0.5	WORLDCOM	910 15th st denver co
0.5	WORLDCOM	401 fieldcrest dr greenburgh ny
0.5	WORLDCOM	1102 grand blvd kansas city mo
0.5	WORLDCOM	1102 grand blvd kansas city mo

sim	name	address
0.568945894389856	TC PAYPHONES	w houston st manhattan ny
0.563179091332391	TC PAYPHONES	w 30th st manhattan ny
0.559800387826103	PSI COLOCATE AMTRAK	w 33rd st manhattan ny
0.559800387826103	LIRR	w 33rd st manhattan ny
0.543052687383282	AMERICAN MUSEUM OF N	w 79th st manhattan ny
0.522269095184392	AUDREY ZUCKNER	manhattan ny
0.522269095184392	TUMBLE INTERACTIVE M	manhattan ny
0.522269095184392	AMS TECHNOLOGIES	manhattan ny
0.522269095184392	EASTERN ELECTRONICS CORP	manhattan ny
0.493439926229466	SPRINT SPECTRUM	64th st manhattan ny

sim	name	address	tid1	tid2
0.540166767574879	WORLDSCO	110 wall st new york ny	1	90213
0.540166767574879	WORLDSCO	110 wall st new york ny	1	90740
0.482300726524807	MANHATTAN CENTER FOR	e 116th manhattan ny	1	97312
0.447666521564224	WORLDCOM POP	750 e main st chattanooga tn	1	28302
0.445944986954701	MANHATTAN COMMUNICATION	963 manhattan ave brooklyn ny	1	96276
0.438408539297653	ALS CAGE AT MANHATTAN	193 manhattan ave new york ny	1	87652

sim	name	address
0.568322017929254	WORLDSCO	110 wall st new york ny
0.568322017929254	WORLDSCO	110 wall st new york ny
0.481778518568938	WORLDSCO	111 8th ave new york ny
0.462023103220271	WORLDSCO	60 hudson st new york ny
0.461736895695425	WORLDSCOMM	140 west st new york ny
0.461223519217079	WORLDCOM POP	750 e main st chattanooga tn
0.45599984346651	LIRR	w 33rd st manhattan ny
0.43316429737569	WORLDCOM	600 s federal st chicago il
0.425581077229952	TC PAYPHONES	w houston st manhattan ny
0.424860726905206	TC PAYPHONES	w 30th st manhattan ny



Functionality Need: Semantic Equivalences

- Problem: same real-world entity has distinct representations
 - “1 ATT Way Bedminster NJ” vs “900 Route 202/206 Bedminster NJ”
 - ‘MCI’ vs “Worldcom”
 - since few tokens shared, not caught by flexible string matching
- Solution: enhance flexible string matching
 - specify semantic equivalences in a binary table S
 - additionally perform flexible string joins with S
 - pre-processing: join with S , augment database string tokens
 - query time: augment query string tokens, do flexible string matching

Semantic Equivalences: Experiments

- Synonym table with address equivalences
 - “Route 25 Forest Hills NY” vs “Queens Blvd Queen NY”
- Query: “4001 Rte 25 Forrest Hills NY”
- Results: robust despite poor data quality
 - in synonym table: “Queen NY” vs “Queens NY”
 - in query: “Forrest” vs “Forest”, “Rte 25” vs “Route 25”

	sim	address	tid1	tid2
Explain sim	0.752483612033064	4001 queens blvd queens ny	6	76542
Explain sim	0.701585125233138	4011 queens blvd queens ny	6	76364
Explain sim	0.692897932355297	queens ny	6	76627
Explain sim	0.692897932355297	queens ny	6	76630
Explain sim	0.692897932355297	queens ny	6	25573
Explain sim	0.692897932355297	queens ny	6	46546

Performance Enhancements: Indexing BaseWeights

- Query to find flexible matches above threshold T

```
BaseWeights(tid, token, weight)      select S.tid, B.tid, SUM(S.weight*B.weight)
SearchWeights(tid, token, weight)    from SearchWeights S, BaseWeights B
                                     where S.token = B.token
                                     group by S.tid, B.tid
                                     having SUM(S.weight*B.weight) > T
```

- Primary key of BaseWeights is (tid, token), join is by token

Table size	Running time (sec)	
	NonIndexed	Indexed
100,000	2	1
7,000,000	48	22
13,000,000	105	42

Performance Enhancements: Pre-selecting Tuples

- Even with indexing, flexible string matching takes time
 - 42sec on 13,000,000 row base table
 - cosine similarity computed with every tuple having a shared token
- Pre-selecting high weight tuples from BaseWeights
 - add condition “B.tid in (SubQuery)” to where clause
 - recall may suffer
 - preserves similarity scores and perfect precision

Performance Enhancements: Pre-selecting Tuples

- Intuition for class of optimizations

- cosine similarity is sum of terms, each term is product of token weights
- pick one or many, high weight token or high weight term

01: High Weight Token

```
select B.tid
from SearchWeights S, BaseWeights B
where B.token = S.token and B.weight > T*F
```

02: High Weight Term

```
select B.tid
from SearchWeights S, BaseWeights B
where B.token = S.token and B.weight > T*G/S.Weight
```

03: Many High Weight Tokens

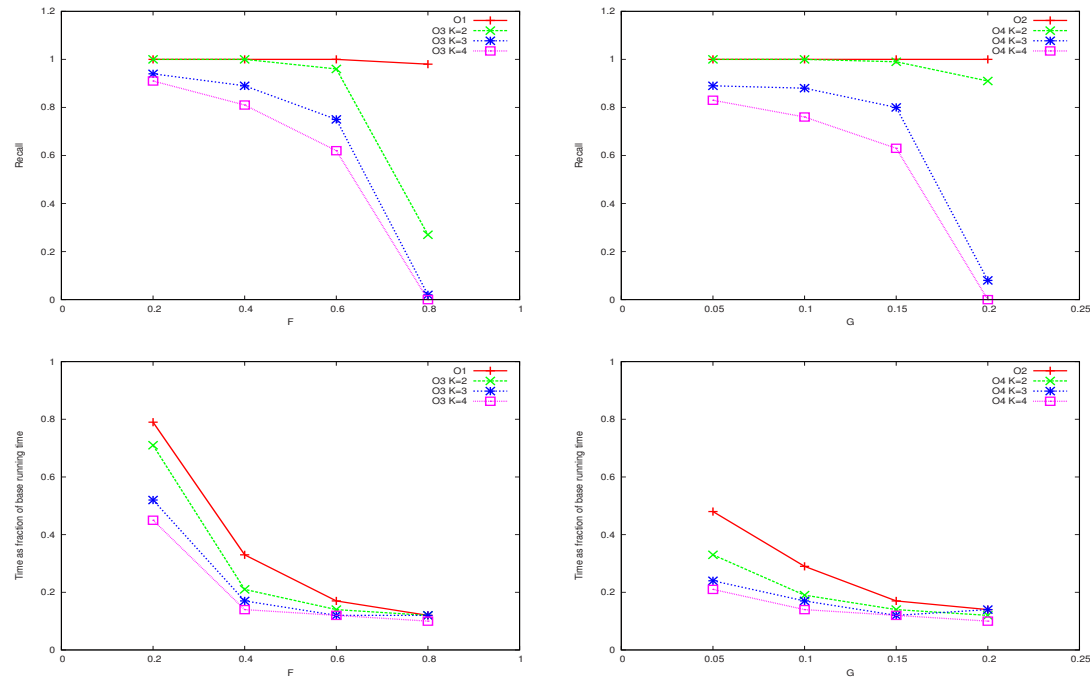
```
select B.tid
from SearchWeights S, BaseWeights B
where B.token = S.token and B.weight > T*F
group by B.tid
having count(*) >= K
```

04: Many High Weight Terms

```
select B.tid
from SearchWeights S, BaseWeights B
where B.token = S.token and B.weight > T*G/S.Weight
group by B.tid
having count(*) >= K
```

Pre-selecting Tuples: Experiments

- Data: company names table with 13 million rows
- Results: much smaller running time with small loss in recall



Open Issues

- Functionality: multi-table, multi-attribute flexible matching
 - issue: can one avoid materializing the multi-table join?
- Functionality: handling semantic dissimilarities
 - issue: principled techniques for semantic negations
- Functionality: numeric data embedded in strings
 - issue: need to handle string context and numeric order
- Performance: dynamically updated databases
 - issue: balance tf.idf accuracy with efficiency

Conclusions

- Technical contributions of paper
 - dynamic weighting for tf.idf on multiple attributes
 - extending tf.idf to incorporate semantic equivalences
 - optimizations that trade off recall for efficiency
- Techniques implemented in Spider system, used at AT&T
- Ongoing work: functionality and performance enhancements