Tree-Pattern Similarity Estimation for Scalable Content-based Routing

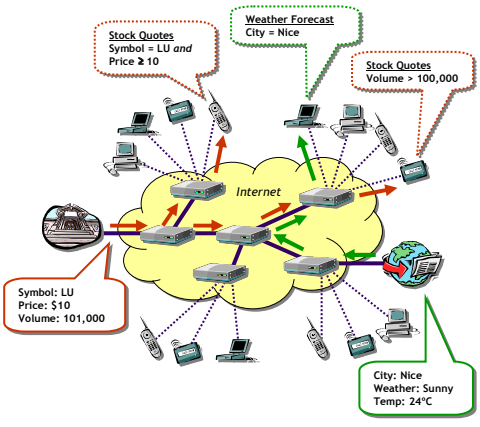
R. Chand
University of Geneva
Raphael.Chand@cui.unige.ch
http://cui.unige.ch/

Based on joint work with:
P. Felber (University of Neuchatel)
M. Garofalakis (Yahoo Research)

UNIVERSITÉ DE GENÈVE
CENTRE UNIVERSITAIRE D'INFORMATIQUE

Content-based publish/subscribe

- Consumers register **subscriptions**
- Producers publish **events** (messages)
- Messages are routed to interested consumers
  - Interested ≡ message **matches** subscription
- Matching based on the **content** of messages

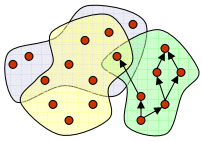


Broker-based approach

- Fixed infrastructure of reliable brokers
- (Subset of) subscriptions stored at brokers in routing tables
  - Typically takes advantage of "containment" relationship
- Filtering engine matches message against subscriptions to determine next hop(s)
- **Cons:** dedicated infrastructure, large routing tables, complex filtering algorithms

P2P approach

- Gather consumers in semantic communities according to interests (subscriptions)
- Disseminate messages in community & stop when reaching boundaries
- **Pros:** broker-less, space-efficient, low filtering cost



- **Challenge**: identify subscription proximity

  "are two distinct subscriptions likely to match the same set of documents?"

Problem statement

Given

S: valid tree patterns (subscriptions)

D: valid documents

$p, q \in S$

compute the similarity between *p* and *q*

$p \sim q$ $\sim: S^2 \rightarrow [0,1]$

(probability that *p* matches the same subset of *D* as *q*)

- Algorithms use
  - $H \subset D$: historical data about document stream
  - *k*: space bound

Basic approach

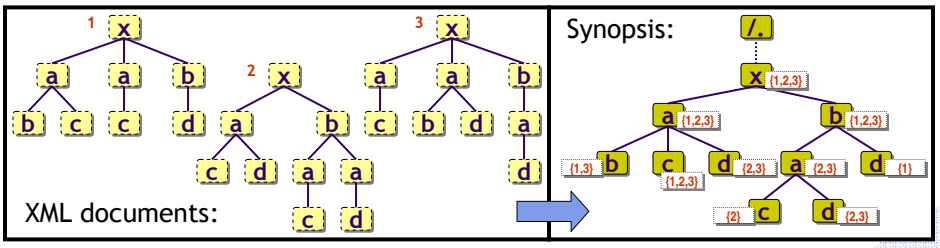
1. **Summarize** the document stream
   - Synopsis maintained incrementally
   - Accurate yet compact (compression, pruning)
2. Evaluate **selectivity** of tree pattern using synopsis
   - Recursive algorithm matches TP against synopsis
3. Estimate **similarity** using various metrics
   - Similarity computed from selectivity

1. Document-tree synopsis

- Maintain a concise, accurate synopsis H_S
  - Built on-line as documents stream by
  - Captures the path distribution of documents in H
  - Captures **cross-pattern correlations** in the stream
    - p, q match the **same** documents (not just the same number)
  - Allows us to estimate the fraction of documents matching different patterns

1. Document-tree synopsis

- **Document-tree synopsis:** tree with paths labeled with **matching sets** (documents containing path)
  - Summary of path-distribution characteristics of documents
- Adding a document to the synopsis:
  - Trace each path from the root of the synopsis, updating the matching sets and adding new nodes where necessary

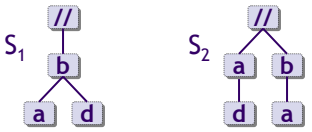


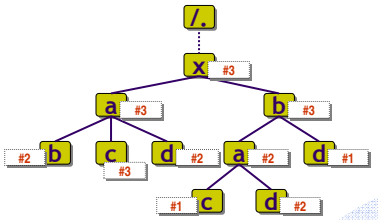
1. Matching set compression

- **Problem:** cannot maintain full matching set
  - With *N* documents: *O(N)*
- **Approach 1:** only maintain **document count**
  - Independence assumption unrealistic (no cross-pattern correlation)

    $P(S_1) = 2/3 * 1/3 = 2/9$ vs. **0**

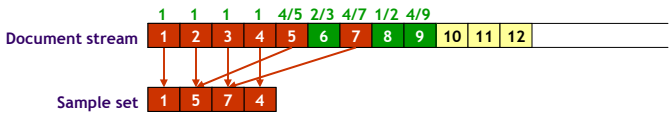
    $P(S_2) = 2/3 * 2/3 = 4/9$ vs. **2/3**





1. Matching set compression

- **Approach 2:** use **fixed-size sample sets**
  - Keep uniform sample of *s* documents
    [Vitter's reservoir-sampling scheme]
  - P(k^{th} document in synopsis) = min(1,*s*/*k*)



  - Once replaced, document ID deleted from whole tree
  - Sampling rate decided uniformly over all nodes

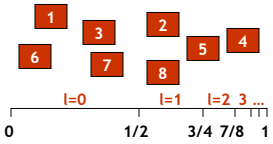
⇒ Inefficient utilization of the space budget

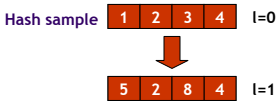
⇒ Poor estimates

1. Matching set compression

- **Approach 3:** use **per-node hash samples**
  - Gibbons' distinct sampling: hash function maps document IDs on logarithmic range of levels

    $\Pr[h(x) \geq l] = 1/2^l$
  - Hash samples start at level $l=0$, keep $d \Leftrightarrow h(d) \geq l$
  - Once sample is full, increment level and "sub-sample"





  - Fine sampling granularity, keep low frequency paths

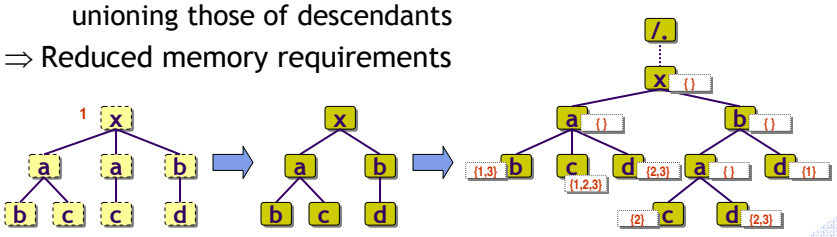
⇒ Much better estimates

⇒ Good utilization of the space budget

1. Matching set compression

- **Approach 3:** (cont'd)
  - Computing union/intersection: sub-sample lower level to higher prior to union/intersection, then possibly once more
  - Estimate cardinality of sample with n elements: $n2^l$
  - Only need to store document ID in hash samples at final nodes of incoming paths
    - Matching set of parent can be reconstructed by recursively unioning those of descendants

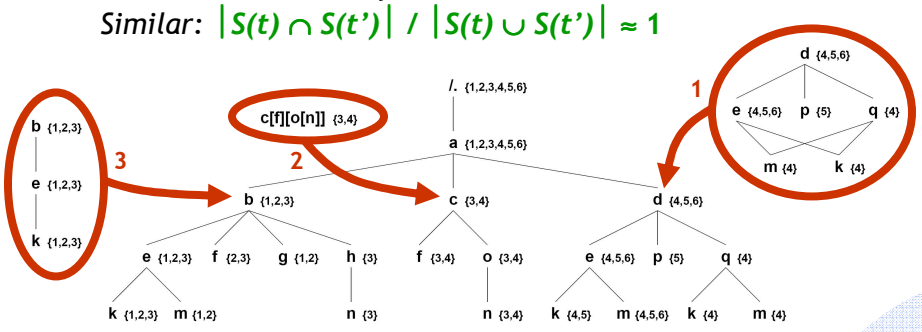
⇒ Reduced memory requirements



1. Synopsis pruning

- Synopsis may grow very large (due to path diversity)
  ⇒ **Prune** nodes with little influence on selectivity estimation
  1. Merge same-label leaf nodes with high similarity
  2. Fold leaf nodes in parent with high similarity
  3. Delete low-cardinality nodes

  Similar: $|S(t) \cap S(t')| \;/\; |S(t) \cup S(t')| \approx 1$



2. Evaluate selectivity

- Recursive algorithm matches TP against synopsis

Algorithm 1 Recursive selectivity function: $\text{SEL}(v, u)$

u: node of tree pattern *p*
v: node of synopsis

1: **if** $label(v) \not\geq label(u)$ **then** — ≥: at least as general (// ≥ * ≥ a)
2: $\text{SEL}(v, u) = \varnothing$ — ⇒ No matching path
3: **else if** u is a leaf **then** — Found matching path for *root*→*u* in synopsis
4: $\text{SEL}(v, u) = S(v)$ — ⇒ Return matching set
5: **else if** $label(u) \neq //$ **then** — Leaf of synopsis, not leaf of TP
6: **if** v is a leaf **then**
7: $\text{SEL}(v, u) = \varnothing$ — ⇒ No matching path
8: **else**
9: $\text{SEL}(v, u) = \bigcap_{u' \in Children(u)} \bigcup_{v' \in Children(v)} \text{SEL}(v', u')$ — Look for any path in synopsis for each branch of TP
10: **end if**
11: **else**
12: $S_0 = \bigcap_{u' \in Children(u)} \text{SEL}(v, u')$ — $\{label(u) = //\}$ Maps // to empty path…
13: $S_{\geq 1} = \bigcup_{v' \in Children(v)} \text{SEL}(v', u)$ — …or to path of length ≥2
14: $\text{SEL}(v, u) = S_0 \cup S_1$
15: **end if**

For *Counters*
∪→*max*
∩→*

Algorithm 2 Selectivity function: $P(p)$

1: $\text{P}(p) = |\text{SEL}(r_s, r_p)|/|S(r_s)|$ — Selectivity is # of matching documents / # total documents

3. Estimate similarity

- Metrics to estimate similarity using selectivity
  - Conditional probability of *p* given *q* (if *p* and *q* match the same set of documents as *q* alone, then *p* ~ *q*)

$$M_1(p,q) = P(p|q) = \frac{P(p \wedge q)}{P(q)}$$

  - Symmetrical conditional probability

$$M_2(p,q) = \frac{P(p|q) + P(q|p)}{2}$$

  - Ratio of joint to union probability (also symmetric)

$$M_3(p,q) = \frac{P(p \wedge q)}{P(p \vee q)} = \frac{P(p \wedge q)}{P(p) + P(q) - P(p \wedge q)}$$

$P(p \wedge q)$ computed by merging root nodes of *p* and *q*

Evaluation: setup

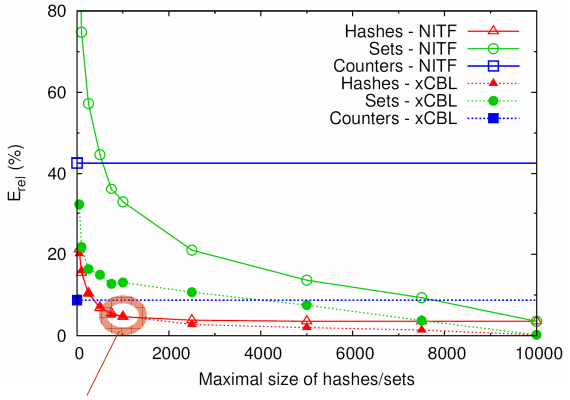
- NITF and xCBL DTDs
  - *D*: 10,000 documents with approx. 100 tag pairs, 10 levels
  - S_p: 1,000 "positive" TPs (some match in *D*)
    * (10%), // (10%), branches (10%), ≤10 levels, Zipf skew (1)
  - S_n: 1,000 "negative" TPs (no match in *D*)
- Synopses with 3 variants for **matching sets**
- Different **space budgets** (sizes of matching sets, compression degrees for pruning)
- Compare result of **proximity metrics** with exact value computed from sets of matching documents

Evaluation: error metrics

- Let
  - *P(p)*: exact selectivity of *p*
  - *P'(p)*: our estimate of the selectivity of *p*
  - $M_i(p,q)$: exact proximity of *p* and *q* using metric M_i
  - $M'_i(p,q)$: our estimate of the proximity of *p* and *q* using M_i
- Positive error: $$E_{rel} = \frac{1}{|S_P|} \sum_{p \in S_P} \frac{|P'(p) - P(p)|}{P(p)}$$
- Negative error: $$E_{sqr} = \sqrt{\frac{1}{|S_N|} \sum_{p \in S_N} (P'(p) - P(p))^2}$$
- Metrics error: $$E_{rel}(M_i) = \frac{1}{|S_P|^2} \sum_{p,q \in S_P} \frac{|M'_i(p,q) - M_i(p,q)|}{M_i(p,q)}$$

Positive error vs. MS size

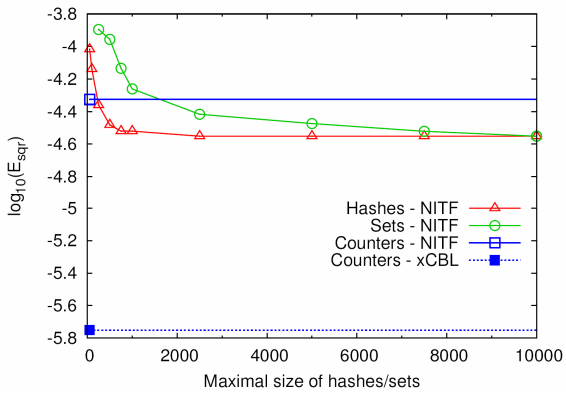
Hashes **outperforms other approaches in terms of accuracy**



Less than 5% with 1,000 entries

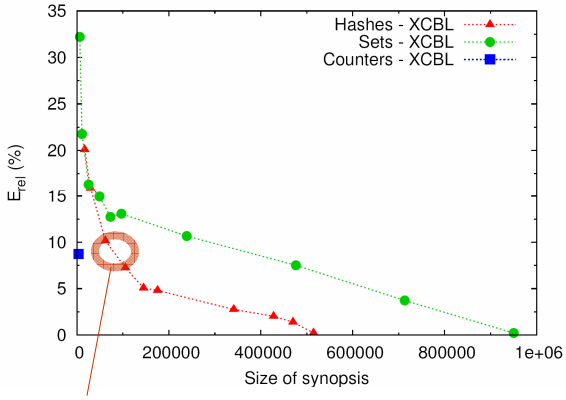
Negative error vs. MS size

***Hashes* also outperforms other approaches (no error with xCBL for *Hashes* & *Sets*)**



Positive error vs. synopsis size

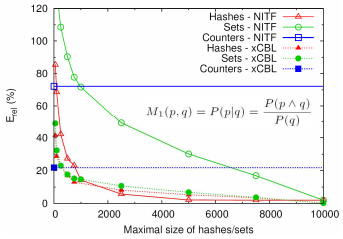
For a given space budget, *Hashes* is the most accurate (after some threshold)

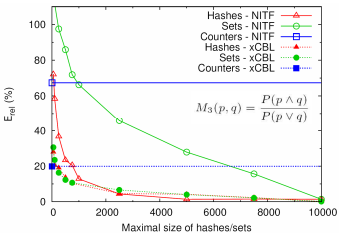


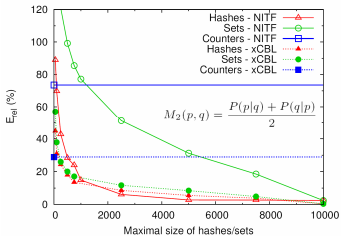
Hashes* becomes more accurate than *Counters

Error of proximity metrics

Hashes **produces the best estimates**

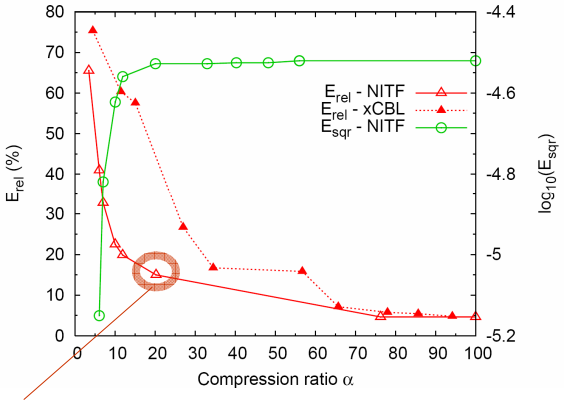






Error vs. compression ratio

Error remains small even for relatively high compression degrees



Less than 15% error with 1:5 compression

Conclusion

- **Goal:** semantic communities for publish/subscribe
- **Problem:** estimate similarity of (seemingly unrelated) tree patterns
- Similarity metric is very accurate and consistent
- Other usages (e.g., approximate XML queries)

Thank You!

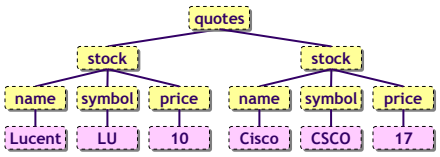
Context: XML (messages)

- Extensible Markup Language: *universal* interchange (meta-)language, standard, semi-structured
- Type/structure (tags, defined by DTD or schema) vs. content (values, data associated with tags)
- *Well-formed:* syntactically correct
- *Valid:* matches DTD or schema
- XML documents : single-rooted trees

```
<quotes>
  <stock>
    <name>Lucent Tech.</name>
    <symbol>LU</symbol>
    <price>10</price>
  </stock>
  <stock>
    <name>Cisco Systems, Inc.</name>
    <symbol>CSCO</symbol>
    <price>17</price>
  </stock>
</quotes>
```

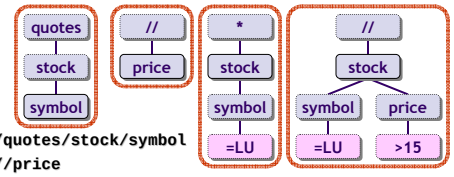
Start/End Tags (properly nested)

Character Data

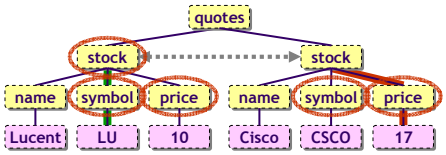


Context: XPath (subscriptions)

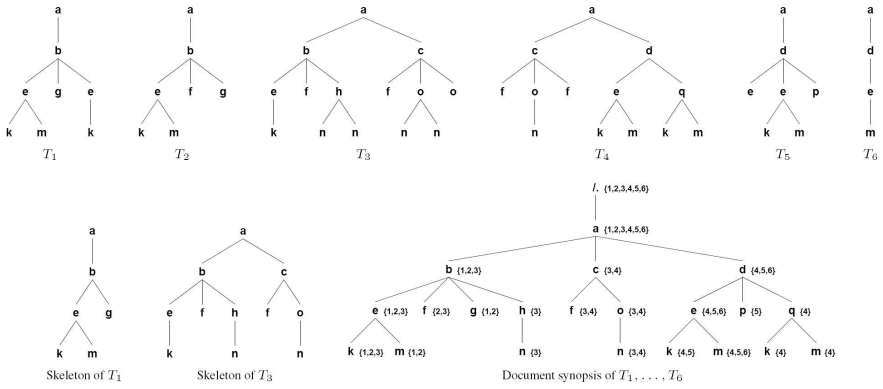
- Simple language: navigate/select parts of XML tree
- XPath Expression: sequence of node tests, child (/), descendant (//), wildcard (*), qualifiers ([...])
  - Constraints on *structure* **and** *content* of messages
  - Using qualifiers, define tree pattern: specifies *existential condition* for paths with *conjunctions at branching nodes*
- XPath fragment, binary output: selection ⇒ match



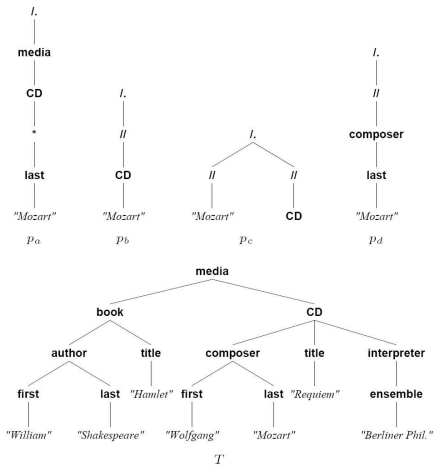

```
/quotes/stock/symbol
//price
/*/stock[symbol="LU"]
//stock[price>15][symbol="LU"]
```



Document-tree synopsis



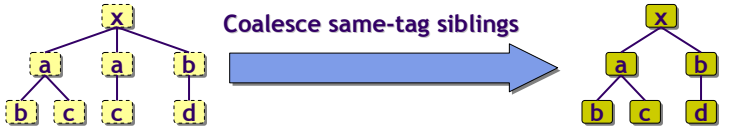
Tree patterns and XML tree



1. Document-tree synopsis

- **Document-tree synopsis:** tree with paths labeled with **matching sets** (IDs of documents containing that path)
  - Summary of path-distribution characteristics of documents
- Adding a document to the synopsis:

I) Identify distinct document paths ⇒ skeleton-tree



1. Document-tree synopsis

II) Install all skeleton-tree paths in synopsis
  - Trace each path from the root of the synopsis, updating the matching sets and adding new nodes where necessary