Approximation: Theory and Algorithms
Windowed $pq$-Grams: Approximate Joins for Unordered Trees

Nikolaus Augsten
Free University of Bozen-Bolzano
Faculty of Computer Science
DIS

Unit 12 – May 29, 2009

Outline

1. Motivation
2. Windowed $pq$-Grams for Data-Centric XML
   - Windowed $pq$-Grams
   - Tree Sorting
   - Forming Bases
3. Efficient Approximate Joins with Windowed $pq$-Gram
4. Experiments
5. Related Work
6. Conclusion

Motivation

How similar are two XML items?

Song Lyric Store
- album
- track
- title
- artist
- artist
- title

2000

CD Warehouse
- album
- track
- title
- artist
- title
- artist
- title

2000

Query: Give me all album pairs that represent the same music CDs.

How similar are two XML items?

- Standard solution $O(n^3)$: tree edit distance
  - Minimum number of node edit operations (insert, delete, rename) that transforms one ordered tree into the other.
- Problem: permuted subtrees are deleted/re-inserted node by node
Motivation

Ordered vs. Unordered Trees

**Ordered Trees**
- Sibling order matters
- Sibling order matters

**Unordered Trees**
- \( \text{data-centric XML} \)
- Sibling order ignored

---

Problem Definition

Find an effective distance for the approximate matching of hierarchical data represented as unordered labeled trees that is efficient for approximate joins.

---

Naive approaches that fail:
- Unordered tree edit distance: NP-complete
- Allow subtree move: NP-hard
- Compute minimum distance between all permutations: \( O(n!) \)
- Sort by label and use ordered tree edit distance: error \( O(n) \)

---

Outline

1. Motivation
2. Windowed \( pq \)-Grams for Data-Centric XML
   - Windowed \( pq \)-Grams
   - Tree Sorting
   - Forming Bases
3. Efficient Approximate Joins with Windowed \( pq \)-Gram
4. Experiments
5. Related Work
6. Conclusion

---

Our Solution: Windowed \( pq \)-Grams

- **Windowed \( pq \)-Gram**: small subtree with stem and base
- **Key Idea**: split unordered tree into set of windowed \( pq \)-grams that is
  - Not sensitive to the sibling order
  - Sensitive to any other change in the tree
- **Intuition**: similar unordered trees have similar windowed \( pq \)-grams
- **Systematic computation** of windowed \( pq \)-grams
  1. Sort the children of each node by their label (works OK for \( pq \)-grams)
  2. Simulate permutations with a window
  3. Split tree into windowed \( pq \)-grams
Implementation of Windowed \( pq \)-Grams

- Set of windowed \( pq \)-grams:
  \[
  \begin{array}{cccccccc}
  \text{a} & \rightarrow & * & * & * & * & * & a \\
  \text{b} & \rightarrow & * & a & * & a & a & a \\
  \text{c} & \rightarrow & b & \text{b} & \text{c} & * & \text{c} & \text{b} \\
  \text{d} & \rightarrow & * & \text{b} & * & \text{c} & \text{b} & \text{c} \\
  \text{e} & \rightarrow & \text{b} & \text{c} & \text{b} & * & \text{c} & \text{b} \\
  \end{array}
  \]

- Hashing: map \( pq \)-gram to integer:
  \[
  \begin{array}{cccc}
  \text{label} & l & h(l) \\
  \text{a} & 9 & 9 \\
  \text{b} & 7 & 3 \\
  \text{c} & 3 & \ldots \\
  \end{array}
  \]

Note: labels may be strings of arbitrary length!

- \( pq \)-Gram index: bag of hashed \( pq \)-grams
  \[
  I(T) = \{0973, 0970, 0930, 0937, 0907, 0903, 9316, 9310, 9360, 9361, 9301, 9306, 3100, 3600\}
  \]

Tree is represented by a bag of integers!

Outline

1. Motivation
2. Windowed \( pq \)-Grams for Data-Centric XML
   - Windowed \( pq \)-Grams
   - Tree Sorting
   - Forming Bases
3. Efficient Approximate Joins with Windowed \( pq \)-Gram
4. Experiments
5. Related Work
6. Conclusion

The Windowed \( pq \)-Gram Distance

- The windowed \( pq \)-gram distance between two trees, \( T \) and \( T' \):
  \[
  \text{dist}^{pq}(T, T') = |I(T) \cup I(T')| - 2|I(T) \cap I(T')|
  \]

- Pseudo-metric properties hold:
  - self-identity: \( x = y \iff \text{dist}^{pq}(x, y) = 0 \)
  - symmetry: \( \text{dist}^{pq}(x, y) = \text{dist}^{pq}(y, x) \)
  - triangle inequality: \( \text{dist}^{pq}(x, z) \leq \text{dist}^{pq}(x, y) + \text{dist}^{pq}(y, z) \)

- Different trees may be at distance zero:
  - \( b' \)
  - \( b \)

- Runtime for the distance computation is \( O(n \log n) \).

Sorting the Tree?

- Idea:
  1. sort the children of each node by their label
  2. apply an ordered tree distance

- \(\times\) Edit distance: tree sorting does not work

- \(\checkmark\) Windowed \( pq \)-Grams: tree sorting works OK
1. Non-unique sorting: edit distance $O(n)$ for identical trees

2. Node renaming: edit distance depends on node label

Theorem (Local Effect of Node Reordering)

If $k$ children of a node are reordered, i.e., their subtrees are moved, only $O(k)$ windowed $pq$-grams change.

Proof (idea):
- $pq$-grams consist of a stem and a base
- stems are invariant to the sibling order
- bases: only the $O(k)$ $pq$-grams with the reordered nodes in the bases change

✓ Non-unique sortings are equivalent: distance is 0 for identical trees
✓ Node renaming is independent of the node label
Goal for windowed \( pq \)-grams:
- not sensitive to the sibling order
- sensitive to any other change in the tree

Stems: ignore sibling order
\[
\begin{align*}
\text{stem} \quad p &= 2 \\
\text{base} \quad q &= 3
\end{align*}
\]

Bases: do not ignore sibling order!

Requirements for bases:
- detection of node moves
- robustness to different sortings
- balanced node weight

Our solution:
- windows: simulate all permutations within a window
- wrapping: wrap windows that extend beyond the right border
- dummies: extend small sibling sets with dummy nodes

Algorithm 1: Form bases from a sorted sibling sequence
1. if sibling sequence < window then extend with dummy nodes;
2. initialize window: start with leftmost node;
3. repeat
4. form bases in window: all \( q \)-permutations that contain start node;
5. shift window to the right by one node;
6. if window extends the right border then wrap window;
7. until processed all window positions

Example: stem, sorted sibling sequence, window \( w = 3 \)
\[
\begin{array}{ccccccc}
a & a & a & a & a & a & a \\
b & c & c & c & c & c & c \\
d & e & d & e & * & e & d & * & e
\end{array}
\]

Theorem (Optimal Windowed \( pq \)-Grams)
For trees with fanout \( f \), windowed \( pq \)-grams with base size \( q = 2 \) and window size \( w = \frac{f + 1}{2} \) have the following properties:

1. Detection of node moves:
   - base recall \( \rho = 1 \) (all sibling pairs are encoded)
   - base precision \( \pi = 1 \) (each pair is encoded only once)

2. Robustness to different sortings: \( (k \) edit operations)
   - base error \( \epsilon \leq \frac{2k}{f} \)

3. Balanced node weight:
   Each non-root node appears in exactly \( 2w - 2 \) bases.
**Illustration: Detection of Node Moves**

- **Single Node:** each node forms a base of size $q = 1$
- **Window:** $q \geq 2$ nodes of a window form a base

![Diagram of single node move]

**Goal:** bases must change

<table>
<thead>
<tr>
<th>Single Node</th>
<th>No bases change</th>
</tr>
</thead>
<tbody>
<tr>
<td>c, d, e</td>
<td>c, d, e</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Window</th>
<th>33% bases change</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd, c*, d*, dc</td>
<td>c*, c*, **, *c, *c, **, de, ...</td>
</tr>
</tbody>
</table>

Windowed $pq$-grams detect node moves.

**Illustration: Robustness to Different Sortings**

- **Consecutive siblings** form a base (no permutation)
- **Window:** all sibling permutations within the window form bases

![Diagram of consecutive sortings]

**Goal:** Same number of bases change for both sortings.

<table>
<thead>
<tr>
<th>Consecutive</th>
<th>Sort A</th>
<th>100% bases change</th>
<th>Sort B</th>
<th>50% bases change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ab bc</td>
<td>ac cd</td>
<td>ad db</td>
<td>ad dc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Window</th>
<th>33% bases change</th>
</tr>
</thead>
<tbody>
<tr>
<td>ad ab</td>
<td>ab db ...</td>
</tr>
</tbody>
</table>

Windowed $pq$-grams: Robust to different sortings.

**Illustration: Balancing the Node Weight**

- **Permutations:** all permutations of size $q$ form a base
- **Window:** only permutations within window form a base

![Diagram of permutation changes]

**Goal:** Same number of bases change for both renames.

<table>
<thead>
<tr>
<th>Permutations</th>
<th>60/137 bases change</th>
<th>6/137 bases change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x e f g h i m n o</td>
<td>x e f g h i m n o</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Window</th>
<th>12/51 bases change</th>
<th>12/51 bases change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d e f g h i m n o</td>
<td>d e f g h i m n o</td>
</tr>
</tbody>
</table>

Winnowed $pq$-grams: Node weight is independent of sibling number.

**Outline**

1. Motivation
2. Windowed $pq$-Grams for Data-Centric XML
   - Windowed $pq$-Grams
   - Tree Sorting
   - Forming Bases
3. Efficient Approximate Joins with Windowed $pq$-Gram
4. Experiments
5. Related Work
6. Conclusion
Efficient Approximate Joins with Windowed pq-Gram

**Approximate Join**

<table>
<thead>
<tr>
<th>tid</th>
<th>tree</th>
<th>F</th>
<th>threshold=2</th>
<th>F'</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a</td>
<td></td>
<td>b</td>
</tr>
<tr>
<td>T1</td>
<td>x</td>
<td>y</td>
<td>w</td>
<td>v</td>
</tr>
<tr>
<td>T2</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e</td>
<td></td>
<td>f</td>
</tr>
<tr>
<td>T3</td>
<td>a</td>
<td>b</td>
<td>h</td>
<td></td>
</tr>
</tbody>
</table>

**Simple approach:** distance join
1. compute distance between all pairs of trees
2. return document pairs within threshold

**Very expensive:** $N^2$ distance computations!

**Usual Join Optimization Does not Apply**

- **Distance join:** expensive
  - nested loop join: evaluate distance function between every input pair
- **Equality join:** efficient
  - implementation as sort-merge or hash join
- **Sort-merge and hash join:**
  - **first step:** treat each join attribute in isolation (sort/hash)
  - **second step:** evaluate equality function
- **Sort-merge and hash not applicable to distance join:**
  - there is no sorting that groups similar trees
  - there is no hash function that partitions similar trees into buckets
- **Solution:** reduce distance join to equality join on pq-grams

**Reducing a Distance Join to an Equality Join**

**Distance join** between trees: $N^2$ intersections between integer bags

- $\{1,7\}_a \times \{1,7\}_d$: $|a \cap d| = 2$
- $\{1,0\}_b \times \{5,5\}_e$: $|b \cap d| = 1$
- $\{4,6\}_c \times \{0,8\}_f$: $|c \cap d| = 0$

**Optimized pq-gram join:** empty intersections are never computed!

1. **union**
   - $\{1_d, 7_d, 1_b, 0_b, 4_c, 6_c\}$, $\{1_d, 7_d, 5_e, 5_e, 0_r, 8_r\}$
2. **sort**
3. **merge-join**
   - $0_b \cup 0_r$
   - $1_d \cup 1_d$
   - $1_b \cup 5_e$
   - $4_c \cup 5_e$
   - $6_c \cup 7_d$
   - $7_d \cup 8_r$

**Outline**

1. **Motivation**
2. **Windowed pq-Grams for Data-Centric XML**
   - Windowed pq-Grams
   - Tree Sorting
   - Forming Bases
3. **Efficient Approximate Joins with Windowed pq-Gram**
4. **Experiments**
5. **Related Work**
6. **Conclusion**
Experiments

Effectiveness of the Windowed $pq$-Gram Join

Experiment: match DBLP articles
- add noise to articles (missing elements and spelling mistakes)
- approximate join between original and noisy data
- measure precision and recall for different thresholds

Datasets:
- **DBLP**: articles
  depth 1.9, 15 nodes (max 1494 nodes)
- **SwissProt**: protein descriptions
  depth 3.5, 104 nodes (max 2640 nodes)
- **Treebank**: tagged English sentences
  depth 6.9 (max depth 30), 43 nodes

Windowed $pq$-grams are effective for data-centric XML

Efficiency of the Optimized $pq$-Gram Join

Optimized $pq$-gram join: very efficient

- compute nested-loop join between trees
- compute optimized $pq$-gram join between trees
- measure wallclock time

Outline

1 Motivation
2 Windowed $pq$-Grams for Data-Centric XML
   - Windowed $pq$-Grams
   - Tree Sorting
   - Forming Bases
3 Efficient Approximate Joins with Windowed $pq$-Gram
4 Experiments
5 Related Work
6 Conclusion

Distances between Unordered Trees

Edit Distances between Unordered Trees
- [ZSS92]: proof for NP-completeness
- [KKSS04]: lower bound for a restricted edit distance
- [CGM97]: $O(n^3)$ heuristics
- Our solution: $O(n \log n)$ heuristics

Approximate Join
- [GIJ+01]: efficient approximate join for strings
Windowed pq-Grams

Windowed pq-grams for unordered trees:
- \(O(n \log n)\) heuristics for NP-complete edit distance
- **Key problem:** all permutations must be considered
- **Our heuristics:** sort trees and simulate permutations with window
- **Sorting:** works for pq-grams, but not for edit distance
- **Window technique** guarantees core properties
  - detection of node moves
  - robustness to different sortings
  - balanced node weight
- **Efficient approximate join:** reduces distance join to equality join

### What’s Next?

- **Project Presentation**
  - Mai 29 (today), 2pm-3pm
  - Implementation demo: 5 minutes
  - Presentation: 10 minutes
  - Discussion: 5-10 minutes
- **Project Report**
  - Deadline: Mai 29 (today)
- **Final Exam**
  - Wed, Jun 23, 11am-1pm
  - 15 minutes oral exam
  - detailed exam procedure are published
  - exam questions will be published soon

---

Sudarshan S. Chawathe and Hector Garcia-Molina.
Meaningful change detection in structured data.

Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava.
Approximate string joins in a database (almost) for free.

Karin Kailing, Hans-Peter Kriegel, Stefan Schönauer, and Thomas Seidl.
Efficient similarity search for hierarchical data in large databases.

Kaizhong Zhang, Richard Statman, and Dennis Shasha.
On the editing distance between unordered labeled trees.