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)

Content-based publish/subscribe

- Consumers register subscriptions
- Producers publish events (messages)
- Messages are routed to interested consumers
  - Interested \iff 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 $q$

$$p - q \quad \text{-- : } S^2 \to [0,1]$$

(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)
    - $P(S_1) = 2/3 \times 1/3 = 2/9$ vs. 0
    - $P(S_2) = 2/3 \times 2/3 = 4/9$ vs. $2/3$

![Tree Diagram](image)

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) = $\min(1,s/k)$

![Sample Sets and Document Stream](image)

- 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”

![Hash sample diagram]

- Fine sampling granularity, keep low frequency paths
  \( \Rightarrow \) Much better estimates
  \( \Rightarrow \) Good utilization of the space budget

1. Matching set compression (cont’d)

**Approach 3:**

- 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
  \( \Rightarrow \) Matching set of parent can be reconstructed by recursively unioning those of descendants

\( \Rightarrow \) 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')| = 1**

2. Evaluate selectivity

- Recursive algorithm matches TP against synopsis

**Algorithm 1** Recursive selectivity function: \(SEL(v, u)\)

1: If \(\text{label}(v) \neq \text{label}(u)\) then
2: \(SEL(v, u) = \emptyset\)
3: Else if \(u\) is a leaf then
4: \(SEL(v, u) = S(v)\)
5: Else if \(\text{label}(u) \neq \emptyset\) then
6: \(SEL(v, u) = \emptyset\)
7: Else
8: \(SEL(v, u) = \bigcap_{e' \in \text{Children}(u)} \bigcup_{e' \in \text{Children}(v)} SEL(e', u')\)
9: End if
10: Else if \(\text{label}(u) = \emptyset\) then
11: \(S_0 = \bigcap_{e' \in \text{Children}(v)} \bigcup_{e'' \in \text{Children}(u)} \bigcup_{e' \in \text{Children}(v)} SEL(e', u')\)
12: \(S_1 = \bigcup_{e' \in \text{Children}(v)} \bigcup_{e'' \in \text{Children}(u)} \bigcup_{e' \in \text{Children}(v)} SEL(e', u')\)
13: \(SEL(v, u) = S_0 \cup S_1\)
14: End if

**Algorithm 2** Selectivity function: \(P(p)\)

1: \(P(p) = |SEL(r_p, r_p) / |S(r_p)|\) 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 \( p \sim q \))
    \[
    M_1(p, q) = P(p|q) = \frac{P(p \land q)}{P(q)}
    \]
  - Symmetrical conditional probability
    \[
    M_2(p, q) = \frac{P(p|q) + P(q|p)}{2}
    \]
  - Ratio of joint to union probability (also symmetric)
    \[
    M_3(p, q) = \frac{P(p \land q)}{P(p \lor q)} = \frac{P(p \land q)}{P(p) + P(q) - P(p \land q)}
    \]

\( P(p \land q) \) 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%), \leq 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:
  \[
  E_{neg} = \sqrt{\frac{1}{|S_N|} \sum_{p \in S_N} (P'(p) - P(p))^2}
  \]

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

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>

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
  /*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:
    1) Identify distinct document paths ⇒ skeleton-tree

```
XML document
a b c c b d
b c c d

Skeleton-tree
a1 b
b c d
```

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